Medical Imaging ToolKit Reference Manual

Medical Image Processing Group
Key Laboratory of Complex Systems and Intelligence Science
Institute of Automation, Chinese Academy of Sciences

Generated by Doxygen 1.4.3

Tue Feb 25 15:00:36 2014

Contents

1 MITK Introduction **1**

2 MITK Directory Hierarchy **3**

2.1 MITK (Medical Imaging ToolKit) Directories 3

3 MITK Namespace Index **5**

3.1 MITK (Medical Imaging ToolKit) Namespace List 5

4 MITK Hierarchical Index **7**

4.1 MITK (Medical Imaging ToolKit) Class Hierarchy 7

5 MITK Class Index **15**

5.1 MITK (Medical Imaging ToolKit) Class List 15

6 MITK Directory Documentation **27**

6.1 Additional/ Directory Reference 27

6.2 Additional/mitkCUDAVolumeRendering/Include/ Directory Reference 28

6.3 mitkSegmentation/Include/ Directory Reference 29

6.4 mitkVisualization2/Include/ Directory Reference 30

6.5 mitkVisualization/Include/ Directory Reference 31

6.6 mitkCommon/Include/ Directory Reference 34

6.7 mitkRegistration/Include/ Directory Reference 37

6.8 Additional/mitkReconstructionCUDA/Include/ Directory Reference 38

6.9 mitkReconstruction/Include/ Directory Reference 39

6.10 Additional/mitkReconstructionCG/Include/ Directory Reference 41

6.11 mitkCommon/ Directory Reference 42

6.12 Additional/mitkCUDAVolumeRendering/ Directory Reference 43

6.13 mitkReconstruction/ Directory Reference 44

6.14 Additional/mitkReconstructionCG/ Directory Reference 45

6.15 Additional/mitkReconstructionCUDA/ Directory Reference 46

6.16 mitkRegistration/ Directory Reference . . . 47
6.17 mitkSegmentation/ Directory Reference . . . 48
6.18 mitkVisualization/ Directory Reference . . . 49
6.19 mitkVisualization2/ Directory Reference . . . 50
6.20 Additional/mitkCUDAVolumeRendering/Include/modules/ Directory Reference . . . 51
6.21 mitkVisualization2/Include/modules/ Directory Reference . . . 52
6.22 Additional/mitkCUDAVolumeRendering/Include/modules/StateRayCaster/ Directory Reference . . . 53

7 MITK Namespace Documentation **55**
7.1 AnalyzeImageIO Namespace Reference . . . 55

8 MITK Class Documentation **59**
8.1 AnalyzeImageIO::data_history Struct Reference . . . 59
8.2 AnalyzeImageIO::dsr Struct Reference . . . 63
8.3 AnalyzeImageIO::header_key Struct Reference . . . 65
8.4 AnalyzeImageIO::image_dimension Struct Reference . . . 67
8.5 mitk2DBackProjectionUsingGPU Class Reference . . . 71
8.6 mitk2DBeamICPD Class Reference . . . 73
8.7 mitk2DBeamOoCPD Class Reference . . . 76
8.8 mitk2DBeamPD Class Reference . . . 79
8.9 mitk3DBackProjectionUsingGPU Class Reference . . . 82
8.10 mitk3DUSPixelBasedGPU Class Reference . . . 84
8.11 mitk3DUSPixelBasedMT Class Reference . . . 86
8.12 mitk3DUSReconstructionFilter Class Reference . . . 89
8.13 mitkAffineTransform Class Reference . . . 91
8.14 mitkAmoebaOptimizer Class Reference . . . 94
8.15 mitkAnalyzeImageReader Class Reference . . . 97
8.16 mitkAnalyzeImageWriter Class Reference . . . 99
8.17 mitkAngleWidgetModel2D Class Reference . . . 101
8.18 mitkAngleWidgetModel3D Class Reference . . . 108
8.19 mitkBackProjection Class Reference . . . 113
8.20 mitkBackProjectionUsingGPU Class Reference . . . 115
8.21 mitkBarrier Class Reference . . . 117
8.22 mitkBinaryFilter Class Reference . . . 119
8.23 mitkBinMarchingCubes Class Reference . . . 121
8.24 mitkBMPReader Class Reference . . . 124
8.25 mitkBMPWriter Class Reference . . . 127

8.26 mitkBSplineInterpolateFilter Class Reference . . . 129
8.27 mitkBSPTree< BSPTreeNode > Class Template Reference . . . 131
8.28 mitkCacheVolumeReader Class Reference . . . 132
8.29 mitkCacheVolumeWriter Class Reference . . . 134
8.30 mitkCamera Class Reference . . . 136
8.31 mitkCBFDKReconCF Class Reference . . . 142
8.32 mitkCBFDKReconCFCPU Class Reference . . . 145
8.33 mitkCBFDKReconCFGPU Class Reference . . . 148
8.34 mitkCBTFDKReconCF Class Reference . . . 151
8.35 mitkClippingPlaneWidgetModel Class Reference . . . 154
8.36 mitkCoherentVolume Class Reference . . . 161
8.37 mitkColorTable Class Reference . . . 167
8.38 mitkColorTransferFunction Class Reference . . . 175
8.39 mitkConditionVariable Class Reference . . . 180
8.40 mitkConeBeamICPD Class Reference . . . 181
8.41 mitkConeBeamOoCPD Class Reference . . . 184
8.42 mitkConeBeamPD Class Reference . . . 187
8.43 mitkConeBeamProjectData Class Reference . . . 190
8.44 mitkConeBeamRealTimePD Class Reference . . . 196
8.45 mitkConeBeamReconFromHelix Class Reference . . . 200
8.46 mitkConnectivityLabelFilter Class Reference . . . 204
8.47 mitkCTProjectionData Class Reference . . . 206
8.48 mitkCTReconstructionFilter Class Reference . . . 209
8.49 mitkCubicInterpolateFilter Class Reference . . . 213
8.50 mitkCUDAAugmentedSurfaceVRScene Class Reference . . . 215
8.51 mitkCUDADecomposedIsosurfaceVRScene Class Reference . . . 222
8.52 mitkCUDADeltaAccStateVRScene Class Reference . . . 229
8.53 mitkCUDAEndoscopySliceScene Class Reference . . . 237
8.54 mitkCUDAFullVRScene Class Reference . . . 242
8.55 mitkCUDAIsosurfaceVRScene Class Reference . . . 250
8.56 mitkCUDALut1D Class Reference . . . 257
8.57 mitkCUDAMIPVRScene Class Reference . . . 258
8.58 mitkCUDAROIDistanceVRScene Class Reference . . . 261
8.59 mitkCUDASegmentedVRScene Class Reference . . . 270
8.60 mitkCUDAStructureFunctionVRScene Class Reference . . . 278
8.61 mitkCUDAValueBasedStateVRScene Class Reference . . . 283

8.62 mitkCUDAVersatileVRScene Class Reference 292
8.63 mitkDataModel Class Reference 299
8.64 mitkDataObject Class Reference 302
8.65 mitkDICOMDIRReader Class Reference 305
8.66 mitkDICOMDIRTree Class Reference 307
8.67 mitkDICOMInfoReader Class Reference 309
8.68 mitkDICOMReader Class Reference 312
8.69 mitkDICOMWriter Class Reference 314
8.70 mitkDiffusionFilter Class Reference 317
8.71 mitkDirectionEncoder Class Reference 320
8.72 mitkDistanceTransformSaito Class Reference 322
8.73 mitkEllipseWidgetModel2D Class Reference 324
8.74 mitkEncodedGradientEstimator Class Reference 333
8.75 mitkEncodedGradientShader Class Reference 342
8.76 mitkEndoscopySliceScene Class Reference 343
8.77 mitkEuclideanDistanceMetric Class Reference 348
8.78 mitkEuclideanDistanceTransformFilter Class Reference 350
8.79 mitkEuclideanSkeletonFeature Class Reference 353
8.80 mitkEuclideanSkeletonFeatureExtraction Class Reference 355
8.81 mitkEuclideanSkeletonGraphCut Class Reference 358
8.82 mitkFanBeamProjectData Class Reference 361
8.83 mitkFanBeamReconFromCirle Class Reference 366
8.84 mitkFbFbp Class Reference 370
8.85 mitkFBFBPReconCF Class Reference 372
8.86 mitkFBFBPReconCFCPU Class Reference 375
8.87 mitkFBFBPReconCFGPU Class Reference 378
8.88 mitkFbFbpSpeedUsingCpu Class Reference 381
8.89 mitkFbFbpSpeedUsingGpu Class Reference 383
8.90 mitkFDK Class Reference 386
8.91 mitkFDKSpeedUsingCpu Class Reference 388
8.92 mitkFDKSpeedUsingGpu Class Reference 390
8.93 mitkFFTFilter Class Reference 392
8.94 mitkFilter Class Reference 394
8.95 mitkFiniteDifferenceGradientEstimator Class Reference 396
8.96 mitkFootprint Class Reference 398
8.97 mitkFootprint1D Class Reference 401

8.98 mitkFootprint1DGaussian Class Reference . . . 403
8.99 mitkFootprint2D Class Reference . . . 407
8.100 mitkFootprint2DGaussian Class Reference . . . 409
8.101 mitkFullVRScene Class Reference . . . 413
8.102 mitkGaussianDerivativeImageFilter Class Reference . . . 421
8.103 mitkGaussianFilter Class Reference . . . 423
8.104 mitkGLSLFullVRScene Class Reference . . . 425
8.105 mitkGLSLIsosurfaceVRScene Class Reference . . . 432
8.106 mitkGLSLLut1D Class Reference . . . 438
8.107 mitkGLSLMIPVRScene Class Reference . . . 439
8.108 mitkGradientDescentOptimizer Class Reference . . . 442
8.109 mitkHEICTriangleMesh Class Reference . . . 445
8.110 mitkHEMesh Class Reference . . . 452
8.111 mitkHEOoCTriangleMesh Class Reference . . . 470
8.112 mitkHETriangleMesh Class Reference . . . 481
8.113 mitkHistogram1D Class Reference . . . 486
8.114 mitkHistogram2D Class Reference . . . 490
8.115 mitkICConeBeamProjectData Class Reference . . . 495
8.116 mitkICFanBeamProjectData Class Reference . . . 498
8.117 mitkICParallelProjectData Class Reference . . . 501
8.118 mitkICTriangleMesh Class Reference . . . 504
8.119 mitkICVolume Class Reference . . . 508
8.120 mitkIM0Reader Class Reference . . . 516
8.121 mitkIM0Writer Class Reference . . . 518
8.122 mitkImageIOFactory Class Reference . . . 520
8.123 mitkImageModel Class Reference . . . 522
8.124 mitkImageScene Class Reference . . . 532
8.125 mitkImageView Class Reference . . . 541
8.126 mitkImageViewManipulatorStandard Class Reference . . . 547
8.127 mitkImageViewManipulatorWithWidgets Class Reference . . . 550
8.128 mitkImplementor Class Reference . . . 553
8.129 mitkInfoReader Class Reference . . . 554
8.130 mitkInterpolateFilter Class Reference . . . 556
8.131 mitkIsosurfaceVRScene Class Reference . . . 561
8.132 mitkIterativeClosestPointOptimizer Class Reference . . . 568
8.133 mitkJPEGReader Class Reference . . . 571

8.134 mitkJPEGWriter Class Reference . . . 574
8.135 mitkKdTree Class Reference . . . 576
8.136 mitkKdTreePointSet Class Reference . . . 578
8.137 mitkLandmarkRegistrationFilter Class Reference . . . 581
8.138 mitkLBFGSBOptimizer Class Reference . . . 583
8.139 mitkLBFGSOptimizer Class Reference . . . 586
8.140 mitkLight Class Reference . . . 588
8.141 mitkLinearInterpolateFilter Class Reference . . . 590
8.142 mitkLineWidgetModel2D Class Reference . . . 592
8.143 mitkLineWidgetModel3D Class Reference . . . 600
8.144 mitkList Class Reference . . . 605
8.145 mitkLiveWireImageFilter Class Reference . . . 607
8.146 mitkLut1D Class Reference . . . 609
8.147 mitkManipulator Class Reference . . . 610
8.148 mitkMarchingCubes Class Reference . . . 614
8.149 mitkMatrix Class Reference . . . 617
8.150 mitkMatrixD Class Reference . . . 620
8.151 mitkMeanSquaresMetric Class Reference . . . 623
8.152 mitkMesh Class Reference . . . 626
8.153 mitkMeshReader Class Reference . . . 636
8.154 mitkMeshToMeshFilter Class Reference . . . 639
8.155 mitkMeshViewManipulatorStandard Class Reference . . . 642
8.156 mitkMeshWriter Class Reference . . . 645
8.157 mitkMetaImageReader Class Reference . . . 647
8.158 mitkMetaImageWriter Class Reference . . . 649
8.159 mitkMetric Class Reference . . . 651
8.160 mitkMIPVRScene Class Reference . . . 656
8.161 mitkModel Class Reference . . . 660
8.162 mitkMutex Class Reference . . . 672
8.163 mitkNearestNeighborInterpolateFilter Class Reference . . . 674
8.164 mitkNeighborhood Class Reference . . . 676
8.165 mitkNetCDFImageReader Class Reference . . . 678
8.166 mitkNetCDFImageWriter Class Reference . . . 680
8.167 mitkNeumannBoundary Class Reference . . . 682
8.168 mitkNode Class Reference . . . 684
8.169 mitkNodeHeap Class Reference . . . 687

8.170 mitkNormalizedCorrelationMetric Class Reference . . . 689
8.171 mitkObject Class Reference . . . 692
8.172 mitkObserver Class Reference . . . 699
8.173 mitkOoCConeBeamProjectData Class Reference . . . 701
8.174 mitkOoCFanBeamProjectData Class Reference . . . 704
8.175 mitkOoCParallelProjectData Class Reference . . . 707
8.176 mitkOoCSurfaceRendererStandard Class Reference . . . 710
8.177 mitkOoCSurfaceRendererUseVA Class Reference . . . 712
8.178 mitkOoCTriangleMesh Class Reference . . . 714
8.179 mitkOoCVolume Class Reference . . . 720
8.180 mitkOoCVolumeRayCastCompositeFunction Class Reference . . . 730
8.181 mitkOoCVolumeRayCastCompositeFunctionEx Class Reference . . . 732
8.182 mitkOoCVolumeRendererRayCasting Class Reference . . . 734
8.183 mitkOoCVolumeRendererRayCastingEx Class Reference . . . 736
8.184 mitkOoCVolumeRendererShearWarp Class Reference . . . 739
8.185 mitkOoCVolumeRendererTexture3D Class Reference . . . 745
8.186 mitkOoCVolumeRendererTexture3DEx Class Reference . . . 748
8.187 mitkOoCVolumeShearFunction Class Reference . . . 751
8.188 mitkOoCVolumeShearParallel Class Reference . . . 753
8.189 mitkOoCVolumeShearPerspective Class Reference . . . 755
8.190 mitkOptimizer Class Reference . . . 757
8.191 mitkOrthogonalImageModel Class Reference . . . 763
8.192 mitkOtsuSegmentationFilter Class Reference . . . 766
8.193 mitkParallelBeamReconFromCirle Class Reference . . . 768
8.194 mitkParallelProjectData Class Reference . . . 771
8.195 mitkPbFbp Class Reference . . . 775
8.196 mitkPBFBPReconCF Class Reference . . . 777
8.197 mitkPickManipulator Class Reference . . . 780
8.198 mitkPlane Class Reference . . . 782
8.199 mitkPLYASCIIWriter Class Reference . . . 783
8.200 mitkPLYBinaryWriter Class Reference . . . 785
8.201 mitkPLYReader Class Reference . . . 787
8.202 mitkPointSet Class Reference . . . 789
8.203 mitkPointSetEstimator Class Reference . . . 801
8.204 mitkPointSetKernelTransformEstimator Class Reference . . . 805
8.205 mitkPointSetMetric Class Reference . . . 807

8.206 mitkPointSetQuaternionEstimator Class Reference . . . 811
8.207 mitkPointSetRegistrationFilter Class Reference . . . 813
8.208 mitkPolygonWidgetModel2D Class Reference . . . 818
8.209 mitkPowellOptimizer Class Reference . . . 825
8.210 mitkProcessObject Class Reference . . . 828
8.211 mitkProjectData Class Reference . . . 830
8.212 mitkProjectDataToVolumeFilter Class Reference . . . 834
8.213 mitkPseudocolorWidgetModel Class Reference . . . 838
8.214 mitkPseudocolorWidgetModelEx Class Reference . . . 845
8.215 mitkQEMSimplification Class Reference . . . 852
8.216 mitkQuaternion Class Reference . . . 855
8.217 mitkRawConeBeamProjectData Class Reference . . . 859
8.218 mitkRawData Class Reference . . . 863
8.219 mitkRawDataReader Class Reference . . . 866
8.220 mitkRawDataToVolumeFilter Class Reference . . . 868
8.221 mitkRawDataWriter Class Reference . . . 870
8.222 mitkRawFilesReader Class Reference . . . 872
8.223 mitkRawFilesWriter Class Reference . . . 882
8.224 mitkRawReader Class Reference . . . 885
8.225 mitkRawWriter Class Reference . . . 891
8.226 mitkRCPtr< T > Class Template Reference . . . 894
8.227 mitkReader Class Reference . . . 895
8.228 mitkRectPlane Class Reference . . . 897
8.229 mitkRectWidgetModel2D Class Reference . . . 898
8.230 mitkRecursiveSphereDirectionEncoder Class Reference . . . 909
8.231 mitkRegionGrowImageFilter Class Reference . . . 912
8.232 mitkRegistrationFilter Class Reference . . . 915
8.233 mitkRenderer Class Reference . . . 921
8.234 mitkReslicePlaneWidgetModel Class Reference . . . 924
8.235 mitkRGBATransferFunction1D Class Reference . . . 938
8.236 mitkRGBToGrayFilter Class Reference . . . 942
8.237 mitkRigidTransform Class Reference . . . 945
8.238 mitkScaleTransform Class Reference . . . 949
8.239 mitkScene Class Reference . . . 952
8.240 mitkSceneBase Class Reference . . . 968
8.241 mitkSceneContainer Class Reference . . . 972

8.242 mitkSceneParam Class Reference . . . 974
8.243 mitkSeedFillFilter Class Reference . . . 975
8.244 mitkSemaphore Class Reference . . . 977
8.245 mitkSglNode Class Reference . . . 979
8.246 mitkSheetFilter Class Reference . . . 980
8.247 mitkSimilarityTransform Class Reference . . . 982
8.248 mitkSobelEdgeDetectFilter Class Reference . . . 986
8.249 mitkSource Class Reference . . . 988
8.250 mitkSpeedImageBuilder Class Reference . . . 990
8.251 mitkSplatCamera Class Reference . . . 992
8.252 mitkSPMD Class Reference . . . 995
8.253 mitkSPMDDelegate Class Reference . . . 997
8.254 mitkSPMDMember< T > Class Template Reference . . . 998
8.255 mitkSPMDStatic Class Reference . . . 1000
8.256 mitkSTLASCIIWriter Class Reference . . . 1002
8.257 mitkSTLBinaryWriter Class Reference . . . 1004
8.258 mitkSTLReader Class Reference . . . 1006
8.259 mitkSubtractImageFilter Class Reference . . . 1008
8.260 mitkSurfaceModel Class Reference . . . 1011
8.261 mitkSurfaceProperty Class Reference . . . 1015
8.262 mitkSurfaceRenderer Class Reference . . . 1025
8.263 mitkSurfaceRendererStandard Class Reference . . . 1027
8.264 mitkSurfaceRendererUseVA Class Reference . . . 1029
8.265 mitkSurfaceRendererUseVBO Class Reference . . . 1031
8.266 mitkTarget Class Reference . . . 1033
8.267 mitkTargetMeasurementScene Class Reference . . . 1035
8.268 mitkTF2DRGBAClassifier Class Reference . . . 1040
8.269 mitkTF2DRGBAEllipsoidClassifier Class Reference . . . 1045
8.270 mitkTF2DRGBAPSPSplineClassifier Class Reference . . . 1046
8.271 mitkTF2DRGBAPyramidClassifier Class Reference . . . 1048
8.272 mitkTF2DRGBARampClassifier Class Reference . . . 1049
8.273 mitkTF2DRGBASlopeClassifier Class Reference . . . 1050
8.274 mitkTFDK Class Reference . . . 1051
8.275 mitkThinPlateSplineTransform Class Reference . . . 1053
8.276 mitkThread Class Reference . . . 1056
8.277 mitkThreadDelegate Class Reference . . . 1058

8.278 mitkThreadMaster Class Reference 1059
8.279 mitkThreadMember< T > Class Template Reference 1061
8.280 mitkThreadStatic Class Reference 1062
8.281 mitkThreadWorker Class Reference 1063
8.282 mitkThreadWorkerCreator Class Reference 1065
8.283 mitkThresholdSegmentationFilter Class Reference 1067
8.284 mitkTIFFReader Class Reference 1070
8.285 mitkTIFFWriter Class Reference 1073
8.286 mitkTrackBall Class Reference 1075
8.287 mitkTrackedBscanData Class Reference 1076
8.288 mitkTrackedBscanDataReader Class Reference 1085
8.289 mitkTrackedBscanDataWriter Class Reference 1087
8.290 mitkTrackedBscanICData Class Reference 1089
8.291 mitkTrackedBscanOoCData Class Reference 1093
8.292 mitkTransferFunction Class Reference 1098
8.293 mitkTransferFunction1D Class Reference 1100
8.294 mitkTransferFunction2DRGBA Class Reference 1104
8.295 mitkTransform Class Reference 1108
8.296 mitkTranslationTransform Class Reference 1116
8.297 mitkTriangleMesh Class Reference 1119
8.298 mitkTriangleMeshSimplification Class Reference 1123
8.299 mitkVector Class Reference 1125
8.300 mitkVersorRigid3DTransform Class Reference 1126
8.301 mitkVersorSimilarity3DTransform Class Reference 1129
8.302 mitkView Class Reference 1133
8.303 mitkVis2Scene3D Class Reference 1154
8.304 mitkVolume Class Reference 1161
8.305 mitkVolumeCropFilter Class Reference 1179
8.306 mitkVolumeDataTypeConvertor Class Reference 1182
8.307 mitkVolumeDivider Class Reference 1184
8.308 mitkVolumeIteratorBase Class Reference 1185
8.309 mitkVolumeModel Class Reference 1186
8.310 mitkVolumeProperty Class Reference 1189
8.311 mitkVolumeRayCastCompositeFunction Class Reference 1197
8.312 mitkVolumeRayCastFunction Class Reference 1199
8.313 mitkVolumeReader Class Reference 1201

8.314 mitkVolumeRenderer Class Reference 1204
8.315 mitkVolumeRendererRayCasting Class Reference 1209
8.316 mitkVolumeRendererRayCastingLoD Class Reference 1216
8.317 mitkVolumeRendererSplatting Class Reference 1223
8.318 mitkVolumeRendererTexture3D Class Reference 1234
8.319 mitkVolumeResizeFilter Class Reference 1237
8.320 mitkVolumeResliceFilter Class Reference 1240
8.321 mitkVolumeSplatFunction Class Reference 1243
8.322 mitkVolumeSplatParallel Class Reference 1245
8.323 mitkVolumeSplatPerspective Class Reference 1247
8.324 mitkVolumeToMeshFilter Class Reference 1249
8.325 mitkVolumeToVolumeFilter Class Reference 1252
8.326 mitkVolumeWriter Class Reference 1256
8.327 mitkWatershedSegFilter Class Reference 1258
8.328 mitkWidgetModel Class Reference 1260
8.329 mitkWidgetModel2D Class Reference 1267
8.330 mitkWidgetModel3D Class Reference 1271
8.331 mitkWidgetsViewManipulator Class Reference 1273
8.332 mitkWriter Class Reference 1276

Chapter 1

MITK Introduction

MITK stands for **M**edical **I**maging **T**ool**K**it. It is a C++ library for integrated medical image processing and analyzing developed by the Medical Image Processing Group, Key Laboratory of Complex Systems and Intelligence Science, Institute of Automation, the Chinese Academy of Sciences. The development of MITK is inspired by the big success of open source softwares VTK and ITK, and its main purpose is to provide medical image community a consistent framework to combine the function of medical image segmentation, registration and visualization. Much as the style of VTK, MITK uses the traditional Object-Oriented design methods, and doesn't use the Generic Programming style used by ITK. So the syntax and interface of MITK is simple and intuitive. Now MITK is a free software and can be used freely for research and education purpose. We hope that MITK will become another available choice for the medical imaging community.

Chapter 2

MITK Directory Hierarchy

2.1 MITK (Medical Imaging ToolKit) Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Additional . . . 27
mitkCUDAVolumeRendering . . . 43
Include . . . 28
modules . . . 51
StateRayCaster . . . 53
mitkReconstructionCG . . . 45
Include . . . 41
mitkReconstructionCUDA . . . 46
Include . . . 38
mitkCommon . . . 42
Include . . . 34
mitkReconstruction . . . 44
Include . . . 39
mitkRegistration . . . 47
Include . . . 37
mitkSegmentation . . . 48
Include . . . 29
mitkVisualization . . . 49
Include . . . 31
mitkVisualization2 . . . 50
Include . . . 30
modules . . . 52

Chapter 3

MITK Namespace Index

3.1 MITK (Medical Imaging ToolKit) Namespace List

Here is a list of all documented namespaces with brief descriptions:

AnalyzeImageIO (MitkAnalyzeImageReader -) 55

Chapter 4

MITK Hierarchical Index

4.1 MITK (Medical Imaging ToolKit) Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AnalyzeImageIO::data_history . . . 59
AnalyzeImageIO::dsr . . . 63
AnalyzeImageIO::header_key . . . 65
AnalyzeImageIO::image_dimension . . . 67
mitkBSPTree< BSPTreeNode > . . . 131
mitkCUDALut1D . . . 257
mitkGLSLLut1D . . . 438
mitkKdTree . . . 576
mitkList . . . 605
mitkLut1D . . . 609
mitkMatrix . . . 617
mitkMatrixD . . . 620
mitkNeighborhood . . . 676
mitkNodeHeap . . . 687
mitkObject . . . 692
- mitkBackProjection . . . 113
  - mitkBackProjectionUsingGPU . . . 115
    - mitk2DBackProjectionUsingGPU . . . 71
    - mitk3DBackProjectionUsingGPU . . . 82
- mitkBarrier . . . 117
- mitkCamera . . . 136
  - mitkSplatCamera . . . 992
- mitkColorTable . . . 167
- mitkColorTransferFunction . . . 175
- mitkConditionVariable . . . 180
- mitkDataObject . . . 302
  - mitkCoherentVolume . . . 161
  - mitkDICOMDIRTree . . . 307
  - mitkMesh . . . 626
    - mitkHEMesh . . . 452
      - mitkHETriangleMesh . . . 481
        - mitkHEICTriangleMesh . . . 445
        - mitkHEOoCTriangleMesh . . . 470

mitkTriangleMesh . . . 1119
mitkICTriangleMesh . . . 504
mitkOoCTriangleMesh . . . 714
mitkProjectData . . . 830
mitkConeBeamProjectData . . . 190
mitkICConeBeamProjectData . . . 495
mitkOoCConeBeamProjectData . . . 701
mitkRawConeBeamProjectData . . . 859
mitkFanBeamProjectData . . . 361
mitkICFanBeamProjectData . . . 498
mitkOoCFanBeamProjectData . . . 704
mitkParallelProjectData . . . 771
mitkICParallelProjectData . . . 501
mitkOoCParallelProjectData . . . 707
mitkRawData . . . 863
mitkCTProjectionData . . . 206
mitk2DBeamPD . . . 79
mitk2DBeamICPD . . . 73
mitk2DBeamOoCPD . . . 76
mitkConeBeamPD . . . 187
mitkConeBeamICPD . . . 181
mitkConeBeamOoCPD . . . 184
mitkConeBeamRealTimePD . . . 196
mitkTrackedBscanData . . . 1076
mitkTrackedBscanICData . . . 1089
mitkTrackedBscanOoCData . . . 1093
mitkVolume . . . 1161
mitkICVolume . . . 508
mitkEuclideanSkeletonFeature . . . 353
mitkOoCVolume . . . 720
mitkDirectionEncoder . . . 320
mitkRecursiveSphereDirectionEncoder . . . 909
mitkEncodedGradientEstimator . . . 333
mitkFiniteDifferenceGradientEstimator . . . 396
mitkEncodedGradientShader . . . 342
mitkFootprint . . . 398
mitkFootprint1D . . . 401
mitkFootprint1DGaussian . . . 403
mitkFootprint2D . . . 407
mitkFootprint2DGaussian . . . 409
mitkHistogram1D . . . 486
mitkHistogram2D . . . 490
mitkImplementor . . . 553
mitkLight . . . 588
mitkManipulator . . . 610
mitkImageViewManipulatorStandard . . . 547
mitkMeshViewManipulatorStandard . . . 642
mitkPickManipulator . . . 780
mitkImageViewManipulatorWithWidgets . . . 550
mitkWidgetsViewManipulator . . . 1273
mitkModel . . . 660
mitkDataModel . . . 299

mitkImageModel . . . 522
mitkOrthogonalImageModel . . . 763
mitkSurfaceModel . . . 1011
mitkVolumeModel . . . 1186
mitkWidgetModel . . . 1260
mitkWidgetModel2D . . . 1267
mitkAngleWidgetModel2D . . . 101
mitkEllipseWidgetModel2D . . . 324
mitkLineWidgetModel2D . . . 592
mitkPolygonWidgetModel2D . . . 818
mitkPseudocolorWidgetModel . . . 838
mitkPseudocolorWidgetModelEx . . . 845
mitkRectWidgetModel2D . . . 898
mitkWidgetModel3D . . . 1271
mitkAngleWidgetModel3D . . . 108
mitkClippingPlaneWidgetModel . . . 154
mitkLineWidgetModel3D . . . 600
mitkReslicePlaneWidgetModel . . . 924
mitkMutex . . . 672
mitkNode . . . 684
mitkObserver . . . 699
mitkOoCVolumeShearFunction . . . 751
mitkOoCVolumeShearParallel . . . 753
mitkOoCVolumeShearPerspective . . . 755
mitkPlane . . . 782
mitkProcessObject . . . 828
mitkFilter . . . 394
mitkEuclideanSkeletonFeatureExtraction . . . 355
mitkMeshToMeshFilter . . . 639
mitkTriangleMeshSimplification . . . 1123
mitkQEMSimplification . . . 852
mitkRawDataToVolumeFilter . . . 868
mitk3DUSReconstructionFilter . . . 89
mitk3DUSPixelBasedGPU . . . 84
mitk3DUSPixelBasedMT . . . 86
mitkCTReconstructionFilter . . . 209
mitkConeBeamReconFromHelix . . . 200
mitkCBFDKReconCF . . . 142
mitkCBFDKReconCFCPU . . . 145
mitkCBFDKReconCFGPU . . . 148
mitkCBTFDKReconCF . . . 151
mitkFanBeamReconFromCirle . . . 366
mitkFBFBPReconCF . . . 372
mitkFBFBPReconCFCPU . . . 375
mitkFBFBPReconCFGPU . . . 378
mitkParallelBeamReconFromCirle . . . 768
mitkPBFBPReconCF . . . 777
mitkProjectDataToVolumeFilter . . . 834
mitkFbFbp . . . 370
mitkFbFbpSpeedUsingCpu . . . 381
mitkFbFbpSpeedUsingGpu . . . 383
mitkFDK . . . 386
mitkFDKSpeedUsingCpu . . . 388

mitkFDKSpeedUsingGpu . . . 390
mitkPbFbp . . . 775
mitkTFDK . . . 1051
mitkVolumeToMeshFilter . . . 1249
mitkBinMarchingCubes . . . 121
mitkMarchingCubes . . . 614
mitkVolumeToVolumeFilter . . . 1252
mitkBinaryFilter . . . 119
mitkConnectivityLabelFilter . . . 204
mitkDiffusionFilter . . . 317
mitkDistanceTransformSaito . . . 322
mitkEuclideanDistanceTransformFilter . . . 350
mitkEuclideanSkeletonGraphCut . . . 358
mitkFFTFilter . . . 392
mitkGaussianDerivativeImageFilter . . . 421
mitkGaussianFilter . . . 423
mitkInterpolateFilter . . . 556
mitkBSplineInterpolateFilter . . . 129
mitkCubicInterpolateFilter . . . 213
mitkLinearInterpolateFilter . . . 590
mitkNearestNeighborInterpolateFilter . . . 674
mitkLiveWireImageFilter . . . 607
mitkNeumannBoundary . . . 682
mitkOtsuSegmentationFilter . . . 766
mitkRegionGrowImageFilter . . . 912
mitkRegistrationFilter . . . 915
mitkPointSetRegistrationFilter . . . 813
mitkLandmarkRegistrationFilter . . . 581
mitkRGBToGrayFilter . . . 942
mitkSeedFillFilter . . . 975
mitkSheetFilter . . . 980
mitkSobelEdgeDetectFilter . . . 986
mitkSpeedImageBuilder . . . 990
mitkSubtractImageFilter . . . 1008
mitkThresholdSegmentationFilter . . . 1067
mitkVolumeCropFilter . . . 1179
mitkVolumeDataTypeConvertor . . . 1182
mitkVolumeResizeFilter . . . 1237
mitkVolumeResliceFilter . . . 1240
mitkWatershedSegFilter . . . 1258
mitkImageIOFactory . . . 520
mitkMetric . . . 651
mitkMeanSquaresMetric . . . 623
mitkNormalizedCorrelationMetric . . . 689
mitkPointSetMetric . . . 807
mitkEuclideanDistanceMetric . . . 348
mitkOptimizer . . . 757
mitkAmoebaOptimizer . . . 94
mitkGradientDescentOptimizer . . . 442
mitkIterativeClosestPointOptimizer . . . 568
mitkLBFGSBOptimizer . . . 583
mitkLBFGSOptimizer . . . 586
mitkPowellOptimizer . . . 825

mitkPointSetEstimator . . . 801
mitkPointSetKernelTransformEstimator . . . 805
mitkPointSetQuaternionEstimator . . . 811
mitkSource . . . 988
mitkReader . . . 895
mitkDICOMDIRReader . . . 305
mitkInfoReader . . . 554
mitkDICOMInfoReader . . . 309
mitkMeshReader . . . 636
mitkPLYReader . . . 787
mitkSTLReader . . . 1006
mitkRawDataReader . . . 866
mitkTrackedBscanDataReader . . . 1085
mitkVolumeReader . . . 1201
mitkAnalyzeImageReader . . . 97
mitkBMPReader . . . 124
mitkCacheVolumeReader . . . 132
mitkDICOMReader . . . 312
mitkIM0Reader . . . 516
mitkJPEGReader . . . 571
mitkMetaImageReader . . . 647
mitkNetCDFImageReader . . . 678
mitkRawFilesReader . . . 872
mitkRawFilesReader . . . 872
mitkRawReader . . . 885
mitkTIFFReader . . . 1070
mitkTarget . . . 1033
mitkWriter . . . 1276
mitkMeshWriter . . . 645
mitkPLYASCIIWriter . . . 783
mitkPLYBinaryWriter . . . 785
mitkSTLASCIIWriter . . . 1002
mitkSTLBinaryWriter . . . 1004
mitkRawDataWriter . . . 870
mitkTrackedBscanDataWriter . . . 1087
mitkVolumeWriter . . . 1256
mitkAnalyzeImageWriter . . . 99
mitkBMPWriter . . . 127
mitkCacheVolumeWriter . . . 134
mitkDICOMWriter . . . 314
mitkIM0Writer . . . 518
mitkJPEGWriter . . . 574
mitkMetaImageWriter . . . 649
mitkNetCDFImageWriter . . . 680
mitkRawFilesWriter . . . 882
mitkRawWriter . . . 891
mitkTIFFWriter . . . 1073
mitkTransform . . . 1108
mitkAffineTransform . . . 91
mitkRigidTransform . . . 945
mitkScaleTransform . . . 949
mitkSimilarityTransform . . . 982

mitkThinPlateSplineTransform . . . 1053
mitkTranslationTransform . . . 1116
mitkVersorRigid3DTransform . . . 1126
mitkVersorSimilarity3DTransform . . . 1129
mitkRenderer . . . 921
mitkSurfaceRenderer . . . 1025
mitkOoCSurfaceRendererStandard . . . 710
mitkOoCSurfaceRendererUseVA . . . 712
mitkSurfaceRendererStandard . . . 1027
mitkSurfaceRendererUseVA . . . 1029
mitkSurfaceRendererUseVBO . . . 1031
mitkVolumeRenderer . . . 1204
mitkOoCVolumeRendererRayCasting . . . 734
mitkOoCVolumeRendererRayCastingEx . . . 736
mitkOoCVolumeRendererShearWarp . . . 739
mitkOoCVolumeRendererTexture3D . . . 745
mitkOoCVolumeRendererTexture3DEx . . . 748
mitkVolumeRendererRayCasting . . . 1209
mitkVolumeRendererRayCastingLoD . . . 1216
mitkVolumeRendererSplatting . . . 1223
mitkVolumeRendererTexture3D . . . 1234
mitkRGBATransferFunction1D . . . 938
mitkSceneBase . . . 968
mitkCUDAEndoscopySliceScene . . . 237
mitkEndoscopySliceScene . . . 343
mitkScene . . . 952
mitkImageScene . . . 532
mitkVis2Scene3D . . . 1154
mitkCUDAAugmentedSurfaceVRScene . . . 215
mitkCUDADecomposedIsosurfaceVRScene . . . 222
mitkCUDADeltaAccStateVRScene . . . 229
mitkCUDAFullVRScene . . . 242
mitkCUDAIsosurfaceVRScene . . . 250
mitkCUDAMIPVRScene . . . 258
mitkCUDAROIDistanceVRScene . . . 261
mitkCUDASegmentedVRScene . . . 270
mitkCUDAStructureFunctionVRScene . . . 278
mitkCUDAValueBasedStateVRScene . . . 283
mitkCUDAVersatileVRScene . . . 292
mitkFullVRScene . . . 413
mitkGLSLFullVRScene . . . 425
mitkGLSLIsosurfaceVRScene . . . 432
mitkGLSLMIPVRScene . . . 439
mitkIsosurfaceVRScene . . . 561
mitkMIPVRScene . . . 656
mitkTargetMeasurementScene . . . 1035
mitkSceneContainer . . . 972
mitkSemaphore . . . 977
mitkSPMD . . . 995
mitkSPMDDelegate . . . 997
mitkSPMDMember< T > . . . 998
mitkSPMDStatic . . . 1000
mitkSurfaceProperty . . . 1015

mitkTF2DRGBAClassifier . . . 1040
mitkTF2DRGBAEllipsoidClassifier . . . 1045
mitkTF2DRGBAPSPSplineClassifier . . . 1046
mitkTF2DRGBAPyramidClassifier . . . 1048
mitkTF2DRGBARampClassifier . . . 1049
mitkTF2DRGBASlopeClassifier . . . 1050
mitkThread . . . 1056
mitkThreadDelegate . . . 1058
mitkThreadMember< T > . . . 1061
mitkThreadStatic . . . 1062
mitkThreadMaster . . . 1059
mitkThreadWorker . . . 1063
mitkThreadWorkerCreator . . . 1065
mitkTransferFunction . . . 1098
mitkTransferFunction1D . . . 1100
mitkTransferFunction2DRGBA . . . 1104
mitkView . . . 1133
mitkImageView . . . 541
mitkVolumeDivider . . . 1184
mitkVolumeProperty . . . 1189
mitkVolumeRayCastFunction . . . 1199
mitkOoCVolumeRayCastCompositeFunction . . . 730
mitkOoCVolumeRayCastCompositeFunctionEx . . . 732
mitkVolumeRayCastCompositeFunction . . . 1197
mitkVolumeSplatFunction . . . 1243
mitkVolumeSplatParallel . . . 1245
mitkVolumeSplatPerspective . . . 1247
mitkPointSet . . . 789
mitkKdTreePointSet . . . 578
mitkQuaternion . . . 855
mitkRCPtr< T > . . . 894
mitkRectPlane . . . 897
mitkSceneParam . . . 974
mitkSglNode . . . 979
mitkTrackBall . . . 1075
mitkVector . . . 1125
mitkVolumeIteratorBase . . . 1185

Chapter 5

MITK Class Index

5.1 MITK (Medical Imaging ToolKit) Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AnalyzeImageIO::data_history . . . 59
AnalyzeImageIO::dsr . . . 63
AnalyzeImageIO::header_key . . . 65
AnalyzeImageIO::image_dimension . . . 67
mitk2DBackProjectionUsingGPU (Mitk2DBackProjectionUsingGPU - abstract base class for C-T back projection) . . . 71
mitk2DBeamICPD (Mitk2DBeamICPD - concret class to represent fan-beam and parallel projection data) . . . 73
mitk2DBeamOoCPD (Mitk2DBeamOoCPD - concret class to represent fan-beam project data) 76
mitk2DBeamPD (Mitk2DBeamPD - abstract class to represent fan-beam project data) . . . 79
mitk3DBackProjectionUsingGPU (Mitk3DBackProjectionUsingGPU - abstract base class for C-T back projection) . . . 82
mitk3DUSPixelBasedGPU (Mitk3DUSPixelBasedGPU -) . . . 84
mitk3DUSPixelBasedMT (Mitk3DUSPixelBasedMT - a concrete class for 3D US Reconstruction) . . . 86
mitk3DUSReconstructionFilter (Mitk3DUSReconstructionFilter - an abstract class for 3D US Reconstruction) . . . 89
mitkAffineTransform (MitkAffineTransform - a concrete transform to perform affine transformation) . . . 91
mitkAmoebaOptimizer (MitkAmoebaOptimizer - a concrete class for implementation of the Nelder-Meade downhill simplex algorithm) . . . 94
mitkAnalyzeImageReader (MitkAnalyzeImageReader -) . . . 97
mitkAnalyzeImageWriter (MitkAnalyzeImageWriter -) . . . 99
mitkAngleWidgetModel2D (MitkAngleWidgetModel2D - a 2D widget for measuring angles) . 101
mitkAngleWidgetModel3D (MitkAngleWidgetModel3D - a 3D widget for measuring angles) . 108
mitkBackProjection (MitkBackProjection - abstract base class for CT back projection) . . . 113
mitkBackProjectionUsingGPU (MitkBackProjectionUsingGPU - abstract base class for CT back projection) . . . 115
mitkBarrier (MitkBarrier - A synchronizer for mitkSPMD) . . . 117
mitkBinaryFilter (MitkBinaryFilter - a filter to get binary data from a source volume) . . . 119
mitkBinMarchingCubes (MitkBinMarchingCubes - a marching cubes algorithm using binary data) . . . 121
mitkBMPReader (MitkBMPReader - a concrete reader for reading BMP image files) . . . 124

mitkBMPWriter (MitkBMPWriter - a concrete writer for writing a volume to BMP image files) 127
mitkBSplineInterpolateFilter (MitkBSplineInterpolateFilter - a concrete interpolator) 129
mitkBSPTree< BSPTreeNode > (MitkBSPTree -) 131
mitkCacheVolumeReader (MitkCacheVolumeReader - a volume reader for reading out-of-core volume cache files) 132
mitkCacheVolumeWriter (MitkCacheVolumeWriter - a volume writer for reusing out-of-core volume cache files) 134
mitkCamera (MitkCamera - a camera in 3D scene) 136
mitkCBFDKReconCF (MitkCBFDKReconCF - concret class implement cone-beam reconstruction using FDK Algorithm) 142
mitkCBFDKReconCFCPU (MitkCBFDKReconCFCPU - concret class implement cone-beam reconstruction using FDK Algorithm) 145
mitkCBFDKReconCFGPU (MitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction using FDK Algorithm) 148
mitkCBTFDKReconCF (MitkCBTFDKReconCF - concret class implement cone-beam reconstruction using TFDK Algorithm) 151
mitkClippingPlaneWidgetModel (MitkClippingPlaneWidgetModel - a 3D widget for clipping plane) 154
mitkCoherentVolume (MitkCoherentVolume - a volume data class optimized for random access using blocks) 161
mitkColorTable (MitkColorTable - a table mapping pixel value to color) 167
mitkColorTransferFunction (MitkColorTransferFunction - a transfer function to map the scalar value to color) 175
mitkConditionVariable (MitkConditionVariable - a class that encapsulates a condition variable) 180
mitkConeBeamICPD (MitkConeBeamICPD - concret class to represent cone-beam project data) 181
mitkConeBeamOoCPD (MitkConeBeamOoCPD - concret class to represent cone-beam OoC project data) 184
mitkConeBeamPD (MitkConeBeamPD - abstract class to represent Cone-beam project data) . . 187
mitkConeBeamProjectData (MitkConeBeamProjectData - abstract class to represent Cone-beam project data) 190
mitkConeBeamRealTimePD (Example) 196
mitkConeBeamReconFromHelix (MitkConeBeamReconFromHelix - concret class implement Cone-beam reconstruction) 200
mitkConnectivityLabelFilter (MitkConnectivityLabelFilter - A class that label each connected region with 1, 2, 3....) 204
mitkCTProjectionData (MitkCTProjectionData - abstract class to represent project data) 206
mitkCTReconstructionFilter (MitkCTReconstructionFilter - abstract class for forward construction) 209
mitkCubicInterpolateFilter (MitkCubicInterpolateFilter - a concrete class for cubic interpolation) 213
mitkCUDAAugmentedSurfaceVRScene (MitkCUDAAugmentedSurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface) . . 215
mitkCUDADecomposedIsosurfaceVRScene (MitkCUDADecomposedIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets, for large data. Accelerated with the CUDA interface) 222
mitkCUDADeltaAccStateVRScene (MitkCUDADeltaAccStateVRScene - a 3D scene for volume rendering with delta accumulation based ray feature analysis. Accelerated with the CUDA interface) 229
mitkCUDAEndoscopySliceScene (MitkCUDAEndoscopySliceScene - A CUDA based slice preview scene for virtual endoscopy navigation) 237
mitkCUDAFullVRScene (MitkCUDAFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface) 242
mitkCUDAIsosurfaceVRScene (MitkCUDAIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface) 250
mitkCUDALut1D (MitkGLSLLut1D - A data class for 1D-LookUPTable,) 257

mitkCUDAMIPVRScene (MitkCUDAMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Accelerated with the CUDA interface) . . . 258
mitkCUDAROIDistanceVRScene (MitkCUDAROIDistanceVRScene - a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface) . . . 261
mitkCUDASegmentedVRScene (MitkCUDASegmentedVRScene - a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface) . . . 270
mitkCUDAStructureFunctionVRScene (MitkCUDAStructureFunctionVRScene - a 3D scene for rendering a structure volume together with a function volume. Accelerated with the CUDA interface) . . . 278
mitkCUDAValueBasedStateVRScene (MitkCUDADeltaAccStateVRScene - a 3D scene for volume rendering with value based ray feature analysis. Accelerated with the CUDA interface) . . . 283
mitkCUDAVersatileVRScene (MitkCUDAVersatileVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface) . . . 292
mitkDataModel (MitkDataModel - abstract class used to represent an data entity in a rendering scene) . . . 299
mitkDataObject (MitkDataObject - an abstract class to represents a data object in MITK) . . . 302
mitkDICOMDIRReader (MitkDICOMDIRReader - A reader class that reads a DICOMDIR file from disk) . . . 305
mitkDICOMDIRTree (MitkDICOMDIRTree - Data Object that stores the DICOMDIR structure of a data base) . . . 307
mitkDICOMInfoReader (MitkDICOMInfoReader - a concrete reader for reading element information in DICOM files) . . . 309
mitkDICOMReader (MitkDICOMReader - a concrete reader for reading DICOM image files) . . . 312
mitkDICOMWriter (MitkDICOMWriter - a concrete writer for writing a volume to DICOM image files) . . . 314
mitkDiffusionFilter (MitkDiffusionFilter - a class used to diffuse the mitkVolume data (2D or 3D)) 317
mitkDirectionEncoder (MitkDirectionEncoder - an abstract class to encode a direction into a one or two byte value) . . . 320
mitkDistanceTransformSaito (MitkDistanceTransformSaito - computes 3D Euclidean DT) . . . 322
mitkEllipseWidgetModel2D (MitkEllipseWidgetModel2D - an ellipse widget used in 2D scenes) 324
mitkEncodedGradientEstimator (MitkEncodedGradientEstimator - an abstract class for gradient estimation) . . . 333
mitkEncodedGradientShader (MitkEncodedGradientShader - Computes shading tables for encoded normals) . . . 342
mitkEndoscopySliceScene (MitkEndoscopySliceScene - A slice preview scene for virtual endoscopy navigation) . . . 343
mitkEuclideanDistanceMetric (MitkEuclideanDistanceMetric - a concrete class computing similarity between two point sets) . . . 348
mitkEuclideanDistanceTransformFilter (MitkEuclideanDistanceTransformFilter - computes 3D Euclidean DT using Meijster's Linear Time Algorithm) . . . 350
mitkEuclideanSkeletonFeature (MitkEuclideanSkeletonFeature - Data structure representing a decomposed volume) . . . 353
mitkEuclideanSkeletonFeatureExtraction (MitkEuclideanSkeletonFeatureExtraction - Decompose a volume based on Euclidean DT and Reverse DT) . . . 355
mitkEuclideanSkeletonGraphCut (MitkEuclideanSkeletonGraphCut - A segmentation class implementing the Skeleton-Cut algorithm) . . . 358
mitkFanBeamProjectData (MitkFanBeamProjectData - abstract class to represent fan-beam project data) . . . 361
mitkFanBeamReconFromCirle (MitkFanBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm) . . . 366

mitkFbFbp (MitkFbFbp - concret class implement fan-beam reconstruction using filtered back-projection Algorithm) . . . 370
mitkFBFBPReconCF (MitkFBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm) . . . 372
mitkFBFBPReconCFCPU (MitkFBFBPReconCFCPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm) . . . 375
mitkFBFBPReconCFGPU (MitkFBFBPReconCFGPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm) . . . 378
mitkFbFbpSpeedUsingCpu (MitkFbFbpSpeedUsingCpu - concret class implement fan-beam reconstruction using filtered backprojection Algorithm) . . . 381
mitkFbFbpSpeedUsingGpu (MitkFbFbpSpeedUsingGpu- concret class implement fan-beam reconstruction using filtered backprojection Algorithm) . . . 383
mitkFDK (MitkFDK - concret class implement cone-beam reconstruction using FDK Algorithm) 386
mitkFDKSpeedUsingCpu (MitkFDKSpeedUsingCpu - concret class implement cone-beam reconstruction using FDK Algorithm) . . . 388
mitkFDKSpeedUsingGpu (MitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction using FDK Algorithm) . . . 390
mitkFFTFilter (MitkFFTFilter - a concrete class for implementation of 3D-FFT Transform) . . 392
mitkFilter (MitkFilter - abstract class specifies interface for filter object) . . . 394
mitkFiniteDifferenceGradientEstimator (MitkFiniteDifferenceGradientEstimator - a concrete class to use finite differences to estimate gradient) . . . 396
mitkFootprint (MitkFootprint - abstract class defines common interface for footprint) . . . 398
mitkFootprint1D (MitkFootprint1D - abstract class specifies interface for one dimensional footprint) . . . 401
mitkFootprint1DGaussian (MitkFootprint1DGaussian - concrete class for one dimensional footprint with Gaussian kernel) . . . 403
mitkFootprint2D (MitkFootprint2D - abstract class specifies interface for two dimensional footprint) . . . 407
mitkFootprint2DGaussian (MitkFootprint2DGaussian - concrete class for two dimensional footprint with Gaussian kernel) . . . 409
mitkFullVRScene (MitkFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Optimized for multicore CPU) . . . 413
mitkGaussianDerivativeImageFilter (MitkGaussianDerivativeImageFilter - a concrete class for implementation of Gaussian Derivative Filter) . . . 421
mitkGaussianFilter (MitkGaussianFilter - a concrete class for implementation of Gaussian Smooth Filter) . . . 423
mitkGLSLFullVRScene (MitkGLSLFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the GLSL interface) . . . 425
mitkGLSLIsosurfaceVRScene (MitkGLSLIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the GLSL interface) . . . 432
mitkGLSLLut1D (MitkGLSLLut1D - A data class for 1D-LookUPTable,) . . . 438
mitkGLSLMIPVRScene (MitkGLSLMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Accelerated with the GLSL interface) . . . 439
mitkGradientDescentOptimizer (MitkGradientDescentOptimizer - a concrete class for implementation of the Gradient Descent algorithm) . . . 442
mitkHEICTriangleMesh (MitkHEICTriangleMesh - a concrete class for in-core triangle meshes represented by Half Edges) . . . 445
mitkHEMesh (MitkHEMesh - a concrete class for polygon meshes represented by Half Edges) . 452
mitkHEOoCTriangleMesh (MitkHEOoCTriangleMesh - a concrete class for out-of-core triangle meshes represented by Half Edges) . . . 470
mitkHETriangleMesh (MitkHETriangleMesh - an abstract class for triangle meshes represented by Half Edges) . . . 481
mitkHistogram1D (MitkHistogram1D - a tool to generate a 1D histogram from a volume) . . . 486
mitkHistogram2D (MitkHistogram2D - a tool to generate a 2D histogram from two volumes) . 490

mitkICConeBeamProjectData (MitkICConeBeamProjectData - concret class to represent cone-beam project data) . . . 495
mitkICFanBeamProjectData (MitkICFanBeamProjectData - concret class to represent fan-beam project data) . . . 498
mitkICParallelProjectData (MitkICParallelProjectData - concret class to represent Parallel project data) . . . 501
mitkICTriangleMesh (MitkICTriangleMesh - a concrete class for in-core triangle meshes) . . . 504
mitkICVolume (MitkICVolume - a concrete data object to represent an in-core multi-dimensional medical image dataset) . . . 508
mitkIM0Reader (MitkIM0Reader - a concrete reader for reading IM0 volume file) . . . 516
mitkIM0Writer (MitkIM0Writer - a concrete writer for writing a volume to IM0 volume file) . . . 518
mitkImageIOFactory (MitkImageIOFactory -) . . . 520
mitkImageModel (MitkImageModel - a concrete model class used to display a 2D image in mitkImageScene) . . . 522
mitkImageScene (MitkImageScene - a Scene to display 2D images) . . . 532
mitkImageView (MitkImageView - a view to display 2D images) . . . 541
mitkImageViewManipulatorStandard (MitkImageViewManipulatorStandard - a concrete manipulator to process mouse events in an image scene) . . . 547
mitkImageViewManipulatorWithWidgets (MitkImageViewManipulatorWithWidgets - manipulator of an image view contains widgets) . . . 550
mitkImplementor (MitkImplementor - an abstract class to define an interface to implement some OS dependent operations) . . . 553
mitkInfoReader (MitkInfoReader - an abstract class represents an information reader) . . . 554
mitkInterpolateFilter (MitkInterpolateFilter - an abstract class specifies interface for interpolating values when objects are resampled through the Transform) . . . 556
mitkIsosurfaceVRScene (MitkIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Optimized for multicore CPU) . . . 561
mitkIterativeClosestPointOptimizer (MitkIterativeClosestPointOptimizer - a concrete class to iteratively align two point sets) . . . 568
mitkJPEGReader (MitkJPEGReader - a concrete reader for reading JPEG image files) . . . 571
mitkJPEGWriter (MitkJPEGWriter - a concrete writer for writing a volume to JPEG image files) 574
mitkKdTree (MitkKdTree - a class for implementation of k-d tree data structure) . . . 576
mitkKdTreePointSet . . . 578
mitkLandmarkRegistrationFilter (MitkLandmarkRegistrationFilter - implementation of landmark-based registration framework) . . . 581
mitkLBFGSBOptimizer (MitkLBFGSBOptimizer -) . . . 583
mitkLBFGSOptimizer (MitkLBFGSOptimizer -) . . . 586
mitkLight (MitkLight - a light object in 3D scene) . . . 588
mitkLinearInterpolateFilter (MitkLinearInterpolateFilter - a concrete class for bilinear and trilinear interpolation) . . . 590
mitkLineWidgetModel2D (MitkLineWidgetModel2D - a line widget used in 2D scene) . . . 592
mitkLineWidgetModel3D (MitkLineWidgetModel3D - a line widget used in a 3D scene) . . . 600
mitkList (MitkList - a utility class for a list of mitkObject) . . . 605
mitkLiveWireImageFilter (MitkLiveWireImageFilter - A class that implement livewire segmentation arithmetic) . . . 607
mitkLut1D (MitkLut1D - A data class for 1D-LookUPTable,) . . . 609
mitkManipulator (MitkManipulator - an abstract class to define an interface to implement mouse events processing in a scene) . . . 610
mitkMarchingCubes (MitkMarchingCubes - implementing 3d reconstruction algorithm) . . . 614
mitkMatrix (MitkMatrix - an encapsulation of a matrix) . . . 617
mitkMatrixD (MitkMatrixD - an encapsulation of a matrix) . . . 620
mitkMeanSquaresMetric (MitkMeanSquaresMetric - a concrete class computing similarity between two objects to be registered) . . . 623
mitkMesh (MitkMesh - an abstract class for mesh types) . . . 626

mitkMeshReader (MitkMeshReader - an abstract class represents a mesh reader to read mesh files)636
mitkMeshToMeshFilter (MitkMeshToMeshFilter - abstract class specifies interface for mesh to mesh filter) . . . 639
mitkMeshViewManipulatorStandard (MitkMeshViewManipulatorStandard - a concrete manipulator to process mouse events in an 3D scene) . . . 642
mitkMeshWriter (MitkMeshWriter - an abstract class represents a mesh writer for writing mesh data to disk) . . . 645
mitkMetaImageReader (MitkMetaImageReader -) . . . 647
mitkMetaImageWriter (MitkMetaImageWriter -) . . . 649
mitkMetric (MitkMetric - an abstract class specifies interface for computing similarity between regions of two volumes) . . . 651
mitkMIPVRScene (MitkMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Optimized for multicore CPU) . . . 656
mitkModel (MitkModel - abstract class used to represent an entity in a rendering scene) . . . 660
mitkMutex (MitkMutex - a class that encapsulates a mutex) . . . 672
mitkNearestNeighborInterpolateFilter (MitkNearestNeighborInterpolateFilter - a concrete interpolator) . . . 674
mitkNeighborhood (A light-weight container object for storing an N-dimensional neighborhood of values) . . . 676
mitkNetCDFImageReader (MitkNetCDFImageReader -) . . . 678
mitkNetCDFImageWriter (MitkNetCDFImageWriter -) . . . 680
mitkNeumannBoundary (A function object that determines a neighborhood of values at an image boundary according to a Neumann boundary condition where first, upwind derivatives on the boundary are zero. This is a useful condition in solving some classes of differential equations) . . . 682
mitkNode (MitkNode - a class that define NodeType used by mitkFastMarchingImageFilter) . . 684
mitkNodeHeap (MitkNodeHeap - a class that define mitkNodeHeap used by mitkFastMarchingImageFilter) . . . 687
mitkNormalizedCorrelationMetric (MitkNormalizedCorrelationMetric - a concrete class computing similarity between two objects to be registered) . . . 689
mitkObject (MitkObject - abstract base class for most objects in MITK) . . . 692
mitkObserver (MitkObserver - abstract base class for observers) . . . 699
mitkOoCConeBeamProjectData (MitkOoCConeBeamProjectData - concret class to represent cone-beam project data) . . . 701
mitkOoCFanBeamProjectData (MitkOoCFanBeamProjectData - concret class to represent fan-beam project data) . . . 704
mitkOoCParallelProjectData (MitkOoCParallelProjectData - concret class to represent Parallel project data) . . . 707
mitkOoCSurfaceRendererStandard (MitkOoCSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel) . . . 710
mitkOoCSurfaceRendererUseVA (MitkOoCSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array) . . . 712
mitkOoCTriangleMesh (MitkOoCTriangleMesh - a concrete class for out-of-core triangle meshes) . . . 714
mitkOoCVolume (MitkOoCVolume - a concrete data object to represent a out-of-core multi-dimensional medical image dataset) . . . 720
mitkOoCVolumeRayCastCompositeFunction (MitkOoCVolumeRayCastCompositeFunction - a concrete ray cast function for compositing) . . . 730
mitkOoCVolumeRayCastCompositeFunctionEx (MitkOoCVolumeRayCastCompositeFunctionEx - a concrete ray cast function for compositing) . . . 732
mitkOoCVolumeRendererRayCasting (MitkOoCVolumeRendererRayCasting - a concrete volume renderer for rendering an out-of-core volume) . . . 734
mitkOoCVolumeRendererRayCastingEx (MitkOoCVolumeRendererRayCastingEx - a concrete volume renderer for rendering an out-of-core volume) . . . 736

mitkOoCVolumeRendererShearWarp (MitkVolumeRendererShearWarp - a concrete volume renderer for rendering a volume) . . . 739
mitkOoCVolumeRendererTexture3D (MitkOoCVolumeRendererTexture3D - a concrete volume renderer for rendering an out-of-core volume) . . . 745
mitkOoCVolumeRendererTexture3DEx (MitkOoCVolumeRendererTexture3DEx - a concrete volume renderer for rendering an out-of-core volume) . . . 748
mitkOoCVolumeShearFunction (MitkOoCVolumeShearFunction - abstract class defines interface for volume shear) . . . 751
mitkOoCVolumeShearParallel (MitkOoCVolumeShearParallel -) . . . 753
mitkOoCVolumeShearPerspective (MitkOoCVolumeShearPerspective -) . . . 755
mitkOptimizer (MitkOptimizer - an abstract class specifies interface for an optimization method) 757
mitkOrthogonalImageModel (MitkOrthogonalImageModel) . . . 763
mitkOtsuSegmentationFilter (MitkOtsuSegmentationFilter - a class implement Ostu segmentation arithmetic) . . . 766
mitkParallelBeamReconFromCirle (MitkParallelBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm) . . . 768
mitkParallelProjectData (MitkParallelProjectData - abstract class to represent Parallel project data) . . . 771
mitkPbFbp (MitkPbFbp - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm) . . . 775
mitkPBFBPReconCF (MitkPBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm) . . . 777
mitkPickManipulator (MitkPickManipulator - a manipulator with picking function enabled) . . 780
mitkPlane (MitkPlane - a class to represent a plane) . . . 782
mitkPLYASCIIWriter (MitkPLYASCIIWriter - a concrete writer for writing a mesh to PLY files with ASCII format) . . . 783
mitkPLYBinaryWriter (MitkPLYBinaryWriter - a concrete writer for writing a mesh to PLY files with binary format) . . . 785
mitkPLYReader (MitkPLYReader - a concrete reader for reading a PLY file) . . . 787
mitkPointSet (MitkPointSet - a class to represent geometry in the form of a set of points in two or three dimensional space) . . . 789
mitkPointSetEstimator (MitkPointSetEstimator - an abstract class specifies interface for an estimation method) . . . 801
mitkPointSetKernelTransformEstimator (MitkPointSetKernelTransformEstimator -) . . . 805
mitkPointSetMetric (MitkPointSetMetric - an abstract class specifies interface for computing similarity between regions of two point sets) . . . 807
mitkPointSetQuaternionEstimator (MitkPointSetQuaternionEstimator - a concrete class for transform estimation based on Unit Quaternion method) . . . 811
mitkPointSetRegistrationFilter (MitkPointSetRegistrationFilter - a class for the implementation of point sets based registration framework) . . . 813
mitkPolygonWidgetModel2D (MitkPolygonWidgetModel2D - a 2D widget for displaying an arbitrary polygon in an image scene) . . . 818
mitkPowellOptimizer . . . 825
mitkProcessObject (MitkProcessObject - abstract base class for source, filter(algorithm) and mapper) . . . 828
mitkProjectData (MitkProjectData - abstract class to represent project data) . . . 830
mitkProjectDataToVolumeFilter (MitkProjectDataToVolumeFilter - abstract class for forward construction) . . . 834
mitkPseudocolorWidgetModel (MitkPseudocolorWidgetModel - a 2D widget for displaying pseudocolor image) . . . 838
mitkPseudocolorWidgetModelEx (MitkPseudocolorWidgetModelEx - a 2D widget for displaying pseudocolor image) . . . 845
mitkQEMSimplification (MitkQEMSimplification - an implementation for the simplification algorithm using Quadric Error Metrics (QEM)) . . . 852

mitkQuaternion (MitkQuaternion - it's used to implement 3d rotation) . . . 855
mitkRawConeBeamProjectData (MitkRawConeBeamProjectData - concret class to represent cone-beam project data) . . . 859
mitkRawData (MitkRawData - an abstract class representing Raw Data) . . . 863
mitkRawDataReader (MitkRawDataReader - An abstract class of reading Raw Data) . . . 866
mitkRawDataToVolumeFilter (MitkRawDataToVolumeFilter - an abstract class for volume reconstruction) . . . 868
mitkRawDataWriter (MitkRawDataWriter - an abstract class of writing Raw Data) . . . 870
mitkRawFilesReader (MitkRawFilesReader - a concrete reader for reading a series of files contain raw data (no header information)) . . . 872
mitkRawFilesWriter (MitkRawFilesWriter - a concrete writer for writing a volume to a a series of raw files) . . . 882
mitkRawReader (MitkRawReader - a concrete reader for reading raw volume file(no header information)) . . . 885
mitkRawWriter (MitkRawWriter - a concrete writer for writing a volume to a single raw file(no any header information)) . . . 891
mitkRCPtr< T > (MitkRCPtr - a smart pointer class) . . . 894
mitkReader (MitkReader - an abstract class represents a reader) . . . 895
mitkRectPlane (MitkRectPlane - an encapsulation of a rectangle in 3D space) . . . 897
mitkRectWidgetModel2D (MitkRectWidgetModel2D - a 2D widget for displaying a rectangle in an image scene) . . . 898
mitkRecursiveSphereDirectionEncoder (MitkRecursiveSphereDirectionEncoder - A direction encoder based on the recursive subdivision of an octahedron) . . . 909
mitkRegionGrowImageFilter (MitkRegionGrowImageFilter - A class that implement region grow segmentation arithmetic) . . . 912
mitkRegistrationFilter (MitkRegistrationFilter - a class for registration filter) . . . 915
mitkRenderer (MitkRenderer - an abstract class to define the common interface of a renderer) . . . 921
mitkReslicePlaneWidgetModel (MitkReslicePlaneWidgetModel - a 3D widget for re-slice plane) 924
mitkRGBATransferFunction1D (MitkRGBATransferFunction1D - 1D-Transfer-Function with RGBA control points) . . . 938
mitkRGBToGrayFilter (MitkRGBToGrayFilter - a filter to transfer RGB volume to gray volume) 942
mitkRigidTransform (MitkRigidTransform - a concrete transform to perform rigid transform) . . . 945
mitkScaleTransform (MitkScaleTransform - a concrete transform to perform anisotropic scaling transform) . . . 949
mitkScene (MitkScene - a scene to display 3D surface rendered or volume rendered image) . . . 952
mitkSceneBase . . . 968
mitkSceneContainer . . . 972
mitkSceneParam (MitkSceneParam - storage class for scene parameters) . . . 974
mitkSeedFillFilter (MitkSeedFillFilter - a concrete filter for seed fill algorithm) . . . 975
mitkSemaphore (MitkSemaphore - a class that encapsulates a semaphore) . . . 977
mitkSglNode (MitkSglNode - a class that define mitkSglNode used by mitkNodeHeap) . . . 979
mitkSheetFilter (MitkSheetFilter - a concrete class for implementation of 3D Sheet Structure Enhancement) . . . 980
mitkSimilarityTransform (MitkSimilarityTransform - a concrete transform to perform Similarity transformation) . . . 982
mitkSobelEdgeDetectFilter (MitkSobelEdgeDetectFilter - a class used to detect the edge in an image) . . . 986
mitkSource (MitkSource - abstract class specifies interface for source object) . . . 988
mitkSpeedImageBuilder (MitkSpeedImageBuilder - a class that build SpeedImage for mitkFastMarchingImageFilter) . . . 990
mitkSplatCamera (MitkSplatCamera - a camera in 3D scene) . . . 992
mitkSPMD . . . 995
mitkSPMDDelegate (MitkSPMDDelegate - The base class of mitkSPMDStatic and mitkSPMDMember,) . . . 997

mitkSPMDMember< T > . . . 998
mitkSPMDStatic . . . 1000
mitkSTLASCIIWriter (MitkSTLASCIIWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using ASCII format) . . . 1002
mitkSTLBinaryWriter (MitkSTLBinaryWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using binary format) . . . 1004
mitkSTLReader (MitkSTLReader - a concrete reader for reading a STL file) . . . 1006
mitkSubtractImageFilter (MitkSubtractImageFilter - a concrete class for subtraction of two volumes) . . . 1008
mitkSurfaceModel (MitkSurfaceModel - an 3D entity in a rendering scene represented in surface) 1011
mitkSurfaceProperty (MitkSurfaceProperty - properties of an mitkSurfaceModel) . . . 1015
mitkSurfaceRenderer (MitkSurfaceRenderer - an abstract class for surface renderer object) . . . 1025
mitkSurfaceRendererStandard (MitkSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel) . . . 1027
mitkSurfaceRendererUseVA (MitkSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array) . . . 1029
mitkSurfaceRendererUseVBO (MitkSurfaceRendererUseVBO - a renderer for mitkSurfaceModel using vertex buffer object) . . . 1031
mitkTarget (MitkTarget - abstract class specifies interface for Target object) . . . 1033
mitkTargetMeasurementScene (MitkTargetMeasurementScene - a 3D scene to display a target region from a volume data, long/short axis are detected) . . . 1035
mitkTF2DRGBAClassifier . . . 1040
mitkTF2DRGBAEllipsoidClassifier . . . 1045
mitkTF2DRGBAPSPSplineClassifier . . . 1046
mitkTF2DRGBAPyramidClassifier . . . 1048
mitkTF2DRGBARampClassifier . . . 1049
mitkTF2DRGBASlopeClassifier . . . 1050
mitkTFDK (MitkTFDK - concret class implement cone-beam reconstruction using TFDK Algorithm) . . . 1051
mitkThinPlateSplineTransform (MitkThinPlateSplineTransform - a concrete transform to perform thin-plate spline transform) . . . 1053
mitkThread (MitkThread - a class that encapsulates a thread) . . . 1056
mitkThreadDelegate (MitkThreadDelegate - The base class of mitkThreadStatic and mitkThreadMember,) . . . 1058
mitkThreadMaster (MitkThreadMaster - encapsulates a master of the master/worker model) . . . 1059
mitkThreadMember< T > . . . 1061
mitkThreadStatic . . . 1062
mitkThreadWorker . . . 1063
mitkThreadWorkerCreator . . . 1065
mitkThresholdSegmentationFilter (MitkThresholdSegmentationFilter - a class implement threshold segmentation arithmetic) . . . 1067
mitkTIFFReader (MitkTIFFReader - a concrete reader for reading TIFF image files) . . . 1070
mitkTIFFWriter (MitkTIFFWriter - a concrete writer for writing a volume to TIFF image files) 1073
mitkTrackBall (MitkTrackBall - a virtual trackball for tracking the movement of mouse) . . . 1075
mitkTrackedBscanData (MitkTrackedBscanData - a concrete class encapsulating Raw Data) . . . 1076
mitkTrackedBscanDataReader (MitkTrackedBscanDataReader - A concrete class to read PAI files) . . . 1085
mitkTrackedBscanDataWriter (MitkTrackedBscanDataWriter - A concrete class to write TrackedBscanData to PAI files) . . . 1087
mitkTrackedBscanICData (MitkTrackedBscanICData - a concrete class encapsulating Raw Data) 1089
mitkTrackedBscanOoCData (MitkTrackedBscanOoCData - a concrete class encapsulating Raw Data) . . . 1093
mitkTransferFunction (MitkTransferFunction - an abstract class to map the data property to opacity) . . . 1098

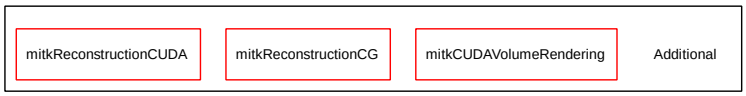
mitkTransferFunction1D (MitkTransferFunction1D - a concrete 1D transfer function to map the data property to opacity) . . . 1100
mitkTransferFunction2DRGBA . . . 1104
mitkTransform (MitkTransform - an abstract class to perform coordinates transformation) . . . 1108
mitkTranslationTransform (MitkTranslationTransform - a concrete transform to perform translation transform) . . . 1116
mitkTriangleMesh (MitkTriangleMesh - an abstract class for triangle meshes) . . . 1119
mitkTriangleMeshSimplification (MitkTriangleMeshSimplification - abstract class for triangle mesh simplification algorithms) . . . 1123
mitkVector (MitkVector - an encapsulation of a 4-element vector) . . . 1125
mitkVersorRigid3DTransform (MitkVersorRigid3DTransform - a concrete transform to perform 3d rigid transformation) . . . 1126
mitkVersorSimilarity3DTransform (MitkVersorSimilarity3DTransform - a concrete transform to perform 3d similarity transform) . . . 1129
mitkView (MitkView - a view to display 3D surface rendered or volume rendered image) . . . 1133
mitkVis2Scene3D (MitkVis2Scene3D - a simple base class for 3D scenes) . . . 1154
mitkVolume (MitkVolume - an abstract class to represent a multi-dimensional medical image dataset) . . . 1161
mitkVolumeCropFilter (MitkVolumeCropFilter - A concrete Filter class to crop a volume) . . . 1179
mitkVolumeDataTypeConvertor (MitkVolumeDataTypeConvertor - a filter to convert the type of volume data) . . . 1182
mitkVolumeDivider (MitkVolumeDivider -) . . . 1184
mitkVolumeIteratorBase (MitkVolumeIterator) . . . 1185
mitkVolumeModel (MitkVolumeModel - an 3D entity in a rendering scene represented in volume) 1186
mitkVolumeProperty (MitkVolumeProperty - properties of an mitkVolumeModel) . . . 1189
mitkVolumeRayCastCompositeFunction (MitkVolumeRayCastCompositeFunction - a concrete ray cast function for compositing) . . . 1197
mitkVolumeRayCastFunction (MitkVolumeRayCastFunction - an abstract class for calculating the ray casting function) . . . 1199
mitkVolumeReader (MitkVolumeReader - an abstract class represents a volume reader to read image/volume files) . . . 1201
mitkVolumeRenderer (MitkVolumeRenderer - an abstract class for volume rendering) . . . 1204
mitkVolumeRendererRayCasting (MitkVolumeRendererRayCasting - a concrete volume renderer for rendering a volume) . . . 1209
mitkVolumeRendererRayCastingLoD (MitkVolumeRendererRayCastingLoD - a concrete volume renderer with LoD function for rendering a volume) . . . 1216
mitkVolumeRendererSplatting (MitkVolumeRendererSplatting - a concrete volume renderer for rendering a volume) . . . 1223
mitkVolumeRendererTexture3D (MitkVolumeRendererTexture3D - a concrete volume renderer for rendering a volume) . . . 1234
mitkVolumeResizeFilter (MitkVolumeResizeFilter - a concrete filter class to zoom the specified volume) . . . 1237
mitkVolumeResliceFilter (MitkVolumeResliceFilter - a volume re-slice filter) . . . 1240
mitkVolumeSplatFunction (MitkVolumeSplatFunction - abstract class defines interface for volume splatting) . . . 1243
mitkVolumeSplatParallel (MitkVolumeSplatParallel - concrete class for parallel volume splatting) 1245
mitkVolumeSplatPerspective (MitkVolumeSplatCompositeFunction - concrete class for perspective volume splatting) . . . 1247
mitkVolumeToMeshFilter (MitkVolumeToMeshFilter - abstract class specifies interface for volume to mesh filter) . . . 1249
mitkVolumeToVolumeFilter (MitkVolumeToVolumeFilter - abstract class specifies interface for volume to volume filter) . . . 1252
mitkVolumeWriter (MitkVolumeWriter - an abstract class represents a volume writer for writing image/volume files to disk) . . . 1256

mitkWatershedSegFilter (MitkWatershedSegFilter - a class implement watershed segmentation arithmetic) 1258
mitkWidgetModel (MitkWidgetModel - abstract class used to represent a widget entity (e.g. a line or an angle) in a rendering scene) 1260
mitkWidgetModel2D (MitkWidgetModel2D - abstract class used to represent a 2D widget entity) 1267
mitkWidgetModel3D (MitkWidgetModel3D - abstract class used to represent a 3D widget entity) 1271
mitkWidgetsViewManipulator (MitkWidgetsViewManipulator - manipulator of a view contains widgets) 1273
mitkWriter (MitkWriter - an abstract class represents a writer) 1276

Chapter 6

MITK Directory Documentation

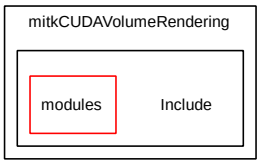
6.1 Additional/ Directory Reference



Directories

- directory mitkCUDAVolumeRendering
- directory mitkReconstructionCG
- directory mitkReconstructionCUDA

6.2 Additional/mitkCUDAVolumeRendering/Include/ Directory Reference



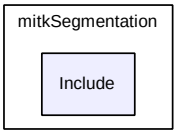
Directories

- directory modules

Files

- file **mitkCUDAAugmentedSurfaceVRScene.h**
- file **mitkCUDADecomposedIsosurfaceVRScene.h**
- file **mitkCUDADeltaAccStateVRScene.h**
- file **mitkCUDAEndoscopySliceScene.h**
- file **mitkCUDAFullVRScene.h**
- file **mitkCUDAIsosurfaceVRScene.h**
- file **mitkCUDAMIPVRScene.h**
- file **mitkCUDAROIDistanceVRScene.h**
- file **mitkCUDASegmentedVRScene.h**
- file **mitkCUDAStructureFunctionVRScene.h**
- file **mitkCUDAValueBasedStateVRScene.h**
- file **mitkCUDAVersatileVRScene.h**
- file **mitkCUDAVolumeRenderingIncludes.h**
- file **mitkCUDAWallCarvingVRScene.h**
- file **mitkCUDAWallPenetrateVRScene.h**

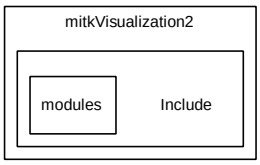
6.3 mitkSegmentation/Include/ Directory Reference



Files

- file **mitkConnectivityLabelFilter.h**
- file **mitkEuclideanSkeletonFeatureExtraction.h**
- file **mitkEuclideanSkeletonGraphCut.h**
- file **mitkGraphCutImageFilter.h**
- file **mitkLiveWireImageFilter.h**
- file **mitkOtsuSegmentationFilter.h**
- file **mitkRegionGrowImageFilter.h**
- file **mitkSeedFillFilter.h**
- file **mitkSegmentationIncludes.h**
- file **mitkSpeedImageBuilder.h**
- file **mitkThresholdSegmentationFilter.h**
- file **mitkWatershedSegFilter.h**

6.4 mitkVisualization2/Include/ Directory Reference



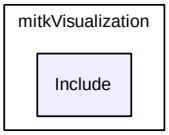
Directories

- directory modules

Files

- file **mitkCoherentVolume.h**
- file **mitkEndoscopySliceScene.h**
- file **mitkFullVRScene.h**
- file **mitkGlewInit.h**
- file **mitkGLSLFullVRScene.h**
- file **mitkGLSLIsosurfaceVRScene.h**
- file **mitkGLSLMIPVRScene.h**
- file **mitkIsosurfaceVRScene.h**
- file **mitkMIPVRScene.h**
- file **mitkMultiLayerThresholds.h**
- file **mitkPathNetworkScene.h**
- file **mitkRGBATransferFunction1D.h**
- file **mitkTargetMeasurementScene.h**
- file **mitkVis2Matrix4x4.h**
- file **mitkVis2Quaternion.h**
- file **mitkVis2Scene3D.h**
- file **mitkVis2SurfaceScene.h**
- file **mitkVis2Vector2.h**
- file **mitkVis2Vector3.h**
- file **mitkVis2Vector4.h**
- file **mitkVisualization2Includes.h**

6.5 mitkVisualization/Include/ Directory Reference



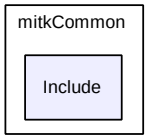
Files

- file **mitkAngleWidgetModel2D.h**
- file **mitkAngleWidgetModel3D.h**
- file **mitkBinMarchingCubes.h**
- file **mitkBSPTree.h**
- file **mitkCamera.h**
- file **mitkClippingPlaneWidgetModel.h**
- file **mitkColorTable.h**
- file **mitkColorTransferFunction.h**
- file **mitkDataModel.h**
- file **mitkDirectionEncoder.h**
- file **mitkEllipseWidgetModel2D.h**
- file **mitkEncodedGradientEstimator.h**
- file **mitkEncodedGradientShader.h**
- file **mitkFiniteDifferenceGradientEstimator.h**
- file **mitkFootprint.h**
- file **mitkFootprint1D.h**
- file **mitkFootprint1DGaussian.h**
- file **mitkFootprint2D.h**
- file **mitkFootprint2DGaussian.h**
- file **mitkGLWindow.h**
- file **mitkImageModel.h**
- file **mitkImageScene.h**
- file **mitkImageView.h**
- file **mitkImageViewManipulatorStandard.h**
- file **mitkImageViewManipulatorWithWidgets.h**
- file **mitkImplementor.h**
- file **mitkLight.h**
- file **mitkLineWidgetModel2D.h**
- file **mitkLineWidgetModel3D.h**
- file **mitkManipulator.h**
- file **mitkMarchingCubes.h**
- file **mitkMeshViewManipulatorStandard.h**
- file **mitkModel.h**
- file **mitkOoCSurfaceRendererStandard.h**
- file **mitkOoCSurfaceRendererUseVA.h**
- file **mitkOoCVolumeRayCastCompositeFunction.h**
- file **mitkOoCVolumeRayCastCompositeFunctionEx.h**
- file **mitkOoCVolumeRendererRayCasting.h**

- file **mitkOoCVolumeRendererRayCastingEx.h**
- file **mitkOoCVolumeRendererShearWarp.h**
- file **mitkOoCVolumeRendererTexture3D.h**
- file **mitkOoCVolumeRendererTexture3DEx.h**
- file **mitkOoCVolumeShearFunction.h**
- file **mitkOoCVolumeShearParallel.h**
- file **mitkOoCVolumeShearPerspective.h**
- file **mitkOrthogonalImageModel.h**
- file **mitkPickManipulator.h**
- file **mitkPolygonWidgetModel2D.h**
- file **mitkPseudocolorWidgetModel.h**
- file **mitkPseudocolorWidgetModelEx.h**
- file **mitkRectWidgetModel2D.h**
- file **mitkRecursiveSphereDirectionEncoder.h**
- file **mitkRenderer.h**
- file **mitkReslicePlaneWidgetModel.h**
- file **mitkScene.h**
- file **mitkSceneBase.h**
- file **mitkSceneContainer.h**
- file **mitkSplatCamera.h**
- file **mitkSurfaceModel.h**
- file **mitkSurfaceProperty.h**
- file **mitkSurfaceRenderer.h**
- file **mitkSurfaceRendererStandard.h**
- file **mitkSurfaceRendererUseVA.h**
- file **mitkSurfaceRendererUseVBO.h**
- file **mitkTF2DRGBAClassifier.h**
- file **mitkTF2DRGBAEllipsoidClassifier.h**
- file **mitkTF2DRGBAPSPSplineClassifier.h**
- file **mitkTF2DRGBAPyramidClassifier.h**
- file **mitkTF2DRGBARampClassifier.h**
- file **mitkTF2DRGBASlopeClassifier.h**
- file **mitkTrackBall.h**
- file **mitkTransferFunction.h**
- file **mitkTransferFunction1D.h**
- file **mitkTransferFunction2DRGBA.h**
- file **mitkView.h**
- file **mitkVisualizationIncludes.h**
- file **mitkVolumeDivider.h**
- file **mitkVolumeModel.h**
- file **mitkVolumeProperty.h**
- file **mitkVolumeRayCastCompositeFunction.h**
- file **mitkVolumeRayCastFunction.h**
- file **mitkVolumeRenderer.h**
- file **mitkVolumeRendererRayCasting.h**
- file **mitkVolumeRendererRayCastingLoD.h**
- file **mitkVolumeRendererSplatting.h**
- file **mitkVolumeRendererTexture3D.h**
- file **mitkVolumeSplatFunction.h**
- file **mitkVolumeSplatParallel.h**

- file **mitkVolumeSplatPerspective.h**
- file **mitkWidgetModel.h**
- file **mitkWidgetModel2D.h**
- file **mitkWidgetModel3D.h**
- file **mitkWidgetsViewManipulator.h**

6.6 mitkCommon/Include/ Directory Reference



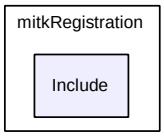
Files

- file **mitkAnalyzeImageReader.h**
- file **mitkAnalyzeImageWriter.h**
- file **mitkBinaryFilter.h**
- file **mitkBMPReader.h**
- file **mitkBMPWriter.h**
- file **mitkCacheVolumeReader.h**
- file **mitkCacheVolumeWriter.h**
- file **mitkCannyEdgeDetectFilter.h**
- file **mitkCommonIncludes.h**
- file **mitkConditionVariable.h**
- file **mitkDataObject.h**
- file **mitkDICOMDIRReader.h**
- file **mitkDICOMDIRTree.h**
- file **mitkDICOMInfoReader.h**
- file **mitkDICOMReader.h**
- file **mitkDICOMStructure.h**
- file **mitkDICOMTags.h**
- file **mitkDICOMWriter.h**
- file **mitkDiffusionFilter.h**
- file **mitkDistanceTransformSaito.h**
- file **mitkEuclideanDistanceTransformFilter.h**
- file **mitkFFTFilter.h**
- file **mitkFilter.h**
- file **mitkGaussianDerivativeImageFilter.h**
- file **mitkGaussianFilter.h**
- file **mitkGeneratePathNetwork.h**
- file **mitkGeometryTypes.h**
- file **mitkGlobal.h**
- file **mitkHalfEdgeStructures.h**
- file **mitkHEICTriangleMesh.h**
- file **mitkHEMesh.h**
- file **mitkHEOoCTriangleMesh.h**
- file **mitkHETriangleMesh.h**
- file **mitkHistogram1D.h**
- file **mitkHistogram2D.h**
- file **mitkICTriangleMesh.h**
- file **mitkICVolume.h**
- file **mitkIM0Reader.h**

- file **mitkIM0Writer.h**
- file **mitkImageIOFactory.h**
- file **mitkInfoReader.h**
- file **mitkIsosurfaceCurvatureFilter.h**
- file **mitkJPEGReader.h**
- file **mitkJPEGWriter.h**
- file **mitkList.h**
- file **mitkMainPathExtraction.h**
- file **mitkMatrix.h**
- file **mitkMatrixD.h**
- file **mitkMesh.h**
- file **mitkMeshReader.h**
- file **mitkMeshToMeshFilter.h**
- file **mitkMeshTransform.h**
- file **mitkMeshWriter.h**
- file **mitkMetaImageReader.h**
- file **mitkMetaImageWriter.h**
- file **mitkMutex.h**
- file **mitkNeighborhood.h**
- file **mitkNeighborhoodIterator.h**
- file **mitkNetCDFImageReader.h**
- file **mitkNetCDFImageWriter.h**
- file **mitkNeumannBoundary.h**
- file **mitkNode.h**
- file **mitkNodeHeap.h**
- file **mitkNumberOfProcessors.h**
- file **mitkObject.h**
- file **mitkObserver.h**
- file **mitkOoCGlobalSettings.h**
- file **mitkOOCIODelegate.h**
- file **mitkOoCTriangleMesh.h**
- file **mitkOoCVolume.h**
- file **mitkPalagyiShrinking_6.h**
- file **mitkPalagyiThinning_6.h**
- file **mitkPath.h**
- file **mitkPathNetwork.h**
- file **mitkPerformanceCounter.h**
- file **mitkPlane.h**
- file **mitkPLYASCIIWriter.h**
- file **mitkPLYBinaryWriter.h**
- file **mitkPLYReader.h**
- file **mitkProcessObject.h**
- file **mitkQEMSimplification.h**
- file **mitkQuaternion.h**
- file **mitkRandom.h**
- file **mitkCommon/Include/mitkRawFilesReader.h**
- file **mitkRawFilesWriter.h**
- file **mitkRawReader.h**
- file **mitkRawWriter.h**
- file **mitkRCPtr.h**

- file **mitkReader.h**
- file **mitkRGBToGrayFilter.h**
- file **mitkSemaphore.h**
- file **mitkSheetFilter.h**
- file **mitkSIMD.h**
- file **mitkSIMDConfig.h**
- file **mitkSobelEdgeDetectFilter.h**
- file **mitkSource.h**
- file **mitkSPMD.h**
- file **mitkSTLASCIIWriter.h**
- file **mitkSTLBinaryWriter.h**
- file **mitkSTLReader.h**
- file **mitkTarget.h**
- file **mitkTetra.h**
- file **mitkThread.h**
- file **mitkThreadMasterWorker.h**
- file **mitkTIFFReader.h**
- file **mitkTIFFWriter.h**
- file **mitkTriangleMesh.h**
- file **mitkTriangleMeshSimplification.h**
- file **mitkVersionCounter.h**
- file **mitkVolume.h**
- file **mitkVolumeCropFilter.h**
- file **mitkVolumeDataTypeConvertor.h**
- file **mitkVolumeIterator.h**
- file **mitkVolumeIteratorOoC.h**
- file **mitkVolumeRandomIterator.h**
- file **mitkVolumeReader.h**
- file **mitkVolumeResizeFilter.h**
- file **mitkVolumeResliceFilter.h**
- file **mitkVolumeToMeshFilter.h**
- file **mitkVolumeToVolumeFilter.h**
- file **mitkVolumeWriter.h**
- file **mitkWriter.h**

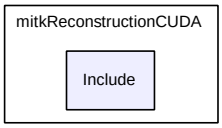
6.7 mitkRegistration/Include/ Directory Reference



Files

- file **mitkAffineTransform.h**
- file **mitkAmoebaOptimizer.h**
- file **mitkBSplineFreeFormTransform.h**
- file **mitkBSplineInterpolateFilter.h**
- file **mitkCubicInterpolateFilter.h**
- file **mitkDeformFieldGenerator.h**
- file **mitkEuclideanDistanceMetric.h**
- file **mitkGlobalRegistrationFramework.h**
- file **mitkGradientDescentOptimizer.h**
- file **mitkInterpolateFilter.h**
- file **mitkIterativeClosestPointOptimizer.h**
- file **mitkKdTree.h**
- file **mitkKdTreePointSet.h**
- file **mitkLandmarkRegistrationFilter.h**
- file **mitkLBFGSBOptimizer.h**
- file **mitkLBFGSOptimizer.h**
- file **mitkLinearInterpolateFilter.h**
- file **mitkMeanSquaresMetric.h**
- file **mitkMetric.h**
- file **mitkNearestNeighborInterpolateFilter.h**
- file **mitkNormalizedCorrelationMetric.h**
- file **mitkOptimizer.h**
- file **mitkPointSet.h**
- file **mitkPointSetEstimator.h**
- file **mitkPointSetKernelTransformEstimator.h**
- file **mitkPointSetMetric.h**
- file **mitkPointSetQuaternionEstimator.h**
- file **mitkPointSetRegistrationFilter.h**
- file **mitkPowellOptimizer.h**
- file **mitkRegistrationFilter.h**
- file **mitkRegistrationIncludes.h**
- file **mitkRigidTransform.h**
- file **mitkScaleTransform.h**
- file **mitkSimilarityTransform.h**
- file **mitkSubtractImageFilter.h**
- file **mitkThinPlateSplineTransform.h**
- file **mitkTransform.h**
- file **mitkTranslationTransform.h**
- file **mitkVersorRigid3DTransform.h**
- file **mitkVersorSimilarity3DTransform.h**

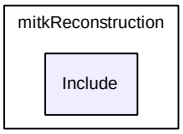
6.8 Additional/mitkReconstructionCUDA/Include/ Directory Reference



Files

- file **mitk3DUSPixelBasedGPU.h**
- file **mitkReconstructionCUDAIncludes.h**

6.9 mitkReconstruction/Include/ Directory Reference

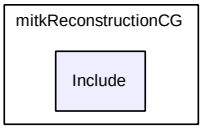


Files

- file **mitk2DBeamICPD.h**
- file **mitk2DBeamOoCPD.h**
- file **mitk2DBeamPD.h**
- file **mitk3DUSPixelBasedMT.h**
- file **mitk3DUSPixelBasedST.h**
- file **mitk3DUSReconstructionFilter.h**
- file **mitkBackProjection.h**
- file **mitkCBFDKReconCF.h**
- file **mitkCBFDKReconCFCPU.h**
- file **mitkCBTFDKReconCF.h**
- file **mitkConeBeamICPD.h**
- file **mitkConeBeamOoCPD.h**
- file **mitkConeBeamPD.h**
- file **mitkConeBeamProjectData.h**
- file **mitkConeBeamRealTimePD.h**
- file **mitkConeBeamReconFromHelix.h**
- file **mitkCTProjectionData.h**
- file **mitkCTReconstructionFilter.h**
- file **mitkFanBeamProjectData.h**
- file **mitkFanBeamReconFromCirle.h**
- file **mitkFbFbp.h**
- file **mitkFBFBPReconCF.h**
- file **mitkFBFBPReconCFCPU.h**
- file **mitkFbFbpSpeedUsingCpu.h**
- file **mitkFDK.h**
- file **mitkFDKSpeedUsingCpu.h**
- file **mitkICConeBeamProjectData.h**
- file **mitkICFanBeamProjectData.h**
- file **mitkICParallelProjectData.h**
- file **mitkOoCConeBeamProjectData.h**
- file **mitkOoCFanBeamProjectData.h**
- file **mitkOoCParallelProjectData.h**
- file **mitkParallelBeamReconFromCirle.h**
- file **mitkParallelProjectData.h**
- file **mitkPbFbp.h**
- file **mitkPBFBPReconCF.h**
- file **mitkPhantom.h**
- file **mitkPhantomToVolumeFilter.h**

- file **mitkProjectData.h**
- file **mitkProjectDataToVolumeFilter.h**
- file **mitkRawConeBeamProjectData.h**
- file **mitkRawData.h**
- file **mitkRawDataReader.h**
- file **mitkRawDataToVolumeFilter.h**
- file **mitkRawDataWriter.h**
- file **mitkReconstruction/Include/mitkRawFilesReader.h**
- file **mitkReconstructionIncludes.h**
- file **mitkTFDK.h**
- file **mitkTrackedBscanData.h**
- file **mitkTrackedBscanDataReader.h**
- file **mitkTrackedBscanDataWriter.h**
- file **mitkTrackedBscanICData.h**
- file **mitkTrackedBscanOoCData.h**

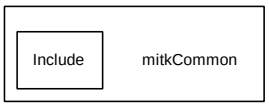
6.10 Additional/mitkReconstructionCG/Include/ Directory Reference



Files

- file **mitk2DBackProjectionUsingGPU.h**
- file **mitk3DBackProjectionUsingGPU.h**
- file **mitkBackProjectionUsingGPU.h**
- file **mitkCBFDKReconCFGPU.h**
- file **mitkFBFBPReconCFGPU.h**
- file **mitkFbfbpSpeedUsingGpu.h**
- file **mitkFDKSpeedUsingGpu.h**
- file **mitkReconstructionCGIncludes.h**

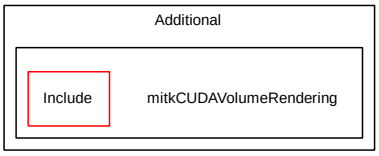
6.11 mitkCommon/ Directory Reference



Directories

- directory Include

6.12 Additional/mitkCUDAVolumeRendering/ Directory Reference



Directories

- directory Include

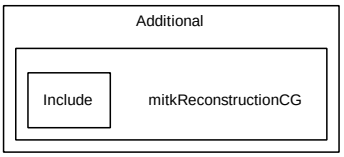
6.13 mitkReconstruction/ Directory Reference



Directories

- directory Include

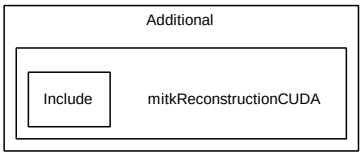
6.14 Additional/mitkReconstructionCG/ Directory Reference



Directories

- directory Include

6.15 Additional/mitkReconstructionCUDA/ Directory Reference



Directories

- directory Include

6.16 mitkRegistration/ Directory Reference



Directories

- directory Include

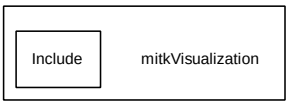
6.17 mitkSegmentation/ Directory Reference



Directories

- directory Include

6.18 mitkVisualization/ Directory Reference



Directories

- directory Include

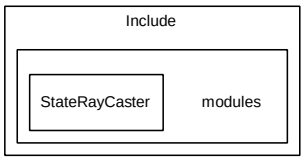
6.19 mitkVisualization2/ Directory Reference



Directories

- directory Include

6.20 Additional/mitkCUDAVolumeRendering/Include/modules/ Directory Reference



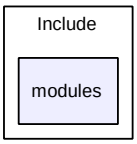
Directories

- directory StateRayCaster

Files

- file **mitkCUDAAugmentedSurfaceRayCasterParam.h**
- file **mitkCUDAAugmentedSurfaceVR.h**
- file **mitkCUDADecomposedSurfaceVR.h**
- file **mitkCUDADecomposedVolume.h**
- file **mitkCUDADepthBuffer.h**
- file **mitkCUDAEndoscopySliceRendering.h**
- file **mitkCUDAFullVR.h**
- file **mitkCUDAGridPartition.h**
- file **mitkCUDAIsosurfaceVR.h**
- file **mitkCUDALut1D.h**
- file **mitkCUDAMIPVR.h**
- file **mitkCUDAPreintegratedLut.h**
- file **mitkCUDAROIDistanceRayCasterParam.h**
- file **mitkCUDAROIDistanceVolume.h**
- file **mitkCUDAROIDistanceVR.h**
- file **mitkCUDASegmentedVolume.h**
- file **mitkCUDASegmentedVolumeLut.h**
- file **mitkCUDASegmentedVR.h**
- file **mitkCUDAShadingLut.h**
- file **mitkCUDAStructureFunctionRayCasterParam.h**
- file **mitkCUDAStructureFunctionVR.h**
- file **mitkCUDATFEmptySpace.h**
- file **mitkCUDAVersatilePreintegratedLut.h**
- file **mitkCUDAVersatileVR.h**
- file **mitkCUDAVolumeTexture.h**
- file **mitkCUDAVRImage.h**
- file **mitkCUDAWallCarvingRayCasterParam.h**
- file **mitkCUDAWallCarvingVolume.h**
- file **mitkCUDAWallCarvingVR.h**
- file **mitkCUDAWallPenetrateRayCasterParam.h**
- file **mitkCUDAWallPenetrateVR.h**

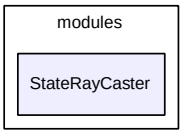
6.21 mitkVisualization2/Include/modules/ Directory Reference



Files

- file **mitkBasicLightParam.h**
- file **mitkDepthBuffer.h**
- file **mitkDepthBufferPBO.h**
- file **mitkEndoscopySliceRendering.h**
- file **mitkFullVR.h**
- file **mitkGLSLFullVR.h**
- file **mitkGLSLIsosurfaceVR.h**
- file **mitkGLSLLut1D.h**
- file **mitkGLSLMIPVR.h**
- file **mitkGLSLProgram.h**
- file **mitkGLSLVolumeTexture.h**
- file **mitkGLSLVRSceneParam.h**
- file **mitkGridPartition.h**
- file **mitkIsosurfaceRayCasterParam.h**
- file **mitkIsosurfaceVR.h**
- file **mitkLut1D.h**
- file **mitkMeshVBO.h**
- file **mitkMIPRayCasterParam.h**
- file **mitkMIPVR.h**
- file **mitkSceneParam.h**
- file **mitkStaticOOP.h**
- file **mitkTFEmptySpace.h**
- file **mitkVR_PBO.h**
- file **mitkVRImage.h**
- file **mitkVRSceneParam.h**

6.22 Additional/mitkCUDAVolumeRendering/Include/modules/StateRayCaster/ Directory Reference



Files

- file **mitkCUDABackwardsStateBuffer.h**
- file **mitkCUDADeltaAccBackwardsTest.h**
- file **mitkCUDADeltaAccStateControl.h**
- file **mitkCUDADeltaAccStateVR.h**
- file **mitkCUDAValueBasedBackwardsTest.h**
- file **mitkCUDAValueBasedStateControl.h**
- file **mitkCUDAValueBasedStateVR.h**

Chapter 7

MITK Namespace Documentation

7.1 AnalyzeImageIO Namespace Reference

mitkAnalyzeImageReader -

Classes

- struct header_key
- struct image_dimension
- struct data_history
- struct dsr

Enumerations

- enum DataTypeKeyValues {

  ANALYZE_DT_UNKNOWN = 0, ANALYZE_DT_BINARY = 1, ANALYZE_DT_UNSIGNED_-CHAR = 2, ANALYZE_DT_SIGNED_SHORT = 4,

  ANALYZE_DT_SIGNED_INT = 8, ANALYZE_DT_FLOAT = 16, ANALYZE_DT_COMPLEX = 32, ANALYZE_DT_DOUBLE = 64,

  ANALYZE_DT_RGB = 128, ANALYZE_DT_ALL = 255, SPMANALYZE_DT_UNSIGNED_-SHORT = 132, SPMANALYZE_DT_UNSIGNED_INT = 136 }
- enum DataTypeIndex
- enum { ANALYZE_ORIENTATION_RPI_TRANSVERSE = 0, ANALYZE_ORIENTATION_-RIP_CORONAL = 1, ANALYZE_ORIENTATION_PIR_SAGITTAL = 2 }

7.1.1 Detailed Description

mitkAnalyzeImageReader -

mitkAnalyzeImageReader

Analyze image file sets consist of at least 2 files:

- an image file ([basename].img or [basename].img.gz or [basename].img.Z) This contains the binary represenation of the raw voxel values. If the file is uncompressed, it should be of of size

(sizeof(storagetype)∗NX∗NY∗NZ(∗NT). The format of the image file is very simple; containing usually uncompressed voxel data for the images in one of the several possible voxel formats:

  - 1 bit packed binary (slices begin on byte boundaries)
  - 8 bit (unsigned char) gray scale unless .lkup file present
  - 16 bit signed short
  - 32 bit signed integers or float
  - 24 bit RGB, 8 bits per channel

- a header file ([basename].hdr) This a 348 byte file 99.99% of all images that contains a binary represenation of the C-struct defined in this file. The analyze 7.5 header structure may, however, be extended beyond this minimal defintion to encompase site specific information, and would have more than 348 bytes. Given that the ability to extend the header has rarely been used, this implementation of the Analyze 7.5 file will only read the first 348 bytes into the structure defined in this file, and all informaiton beyond the 348 bytes will be ignored.

7.1.2 Enumeration Type Documentation

7.1.2.1 anonymous enum

Enumerator:

ANALYZE_ORIENTATION_RPI_TRANSVERSE Denotes a transverse data orientation Right–>Left,

ANALYZE_ORIENTATION_RIP_CORONAL Denotes a coronal data orientation

ANALYZE_ORIENTATION_PIR_SAGITTAL Denotes a sagittal data orientation

7.1.2.2 enum AnalyzeImageIO::DataTypeIndex

The index into the DataTypes array for each type.

7.1.2.3 enum AnalyzeImageIO::DataTypeKeyValues

Acceptable values for hdr.dime.datatype

Enumerator:

ANALYZE_DT_UNKNOWN Deontes that the data type is unknon

ANALYZE_DT_BINARY Deontes that the data type is binary

ANALYZE_DT_UNSIGNED_CHAR Deontes that the data type is unsigned char

ANALYZE_DT_SIGNED_SHORT Deontes that the data type is signed short

ANALYZE_DT_SIGNED_INT Deontes that the data type is signed int

ANALYZE_DT_FLOAT Deontes that the data type is single precision floating point

ANALYZE_DT_COMPLEX Deontes that the data type is pairs of single precision floating point numbers

ANALYZE_DT_DOUBLE Deontes that the data type is double precision floating point

ANALYZE_DT_RGB Deontes that the data type is triples of unsigned char

ANALYZE_DT_ALL Deontes that the data type is unknon

SPMANALYZE_DT_UNSIGNED_SHORT Deontes that the data type is unsigned short in SPM analyze extensions

SPMANALYZE_DT_UNSIGNED_INT Deontes that the data type is unsigned int in SPM analyze extensions

Chapter 8

MITK Class Documentation

8.1 AnalyzeImageIO::data_history Struct Reference

`#include <mitkAnalyzeImageReader.h>`

Public Attributes

- char descrip [80]
- char aux_file [24]
- char orient
- char originator [10]
- char generated [10]
- char scannum [10]
- char patient_id [10]
- char exp_date [10]
- char exp_time [10]
- char hist_un0 [3]
- int views
- int vols_added
- int start_field
- int field_skip
- int omax
- int omin
- int smax
- int smin

8.1.1 Detailed Description

Most of these are historical, relevant to use with the Dynamic Spatial Reconstructor scanner developed at the Mayo BIR in the late 70's and through the 80's. However, the 'orient' field is very important and is used to indicate individual slice orientation and determines whether the Analyze 'Movie' program will attempt to flip the images before displaying a movie sequence. It is perfectly acceptable to put any values you want into those fields because they do not(should not) affect image processing.

- (c) Copyright, 1986-1995

- Biomedical Imaging Resource
- Mayo Foundation

8.1.2 Member Data Documentation

8.1.2.1 char AnalyzeImageIO::data_history::aux_file[24]

A place to put the name of an auxillary file to use instead of the default .img file. This is not currently used by the Analyze program.

8.1.2.2 char AnalyzeImageIO::data_history::descrip[80]

A place to put a short description of the data

8.1.2.3 char AnalyzeImageIO::data_history::exp_date[10]

Experiment date - these were used for DSR experiments

8.1.2.4 char AnalyzeImageIO::data_history::exp_time[10]

Experiment time - these were used for DSR experiments

8.1.2.5 int AnalyzeImageIO::data_history::field_skip

Video field skip factor used in the reconstructed image - these were used for DSR experiments

8.1.2.6 char AnalyzeImageIO::data_history::generated[10]

The date the image was generated.

8.1.2.7 char AnalyzeImageIO::data_history::hist_un0[3]

Unused padding of the structure

8.1.2.8 int AnalyzeImageIO::data_history::omax

The omax/omin and smax/smin relate to rescaling of the value range via the Load process, where both the 'original' omax/omin and the 'scaled' omax/omin could be set to rescale the value range of the data when it is loaded.

8.1.2.9 int AnalyzeImageIO::data_history::omin

See also:
data_history::omax

8.1.2.10 char AnalyzeImageIO::data_history::orient

The 'orient' field in the data_history structure specifies the primary orientation of the data as it is stored in the file on disk. This usually corresponds to the orientation in the plane of acquisition, given that this would correspond to the order in which the data is written to disk by the scanner or other software application.

It would be vary rare that you would ever encounter any old ANALYZE 7.5 files that contain values of 'orient' which indicate that the data has been 'flipped'. The 'flipped flag' values were really only used internal to the Analyze program to precondition data for fast display in the Movie module, where the images were actually flipped vertically in order to accommodate the raster paint order on older graphics devices. The only cases you will encounter will have values of 0, 1, or 2.

- hdr->orient "MayoClinic/Analyze" Origin dims[1] dims[2] dims[3]
- ==========
- 0 transverse-unflipped IRP R->L P->A I->S
- 1 coronal-unflipped IRP R->L I->S P->A
- 2 sagittal-unflipped IRP P->A I->S R->L
- 3 transverse-flipped IRA R->L A->P I->S
- 4 coronal-flipped SRP R->L S->I P->A
- 5 sagittal-flipped ILP P->A I->S L->R
- Where the Origin disignators are with respect to the patient
- [(I)nferior|(S)uperior] [(L}eft|(R)ight] [(A)nterior|(P)osterior]

SPECIAL NOTE: THE BEHAVIOR OF THIS IO ROUTINE DIFFERS FROM THAT OF ANALYZE! NO RE_ORIENTATION OF THE DATA WILL OCCUR IN THE ITK IMPLEMENTATION. Upon loading into the Analyze program, all data is reoriented into the 3D Analyze Coordinate System. The orientation of patient anatomy in the 3D Analyze Coordinate System has a fixed orientation relative to each of the orthogonal axes. This coordinate system does fix the origin in the subject's Inferior-Right-Posterior location relative to anatomy, with a left-handed coordinate system for specification of the other anatomy relative to the axes as given here:

- X-Y plane is Transverse
- X-Z plane is Coronal
- Y-Z plane is Sagittal where:
- X axis runs from patient right (low X) to patient left (high X)
- Y axis runs from posterior (low Y) to anterior (high Y)
- Z axis runs from inferior (low Z) to superior (high Z)

8.1.2.11 char AnalyzeImageIO::data_history::originator[10]

The person or group that generated this image

8.1.2.12 char AnalyzeImageIO::data_history::patient_id[10]

An instituion independant identifier for the subject of the scan

8.1.2.13 char AnalyzeImageIO::data_history::scannum[10]

An instituion independent identifier for the image

8.1.2.14 int AnalyzeImageIO::data_history::smax

See also:
data_history::omax

8.1.2.15 int AnalyzeImageIO::data_history::smin

See also:
data_history::omax

8.1.2.16 int AnalyzeImageIO::data_history::start_field

Video field for first view used in reconstruction - these were used for DSR experiments

8.1.2.17 int AnalyzeImageIO::data_history::views

Number of views in reconstruction - these were used for DSR experiments

8.1.2.18 int AnalyzeImageIO::data_history::vols_added

Number of time points summed together - these were used for DSR experiments

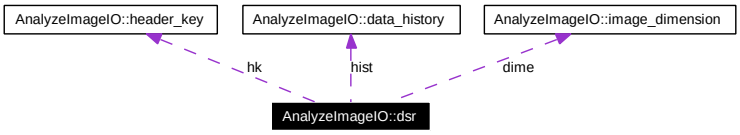
The documentation for this struct was generated from the following file:

- mitkAnalyzeImageReader.h

8.2 AnalyzeImageIO::dsr Struct Reference

`#include <mitkAnalyzeImageReader.h>`

Collaboration diagram for AnalyzeImageIO::dsr:



Public Attributes

- header_key hk
- image_dimension dime
- data_history hist

8.2.1 Detailed Description

Analyze 7.5 header structure The header file is a 'C' structure which describes the dimensions and properties of the voxel data. NOTE: Values in convenience fields should have no effect on how this file is interpreted. These fields are not used read by the Analyze Program, but are written as a convenience for interpreting the file without an analyze compliant file reader.

- (c) Copyright, 1986-1995
- Biomedical Imaging Resource
- Mayo Foundation

8.2.2 Member Data Documentation

8.2.2.1 struct image_dimension AnalyzeImageIO::dsr::dime

The image_dimension structure.

See also:
image_dimension

8.2.2.2 struct data_history AnalyzeImageIO::dsr::hist

The data_history structure.

See also:
data_history

8.2.2.3 struct header_key AnalyzeImageIO::dsr::hk

The header_key structure.

See also:
header_key

The documentation for this struct was generated from the following file:

- mitkAnalyzeImageReader.h

8.3 AnalyzeImageIO::header_key Struct Reference

`#include <mitkAnalyzeImageReader.h>`

Public Attributes

- int sizeof_hdr
- char data_type [10]
- char db_name [18]
- int extents
- short int session_error
- char regular
- char hkey_un0

8.3.1 Detailed Description

- (c) Copyright, 1986-1995
- Biomedical Imaging Resource
- Mayo Foundation

8.3.2 Member Data Documentation

8.3.2.1 char AnalyzeImageIO::header_key::data_type[10]

A convenience character string that has a 1 to 1 correspondence with the short int datatype field of the image_dimension.

See also:
DataTypes

8.3.2.2 char AnalyzeImageIO::header_key::db_name[18]

A convenience character string that should be the same as the the file name without th e∗.hdr or ∗.img extension.

8.3.2.3 int AnalyzeImageIO::header_key::extents

Should be 16384, the image file is created as contiguous with a minium extent size. This field may be used to check endedness of the file

8.3.2.4 char AnalyzeImageIO::header_key::hkey_un0

Unused field for future expansion.

8.3.2.5 char AnalyzeImageIO::header_key::regular

This must be 'r' to indicate that all images and volumes are the same size.

8.3.2.6 short int AnalyzeImageIO::header_key::session_error

This feild is not used for anything other than internal use in Pre 1995 versions of Analyze. Setting this to 0 causes no problems.

8.3.2.7 int AnalyzeImageIO::header_key::sizeof_hdr

Must indicate the byte size of header file, Almost always 348 bytes, but may be larger for sites that have extended the default implementation. This feild is used by analyze to determine if the file is big endian or little endian. If the size of the $*$.hdr file does not equal this value, then the structure needs to be byte swapped.

The documentation for this struct was generated from the following file:

- mitkAnalyzeImageReader.h

8.4 AnalyzeImageIO::image_dimension Struct Reference

Public Attributes

- short int dim [8]
- char vox_units [4]
- char cal_units [8]
- short int unused1
- short int datatype
- short int bitpix
- short int dim_un0
- float pixdim [8]
- float vox_offset
- float roi_scale
- float funused1
- float funused2
- float cal_max
- float cal_min
- int compressed
- int verified
- int glmax
- int glmin

8.4.1 Detailed Description

the organization and size of images. These elements enable IO routines to reference images by volume and slice number.

- (c) Copyright, 1986-1995
- Biomedical Imaging Resource
- Mayo Foundation

8.4.2 Member Data Documentation

8.4.2.1 short int AnalyzeImageIO::image_dimension::bitpix

Bits per pixel. This field must agree with the datatype and data_type fields.

See also:
DataTypeSizes

8.4.2.2 float AnalyzeImageIO::image_dimension::cal_max

The parameters 'cal_max' and 'cal_min' provided a mechanism for rescaling the voxel values into a different range of value representation. This string allowed whatever the range of value representation was to be expressed with these string characters. Never really used in a robust manner.

8.4.2.3 float AnalyzeImageIO::image_dimension::cal_min

See also:
image_dimension::cal_max

8.4.2.4 char AnalyzeImageIO::image_dimension::cal_units[8]

Specifies the name of the calibration unit, valid values are "mm","cm" and "in". NOTE: if no match is found, "mm" is assumed.

8.4.2.5 int AnalyzeImageIO::image_dimension::compressed

Valid values are only 0 and 1. A 0 value indicates that the file is uncompressed. A 1 value indicates that the ∗.img file has been compress with the standard Unix 'compress' utility.

- filename, compressed value, description
- basename.img, 0, uncompressed image file of size
- basename.img, 1, compressed image file using unix 'compress' utility
- basename.img.Z, 0, compressed image using unix 'compress' utiltiy
- basename.img.gz, 0, compressed image file gzip utility
- basename.img.Z, 1, invalid setting
- basename.img.gz, 1, invalid setting

8.4.2.6 short int AnalyzeImageIO::image_dimension::datatype

A short int that defines the type of data being stored. NOTE: THIS MUST BE FILLED IN. This is the field that most applications use to determine the data storage type. Valid values are defined by DataTypeKeyValues.

See also:
DataTypeKeyValues

8.4.2.7 short int AnalyzeImageIO::image_dimension::dim[8]

Array of image dimensions

- dim[0] number of dimensions; usually 4
- dim[1] image X dimension, i.e. number of voxels per row (adjacent memory locations)
- dim[2] image Y dimension, i.e. number of rows per slice
- dim[3] Volume Z dimension, i.e. number of slices per volume
- dim[4] Number of time points, i.e. number of volumes per series
- . . .
- dim[7] volumes in file, i.e. number of volumes per series

8.4.2.8 short int AnalyzeImageIO::image_dimension::dim_un0

Unused value for future expansion.

8.4.2.9 float AnalyzeImageIO::image_dimension::funused1

Unused, for future expansion

8.4.2.10 float AnalyzeImageIO::image_dimension::funused2

Unused, for future expansion

8.4.2.11 int AnalyzeImageIO::image_dimension::glmax

This is the global max/min for the entire volume image (or sets of volume images if there are multiple volumes in this file). The cal_max/min are used only if the desired representation range of the data is different from the true range of the data. For example, CT data may be stored in signed shorts with a range from 0 - 2000, but an appropriate representation range for these values in in Hounsfield units from -1000 to 1000. Any reported values in Analyze were scaled via this value scaling function. UNSIGNED INT and COMPLEX were really not supported under ANALYZE 7.5., so these should be avoided in the Analyze 7.5 file format. The max/min for FLOATs was the nearest integer to the floating point values.

8.4.2.12 int AnalyzeImageIO::image_dimension::glmin

See also:
image_dimension::glmax

8.4.2.13 float AnalyzeImageIO::image_dimension::pixdim[8]

Parallel array to dim giving voxel dimensions NOTE: pixdim[0] is ignored, and the number of dims are taken from dims[0], and pixdim[1..7] are the actual pixdims.

- pixdim[0] this field is not used
- pixdim[1] voxel width
- pixdim[2] voxel height
- pixdim[3] voxel depth or slice thickness
- pixdim[4]
- . . .
- pixdim[7]

8.4.2.14 float AnalyzeImageIO::image_dimension::roi_scale

The "ROI Scale Factor" was really added in by the developers of SPM. This is used as a multiplicative factor to scale all values measured in the ROI module by this multiplicative factor. This really shouldn't be set by any other application outside of Analyze and SPM.

8.4.2.15 short int AnalyzeImageIO::image_dimension::unused1

Unused field for future expansion.

8.4.2.16 int AnalyzeImageIO::image_dimension::verified

This was only used internally by Pre 1995 versions of analyze.

8.4.2.17 float AnalyzeImageIO::image_dimension::vox_offset

Byte offset in the .img file at which voxels start. If value is negative specifies that the absolute value is applied for every image in the file.

8.4.2.18 char AnalyzeImageIO::image_dimension::vox_units[4]

Specifies the spatial units of measure for a voxel, valid values are "mm","cm" and "in". NOTE: if no match is found, "mm" is assumed.

The documentation for this struct was generated from the following file:

- mitkAnalyzeImageReader.h

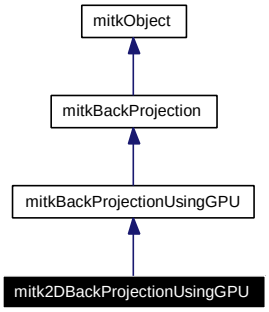
8.5 mitk2DBackProjectionUsingGPU Class Reference

mitk2DBackProjectionUsingGPU - abstract base class for CT back projection

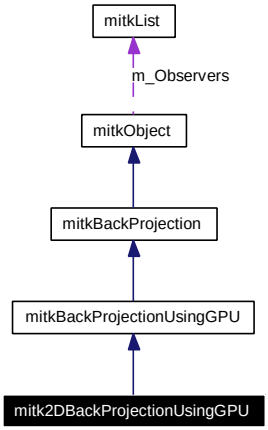
`#include <mitk2DBackProjectionUsingGPU.h>`

Inherits mitkBackProjectionUsingGPU.

Inheritance diagram for mitk2DBackProjectionUsingGPU:



Collaboration diagram for mitk2DBackProjectionUsingGPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.5.1 Detailed Description

mitk2DBackProjectionUsingGPU - abstract base class for CT back projection

mitk2DBackProjectionUsingGPU is an abstract object that specifies behavior and interface of CT backprojection.

8.5.2 Member Function Documentation

8.5.2.1 virtual void mitk2DBackProjectionUsingGPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkBackProjectionUsingGPU.

The documentation for this class was generated from the following file:

- mitk2DBackProjectionUsingGPU.h

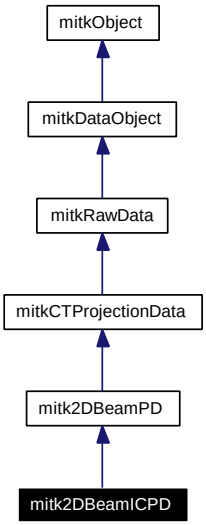
8.6 mitk2DBeamICPD Class Reference

mitk2DBeamICPD - concret class to represent fan-beam and parallel projection data.

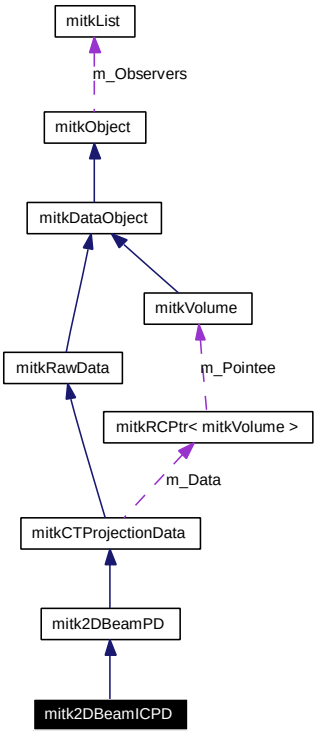
`#include <mitk2DBeamICPD.h>`

Inherits mitk2DBeamPD.

Inheritance diagram for mitk2DBeamICPD:



Collaboration diagram for mitk2DBeamICPD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void Initialize ()
- virtual void const * GetSliceForRead (int sliceIdx)

8.6.1 Detailed Description

mitk2DBeamICPD - concret class to represent fan-beam and parallel projection data.

mitk2DBeamICPD - concret class to represent fan-beam and parallel projection data.

8.6.2 Member Function Documentation

8.6.2.1 virtual void mitk2DBeamICPD::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitk2DBeamPD.

8.6.2.2 virtual unsigned long long mitk2DBeamICPD::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitk2DBeamPD.

8.6.2.3 virtual void const* mitk2DBeamICPD::GetSliceForRead (int *sliceIdx*) [virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implements mitk2DBeamPD.

8.6.2.4 virtual void mitk2DBeamICPD::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitk2DBeamPD.

8.6.2.5 virtual void mitk2DBeamICPD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitk2DBeamPD.

8.6.2.6 virtual void mitk2DBeamICPD::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitk2DBeamPD.

The documentation for this class was generated from the following file:

- mitk2DBeamICPD.h

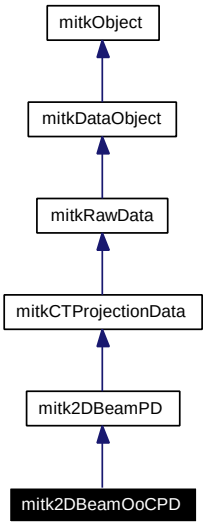
8.7 mitk2DBeamOoCPD Class Reference

mitk2DBeamOoCPD - concret class to represent fan-beam project data.

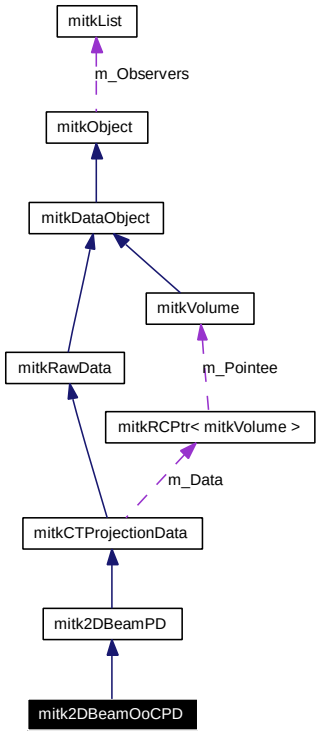
`#include <mitk2DBeamOoCPD.h>`

Inherits mitk2DBeamPD.

Inheritance diagram for mitk2DBeamOoCPD:



Collaboration diagram for mitk2DBeamOoCPD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void Initialize ()
- virtual void const * GetSliceForRead (int sliceIdx)

8.7.1 Detailed Description

mitk2DBeamOoCPD - concret class to represent fan-beam project data.

mitk2DBeamOoCPD - concret class to represent fan-beam project data.

8.7.2 Member Function Documentation

8.7.2.1 virtual void mitk2DBeamOoCPD::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitk2DBeamPD.

8.7.2.2 virtual unsigned long long mitk2DBeamOoCPD::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitk2DBeamPD.

8.7.2.3 virtual void const* mitk2DBeamOoCPD::GetSliceForRead (int *sliceIdx*) [virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implements mitk2DBeamPD.

8.7.2.4 virtual void mitk2DBeamOoCPD::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitk2DBeamPD.

8.7.2.5 virtual void mitk2DBeamOoCPD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitk2DBeamPD.

8.7.2.6 virtual void mitk2DBeamOoCPD::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitk2DBeamPD.

The documentation for this class was generated from the following file:

- mitk2DBeamOoCPD.h

8.8 mitk2DBeamPD Class Reference

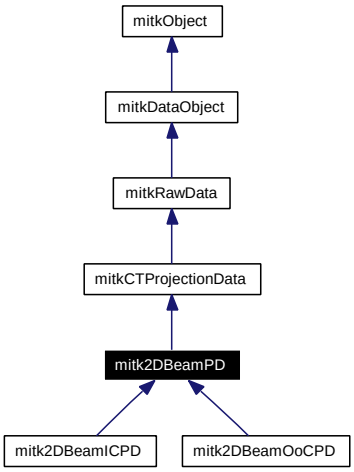
mitk2DBeamPD - abstract class to represent fan-beam project data.

`#include <mitk2DBeamPD.h>`

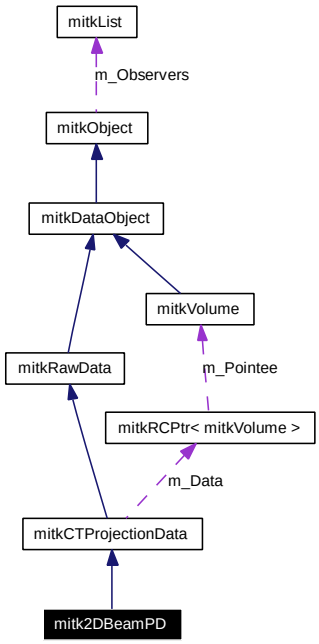
Inherits mitkCTProjectionData.

Inherited by mitk2DBeamICPD, and mitk2DBeamOoCPD.

Inheritance diagram for mitk2DBeamPD:



Collaboration diagram for mitk2DBeamPD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()
- virtual void const * GetSliceForRead (int sliceIdx)=0
- virtual unsigned long long GetActualMemorySize () const =0
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0
- virtual void const * GetVolumeData () const
- virtual void * GetVolumeData ()
- void SetVolumeData (mitkVolume *)

8.8.1 Detailed Description

mitk2DBeamPD - abstract class to represent fan-beam project data.

mitk2DBeamPD - abstract class to represent fan-beam project data. . ... sod . . . . . o . ..... sdd . . . .

8.8.2 Member Function Documentation

8.8.2.1 virtual void mitk2DBeamPD::DeepCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitk2DBeamICPD, and mitk2DBeamOoCPD.

8.8.2.2 virtual unsigned long long mitk2DBeamPD::GetActualMemorySize () const [pure virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Note:
Pure virtual function. Its concrete subclass must implement this function and return its memory size.

Implements mitkDataObject.

Implemented in mitk2DBeamICPD, and mitk2DBeamOoCPD.

8.8.2.3 virtual void const* mitk2DBeamPD::GetSliceForRead (int *sliceIdx*) [pure virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implemented in mitk2DBeamICPD, and mitk2DBeamOoCPD.

8.8.2.4 virtual void* mitk2DBeamPD::GetVolumeData () [inline, virtual]

get volume pointer

Implements mitkCTProjectionData.

8.8.2.5 virtual void const* mitk2DBeamPD::GetVolumeData () const [inline, virtual]

get a const volume pointer

Implements mitkCTProjectionData.

8.8.2.6 virtual void mitk2DBeamPD::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkCTProjectionData.

Reimplemented in mitk2DBeamICPD, and mitk2DBeamOoCPD.

8.8.2.7 virtual void mitk2DBeamPD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkCTProjectionData.

Reimplemented in mitk2DBeamICPD, and mitk2DBeamOoCPD.

8.8.2.8 void mitk2DBeamPD::SetVolumeData (mitkVolume *)

Set data pointer of the project data.

8.8.2.9 virtual void mitk2DBeamPD::ShallowCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitk2DBeamICPD, and mitk2DBeamOoCPD.

The documentation for this class was generated from the following file:

- mitk2DBeamPD.h

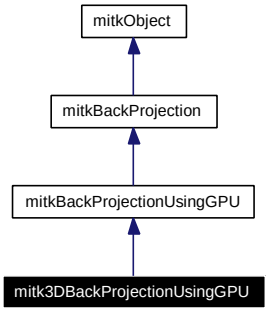
8.9 mitk3DBackProjectionUsingGPU Class Reference

mitk3DBackProjectionUsingGPU - abstract base class for CT back projection

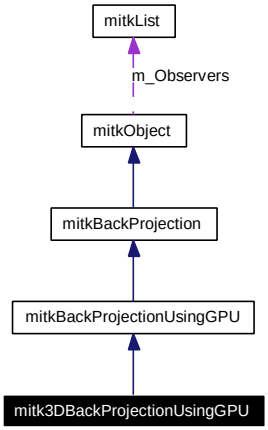
`#include <mitk3DBackProjectionUsingGPU.h>`

Inherits mitkBackProjectionUsingGPU.

Inheritance diagram for mitk3DBackProjectionUsingGPU:



Collaboration diagram for mitk3DBackProjectionUsingGPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.9.1 Detailed Description

mitk3DBackProjectionUsingGPU - abstract base class for CT back projection

mitk3DBackProjectionUsingGPU is an abstract object that specifies behavior and interface of CT back projection.

8.9.2 Member Function Documentation

8.9.2.1 virtual void mitk3DBackProjectionUsingGPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkBackProjectionUsingGPU.

The documentation for this class was generated from the following file:

- mitk3DBackProjectionUsingGPU.h

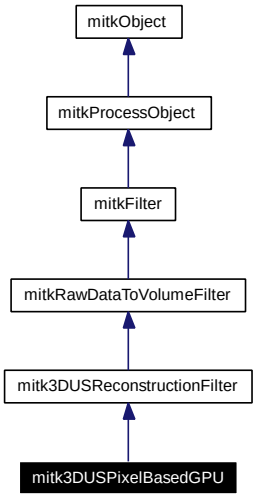
8.10 mitk3DUSPixelBasedGPU Class Reference

mitk3DUSPixelBasedGPU -

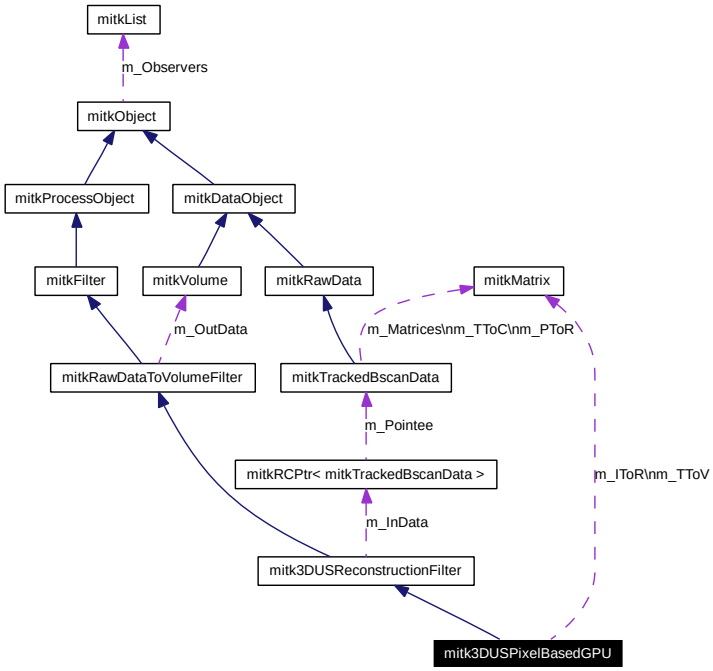
`#include <mitk3DUSPixelBasedGPU.h>`

Inherits mitk3DUSReconstructionFilter.

Inheritance diagram for mitk3DUSPixelBasedGPU:



Collaboration diagram for mitk3DUSPixelBasedGPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void UsePNN ()
- void UseP3DK ()
- virtual mitkVolume * GetOutput ()

8.10.1 Detailed Description

mitk3DUSPixelBasedGPU -

mitk3DUSPixelBasedGPU

8.10.2 Member Function Documentation

8.10.2.1 virtual mitkVolume* mitk3DUSPixelBasedGPU::GetOutput () [virtual]

Get the output volume

Returns:
Return the output volume

Implements mitkRawDataToVolumeFilter.

8.10.2.2 virtual void mitk3DUSPixelBasedGPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitk3DUSReconstructionFilter.

8.10.2.3 void mitk3DUSPixelBasedGPU::UseP3DK () [inline]

Use P3DK method for volume reconstruction.

8.10.2.4 void mitk3DUSPixelBasedGPU::UsePNN () [inline]

Use PNN method for volume reconstruction (Default is PNN).

The documentation for this class was generated from the following file:

- mitk3DUSPixelBasedGPU.h

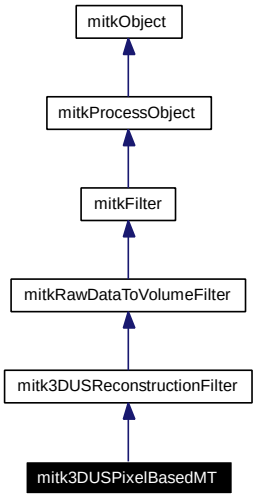
8.11 mitk3DUSPixelBasedMT Class Reference

mitk3DUSPixelBasedMT - a concrete class for 3D US Reconstruction

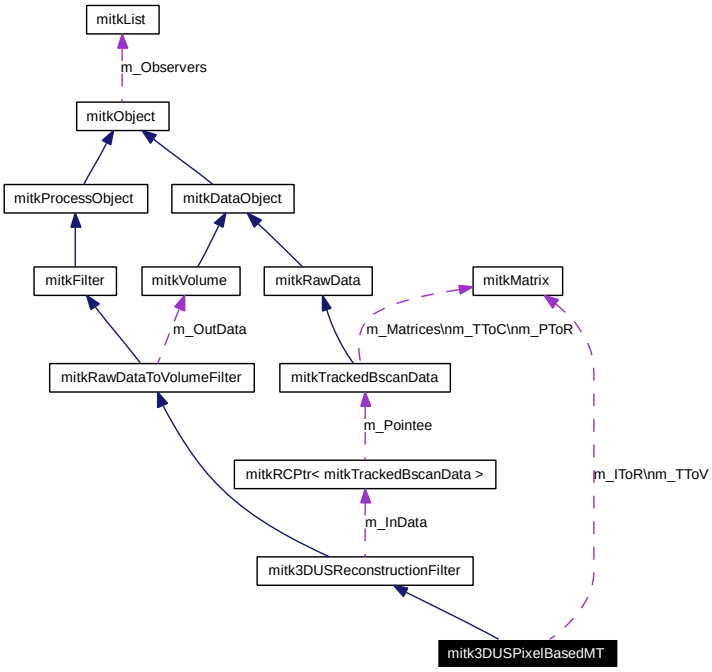
`#include <mitk3DUSPixelBasedMT.h>`

Inherits mitk3DUSReconstructionFilter.

Inheritance diagram for mitk3DUSPixelBasedMT:



Collaboration diagram for mitk3DUSPixelBasedMT:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void UsePNN ()
- void UseP3DK ()
- virtual mitkVolume * GetOutput ()

8.11.1 Detailed Description

mitk3DUSPixelBasedMT - a concrete class for 3D US Reconstruction

mitk3DUSPixelBasedMT is a concrete class for 3D US Reconstruction. It encapsulates several Pixel Based Reconstruction algorithms, including the Pixel Nearest Neighbor (PNN) interpolation with hole filling (3*3*3), and Pixel 3D Kernel (P3DK)(2*2*2 kernel) interpolation with hole filling (3*3*3).

To use this filter, the code snippet is:

```
mitk3DUSPixelBasedMT *filter = new mitk3DUSPixelBasedMT;
filter->SetInput(inTrackedBscanData);
filter->UsePNN() or filter->UseP3DK();
if (filter->Run())
{
      mitkVolume *outVolume = filter->GetOutput();
      Using  outVolume...
}
```

Note: If the mitkTrackedBscanData object was generated by a mitkTrackedBscanDataReader object, the parameters about the reconstruction volume (TTOC matrix, volume size and voxel size) were given. Then you need not set anything to the object, just use it. BUT, If you create and set the mitkTrackedBscanData object by yourself, please remember to compute the TTOC matrix, volume size and voxel size and set them to the mitkTrackedBscanData object, and use SetPredetermined(true) to notify the mitkTrackedBscanData object you have set these parameters. If you use SetPredetermined(false) after you have set these parameters, the mitk3DUSPixelBasedMT object will automatically compute these parameters and use the automatically computed ones for volume reconstruction. If you really want the mitk3DUSPixelBasedMT object to compute these parameters, just use SetPredetermined(false). If you forget to use SetPredetermined(), this filter will use the default status (false), that is, the filter will compute the parameters automatically.

8.11.2 Member Function Documentation

8.11.2.1 virtual mitkVolume* mitk3DUSPixelBasedMT::GetOutput () [virtual]

Get the output volume

Returns:
Return the output volume

Implements mitkRawDataToVolumeFilter.

8.11.2.2 virtual void mitk3DUSPixelBasedMT::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitk3DUSReconstructionFilter.

8.11.2.3 void mitk3DUSPixelBasedMT::UseP3DK () [inline]

Use P3DK method for volume reconstruction.

8.11.2.4 void mitk3DUSPixelBasedMT::UsePNN () [inline]

Use PNN method for volume reconstruction (Default is PNN).

The documentation for this class was generated from the following file:

- mitk3DUSPixelBasedMT.h

8.12 mitk3DUSReconstructionFilter Class Reference

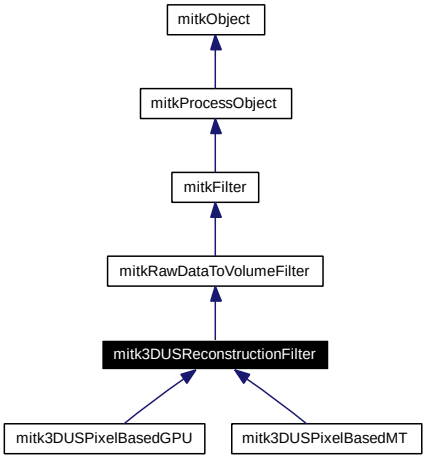
mitk3DUSReconstructionFilter - an abstract class for 3D US Reconstruction

`#include <mitk3DUSReconstructionFilter.h>`

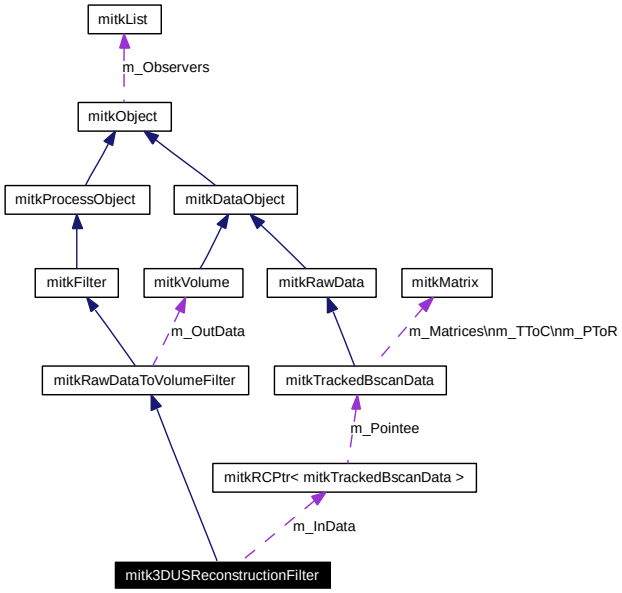
Inherits mitkRawDataToVolumeFilter.

Inherited by mitk3DUSPixelBasedGPU, mitk3DUSPixelBasedMT, and mitk3DUSPixelBasedST.

Inheritance diagram for mitk3DUSReconstructionFilter:



Collaboration diagram for mitk3DUSReconstructionFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- void SetInput (mitkTrackedBscanData *inData)
- mitkTrackedBscanData * GetInput ()

8.12.1 Detailed Description

mitk3DUSReconstructionFilter - an abstract class for 3D US Reconstruction

mitk3DUSReconstructionFilter is an abstract class specifying input interfaces for 3D Ultrasound Reconstruction classes. Concrete 3D US reconstruction classes process an input mitkTrackedBscanData object and generate an output mitkVolume object.

8.12.2 Member Function Documentation

8.12.2.1 mitkTrackedBscanData* mitk3DUSReconstructionFilter::GetInput () [inline]

Get the input Tracked B-Scan Data

Returns:
Return the input Tracked B-Scan Data

8.12.2.2 virtual void mitk3DUSReconstructionFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRawDataToVolumeFilter.

Reimplemented in mitk3DUSPixelBasedMT, and mitk3DUSPixelBasedGPU.

8.12.2.3 void mitk3DUSReconstructionFilter::SetInput (mitkTrackedBscanData * *inData*) [inline]

Set the input Tracked B-Scan Data

Parameters:
inData Specify the input Tracked B-Scan Data

The documentation for this class was generated from the following file:

- mitk3DUSReconstructionFilter.h

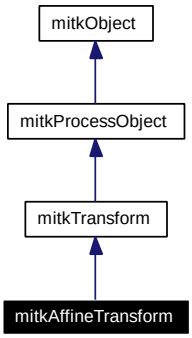
8.13 mitkAffineTransform Class Reference

mitkAffineTransform - a concrete transform to perform affine transformation

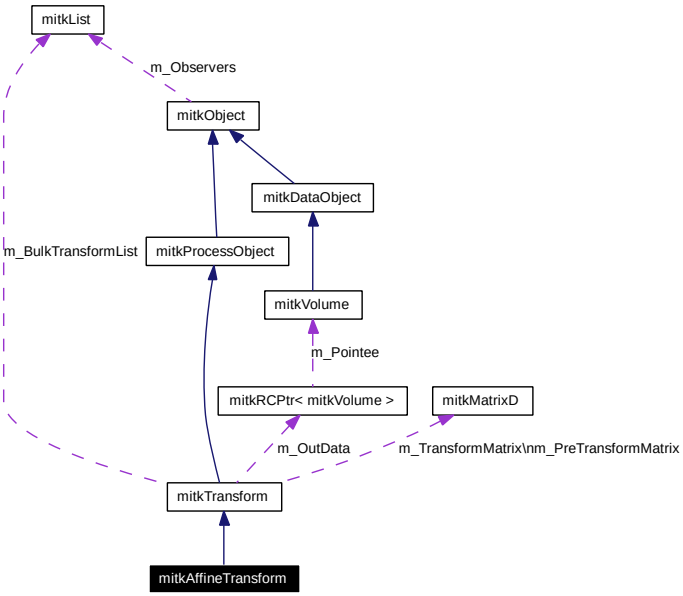
`#include <mitkAffineTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkAffineTransform:



Collaboration diagram for mitkAffineTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkAffineTransform ()
- mitkAffineTransform (unsigned int dim)
- const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- virtual void ConvertTransformMaxtrixToVector ()

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.13.1 Detailed Description

mitkAffineTransform - a concrete transform to perform affine transformation

mitkAffineTransform is a concrete transform to perform affine transformation. There are six transform freedoms in two dimensional affine transform and twelve transform freedoms in three dimensional affine transform. User should use the constructor to specify the image dimensions, e.g. mitkAffineTransform(2) (default) for 2d image, mitkAffineTransform(3) for 3d image.

Transform Mode Number of parameters p[0] p[1] p[2] p[3] p[4] p[5] p[6] p[7] p[8] p[9] p[10] p[11]

MITK_TRANSFORM_AFFINE (2D) 6 m0 m4 m1 m5 tx ty - - - - - -

MITK_TRANSFORM_AFFINE (3D) 12 m0 m4 m8 m1 m5 m9 m2 m6 m10 tx ty tz

8.13.2 Constructor & Destructor Documentation

8.13.2.1 mitkAffineTransform::mitkAffineTransform ()

Default Constructor.

8.13.2.2 mitkAffineTransform::mitkAffineTransform (unsigned int *dim*)

Constructor with specific image dimensions setting.

8.13.3 Member Function Documentation

8.13.3.1 virtual void mitkAffineTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.13.3.2 virtual void mitkAffineTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.13.3.3 const MatrixType& mitkAffineTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:
inPoint The index of the point in image.

Returns:
Return the pointer to the Jacobian matrix.

Reimplemented from mitkTransform.

8.13.3.4 virtual void mitkAffineTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkAffineTransform.h

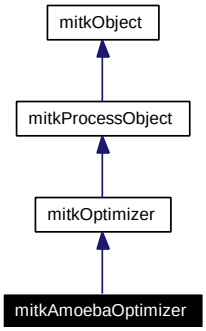
8.14 mitkAmoebaOptimizer Class Reference

mitkAmoebaOptimizer - a concrete class for implementation of the Nelder-Meade downhill simplex algorithm

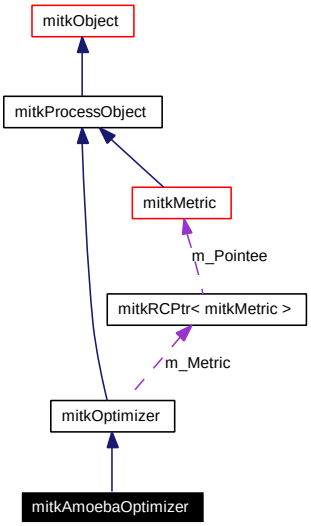
`#include <mitkAmoebaOptimizer.h>`

Inherits mitkOptimizer.

Inheritance diagram for mitkAmoebaOptimizer:



Collaboration diagram for mitkAmoebaOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkAmoebaOptimizer ()
- ScalarParameterType GetValue (const VectorParameterType &p)
- void SetParametersConvergenceTolerance (ScalarParameterType tol)
- void SetFunctionConvergenceTolerance (ScalarParameterType tol)
- void SetAutomaticInitialSimplexFlag (bool flag)
- void SetInitialSimplexDelta (ScalarParameterType *delta, int dim)
- VectorParameterType & GetInitialSimplexDelta ()

8.14.1 Detailed Description

mitkAmoebaOptimizer - a concrete class for implementation of the Nelder-Meade downhill simplex algorithm

mitkAmoebaOptimizer - a concrete class for implementation of the Nelder-Meade downhill simplex algorithm. It works by creating a simplex (n+1 points in n-D space) which then crawls about the space searching for the solution.

By default the set of (n+1) starting points are generated by applying a scaling (relative_diameter) to each element of the supplied starting vector, with a small offset used instead if the value is zero.

Alternatively, if one uses minimize(x,dx), then the starting points are obtained by adding each dx[i] to the elements of x, one at a time. This is useful if you know roughly the scale of your space.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 mitkAmoebaOptimizer::mitkAmoebaOptimizer ()

Constructor.

8.14.3 Member Function Documentation

8.14.3.1 VectorParameterType& mitkAmoebaOptimizer::GetInitialSimplexDelta ()

Get the pointer to the vector of initialized simplex data.

Returns:
The pointer to initialized simplex vector.

8.14.3.2 ScalarParameterType mitkAmoebaOptimizer::GetValue (const VectorParameterType & *p*)

Get the cost function value at the given parameters.

Parameters:
p The given transform parameters.

Returns:
The cost function value.

8.14.3.3 virtual void mitkAmoebaOptimizer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkOptimizer.

8.14.3.4 void mitkAmoebaOptimizer::SetAutomaticInitialSimplexFlag (bool *flag*) [inline]

Set the mode which determines how the amoeba algorithm defines the initial simplex. Default is on.

Parameters:
flag Automatically initialize simplex if flag is true.

8.14.3.5 void mitkAmoebaOptimizer::SetFunctionConvergenceTolerance (ScalarParameterType *tol*) [inline]

Set the cost function convergence threshold.

Parameters:
tol The function convergence value.

8.14.3.6 void mitkAmoebaOptimizer::SetInitialSimplexDelta (ScalarParameterType * *delta*, int *dim*)

Define the initial simplex, setting the ith corner of the simplex as [x0[0], x0[1], ..., x0[i]+InitialSimplex-Delta[i], ...,* x0[d-1]].

Parameters:
delta An array carried initialized simplex data.

dim Parameter space dimensions.

8.14.3.7 void mitkAmoebaOptimizer::SetParametersConvergenceTolerance (ScalarParameterType *tol*) [inline]

Set the simplex diameter threshold value, which will terminate optimization algorithm when the simplex diameter and the difference in cost function at the corners of the simplex falls below user specified thresholds.

Parameters:
tol The simplex convergence tolerance value.

The documentation for this class was generated from the following file:

- mitkAmoebaOptimizer.h

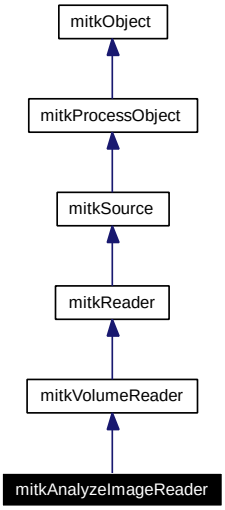
8.15 mitkAnalyzeImageReader Class Reference

mitkAnalyzeImageReader -

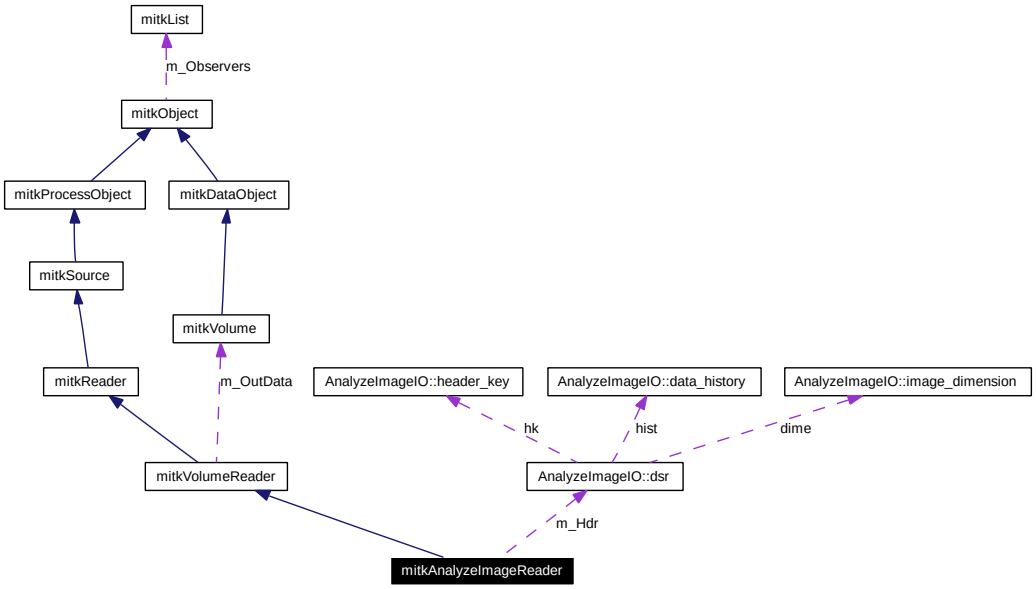
`#include <mitkAnalyzeImageReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkAnalyzeImageReader:



Collaboration diagram for mitkAnalyzeImageReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.15.1 Detailed Description

mitkAnalyzeImageReader -

mitkAnalyzeImageReader

Analyze image file sets consist of at least 2 files:

- an image file ([basename].img or [basename].img.gz or [basename].img.Z) This contains the binary represenation of the raw voxel values. If the file is uncompressed, it should be of of size (sizeof(storagetype)∗NX∗NY∗NZ(∗NT). The format of the image file is very simple; containing usually uncompressed voxel data for the images in one of the several possible voxel formats:
  - 1 bit packed binary (slices begin on byte boundaries)
  - 8 bit (unsigned char) gray scale unless .lkup file present
  - 16 bit signed short
  - 32 bit signed integers or float
  - 24 bit RGB, 8 bits per channel
- a header file ([basename].hdr) This a 348 byte file 99.99% of all images that contains a binary represenation of the C-struct defined in this file. The analyze 7.5 header structure may, however, be extended beyond this minimal defintion to encompase site specific information, and would have more than 348 bytes. Given that the ability to extend the header has rarely been used, this implementation of the Analyze 7.5 file will only read the first 348 bytes into the structure defined in this file, and all informaiton beyond the 348 bytes will be ignored.

8.15.2 Member Function Documentation

8.15.2.1 virtual void mitkAnalyzeImageReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

The documentation for this class was generated from the following file:

- mitkAnalyzeImageReader.h

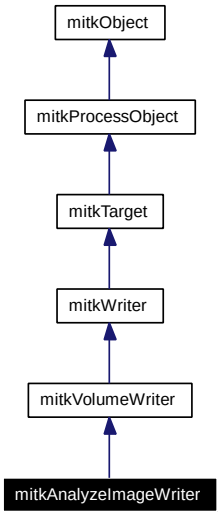
8.16 mitkAnalyzeImageWriter Class Reference

mitkAnalyzeImageWriter -

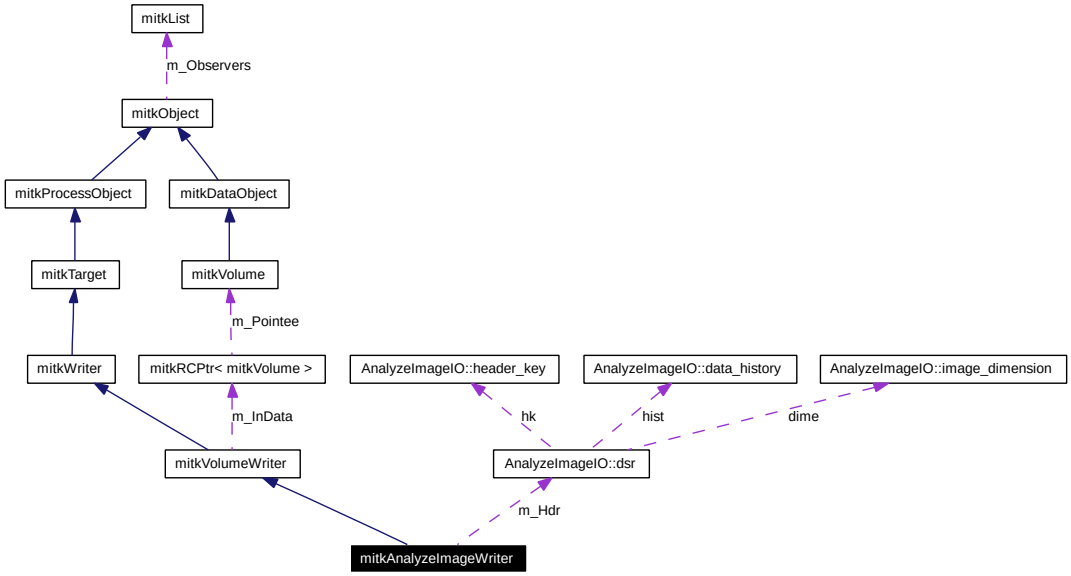
`#include <mitkAnalyzeImageWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkAnalyzeImageWriter:



Collaboration diagram for mitkAnalyzeImageWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.16.1 Detailed Description

mitkAnalyzeImageWriter -

mitkAnalyzeImageWriter

8.16.2 Member Function Documentation

8.16.2.1 virtual void mitkAnalyzeImageWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

The documentation for this class was generated from the following file:

- mitkAnalyzeImageWriter.h

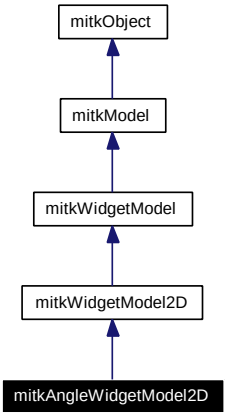
8.17 mitkAngleWidgetModel2D Class Reference

mitkAngleWidgetModel2D - a 2D widget for measuring angles

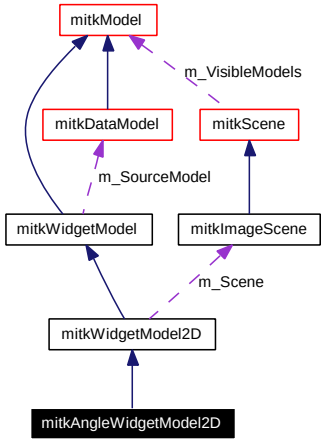
`#include <mitkAngleWidgetModel2D.h>`

Inherits mitkWidgetModel2D.

Inheritance diagram for mitkAngleWidgetModel2D:



Collaboration diagram for mitkAngleWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- void SetStartPoint (float p[2])
- void SetStartPoint (float x, float y)
- void SetStartPoint (int sx, int sy)
- void SetEndPoint0 (float p[2])

- void SetEndPoint0 (float x, float y)
- void SetEndPoint0 (int sx, int sy)
- void SetEndPoint1 (float p[2])
- void SetEndPoint1 (float x, float y)
- void SetEndPoint1 (int sx, int sy)
- float GetAngleInDegree ()
- float GetAngleInRadian ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.17.1 Detailed Description

mitkAngleWidgetModel2D - a 2D widget for measuring angles

mitkAngleWidgetModel2D is a 2D widget for measuring angles. It can respond the mouse events and return the current angle in units of degrees or radian. It is supposed to be attached to a 2D data model (e.g. mitkImageModel) and add to a 2D scene (e.g. mitkImageScene), and in other conditions the display could be improper.

8.17.2 Member Function Documentation

8.17.2.1 virtual void mitkAngleWidgetModel2D::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.17.2.2 virtual void mitkAngleWidgetModel2D::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.17.2.3 virtual void mitkAngleWidgetModel2D::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.17.2.4 float mitkAngleWidgetModel2D::GetAngleInDegree ()

Get the value of angle in degree.

Returns:

Return value of this angle in degree.

8.17.2.5 float mitkAngleWidgetModel2D::GetAngleInRadian ()

Get the value of angle in radian.

Returns:

Return value of this angle in radian.

8.17.2.6 virtual void mitkAngleWidgetModel2D::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:

names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.17.2.7 virtual void mitkAngleWidgetModel2D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel2D.

8.17.2.8 virtual void mitkAngleWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.17.2.9 virtual int mitkAngleWidgetModel2D::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.17.2.10 void mitkAngleWidgetModel2D::SetEndPoint0 (int *sx*, int *sy*) [inline]

Set ont end point of the angle.

Parameters:
sx x-coordinate of this point on screen

sy y-coordinate of this point on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.17.2.11 void mitkAngleWidgetModel2D::SetEndPoint0 (float *x*, float *y*) [inline]

Set ont end point of the angle in object coordinate.

Parameters:
x x-coordinate of this point in the object space

y y-coordinate of this point in the object space

Note:
Because this widget is for 2D image, the z-coordinate will always be zero.

8.17.2.12 void mitkAngleWidgetModel2D::SetEndPoint0 (float *p*[2]) [inline]

Set ont end point of the angle in object coordinate.

Parameters:
p[0] x-coordinate of this point in the object space

p[1] y-coordinate of this point in the object space

Note:
Because this widget is for 2D image, the z-coordinate will always be zero.

8.17.2.13 void mitkAngleWidgetModel2D::SetEndPoint1 (int *sx*, int *sy*) [inline]

Set the other end point of the angle.

Parameters:
sx x-coordinate of this point on screen

sy y-coordinate of this point on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.17.2.14 void mitkAngleWidgetModel2D::SetEndPoint1 (float *x*, float *y*) [inline]

Set the other end point of the angle in object coordinate.

Parameters:
x x-coordinate of this point in the object space

y y-coordinate of this point in the object space

Note:
Because this widget is for 2D image, the z-coordinate will always be zero.

8.17.2.15 void mitkAngleWidgetModel2D::SetEndPoint1 (float *p*[2]) [inline]

Set the other end point of the angle in object coordinate.

Parameters:

p[0] x-coordinate of this point in the object space

p[1] y-coordinate of this point in the object space

Note:

Because this widget is for 2D image, the z-coordinate will always be zero.

8.17.2.16 void mitkAngleWidgetModel2D::SetStartPoint (int *sx*, int *sy*) [inline]

Set the start point of the angle.

Parameters:

sx x-coordinate of this point on screen

sy y-coordinate of this point on screen

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.17.2.17 void mitkAngleWidgetModel2D::SetStartPoint (float *x*, float *y*) [inline]

Set the start point of the angle in object coordinate.

Parameters:

x x-coordinate of this point in the object space

y y-coordinate of this point in the object space

Note:

Because this widget is for 2D image, the z-coordinate will always be zero.

8.17.2.18 void mitkAngleWidgetModel2D::SetStartPoint (float *p*[2]) [inline]

Set the start point of the angle in object coordinate.

Parameters:

p[0] x-coordinate of this point in the object space

p[1] y-coordinate of this point in the object space

Note:

Because this widget is for 2D image, the z-coordinate will always be zero.

The documentation for this class was generated from the following file:

- mitkAngleWidgetModel2D.h

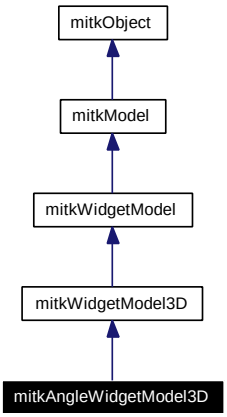
8.18 mitkAngleWidgetModel3D Class Reference

mitkAngleWidgetModel3D - a 3D widget for measuring angles

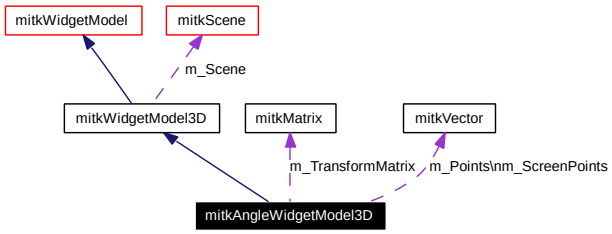
`#include <mitkAngleWidgetModel3D.h>`

Inherits mitkWidgetModel3D.

Inheritance diagram for mitkAngleWidgetModel3D:



Collaboration diagram for mitkAngleWidgetModel3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkAngleWidgetModel3D (float startPoint[3], float endPoint0[3], float endPoint1[3])
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- void SetUnits (float ux, float uy, float uz)
- void SetUnits (float units[3])
- float GetAngleInDegree ()
- float GetAngleInRadian ()
- virtual void Update ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)

- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.18.1 Detailed Description

mitkAngleWidgetModel3D - a 3D widget for measuring angles

mitkAngleWidgetModel3D is a 3D widget for measuring angles. It can respond the mouse events and return the current angle in units of degrees or radian. It is supposed to be attached to a 3D data model (e.g. mitkVolumeModel, mitkSurfaceModel) and add to a 3D scene (e.g. mitkScene), and in other conditions the display could be improper.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 mitkAngleWidgetModel3D::mitkAngleWidgetModel3D (float *startPoint*[3], float *endPoint0*[3], float *endPoint1*[3])

One and only constructor of mitkAngleWidgetModel3D.

Parameters:

startPoint[0] x-coordinate of start point

startPoint[1] y-coordinate of start point

startPoint[2] z-coordinate of start point

endPoint0[0] x-coordinate of one end point

endPoint0[1] y-coordinate of one end point

endPoint0[2] z-coordinate of one end point

endPoint1[0] x-coordinate of the other end point

endPoint1[1] y-coordinate of the other end point

endPoint1[2] z-coordinate of the other end point

8.18.3 Member Function Documentation

8.18.3.1 virtual void mitkAngleWidgetModel3D::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.18.3.2 virtual void mitkAngleWidgetModel3D::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.18.3.3 virtual void mitkAngleWidgetModel3D::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.18.3.4 float mitkAngleWidgetModel3D::GetAngleInDegree ()

Get the value of angle in degree.

Returns:

Return value of this angle in degree.

8.18.3.5 float mitkAngleWidgetModel3D::GetAngleInRadian ()

Get the value of angle in radian.

Returns:

Return value of this angle in radian.

8.18.3.6 virtual void mitkAngleWidgetModel3D::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:

names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.18.3.7 virtual void mitkAngleWidgetModel3D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkWidgetModel3D.

8.18.3.8 virtual void mitkAngleWidgetModel3D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.18.3.9 virtual int mitkAngleWidgetModel3D::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:

scene the pointer of an mitkScene in which this model will be shown

Returns:

Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.18.3.10 void mitkAngleWidgetModel3D::SetUnits (float *units*[3]) [inline]

Set unit length on axes.

Parameters:

units[0] unit length in x-axis

units[1] unit length in y-axis

units[2] unit length in z-axis

8.18.3.11 void mitkAngleWidgetModel3D::SetUnits (float *ux*, float *uy*, float *uz*) [inline]

Set unit length on axes.

Parameters:

ux unit length in x-axis

uy unit length in y-axis

uz unit length in z-axis

8.18.3.12 virtual void mitkAngleWidgetModel3D::Update () [virtual]

Update the parameters of the widget.

Reimplemented from mitkWidgetModel3D.

The documentation for this class was generated from the following file:

- mitkAngleWidgetModel3D.h

8.19 mitkBackProjection Class Reference

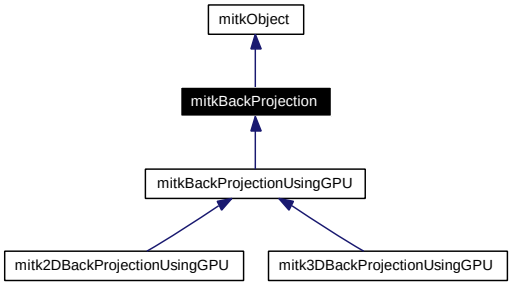
mitkBackProjection - abstract base class for CT back projection

`#include <mitkBackProjection.h>`

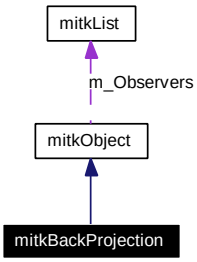
Inherits mitkObject.

Inherited by mitkBackProjectionUsingGPU.

Inheritance diagram for mitkBackProjection:



Collaboration diagram for mitkBackProjection:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInterpolationModel (int m)
- int GetInterpolationModel ()

8.19.1 Detailed Description

mitkBackProjection - abstract base class for CT back projection

mitkBackProjection is an abstract object that specifies behavior and interface of CT back projection.

8.19.2 Member Function Documentation

8.19.2.1 int mitkBackProjection::GetInterpolationModel () `[inline]`

Get the interpolation model

8.19.2.2 virtual void mitkBackProjection::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitk2DBackProjectionUsingGPU, mitk3DBackProjectionUsingGPU, and mitkBackProjectionUsingGPU.

8.19.2.3 void mitkBackProjection::SetInterpolationModel (int *m*) [inline]

Set the interpolation model

The documentation for this class was generated from the following file:

- mitkBackProjection.h

8.20 mitkBackProjectionUsingGPU Class Reference

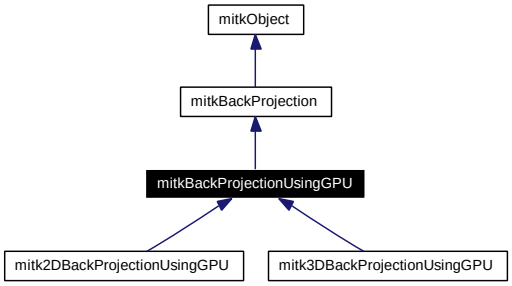
mitkBackProjectionUsingGPU - abstract base class for CT back projection

`#include <mitkBackProjectionUsingGPU.h>`

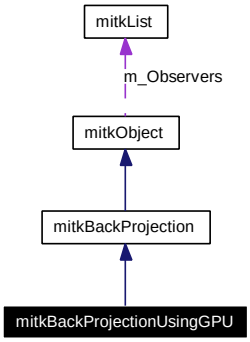
Inherits mitkBackProjection.

Inherited by mitk2DBackProjectionUsingGPU, and mitk3DBackProjectionUsingGPU.

Inheritance diagram for mitkBackProjectionUsingGPU:



Collaboration diagram for mitkBackProjectionUsingGPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.20.1 Detailed Description

mitkBackProjectionUsingGPU - abstract base class for CT back projection

mitkBackProjectionUsingGPU is an abstract object that specifies behavior and interface of CT back projection.

8.20.2 Member Function Documentation

8.20.2.1 virtual void mitkBackProjectionUsingGPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkBackProjection.

Reimplemented in mitk2DBackProjectionUsingGPU, and mitk3DBackProjectionUsingGPU.

The documentation for this class was generated from the following file:

- mitkBackProjectionUsingGPU.h

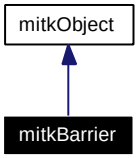
8.21 mitkBarrier Class Reference

mitkBarrier - A synchronizer for mitkSPMD

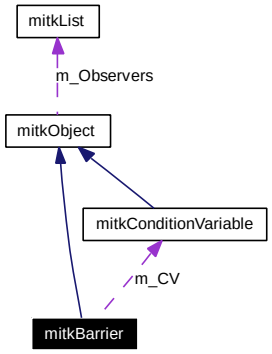
`#include <mitkSPMD.h>`

Inherits mitkObject.

Inheritance diagram for mitkBarrier:



Collaboration diagram for mitkBarrier:



Public Member Functions

- void Synchronize ()

Friends

- class mitkSPMD

8.21.1 Detailed Description

mitkBarrier - A synchronizer for mitkSPMD

mitkBarrier is a synchronizer for mitkSPMD, it can be created only by SPMD. Is you use SPMD by sub-classing mitkSPMD, please use the protected member function Synchronize() of mitkSPMD. This class is explicitly used only when you are using one of the delegates of SPMD (mitkSPMDStatic or mitkSPMDMember)

8.21.2 Member Function Documentation

8.21.2.1 void mitkBarrier::Synchronize ()

Called to synchronize threads. The calling thread waits until other threads created by the same SPMD reach the calling point.

The documentation for this class was generated from the following file:

- mitkSPMD.h

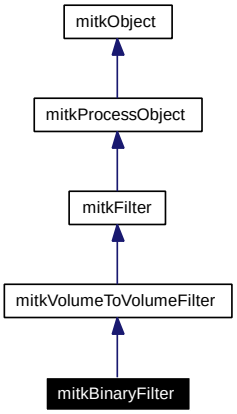
8.22 mitkBinaryFilter Class Reference

mitkBinaryFilter - a filter to get binary data from a source volume

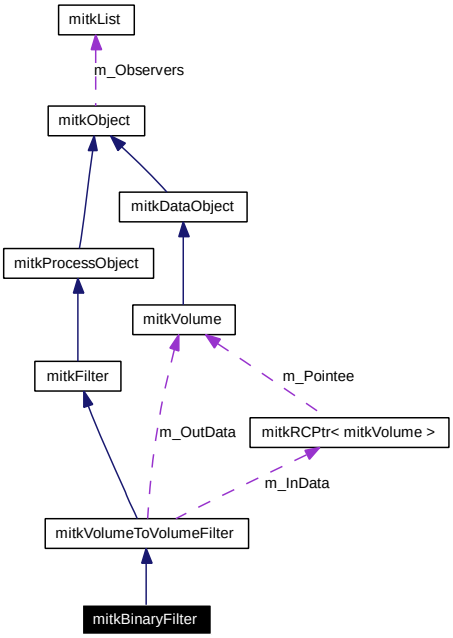
`#include <mitkBinaryFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkBinaryFilter:



Collaboration diagram for mitkBinaryFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkBinaryFilter ()
- float GetCubage ()

8.22.1 Detailed Description

mitkBinaryFilter - a filter to get binary data from a source volume

mitkBinaryFilter is a filter to get binary data of a source volume. The data type of the output volume is MITK_UNSIGNED_CHAR. All non-zero values of the source volume are set to 255. Zeros are not changed.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 mitkBinaryFilter::mitkBinaryFilter ()

Default constructor.

8.22.3 Member Function Documentation

8.22.3.1 float mitkBinaryFilter::GetCubage ()

Get the cubage of the output object (non zero parts).

8.22.3.2 virtual void mitkBinaryFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkBinaryFilter.h

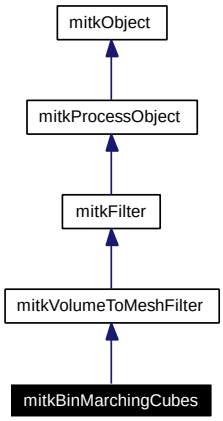
8.23 mitkBinMarchingCubes Class Reference

mitkBinMarchingCubes - a marching cubes algorithm using binary data

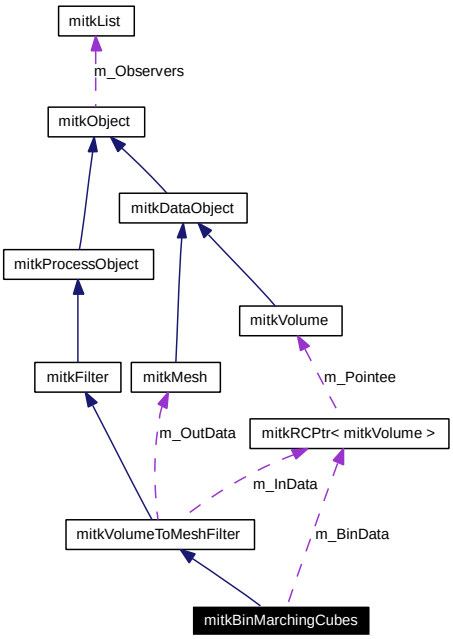
`#include <mitkBinMarchingCubes.h>`

Inherits mitkVolumeToMeshFilter.

Inheritance diagram for mitkBinMarchingCubes:



Collaboration diagram for mitkBinMarchingCubes:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkBinMarchingCubes ()

- void SetBinData (mitkVolume *binData)

8.23.1 Detailed Description

mitkBinMarchingCubes - a marching cubes algorithm using binary data

mitkBinMarchingCubes is is a process object that process Volume Data to generate 3D surface using Marching Cubes algorithm. This algorithm uses binary data volume to get vertices and source volume to get normals. The binary data volume is generated from segmentation result of the source volume via mitkBinaryFilter. Its data type is MITK_UNSIGNED_CHAR and background is set to 0 and foreground is set to 255. The midpoint of the cube side whose one end is 0 and the other is 255 will be taken as a vertex. The calculation of the normals is the same as mitkMarchingCubes.

Warning:
Marching Cubes algorithm may have the protect of United States patent, please use it at your own risk.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 mitkBinMarchingCubes::mitkBinMarchingCubes ()

Default constructor.

8.23.3 Member Function Documentation

8.23.3.1 virtual void mitkBinMarchingCubes::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToMeshFilter.

8.23.3.2 void mitkBinMarchingCubes::SetBinData (mitkVolume * *binData*) [inline]

Set binary data of the source volume.

Parameters:
binData pointer to an mitkVolume contains binary data.

Warning:
The data type of binData must be MITK_UNSIGNED_CHAR. And the value of the pixel of the object must be 255 and background must be 0.

Note:
Use mitkBinaryFilter to get an binary data volume of the source volume and set it to mitkBinMarchingCubes using this function. Then you can run mitkBinMarchingCubes. Note that binary data gotten from the source volume directly makes no sense. In order to get a proper result, you should use a kind of segmentation filter to get a segmented volume of the source volume and filter it with mitkBinaryFilter to get the binary data finally.

The documentation for this class was generated from the following file:

- mitkBinMarchingCubes.h

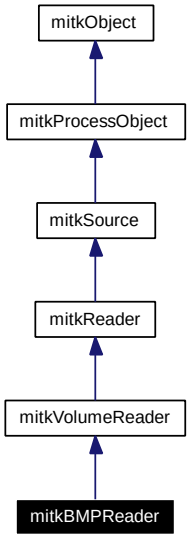
8.24 mitkBMPReader Class Reference

mitkBMPReader - a concrete reader for reading BMP image files

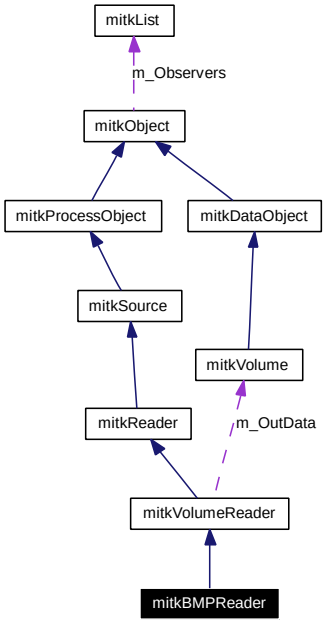
`#include <mitkBMPReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkBMPReader:



Collaboration diagram for mitkBMPReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetSpacingX (float px)
- void SetSpacingY (float py)
- void SetSpacingZ (float pz)
- void SetSpacings (float s[3])

8.24.1 Detailed Description

mitkBMPReader - a concrete reader for reading BMP image files

mitkBMPReader reads a set of BMP image files to a volume. Because BMP file doesn't have the spacing information, you must set them using the SetSpacingX, SetSpacingY, SetSpacingZ functions. To use this reader, the code snippet is:

```
mitkBMPReader *aReader = new mitkBMPReader;
aReader->SetSpacingX(sx);
aReader->SetSpacingY(sy);
aReader->SetSpacingZ(sz);
aReader->AddFileName(file1);
aReader->AddFileName(file2);
...  ...  ...
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

Warning:
All of the images must have equal width and height. Otherwise they can't form a volume. Now MITK only supports 8 bits and 24 bits BMP files.

8.24.2 Member Function Documentation

8.24.2.1 virtual void mitkBMPReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

8.24.2.2 void mitkBMPReader::SetSpacings (float *s*[3]) [inline]

Set spacing information in x, y, z axis respectively, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.
py the spacing (mm) in two adjacent voxels in y axis.
pz the spacing (mm) in two adjacent voxels in z axis.

8.24.2.3 void mitkBMPReader::SetSpacingX (float *px*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.

8.24.2.4 void mitkBMPReader::SetSpacingY (float *py*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:
py the spacing (mm) in two adjacent voxels in y axis.

8.24.2.5 void mitkBMPReader::SetSpacingZ (float *pz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:
pz the spacing (mm) in two adjacent voxels in z axis.

The documentation for this class was generated from the following file:

- mitkBMPReader.h

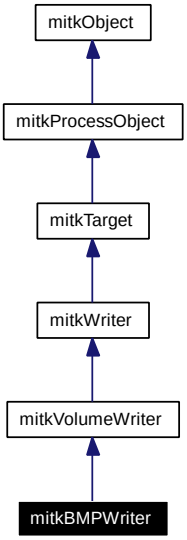
8.25 mitkBMPWriter Class Reference

mitkBMPWriter - a concrete writer for writing a volume to BMP image files

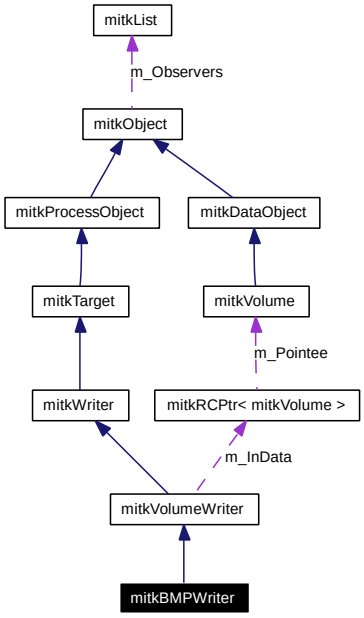
`#include <mitkBMPWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkBMPWriter:



Collaboration diagram for mitkBMPWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.25.1 Detailed Description

mitkBMPWriter - a concrete writer for writing a volume to BMP image files

mitkBMPWriter writes a volume to a set of BMP image files. Because the volume is a 3D dataset, it may contain many slices. So the file names must be generated and passed to writer properly. To use this writer, the code snippet is:

```
mitkBMPWriter *aWriter = new mitkBMPWriter;
aWriter->SetInput(aVolume);
int imageNum = aVolume->GetImageNum();
Gerenate file names into files[imageNum];
for(int i = 0; i < imageNum; i++)
    aWriter->AddFileName(files[i]);
aWriter->Run();
```

8.25.2 Member Function Documentation

8.25.2.1 virtual void mitkBMPWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

The documentation for this class was generated from the following file:

- mitkBMPWriter.h

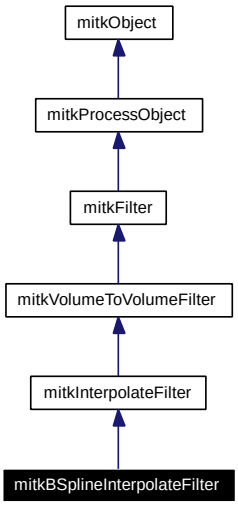
8.26 mitkBSplineInterpolateFilter Class Reference

mitkBSplineInterpolateFilter - a concrete interpolator

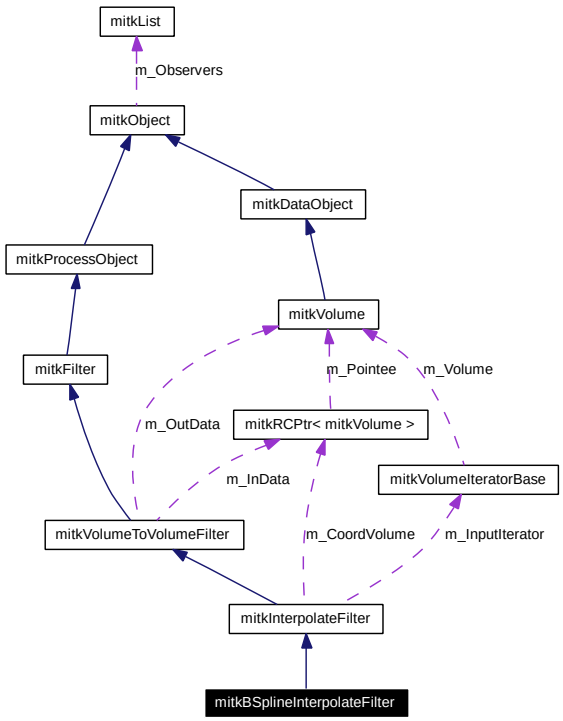
`#include <mitkBSplineInterpolateFilter.h>`

Inherits mitkInterpolateFilter.

Inheritance diagram for mitkBSplineInterpolateFilter:



Collaboration diagram for mitkBSplineInterpolateFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkBSplineInterpolateFilter (unsigned int splineDegree)
- mitkBSplineInterpolateFilter ()

8.26.1 Detailed Description

mitkBSplineInterpolateFilter - a concrete interpolator

mitkBSplineInterpolateFilter is a concrete class for implementation of B-Spline interpolation algorithem. A point's intensity can be estimated using a B-spline curve. Several B-spline basis funcitons were implemented from 2-degree to 9-degree. Cubic(3-degree) B-spline is the default interpolation kernel algorithm which requires 4x4 neighborhood in 2D, 4x4x4 neighborhood in 3D.

This filter can perform point interpolation by using InterpolatePoint() and perform volume interpolation by using Run().

User should specify the input volume and run Update() first before interpolation.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 mitkBSplineInterpolateFilter::mitkBSplineInterpolateFilter (unsigned int *splineDegree*)

Constructor with specific spline degree. param splineDegree Specify the degree for B-Spline kernel.

8.26.2.2 mitkBSplineInterpolateFilter::mitkBSplineInterpolateFilter ()

Constructor.

8.26.3 Member Function Documentation

8.26.3.1 virtual void mitkBSplineInterpolateFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkInterpolateFilter.

The documentation for this class was generated from the following file:

- mitkBSplineInterpolateFilter.h

8.27 mitkBSPTree< BSPTreeNode > Class Template Reference

mitkBSPTree -

`#include <mitkBSPTree.h>`

Collaboration diagram for mitkBSPTree< BSPTreeNode >:



8.27.1 Detailed Description

template<class BSPTreeNode> class mitkBSPTree< BSPTreeNode >

mitkBSPTree -

mitkBSPTree

The documentation for this class was generated from the following file:

- mitkBSPTree.h

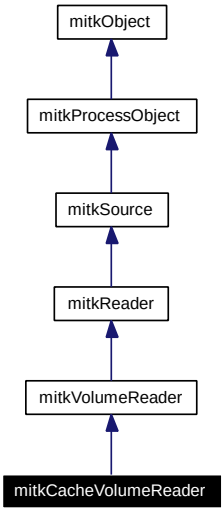
8.28 mitkCacheVolumeReader Class Reference

mitkCacheVolumeReader - a volume reader for reading out-of-core volume cache files

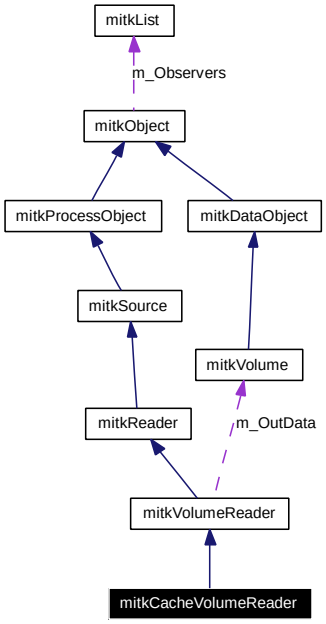
`#include <mitkCacheVolumeReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkCacheVolumeReader:



Collaboration diagram for mitkCacheVolumeReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetConfigFileToBinary (bool b=true)

8.28.1 Detailed Description

mitkCacheVolumeReader - a volume reader for reading out-of-core volume cache files

mitkCacheVolumeReader is a volume reader to read out-of-core volume cache files. It directly reuses the existed cache files of an out-of-core volume and makes the reading process fast. To use this reader, the code snippet is:

```
mitkCacheVolumeReader *aReader = new mitkCacheVolumeReader;
aReader->AddFileName(file); // a config file contains the volume parameters
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

Note:
Use mitkCacheVolumeWriter to generate the config file which contains the volume parameters and tell the mitkOoCVolume keep the cache files.

8.28.2 Member Function Documentation

8.28.2.1 virtual void mitkCacheVolumeReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

8.28.2.2 void mitkCacheVolumeReader::SetConfigFileToBinary (bool *b* = true) [inline]

Set if the config file is a binary file. Binary config file gives a quick and accurate reading.

Parameters:
b whether the config file is binary

The documentation for this class was generated from the following file:

- mitkCacheVolumeReader.h

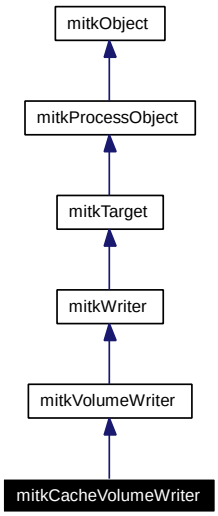
8.29 mitkCacheVolumeWriter Class Reference

mitkCacheVolumeWriter - a volume writer for reusing out-of-core volume cache files

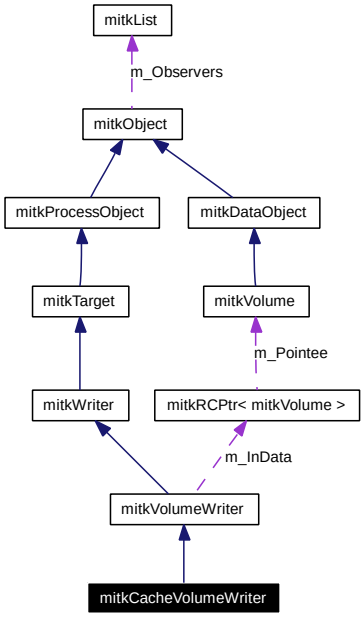
`#include <mitkCacheVolumeWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkCacheVolumeWriter:



Collaboration diagram for mitkCacheVolumeWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetConfigFileToBinary (bool b=true)

8.29.1 Detailed Description

mitkCacheVolumeWriter - a volume writer for reusing out-of-core volume cache files

mitkCacheVolumeWriter is a volume writer for reusing out-of-core volume cache files. It will generate a config file which contains the volume parameters and the out-of-core settings. It also tell the input volume to keep the cache files. This file can be read and resolved by mitkCacheVolumeReader. To use this writer, the code snippet is:

```
mitkCacheVolumeWriter *aWriter = new mitkCacheVolumeWriter;
aWriter->SetInput(aVolume);
aWriter->AddFileName(filename);
aWriter->SetConfigFileToBinary(isBin);
aWriter->Run();
```

8.29.2 Member Function Documentation

8.29.2.1 virtual void mitkCacheVolumeWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

8.29.2.2 void mitkCacheVolumeWriter::SetConfigFileToBinary (bool *b* = true) [inline]

Set the config file to binary file. Binary config file gives a quick and accurate reading.

Parameters:
b whether or not to set the config file to binary

The documentation for this class was generated from the following file:

- mitkCacheVolumeWriter.h

8.30 mitkCamera Class Reference

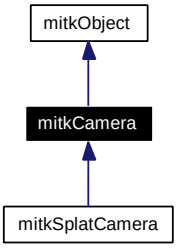
mitkCamera - a camera in 3D scene

`#include <mitkCamera.h>`

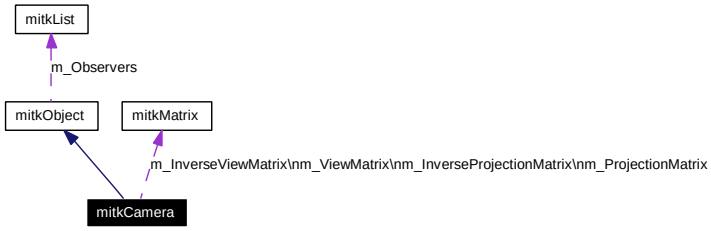
Inherits mitkObject.

Inherited by mitkSplatCamera.

Inheritance diagram for mitkCamera:



Collaboration diagram for mitkCamera:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void GetPosition (float &x, float &y, float &z)
- void GetPosition (float a[3])
- void GetFocalPoint (float &x, float &y, float &z)
- void GetFocalPoint (float a[3])
- float GetThickness ()
- bool IsParallelProjection ()
- void SetProjectionToParallel ()
- void SetProjectionToPerspective ()
- bool IsModified ()
- virtual void LookAt (float eyex, float eyey, float eyez, float centerx, float centery, float centerz, float upx, float upy, float upz)
- virtual void Ortho (float left, float right, float bottom, float top, float zNear, float zFar)
- virtual void Frustum (float left, float right, float bottom, float top, float zNear, float zFar)
- virtual void Perspective (float fovy, float aspect, float zNear, float zFar)
- void GetViewMatrix (mitkMatrix *m)
- void GetViewMatrix (float m[16])
- const mitkMatrix * GetViewMatrix ()

- void GetProjectionMatrix (mitkMatrix *m)
- void GetProjectionMatrix (float m[16])
- const mitkMatrix * GetProjectionMatrix ()
- void GetInverseOfProjectionMatrix (mitkMatrix *m)
- void GetInverseOfProjectionMatrix (float m[16])
- const mitkMatrix * GetInverseOfProjectionMatrix ()
- void GetInverseOfViewMatrix (mitkMatrix *m)
- void GetInverseOfViewMatrix (float m[16])
- const mitkMatrix * GetInverseOfViewMatrix ()
- void WorldToCamera (const float worldPoint[4], float cameraPoint[4])
- void WorldToView (const float worldPoint[4], float viewPoint[4])
- void CameraToView (const float cameraPoint[4], float viewPoint[4])
- void CameraToWorld (const float cameraPoint[4], float worldPoint[4])
- void ViewToWorld (const float viewPoint[4], float worldPoint[4])
- void ViewToCamera (const float viewPoint[4], float cameraPoint[4])

8.30.1 Detailed Description

mitkCamera - a camera in 3D scene

mitkCamera is a camera in 3D scene. The only way you can access it is to get a pointer to it by using the member function GetCamera() of mitkScene. Then you can control it through its interface, and the change will affect the image rendered in the scene.

8.30.2 Member Function Documentation

8.30.2.1 void mitkCamera::CameraToView (const float *cameraPoint*[4], float *viewPoint*[4]) [inline]

Transform a point from camera coordinate to view(NDC) coordinate.

8.30.2.2 void mitkCamera::CameraToWorld (const float *cameraPoint*[4], float *worldPoint*[4]) [inline]

Transform a point from camera coordinate to world coordinate.

8.30.2.3 virtual void mitkCamera::Frustum (float *left*, float *right*, float *bottom*, float *top*, float *zNear*, float *zFar*) [virtual]

Perspective Projection

Reimplemented in mitkSplatCamera.

8.30.2.4 void mitkCamera::GetFocalPoint (float *a*[3]) [inline]

Get the focal point of the camera in world coordinates.

Parameters:

a[0] Return the x coordinate of the focal point in world coordinates

a[1] Return the y coordinate of the focal point in world coordinates

a[2] Return the z coordinate of the focal point in world coordinates

8.30.2.5 void mitkCamera::GetFocalPoint (float & *x*, float & *y*, float & *z*) [inline]

Get the focal point of the camera in world coordinates.

Parameters:

x Return the x coordinate of the focal point in world coordinates

y Return the y coordinate of the focal point in world coordinates

z Return the z coordinate of the focal point in world coordinates

8.30.2.6 const mitkMatrix* mitkCamera::GetInverseOfProjectionMatrix () [inline]

Get the inverse of projection transform matrix.

8.30.2.7 void mitkCamera::GetInverseOfProjectionMatrix (float *m*[16])

Get the inverse of projection transform matrix.

8.30.2.8 void mitkCamera::GetInverseOfProjectionMatrix (mitkMatrix * *m*)

Get the inverse of projection transform matrix.

8.30.2.9 const mitkMatrix* mitkCamera::GetInverseOfViewMatrix () [inline]

Get the inverse of viewing transform matrix.

8.30.2.10 void mitkCamera::GetInverseOfViewMatrix (float *m*[16])

Get the inverse of viewing transform matrix.

8.30.2.11 void mitkCamera::GetInverseOfViewMatrix (mitkMatrix * *m*)

Get the inverse of viewing transform matrix.

8.30.2.12 void mitkCamera::GetPosition (float *a*[3]) [inline]

Get the position of the camera in world coordinates.

Parameters:

a[0] Return the x position of the camera in world coordinates

a[1] Return the y position of the camera in world coordinates

a[2] Return the z position of the camera in world coordinates

8.30.2.13 void mitkCamera::GetPosition (float & *x*, float & *y*, float & *z*) [inline]

Get the position of the camera in world coordinates.

Parameters:

- ***x*** Return the x position of the camera in world coordinates
- ***y*** Return the y position of the camera in world coordinates
- ***z*** Return the z position of the camera in world coordinates

8.30.2.14 const mitkMatrix* mitkCamera::GetProjectionMatrix () [inline]

Get the projection transform matrix.

8.30.2.15 void mitkCamera::GetProjectionMatrix (float *m*[16])

Get the projection transform matrix.

8.30.2.16 void mitkCamera::GetProjectionMatrix (mitkMatrix * *m*)

Get the projection transform matrix.

8.30.2.17 float mitkCamera::GetThickness () [inline]

Get the camera thickness.

Returns:

Return the camera thickness

8.30.2.18 const mitkMatrix* mitkCamera::GetViewMatrix () [inline]

Get the viewing transform matrix.

8.30.2.19 void mitkCamera::GetViewMatrix (float *m*[16])

Get the viewing transform matrix.

8.30.2.20 void mitkCamera::GetViewMatrix (mitkMatrix * *m*)

Get the viewing transform matrix.

8.30.2.21 bool mitkCamera::IsModified () [inline]

Test whether this camera is modified.

Returns:

Return true if this camera is modified.

8.30.2.22 bool mitkCamera::IsParallelProjection () [inline]

Get the projection mode.

Returns:
Return non-zero value, the projection mode is parallel projection, otherwise perspective projection

8.30.2.23 virtual void mitkCamera::LookAt (float *eyex*, float *eyey*, float *eyez*, float *centerx*, float *centery*, float *centerz*, float *upx*, float *upy*, float *upz*) [virtual]

Set the look at point. It is equal to the glLookAt() function in OpenGL API.

8.30.2.24 virtual void mitkCamera::Ortho (float *left*, float *right*, float *bottom*, float *top*, float *zNear*, float *zFar*) [virtual]

Parallel Projection

8.30.2.25 virtual void mitkCamera::Perspective (float *fovy*, float *aspect*, float *zNear*, float *zFar*) [virtual]

Perspective Projection

Reimplemented in mitkSplatCamera.

8.30.2.26 virtual void mitkCamera::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkSplatCamera.

8.30.2.27 void mitkCamera::SetProjectionToParallel () [inline]

Set the projection mode to parallel.

8.30.2.28 void mitkCamera::SetProjectionToPerspective () [inline]

Set the projection mode to perspective.

8.30.2.29 void mitkCamera::ViewToCamera (const float *viewPoint*[4], float *cameraPoint*[4]) [inline]

Transform a point from view(NDC) coordinate to camera coordinate.

8.30.2.30 void mitkCamera::ViewToWorld (const float *viewPoint*[4], float *worldPoint*[4]) [inline]

Transform a point from view(NDC) coordinate to world coordinate.

8.30.2.31 void mitkCamera::WorldToCamera (const float *worldPoint*[4], float *cameraPoint*[4]) [inline]

Transform a point from world coordinate to camera coordinate.

8.30.2.32 void mitkCamera::WorldToView (const float *worldPoint*[4], float *viewPoint*[4]) [inline]

Transform a point from world coordinate to view(NDC) coordinate.

The documentation for this class was generated from the following file:

- mitkCamera.h

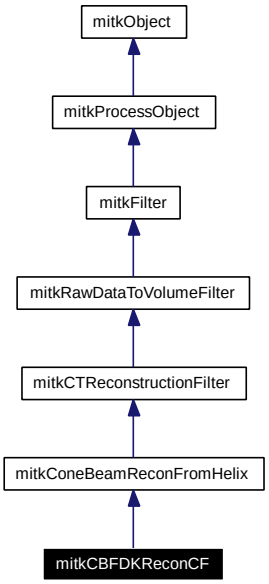
8.31 mitkCBFDKReconCF Class Reference

mitkCBFDKReconCF - concret class implement cone-beam reconstruction using FDK Algorithm.

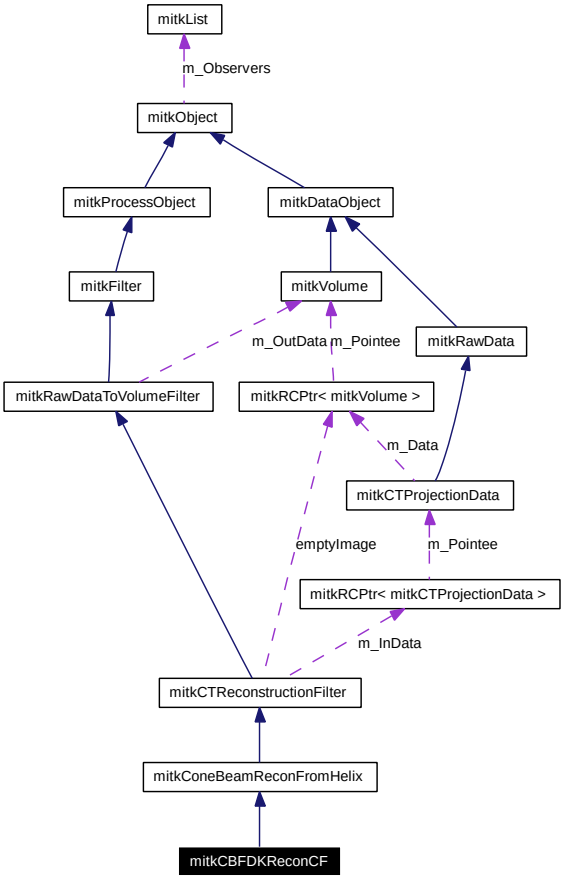
`#include <mitkCBFDKReconCF.h>`

Inherits mitkConeBeamReconFromHelix.

Inheritance diagram for mitkCBFDKReconCF:



Collaboration diagram for mitkCBFDKReconCF:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.31.1 Detailed Description

mitkCBFDKReconCF - concret class implement cone-beam reconstruction using FDK Algorithm.

mitkCBFDKReconCF - concret class implement cone-beam reconstruction FDK, this implemention just using C++ without any acceleration

8.31.2 Member Function Documentation

8.31.2.1 virtual void mitkCBFDKReconCF::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkConeBeamReconFromHelix.

The documentation for this class was generated from the following file:

- mitkCBFDKReconCF.h

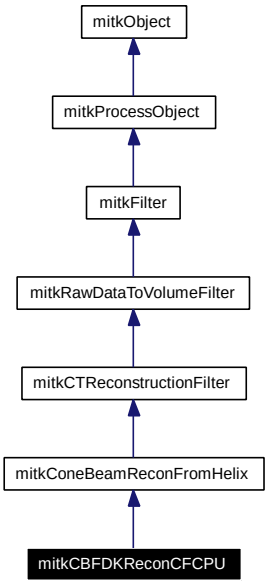
8.32 mitkCBFDKReconCFCPU Class Reference

mitkCBFDKReconCFCPU - concret class implement cone-beam reconstruction using FDK Algorithm.

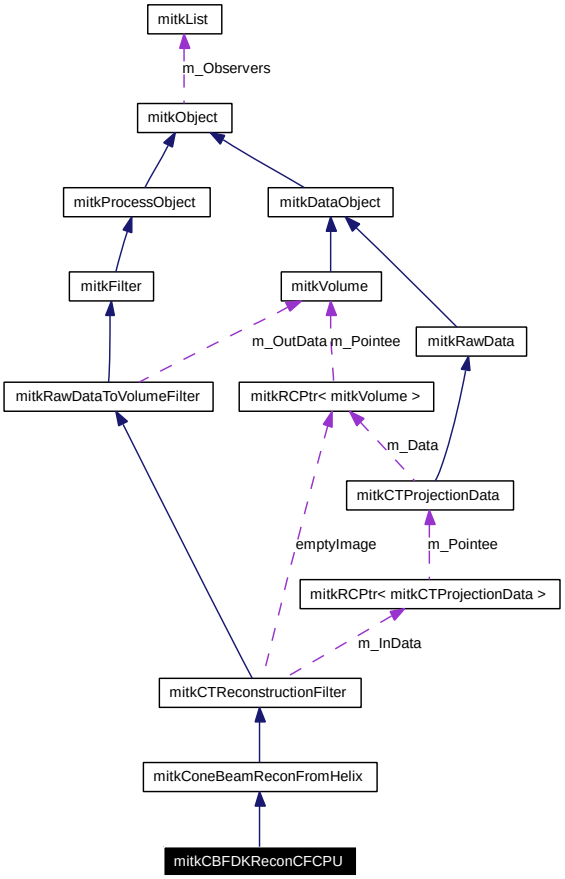
`#include <mitkCBFDKReconCFCPU.h>`

Inherits mitkConeBeamReconFromHelix.

Inheritance diagram for mitkCBFDKReconCFCPU:



Collaboration diagram for mitkCBFDKReconCFCPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.32.1 Detailed Description

mitkCBFDKReconCFCPU - concret class implement cone-beam reconstruction using FDK Algorithm.

mitkCBFDKReconCFCPU - concret class implement cone-beam reconstruction FDK, This implementation using SIMD for accelerating

8.32.2 Member Function Documentation

8.32.2.1 virtual void mitkCBFDKReconCFCPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkConeBeamReconFromHelix.

The documentation for this class was generated from the following file:

- mitkCBFDKReconCFCPU.h

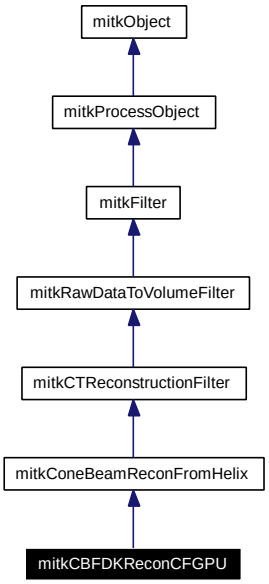
8.33 mitkCBFDKReconCFGPU Class Reference

mitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction using FDK Algorithm.

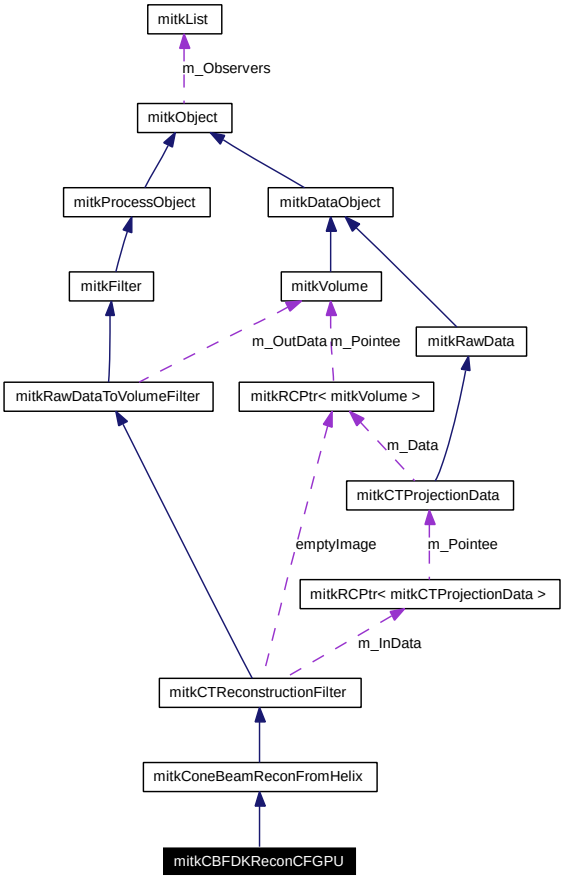
`#include <mitkCBFDKReconCFGPU.h>`

Inherits mitkConeBeamReconFromHelix.

Inheritance diagram for mitkCBFDKReconCFGPU:



Collaboration diagram for mitkCBFDKReconCFGPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.33.1 Detailed Description

mitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction using FDK Algorithm.

mitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction FDK,This implementation

8.33.2 Member Function Documentation

8.33.2.1 virtual void mitkCBFDKReconCFGPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkConeBeamReconFromHelix.

The documentation for this class was generated from the following file:

- mitkCBFDKReconCFGPU.h

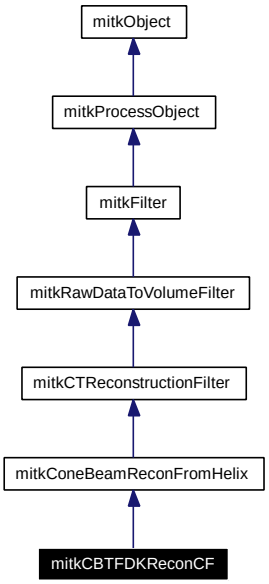
8.34 mitkCBTFDKReconCF Class Reference

mitkCBTFDKReconCF - concret class implement cone-beam reconstruction using TFDK Algorithm.

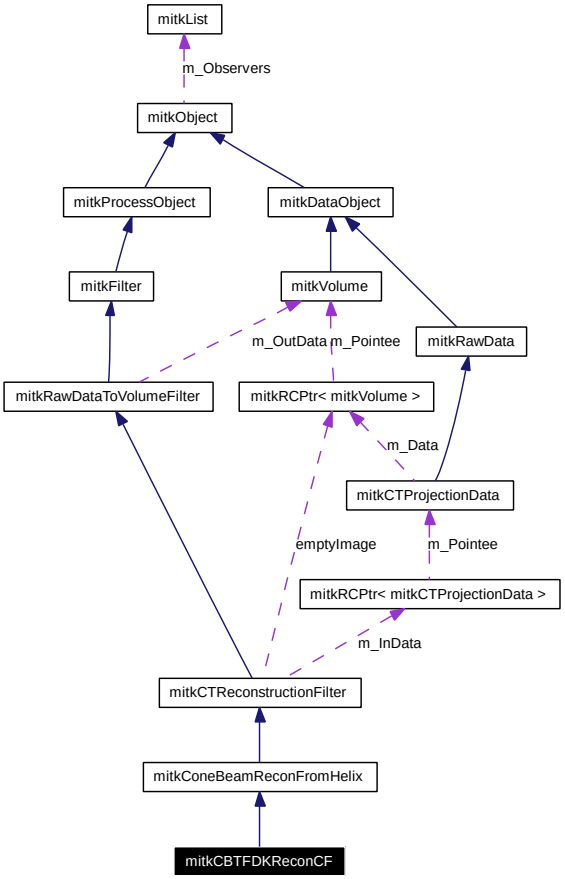
`#include <mitkCBTFDKReconCF.h>`

Inherits mitkConeBeamReconFromHelix.

Inheritance diagram for mitkCBTFDKReconCF:



Collaboration diagram for mitkCBTFDKReconCF:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.34.1 Detailed Description

mitkCBTFDKReconCF - concret class implement cone-beam reconstruction using TFDK Algorithm.

mitkCBTFDKReconCF - concret class implement cone-beam reconstruction TFDK

8.34.2 Member Function Documentation

8.34.2.1 virtual void mitkCBTFDKReconCF::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkConeBeamReconFromHelix.

The documentation for this class was generated from the following file:

- mitkCBTFDKReconCF.h

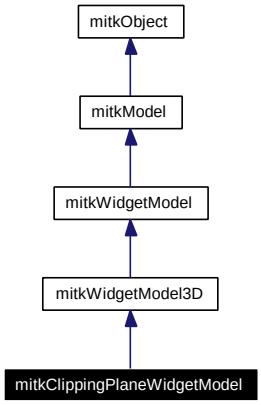
8.35 mitkClippingPlaneWidgetModel Class Reference

mitkClippingPlaneWidgetModel - a 3D widget for clipping plane

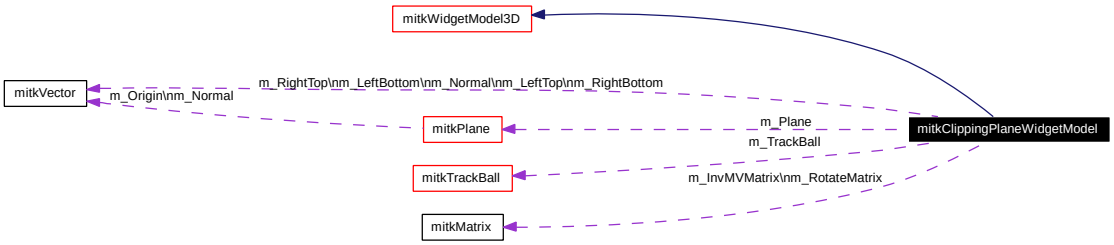
`#include <mitkClippingPlaneWidgetModel.h>`

Inherits mitkWidgetModel3D.

Inheritance diagram for mitkClippingPlaneWidgetModel:



Collaboration diagram for mitkClippingPlaneWidgetModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkClippingPlaneWidgetModel (float ox, float oy, float oz, float nx=0.0f, float ny=0.0f, float nz=1.0f, float width=100.0f, float height=100.0f)
- mitkClippingPlaneWidgetModel (coord_type v0x, coord_type v0y, coord_type v0z, coord_type v1x, coord_type v1y, coord_type v1z, coord_type cx, coord_type cy, coord_type cz)
- void SetPlanePosition (coord_type v0x, coord_type v0y, coord_type v0z, coord_type v1x, coord_type v1y, coord_type v1z, coord_type cx, coord_type cy, coord_type cz)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- virtual void SetSourceModel (mitkDataModel *model)
- void SetOpacity (float opacity)
- virtual void Update ()
- void RotateRadAroundXAxisOfPlane (float angle)

- void RotateRadAroundYAxisOfPlane (float angle)
- void RotateRadAroundZAxisOfPlane (float angle)
- void RotateDegAroundXAxisOfPlane (float angle)
- void RotateDegAroundYAxisOfPlane (float angle)
- void RotateDegAroundZAxisOfPlane (float angle)
- void TranslatePlane (float tx, float ty, float tz)
- void SetPlaneCenter (float ox, float oy, float oz)

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.35.1 Detailed Description

mitkClippingPlaneWidgetModel - a 3D widget for clipping plane

mitkClippingPlaneWidgetModel is a 3D widget for clipping plane. It provides an appearance and an interaction way for clipping plane. It is supposed to be attached to a 3D data model (e.g. mitkVolumeModel, mitkSurfaceModel), and in other conditions the display could be improper.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 mitkClippingPlaneWidgetModel::mitkClippingPlaneWidgetModel (float *ox*, float *oy*, float *oz*, float *nx* = `0.0f`, float *ny* = `0.0f`, float *nz* = `1.0f`, float *width* = `100.0f`, float *height* = `100.0f`)

A constructor of mitkClippingPlaneWidgetModel. The default normal of this plane is (0.0, 0.0, 1.0) (XoY plane) and the default size of this plane is 100.0∗100.0.

Parameters:

ox x-coordinate of the clipping plane's center point

oy y-coordinate of the clipping plane's center point

oz z-coordinate of the clipping plane's center point

nx x-coordinate of the clipping plane's normal, the default value is 0.0

ny y-coordinate of the clipping plane's normal, the default value is 0.0

nz z-coordinate of the clipping plane's normal, the default value is 1.0

width width of the plane shown in the scene, the default value is 100.0

height height of the plane shown in the scene, the default value is 100.0

8.35.2.2 mitkClippingPlaneWidgetModel::mitkClippingPlaneWidgetModel (coord_type *v0x*, coord_type *v0y*, coord_type *v0z*, coord_type *v1x*, coord_type *v1y*, coord_type *v1z*, coord_type *cx*, coord_type *cy*, coord_type *cz*)

A constructor. The parameters specify a rectangle in the model space via its left-bottom point, right-bottom point and center point.

Parameters:

v0x the x-coordinate of the left-bottom point

v0y the y-coordinate of the left-bottom point

v0z the z-coordinate of the left-bottom point

v1x the x-coordinate of the right-bottom point

v1y the y-coordinate of the right-bottom point

v1z the z-coordinate of the right-bottom point

cx the x-coordinate of the center point

cy the y-coordinate of the center point

cz the z-coordinate of the center point

8.35.3 Member Function Documentation

8.35.3.1 virtual void mitkClippingPlaneWidgetModel::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.35.3.2 virtual void mitkClippingPlaneWidgetModel::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.35.3.3 virtual void mitkClippingPlaneWidgetModel::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.35.3.4 virtual void mitkClippingPlaneWidgetModel::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:

names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.35.3.5 virtual void mitkClippingPlaneWidgetModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkWidgetModel3D.

8.35.3.6 virtual void mitkClippingPlaneWidgetModel::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.35.3.7 virtual int mitkClippingPlaneWidgetModel::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:

scene the pointer of an mitkScene in which this model will be shown

Returns:

Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.35.3.8 void mitkClippingPlaneWidgetModel::RotateDegAroundXAxisOfPlane (float *angle*)

Rotate the plane around the x axis of the plane. This function is provided for convenience. It behaves essentially like RotateRadAroundXAxisOfPlane(), but the parameter angle is in degree.

Parameters:
angle rotation angle in degree

8.35.3.9 void mitkClippingPlaneWidgetModel::RotateDegAroundYAxisOfPlane (float *angle*)

Rotate the plane around the y axis of the plane. This function is provided for convenience. It behaves essentially like RotateRadAroundYAxisOfPlane(), but the parameter angle is in degree.

Parameters:
angle rotation angle in degree

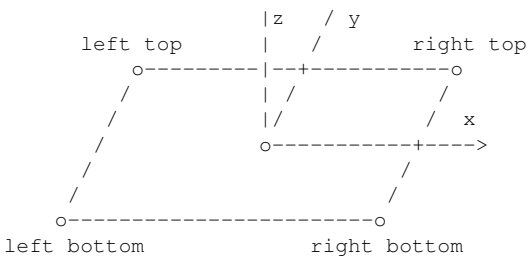
8.35.3.10 void mitkClippingPlaneWidgetModel::RotateDegAroundZAxisOfPlane (float *angle*)

Rotate the plane around the z axis of the plane. This function is provided for convenience. It behaves essentially like RotateRadAroundZAxisOfPlane(), but the parameter angle is in degree.

Parameters:
angle rotation angle in degree

8.35.3.11 void mitkClippingPlaneWidgetModel::RotateRadAroundXAxisOfPlane (float *angle*)

Rotate the plane around the x axis of the plane.

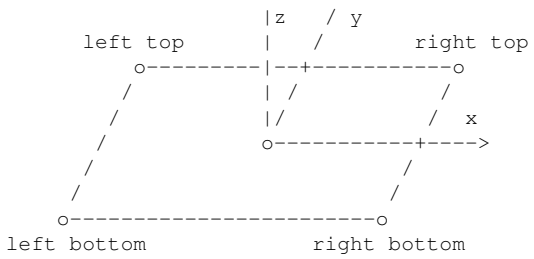


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

8.35.3.12 void mitkClippingPlaneWidgetModel::RotateRadAroundYAxisOfPlane (float *angle*)

Rotate the plane around the y axis of the plane.

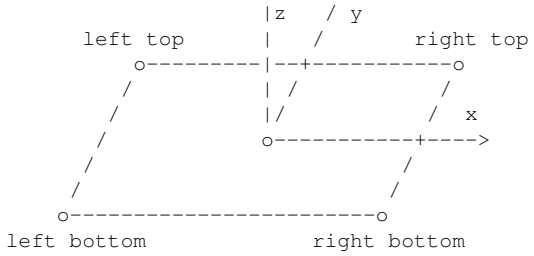


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

8.35.3.13 void mitkClippingPlaneWidgetModel::RotateRadAroundZAxisOfPlane (float *angle*)

Rotate the plane around the z axis of the plane.



Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

8.35.3.14 void mitkClippingPlaneWidgetModel::SetOpacity (float *opacity*)

Set the opacity of the plane.

Parameters:
opacity the opacity of the plane (the value is between 0.0f and 1.0f)

8.35.3.15 void mitkClippingPlaneWidgetModel::SetPlaneCenter (float *ox*, float *oy*, float *oz*)

Set the center of the plane.

Parameters:
ox the x coordinate of the center
oy the y coordinate of the center
oz the z coordinate of the center

8.35.3.16 void mitkClippingPlaneWidgetModel::SetPlanePosition (coord_type *v0x*, coord_type *v0y*, coord_type *v0z*, coord_type *v1x*, coord_type *v1y*, coord_type *v1z*, coord_type *cx*, coord_type *cy*, coord_type *cz*)

Set the position of this plane. The parameters specify a rectangle in the model space via its left-bottom point, right-bottom point and center point.

Parameters:

- ***v0x*** the x-coordinate of the left-bottom point
- ***v0y*** the y-coordinate of the left-bottom point
- ***v0z*** the z-coordinate of the left-bottom point
- ***v1x*** the x-coordinate of the right-bottom point
- ***v1y*** the y-coordinate of the right-bottom point
- ***v1z*** the z-coordinate of the right-bottom point
- ***cx*** the x-coordinate of the center point
- ***cy*** the y-coordinate of the center point
- ***cz*** the z-coordinate of the center point

8.35.3.17 virtual void mitkClippingPlaneWidgetModel::SetSourceModel (mitkDataModel * *model*) [virtual]

Associate this widget with a data model.

Parameters:

model pointer to an mitkDataModel with which this widget is associated

Note:

The parameter model can be NULL. It indicates that manipulation on this widget does not have an effect on other data models.

Reimplemented from mitkWidgetModel3D.

8.35.3.18 void mitkClippingPlaneWidgetModel::TranslatePlane (float *tx*, float *ty*, float *tz*)

Translate the plane with translation vector (tx, ty, tz).

Parameters:

- ***tx*** the x coordinate of the translation vector
- ***ty*** the y coordinate of the translation vector
- ***tz*** the z coordinate of the translation vector

8.35.3.19 virtual void mitkClippingPlaneWidgetModel::Update () [virtual]

Update the parameters of the widget.

Reimplemented from mitkWidgetModel3D.

The documentation for this class was generated from the following file:

- mitkClippingPlaneWidgetModel.h

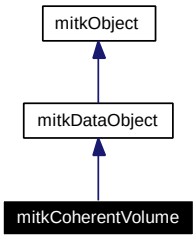
8.36 mitkCoherentVolume Class Reference

mitkCoherentVolume - a volume data class optimized for random access using blocks

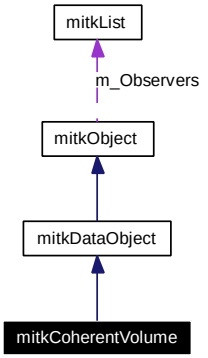
`#include <mitkCoherentVolume.h>`

Inherits mitkDataObject.

Inheritance diagram for mitkCoherentVolume:



Collaboration diagram for mitkCoherentVolume:



Public Member Functions

- virtual void Initialize ()
- virtual int GetDataObjectType () const
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject ∗src)
- virtual void DeepCopy (mitkDataObject ∗src)
- void SetBlockSize (const int bs[3])
- void GetBlcokSize (int bs[3])
- void SetDimensions (const int dims[3])
- void GetDimensions (int dims[3]) const
- void SetSpacings (const float s[3])
- void GetSpacings (float s[3]) const
- void GetBlockNum (int bn[3])
- int GetDataTypeSize () const
- void SetDataType (int data_type)
- int GetDataType () const
- void AllocateBlocks ()

- void SetBlockSlice (void *sliceData, int sliceID, int dimz=-1)
- void SetData (void *data)
- void SetICVolume (mitkICVolume *icv)
- float GetValue (int x, int y, int z)
- void GetMinMaxValue (double &minValue, double &maxValue)

8.36.1 Detailed Description

mitkCoherentVolume - a volume data class optimized for random access using blocks

mitkCoherentVolume encapsulates a blocked volume data structure Call SetDimensions, SetSpacings, SetDataType, before you call AllocateBlocks There are three methods to fill contents into this structure Call SetData if you have the sequentially stored data as a whole block Call SetBlockSlice if you want to set the data content using a block slice as the unit. A block slice is part of the whole volume with the thickness equal to the z-dimension of a block. Call SetICVolume if you have an allocated ICVolume and just want to have it converted Use GetValue to access the data value at a given position

8.36.2 Member Function Documentation

8.36.2.1 void mitkCoherentVolume::AllocateBlocks ()

Allocate space for the blocks

8.36.2.2 virtual void mitkCoherentVolume::DeepCopy (mitkDataObject * *src*) [inline, virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

8.36.2.3 virtual unsigned long long mitkCoherentVolume::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by the volume data. The unit is BYTE.

Returns:
Return the actual memory size occupied by this volume. The unit is BYTE.

Implements mitkDataObject.

8.36.2.4 void mitkCoherentVolume::GetBlcokSize (int *bs*[3]) [inline]

Get the size of the blocks used for storage.

Parameters:
bs[0] Return the block size in x direction
bs[1] Return the block size in y direction
bs[2] Return the block size in z direction

8.36.2.5 void mitkCoherentVolume::GetBlockNum (int *bn*[3]) [inline]

Get the number of blocks in x, y, z direction of this volume.

Parameters:

bn[0] Return the number of blocks in x axis.

bn[1] Return the number of blocks in y axis.

bn[2] Return the number of blocks in z axis.

8.36.2.6 virtual int mitkCoherentVolume::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:

Always return MITK_COHERENT_VOLUME

Reimplemented from mitkDataObject.

8.36.2.7 int mitkCoherentVolume::GetDataType () const [inline]

Get data type of this volume. MITK supports various data type.

Returns:

All of the return values and their meaning are shown as follows:
MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

8.36.2.8 int mitkCoherentVolume::GetDataTypeSize () const [inline]

Get the size of the data type in bytes.

Returns:

Return the size of the data type in bytes.

8.36.2.9 void mitkCoherentVolume::GetDimensions (int *dims*[3]) const [inline]

Get dimension in x, y, z direction of this volume.

Parameters:

dims[0] Return the dimension in x direction.

dims[1] Return the dimension in y direction.

dims[2] Return the dimension in z direction.

8.36.2.10 void mitkCoherentVolume::GetMinMaxValue (double & *minValue*, double & *maxValue*)

Get the minimum and maximum data value of all the volume data.

Parameters:

minValue return the minimum data value

maxValue return the maximum data value

8.36.2.11 void mitkCoherentVolume::GetSpacings (float *s*[3]) const [inline]

Get spacing information in x, y and z axis, the unit is mm.

Parameters:

s[0] Return the spacing (mm) in two adjacent voxels in x axis.

s[1] Return the spacing (mm) in two adjacent voxels in y axis.

s[2] Return the spacing (mm) in two adjacent voxels in z axis.

8.36.2.12 float mitkCoherentVolume::GetValue (int *x*, int *y*, int *z*)

Get value at a given posistion

Parameters:

x the x-coordinate of the voxel in the volume space (in [0, GetWidth()-1])

y the y-coordinate of the voxel in the volume space (in [0, GetHeight()-1])

z the y-coordinate of the voxel in the volume space (in [0, GetImageNum()-1])

Returns:

Return the value at the given posistion

8.36.2.13 virtual void mitkCoherentVolume::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Implements mitkDataObject.

8.36.2.14 void mitkCoherentVolume::SetBlockSize (const int *bs*[3]) [inline]

Set the size of the blocks used for storage. Not required to call, but call it before anything else.

Parameters:

bs[0] the block size in x direction

bs[1] the block size in y direction

bs[2] the block size in z direction

8.36.2.15 void mitkCoherentVolume::SetBlockSlice (void * *sliceData*, int *sliceID*, int *dimz* = -1)

Fill in a slice of the blocks in the data structure

Parameters:

sliceData Pointer to the sequentially stored data

sliceID Slice ID of the slice of blocks to be filled in. (0∼m_BlockNum[2]-1)

dimz The number of the slices of voxels in sliceData. (1∼m_BlockSize[2])

8.36.2.16 void mitkCoherentVolume::SetData (void * *data*)

Fill in the whole volume of the data structure

Parameters:

data Pointer to the sequentially stored data

8.36.2.17 void mitkCoherentVolume::SetDataType (int *data_type*)

Set data type of this volume. MITK supports various data type.

Parameters:

data_type Its valid value and meaning is shown as follows:

MITK_CHAR The data type is char

MITK_UNSIGNED_CHAR The data type is unsigned char

MITK_SHORT The data type is short

MITK_UNSIGNED_SHORT The data type is unsigned short

MITK_INT The data type is int

MITK_UNSIGNED_INT The data type is unsigned int

MITK_LONG The data type is long

MITK_UNSIGNED_LONG The data type is unsigned long

MITK_FLOAT The data type is float

MITK_DOUBLE The data type is double

8.36.2.18 void mitkCoherentVolume::SetDimensions (const int *dims*[3]) [inline]

Set dimension in x, y, z direction of this volume.

Parameters:

dims[0] the dimension in x direction

dims[1] the dimension in y direction

dims[2] the dimension in z direction

8.36.2.19 void mitkCoherentVolume::SetICVolume (mitkICVolume * *icv*)

Setup the coherent volume object using an allocated ICVolume object icv Pointer to the ICVolume object.

8.36.2.20 void mitkCoherentVolume::SetSpacings (const float *s*[3]) [inline]

Set spacing information in x, y and z axis, the unit is mm.

Parameters:

s[0] the spacing (mm) in two adjacent voxels in x axis.

s[1] the spacing (mm) in two adjacent voxels in y axis.

s[2] the spacing (mm) in two adjacent voxels in z axis.

8.36.2.21 virtual void mitkCoherentVolume::ShallowCopy (mitkDataObject * *src*) [inline, virtual]

Warning:

Internal function. Don't call it directly.

Implements mitkDataObject.

The documentation for this class was generated from the following file:

- mitkCoherentVolume.h

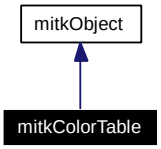
8.37 mitkColorTable Class Reference

mitkColorTable - a table mapping pixel value to color

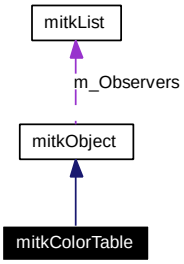
`#include <mitkColorTable.h>`

Inherits mitkObject.

Inheritance diagram for mitkColorTable:



Collaboration diagram for mitkColorTable:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkColorTable ()
- void SetDefaultColor (int r, int g, int b, int a=0)
- void AddColor (double pixVal, int r, int g, int b, int a=0)
- void RemoveColor (double pixVal)
- void RemoveColor (int index)
- void RemoveAllColors ()
- int GetColorCount ()
- bool SetColor (int index, int r, int g, int b, int a=0)
- bool SetValue (int index, double pixVal)
- bool SetValueColor (int index, double pixVal, int r, int g, int b, int a=0)
- bool GetColor (double pixVal, int &r, int &g, int &b)
- bool GetColor (double pixVal, int &r, int &g, int &b, int &a)
- bool GetColor (double pixVal, unsigned char &r, unsigned char &g, unsigned char &b)
- bool GetColor (double pixVal, unsigned char color[3])
- bool GetColor (int index, int &r, int &g, int &b)
- bool GetColor (int index, int &r, int &g, int &b, int &a)
- bool GetColor (int index, unsigned char &r, unsigned char &g, unsigned char &b)
- bool GetColor (int index, unsigned char color[3])
- bool GetValue (int index, double &pixVal)
- bool GetValueColor (int index, double &pixVal, int &r, int &g, int &b)

- bool GetValueColor (int index, double &pixVal, int &r, int &g, int &b, int &a)
- bool GetValueColor (int index, double &pixVal, unsigned char &r, unsigned char &g, unsigned char &b)
- bool GetValueColor (int index, double &pixVal, unsigned char color[3])
- void Copy (mitkColorTable *src)

8.37.1 Detailed Description

mitkColorTable - a table mapping pixel value to color

mitkColorTable is a table mapping pixel value to color. The data type of the pixel value is double. It records some double value regions. Each region is mapped to one RGBA color.

8.37.2 Constructor & Destructor Documentation

8.37.2.1 mitkColorTable::mitkColorTable ()

Default constructor.

8.37.3 Member Function Documentation

8.37.3.1 void mitkColorTable::AddColor (double *pixVal*, int *r*, int *g*, int *b*, int *a* = 0)

Add a mapping entity into the table. After this operation, the pixel values which >= pixVal and < the minimum pixel value which > pixVal in the former table will be mapper to color (r,g,b,a).

Parameters:

pixVal the new pixel value delimiter to be added to the table

r the red component of the color

g the green component of the color

b the blue component of the color

a the alpha component of the color, the default value is 0

8.37.3.2 void mitkColorTable::Copy (mitkColorTable * *src*)

Copy the entities from another color table.

Parameters:

src the source color table to copy from

8.37.3.3 bool mitkColorTable::GetColor (int *index*, unsigned char *color*[3])

Get the color of the entity indicated by index.

Parameters:

index the index of the entity

color[0] return the red component of the color

color[1] return the green component of the color

color[2] return the blue component of the color

Returns:
Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.4 bool mitkColorTable::GetColor (int *index*, unsigned char & *r*, unsigned char & *g*, unsigned char & *b*)

Get the color of the entity indicated by index.

Parameters:
index the index of the entity

r return the red component of the color

g return the green component of the color

b return the blue component of the color

Returns:
Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.5 bool mitkColorTable::GetColor (int *index*, int & *r*, int & *g*, int & *b*, int & *a*)

Get the color of the entity indicated by index.

Parameters:
index the index of the entity

r return the red component of the color

g return the green component of the color

b return the blue component of the color

a return the alpha component of the color

Returns:
Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.6 bool mitkColorTable::GetColor (int *index*, int & *r*, int & *g*, int & *b*)

Get the color of the entity indicated by index.

Parameters:
index the index of the entity

r return the red component of the color

g return the green component of the color

b return the blue component of the color

Returns:
Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.7 bool mitkColorTable::GetColor (double *pixVal*, unsigned char *color*[3])

Get the color pixVal is mapping to according to the table.

Parameters:

pixVal the pixel value

color[0] return the red component of the color

color[1] return the green component of the color

color[2] return the blue component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.8 bool mitkColorTable::GetColor (double *pixVal*, unsigned char & *r*, unsigned char & *g*, unsigned char & *b*)

Get the color pixVal is mapping to according to the table.

Parameters:

pixVal the pixel value

r return the red component of the color

g return the green component of the color

b return the blue component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.9 bool mitkColorTable::GetColor (double *pixVal*, int & *r*, int & *g*, int & *b*, int & *a*)

Get the color pixVal is mapping to according to the table.

Parameters:

pixVal the pixel value

r return the red component of the color

g return the green component of the color

b return the blue component of the color

a return the alpha component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.10 bool mitkColorTable::GetColor (double *pixVal*, int & *r*, int & *g*, int & *b*)

Get the color pixVal is mapping to according to the table.

Parameters:

pixVal the pixel value

r return the red component of the color

g return the green component of the color

b return the blue component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.11 int mitkColorTable::GetColorCount ()

Get the number of mapping entities in the table.

8.37.3.12 bool mitkColorTable::GetValue (int *index*, double & *pixVal*)

Get the pixel value delimiter of the entity indicated by index.

Parameters:

index the index of the entity

pixVal return the pixel value delimiter of the entity

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.13 bool mitkColorTable::GetValueColor (int *index*, double & *pixVal*, unsigned char *color*[3])

Get the pixel value delimiter and the color of the entity indicated by index.

Parameters:

index the index of the entity

pixVal return the pixel value delimiter of the entity

color[0] return the red component of the color

color[1] return the green component of the color

color[2] return the blue component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.14 bool mitkColorTable::GetValueColor (int *index*, double & *pixVal*, unsigned char & *r*, unsigned char & *g*, unsigned char & *b*)

Get the pixel value delimiter and the color of the entity indicated by index.

Parameters:

index the index of the entity

pixVal return the pixel value delimiter of the entity

r return the red component of the color

g return the green component of the color

b return the blue component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.15 bool mitkColorTable::GetValueColor (int *index*, double & *pixVal*, int & *r*, int & *g*, int & *b*, int & *a*)

Get the pixel value delimiter and the color of the entity indicated by index.

Parameters:

index the index of the entity

pixVal return the pixel value delimiter of the entity

r return the red component of the color

g return the green component of the color

b return the blue component of the color

a return the alpha component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.16 bool mitkColorTable::GetValueColor (int *index*, double & *pixVal*, int & *r*, int & *g*, int & *b*)

Get the pixel value delimiter and the color of the entity indicated by index.

Parameters:

index the index of the entity

pixVal return the pixel value delimiter of the entity

r return the red component of the color

g return the green component of the color

b return the blue component of the color

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.17 virtual void mitkColorTable::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

8.37.3.18 void mitkColorTable::RemoveAllColors ()

Remove all mapping entities.

8.37.3.19 void mitkColorTable::RemoveColor (int *index*)

Remove a mapping entity by index.

Parameters:
index the index of the entity to be removed from the table

8.37.3.20 void mitkColorTable::RemoveColor (double *pixVal*)

Remove a mapping entity decided by the pixel value delimiter pixVal from the table.

Parameters:
pixVal the pixel value delimiter to be removed from the table

8.37.3.21 bool mitkColorTable::SetColor (int *index*, int *r*, int *g*, int *b*, int *a* = 0)

Set the color of the mapping entity indicated by index to (r,g,b,a).

Parameters:
index the index of the entity
r the red component of the color to be set
g the green component of the color to be set
b the blue component of the color to be set
a the alpha component of the color to be set, the default value is 0

Returns:
Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.22 void mitkColorTable::SetDefaultColor (int *r*, int *g*, int *b*, int *a* = 0)

Set the default color.

Parameters:
r the red component of the color
g the green component of the color
b the blue component of the color
a the alpha component of the color, the default value is 0

8.37.3.23 bool mitkColorTable::SetValue (int *index*, double *pixVal*)

Set the pixel value delimiter of the mapping entity indicated by index to pixVal.

Parameters:

index the index of the entity

pixVal the pixel value delimiter to be set

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

8.37.3.24 bool mitkColorTable::SetValueColor (int *index*, double *pixVal*, int *r*, int *g*, int *b*, int *a* = 0)

Set the pixel value delimiter and the color of the mapping entity indicated by index to pixVal and (r,g,b,a).

Parameters:

index the index of the entity

pixVal the pixel value delimiter to be set

r the red component of the color to be set

g the green component of the color to be set

b the blue component of the color to be set

a the alpha component of the color to be set, the default value is 0

Returns:

Return true if the operation is accomplished successfully, otherwise return false.

The documentation for this class was generated from the following file:

- mitkColorTable.h

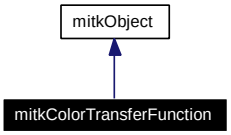
8.38 mitkColorTransferFunction Class Reference

mitkColorTransferFunction - a transfer function to map the scalar value to color

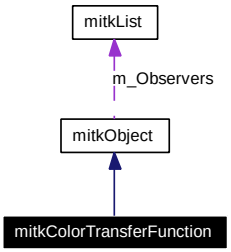
`#include <mitkColorTransferFunction.h>`

Inherits mitkObject.

Inheritance diagram for mitkColorTransferFunction:



Collaboration diagram for mitkColorTransferFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddPoint (int x, float rColor, float gColor, float bColor)
- void RemovePoint (int x)
- void RemoveAllPoints ()
- int GetPointsCount ()
- int GetDataValueAtPoint (int pointIndex)
- float GetRedColorAtPoint (int pointIndex)
- float GetGreenColorAtPoint (int pointIndex)
- float GetBlueColorAtPoint (int pointIndex)
- void SetMax (int xMax, float rMax, float gMax, float bMax)
- int GetMaxX ()
- float GetRedValue (int x)
- float GetGreenValue (int x)
- float GetBlueValue (int x)
- float * GetRData ()
- float * GetGData ()
- float * GetBData ()
- bool IsModified () const
- void SetUnmodified ()

8.38.1 Detailed Description

mitkColorTransferFunction - a transfer function to map the scalar value to color

mitkColorTransferFunction is a transfer function to map the scalar value to color. The scalar value has a range. The minimum value is always 0, the maximum value can be set by calling the function SetMax(). In general, you firstly get the maximum value from a volume, and then pass it to mitkColorTransferFunction.

8.38.2 Member Function Documentation

8.38.2.1 void mitkColorTransferFunction::AddPoint (int *x*, float *rColor*, float *gColor*, float *bColor*)

Add a point to the color transfer function

Parameters:

x Specify the scalar value

rColor Specify the red component of mapped color

gColor Specify the green component of mapped color

bColor Specify the blue component of mapped color

8.38.2.2 float* mitkColorTransferFunction::GetBData () `[inline]`

Get the address of the blue component of the colors in the transfer function.

Returns:

Return a pointer to the address of the blue component of the colors in the transfer function. The number of element in this array is

```
GetMaxX() + 1
```

.

8.38.2.3 float mitkColorTransferFunction::GetBlueColorAtPoint (int *pointIndex*)

Get the blue color component at pointIndex point

Parameters:

pointIndex Specify zero-based point index

Returns:

Return the blue color component at the specified point

8.38.2.4 float mitkColorTransferFunction::GetBlueValue (int *x*)

Get the blue component of the color corresponding to the specified scalar value.

Parameters:

x The specified scalar value.

Returns:

Return the blue component of the color corresponding to the scalar value x.

8.38.2.5 int mitkColorTransferFunction::GetDataValueAtPoint (int *pointIndex*)

Get the data value at pointIndex point

Parameters:
pointIndex Specify zero-based point index

Returns:
Return the data value at the specified point

8.38.2.6 float* mitkColorTransferFunction::GetGData () [inline]

Get the address of the green component of the colors in the transfer function.

Returns:
Return a pointer to the address of the green component of the colors in the transfer function. The number of element in this array is

```
GetMaxX() + 1
```

.

8.38.2.7 float mitkColorTransferFunction::GetGreenColorAtPoint (int *pointIndex*)

Get the green color component at pointIndex point

Parameters:
pointIndex Specify zero-based point index

Returns:
Return the green color component at the specified point

8.38.2.8 float mitkColorTransferFunction::GetGreenValue (int *x*)

Get the green component of the color corresponding to the specified scalar value.

Parameters:
x The specified scalar value.

Returns:
Return the green component of the color corresponding to the scalar value x.

8.38.2.9 int mitkColorTransferFunction::GetMaxX () [inline]

Get the maximum scalar value of this transfer function.

Returns:
Return the maximum scalar value of this transfer function.

8.38.2.10 int mitkColorTransferFunction::GetPointsCount ()

Get the number of points in this transfer function

Returns:
Return the points number

8.38.2.11 float* mitkColorTransferFunction::GetRData () [inline]

Get the address of the red component of the colors in the transfer function.

Returns:
Return a pointer to the address of the red component of the colors in the transfer function. The number of element in this array is

```
GetMaxX() + 1
```

.

8.38.2.12 float mitkColorTransferFunction::GetRedColorAtPoint (int *pointIndex*)

Get the red color component at pointIndex point

Parameters:
pointIndex Specify zero-based point index

Returns:
Return the red color component at the specified point

8.38.2.13 float mitkColorTransferFunction::GetRedValue (int *x*)

Get the red component of the color corresponding to the specified scalar value.

Parameters:
x The specified scalar value.

Returns:
Return the red component of the color corresponding to the scalar value x.

8.38.2.14 bool mitkColorTransferFunction::IsModified () const [inline]

Test if some of the transfer function is modified.

Returns:
Return true if some of the properties are modified. Otherwise return false.

8.38.2.15 virtual void mitkColorTransferFunction::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

8.38.2.16 void mitkColorTransferFunction::RemoveAllPoints ()

Remove all points from the color transfer function

8.38.2.17 void mitkColorTransferFunction::RemovePoint (int *x*)

Remove a point from the color transfer function

Parameters:

x Specify the scalar value. If the color transfer function has a point which scalar value is equal to x, then this point will be removed. Otherwise nothing happens.

8.38.2.18 void mitkColorTransferFunction::SetMax (int *xMax*, float *rMax*, float *gMax*, float *bMax*)

Set the maximum scalar value and its corresponding color.

Parameters:

xMax Specify the maximum scalar value. The scalar value range of this transfer function is from 0 to xMax.

rMax Specify the red component of the corresponding color

gMax Specify the green component of the corresponding color

bMax Specify the blue component of the corresponding color

Note:

This function will call RemoveAllPoints(), so it will clear the previous transfer function. After the calling of this function, the transfer function has two points. One is (0, 0.0, 0.0, 0.0), and the other is (xMax, rMax, gMax, bMax)

8.38.2.19 void mitkColorTransferFunction::SetUnmodified () [inline]

Reset to unmodified after changes have been done according to the new transfer function.

The documentation for this class was generated from the following file:

- mitkColorTransferFunction.h

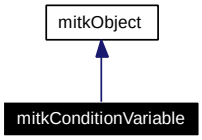
8.39 mitkConditionVariable Class Reference

mitkConditionVariable - a class that encapsulates a condition variable

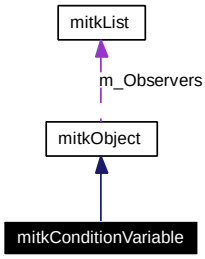
`#include <mitkConditionVariable.h>`

Inherits mitkObject.

Inheritance diagram for mitkConditionVariable:



Collaboration diagram for mitkConditionVariable:



8.39.1 Detailed Description

mitkConditionVariable - a class that encapsulates a condition variable

mitkConditionVariable encapsulates a condition variable.

The documentation for this class was generated from the following file:

- mitkConditionVariable.h

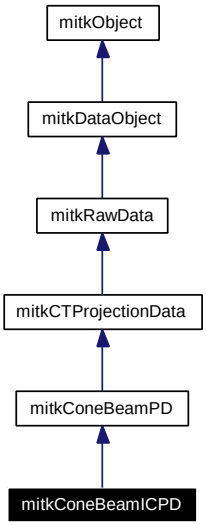
8.40 mitkConeBeamICPD Class Reference

mitkConeBeamICPD - concret class to represent cone-beam project data.

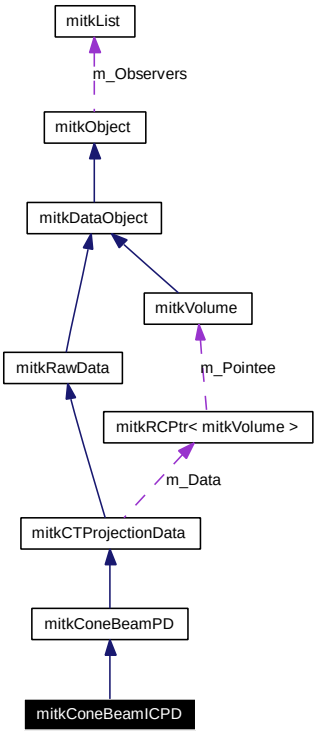
`#include <mitkConeBeamICPD.h>`

Inherits mitkConeBeamPD.

Inheritance diagram for mitkConeBeamICPD:



Collaboration diagram for mitkConeBeamICPD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void Initialize ()
- virtual void const * GetFrameForRead (int sliceIdx)

8.40.1 Detailed Description

mitkConeBeamICPD - concret class to represent cone-beam project data.

mitkConeBeamICPD - concret class to represent cone-beam project data.

8.40.2 Member Function Documentation

8.40.2.1 virtual void mitkConeBeamICPD::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamPD.

8.40.2.2 virtual unsigned long long mitkConeBeamICPD::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkConeBeamPD.

8.40.2.3 virtual void const* mitkConeBeamICPD::GetFrameForRead (int *sliceIdx*) [virtual]

GetAngleSliceForRead GetAngleSliceForWrite and GetAngleSliceForReadWrite

Implements mitkConeBeamPD.

8.40.2.4 virtual void mitkConeBeamICPD::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkConeBeamPD.

8.40.2.5 virtual void mitkConeBeamICPD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkConeBeamPD.

8.40.2.6 virtual void mitkConeBeamICPD::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamPD.

The documentation for this class was generated from the following file:

- mitkConeBeamICPD.h

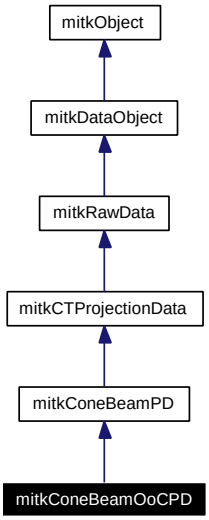
8.41 mitkConeBeamOoCPD Class Reference

mitkConeBeamOoCPD - concret class to represent cone-beam OoC project data .

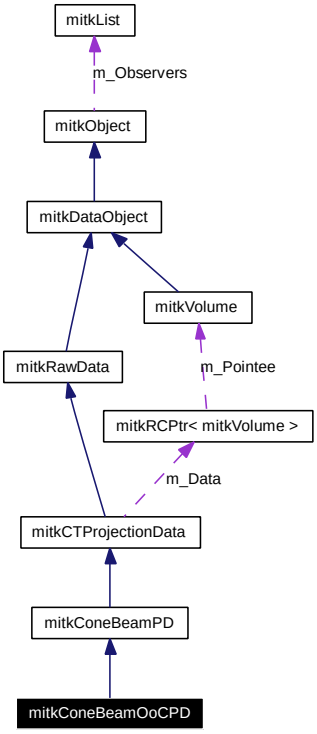
`#include <mitkConeBeamOoCPD.h>`

Inherits mitkConeBeamPD.

Inheritance diagram for mitkConeBeamOoCPD:



Collaboration diagram for mitkConeBeamOoCPD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void const * GetFrameForRead (int sliceIdx)
- virtual void Initialize ()

8.41.1 Detailed Description

mitkConeBeamOoCPD - concret class to represent cone-beam OoC project data .

mitkConeBeamOoCPD - concret class to represent cone-beam OoC project data.

8.41.2 Member Function Documentation

8.41.2.1 virtual void mitkConeBeamOoCPD::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamPD.

8.41.2.2 virtual unsigned long long mitkConeBeamOoCPD::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkConeBeamPD.

8.41.2.3 virtual void const* mitkConeBeamOoCPD::GetFrameForRead (int *sliceIdx*) [virtual]

GetAngleSliceForRead GetAngleSliceForWrite and GetAngleSliceForReadWrite

Implements mitkConeBeamPD.

8.41.2.4 virtual void mitkConeBeamOoCPD::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkConeBeamPD.

8.41.2.5 virtual void mitkConeBeamOoCPD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkConeBeamPD.

8.41.2.6 virtual void mitkConeBeamOoCPD::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamPD.

The documentation for this class was generated from the following file:

- mitkConeBeamOoCPD.h

8.42 mitkConeBeamPD Class Reference

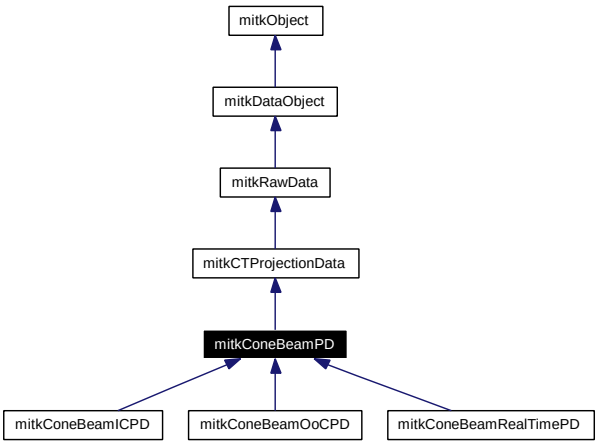
mitkConeBeamPD - abstract class to represent Cone-beam project data.

`#include <mitkConeBeamPD.h>`

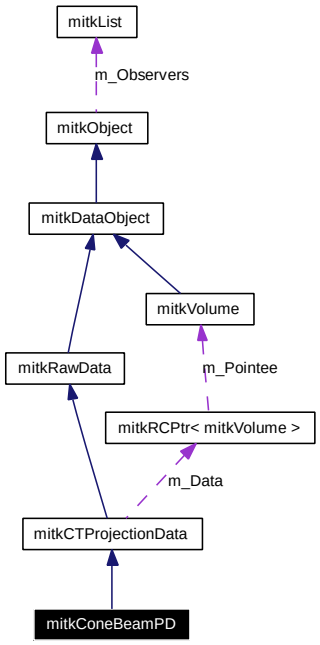
Inherits mitkCTProjectionData.

Inherited by mitkConeBeamICPD, mitkConeBeamOoCPD, and mitkConeBeamRealTimePD.

Inheritance diagram for mitkConeBeamPD:



Collaboration diagram for mitkConeBeamPD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const =0
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0
- virtual void const * GetVolumeData () const
- virtual void * GetVolumeData ()
- void SetVolumeData (mitkVolume *)
- virtual void Initialize ()

8.42.1 Detailed Description

mitkConeBeamPD - abstract class to represent Cone-beam project data.

mitkConeBeamPD - abstract class to represent Cone-beam project data. . ... sod . . . . . o . ..... sdd . . . .

z| | /y | / | / o|/ ————————-x detector is parallel with plane xoz

8.42.2 Member Function Documentation

8.42.2.1 virtual void mitkConeBeamPD::DeepCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkConeBeamICPD, mitkConeBeamOoCPD, and mitkConeBeamRealTimePD.

8.42.2.2 virtual unsigned long long mitkConeBeamPD::GetActualMemorySize () const [pure virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Note:
Pure virtual function. Its concrete subclass must implement this function and return its memory size.

Implements mitkDataObject.

Implemented in mitkConeBeamICPD, mitkConeBeamOoCPD, and mitkConeBeamRealTimePD.

8.42.2.3 virtual void* mitkConeBeamPD::GetVolumeData () [inline, virtual]

get volume pointer

Implements mitkCTProjectionData.

8.42.2.4 virtual void const* mitkConeBeamPD::GetVolumeData () const [inline, virtual]

get a const volume pointer

Implements mitkCTProjectionData.

8.42.2.5 virtual void mitkConeBeamPD::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkCTProjectionData.

Reimplemented in mitkConeBeamICPD, mitkConeBeamOoCPD, and mitkConeBeamRealTimePD.

8.42.2.6 virtual void mitkConeBeamPD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkCTProjectionData.

Reimplemented in mitkConeBeamICPD, mitkConeBeamOoCPD, and mitkConeBeamRealTimePD.

8.42.2.7 void mitkConeBeamPD::SetVolumeData (mitkVolume *)

Set data pointer of the project data.

8.42.2.8 virtual void mitkConeBeamPD::ShallowCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkConeBeamICPD, mitkConeBeamOoCPD, and mitkConeBeamRealTimePD.

The documentation for this class was generated from the following file:

- mitkConeBeamPD.h

8.43 mitkConeBeamProjectData Class Reference

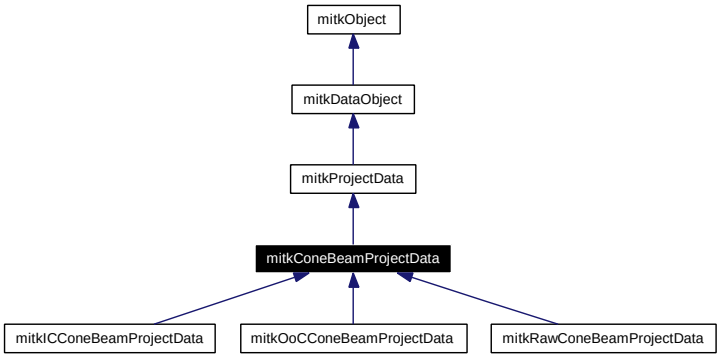
mitkConeBeamProjectData - abstract class to represent Cone-beam project data.

`#include <mitkConeBeamProjectData.h>`

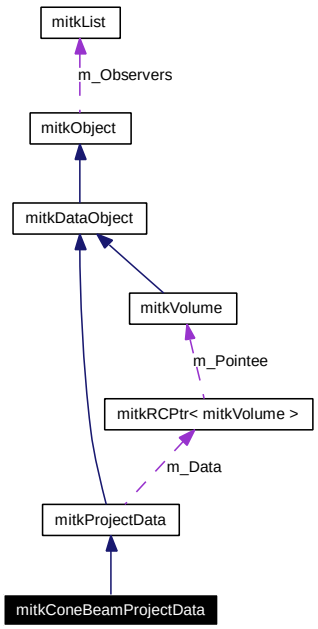
Inherits mitkProjectData.

Inherited by mitkICConeBeamProjectData, mitkOoCConeBeamProjectData, and mitkRawConeBeamProjectData.

Inheritance diagram for mitkConeBeamProjectData:



Collaboration diagram for mitkConeBeamProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()

- virtual void const * GetData () const
- virtual void * GetData ()
- void SetData (mitkVolume *m)
- float GetDetectorMoveX ()
- void SetDetectorMoveX (float m)
- float GetDetectorMoveZ ()
- void SetDetectorMoveZ (float m)
- void SetDetectorLengthX (double m)
- double GetDetectorLengthX ()
- void SetDetectorLengthZ (double m)
- double GetDetectorLengthZ ()
- void SetFov (double *m)
- double * GetFov ()
- void SetXrayNumX (int m)
- int GetXrayNumX () const
- void SetXrayNumZ (int m)
- int GetXrayNumZ () const
- void SetSourceToCenterDistance (double m)
- double GetSourceToCenterDistance () const
- void SetSourceToDetectorDistance (double m)
- double GetSourceToDetectorDistance () const
- virtual unsigned long long GetActualMemorySize () const =0
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0
- virtual void const * GetAngleSliceForRead (int sliceIdx)=0

8.43.1 Detailed Description

mitkConeBeamProjectData - abstract class to represent Cone-beam project data.

mitkConeBeamProjectData - abstract class to represent Cone-beam project data. . ... sod . . . . . o . ..... sdd . . . .

z| | /y | / | / o|/ ————————-x detector is parallel with plane xoz

8.43.2 Member Function Documentation

8.43.2.1 virtual void mitkConeBeamProjectData::DeepCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkICConeBeamProjectData, mitkOoCConeBeamProjectData, and mitkRawConeBeamProjectData.

8.43.2.2 virtual unsigned long long mitkConeBeamProjectData::GetActualMemorySize () const [pure virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Note:
Pure virtual function. Its concrete subclass must implement this function and return its memory size.

Implements mitkDataObject.

Implemented in mitkICConeBeamProjectData, mitkOoCConeBeamProjectData, and mitkRawConeBeamProjectData.

8.43.2.3 virtual void const* mitkConeBeamProjectData::GetAngleSliceForRead (int *sliceIdx*) [pure virtual]

GetAngleSliceForRead GetAngleSliceForWrite and GetAngleSliceForReadWrite

Implemented in mitkICConeBeamProjectData, mitkOoCConeBeamProjectData, and mitkRawConeBeamProjectData.

8.43.2.4 virtual void* mitkConeBeamProjectData::GetData () [inline, virtual]

get volume pointer

Implements mitkProjectData.

8.43.2.5 virtual void const* mitkConeBeamProjectData::GetData () const [inline, virtual]

get a const volume pointer

Implements mitkProjectData.

8.43.2.6 double mitkConeBeamProjectData::GetDetectorLengthX () [inline]

Get the length of detector at x axis

8.43.2.7 double mitkConeBeamProjectData::GetDetectorLengthZ () [inline]

Get the length of detector at y axis

8.43.2.8 float mitkConeBeamProjectData::GetDetectorMoveX () [inline]

get detector excursion at x axis (horizontal)

8.43.2.9 float mitkConeBeamProjectData::GetDetectorMoveZ () [inline]

Get detector excursion at z axis (vertical)

8.43.2.10 double* mitkConeBeamProjectData::GetFov () [inline]

Set field of view [0] left of x axis [1] right of x axis [2] min of y axis [3] max of y axis [4] bottom of z axis [5] top of z axis

8.43.2.11 double mitkConeBeamProjectData::GetSourceToCenterDistance () const [inline]

Get the distance between x-ray source and rotate center

8.43.2.12 double mitkConeBeamProjectData::GetSourceToDetectorDistance () const [inline]

Get the distance between x-ray source and detector center

8.43.2.13 int mitkConeBeamProjectData::GetXrayNumX () const [inline]

Get number of Xray at x axis

8.43.2.14 int mitkConeBeamProjectData::GetXrayNumZ () const [inline]

Get number of Xray at z axis

8.43.2.15 virtual void mitkConeBeamProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkProjectData.

Reimplemented in mitkICConeBeamProjectData, mitkOoCConeBeamProjectData, and mitkRawConeBeamProjectData.

8.43.2.16 virtual void mitkConeBeamProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkProjectData.

Reimplemented in mitkICConeBeamProjectData, mitkOoCConeBeamProjectData, and mitkRawConeBeamProjectData.

8.43.2.17 void mitkConeBeamProjectData::SetData (mitkVolume * *m*)

Set volume pointer

8.43.2.18 void mitkConeBeamProjectData::SetDetectorLengthX (double *m*) [inline]

Set the length of detector at x axis

8.43.2.19 void mitkConeBeamProjectData::SetDetectorLengthZ (double *m*) [inline]

Set the length of detector at z axis

8.43.2.20 void mitkConeBeamProjectData::SetDetectorMoveX (float *m*) [inline]

Set detector excursion at x axis (horizontal)

8.43.2.21 void mitkConeBeamProjectData::SetDetectorMoveZ (float *m*) [inline]

Set detector excursion at z axis (vertical)

8.43.2.22 void mitkConeBeamProjectData::SetFov (double * *m*) [inline]

Set field of view [0] left of x axis [1] right of x axis [2] min of y axis [3] max of y axis [4] bottom of z axis [5] top of z axis

8.43.2.23 void mitkConeBeamProjectData::SetSourceToCenterDistance (double *m*) [inline]

Set the distance between x-ray source and rotate center

8.43.2.24 void mitkConeBeamProjectData::SetSourceToDetectorDistance (double *m*) [inline]

Set the distance between x-ray source and detector center

8.43.2.25 void mitkConeBeamProjectData::SetXrayNumX (int *m*) [inline]

Set number of Xray at x axis

8.43.2.26 void mitkConeBeamProjectData::SetXrayNumZ (int *m*) [inline]

Set number of Xray at z axis

8.43.2.27 virtual void mitkConeBeamProjectData::ShallowCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkICConeBeamProjectData, mitkOoCConeBeamProjectData, and mitkRawConeBeamProjectData.

The documentation for this class was generated from the following file:

- mitkConeBeamProjectData.h

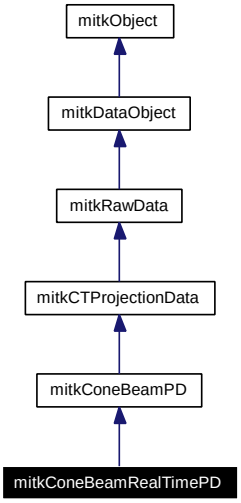
8.44 mitkConeBeamRealTimePD Class Reference

example

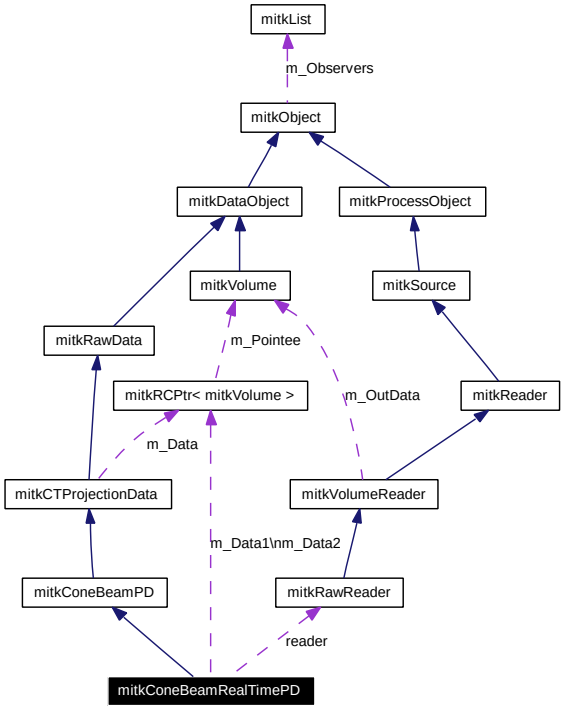
`#include <mitkConeBeamRealTimePD.h>`

Inherits mitkConeBeamPD.

Inheritance diagram for mitkConeBeamRealTimePD:



Collaboration diagram for mitkConeBeamRealTimePD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- void SetBlockSize (int m)
- bool SetFileName (const char *m)
- void SetIsOneFile (bool onefile)
- void SetFilePostfixNum (int m)
- virtual void const * GetFrameForRead (int frameIdx)
- virtual void * GetFrameForWrite (int frameIdx)
- virtual void * GetFrameForReadWrite (int frameIdx)
- int GetRecycleNum () const
- void SetRecycleNum (int m)
- virtual void Initialize ()

8.44.1 Detailed Description

example

mitkConeBeamRealTimePD - concret class to represent cone-beam project data.

8.44.2 Member Function Documentation

8.44.2.1 virtual void mitkConeBeamRealTimePD::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamPD.

8.44.2.2 virtual unsigned long long mitkConeBeamRealTimePD::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkConeBeamPD.

8.44.2.3 virtual void const* mitkConeBeamRealTimePD::GetFrameForRead (int *frameIdx*) [virtual]

GetFrameForRead

Implements mitkConeBeamPD.

8.44.2.4 virtual void* mitkConeBeamRealTimePD::GetFrameForReadWrite (int *frameIdx*) [virtual]

GetFrameForReadWrite dosn't write to harddisk

Implements mitkConeBeamPD.

8.44.2.5 virtual void* mitkConeBeamRealTimePD::GetFrameForWrite (int *frameIdx*) [inline, virtual]

Never call it

Implements mitkConeBeamPD.

8.44.2.6 int mitkConeBeamRealTimePD::GetRecycleNum () const [inline]

Get number of Gpu Block

8.44.2.7 virtual void mitkConeBeamRealTimePD::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkConeBeamPD.

8.44.2.8 virtual void mitkConeBeamRealTimePD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkConeBeamPD.

8.44.2.9 void mitkConeBeamRealTimePD::SetBlockSize (int *m*) [inline]

Set how many slices one block include and slice numbers should be devide by m

8.44.2.10 bool mitkConeBeamRealTimePD::SetFileName (const char * *m*)

Set file name

8.44.2.11 void mitkConeBeamRealTimePD::SetFilePostfixNum (int *m*) [inline]

Set File Postfix Num.

8.44.2.12 void mitkConeBeamRealTimePD::SetIsOneFile (bool *onefile*) [inline]

Set is one file

8.44.2.13 void mitkConeBeamRealTimePD::SetRecycleNum (int *m*) [inline]

Set number of Gpu Block

8.44.2.14 virtual void mitkConeBeamRealTimePD::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamPD.

The documentation for this class was generated from the following file:

- mitkConeBeamRealTimePD.h

8.45 mitkConeBeamReconFromHelix Class Reference

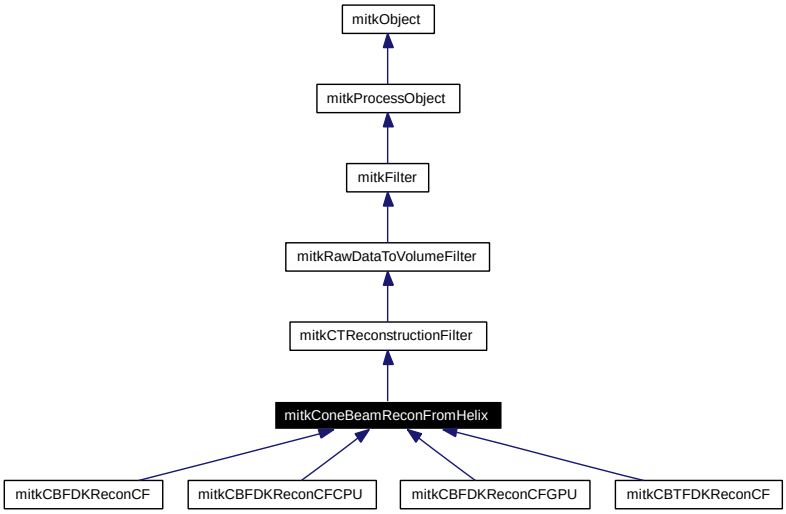
mitkConeBeamReconFromHelix - concret class implement Cone-beam reconstruction.

`#include <mitkConeBeamReconFromHelix.h>`

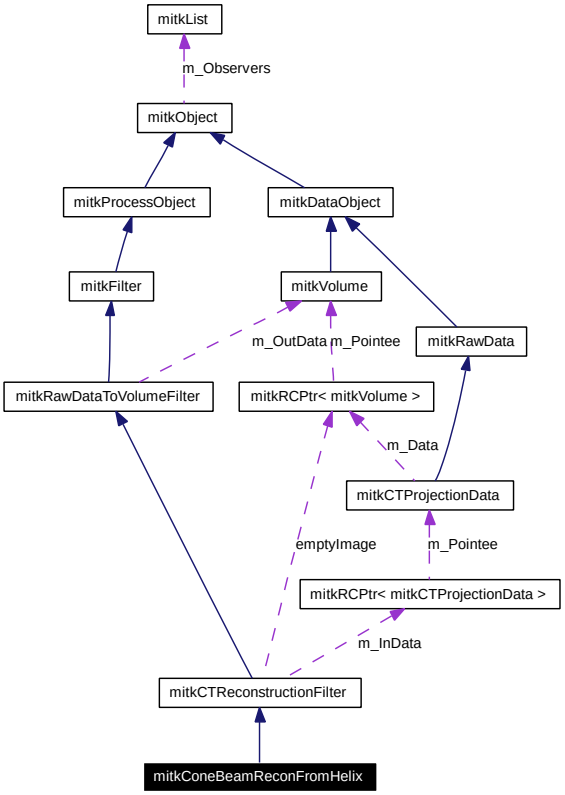
Inherits mitkCTReconstructionFilter.

Inherited by mitkCBFDKReconCF, mitkCBFDKReconCFCPU, mitkCBFDKReconCFGPU, and mitkCBTFDKReconCF.

Inheritance diagram for mitkConeBeamReconFromHelix:



Collaboration diagram for mitkConeBeamReconFromHelix:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- float GetDetectorMoveX ()
- void SetDetectorMoveX (float m)
- float GetDetectorMoveZ ()
- void SetDetectorMoveZ (float m)
- void SetDetectorLengthX (double m)
- double GetDetectorLengthX ()
- void SetDetectorLengthZ (double m)
- double GetDetectorLengthZ ()
- void SetFov (double *m)
- double * GetFov ()
- void SetSourceToCenterDistance (double m)
- double GetSourceToCenterDistance () const
- void SetSourceToDetectorDistance (double m)
- double GetSourceToDetectorDistance () const

8.45.1 Detailed Description

mitkConeBeamReconFromHelix - concret class implement Cone-beam reconstruction.

mitkConeBeamReconFromHelix - concret class implement Cone-beam reconstruction

8.45.2 Member Function Documentation

8.45.2.1 double mitkConeBeamReconFromHelix::GetDetectorLengthX () [inline]

// Get the length of detector of x axis

8.45.2.2 double mitkConeBeamReconFromHelix::GetDetectorLengthZ () [inline]

// Get the length of detector of z axis

8.45.2.3 float mitkConeBeamReconFromHelix::GetDetectorMoveX () [inline]

??

8.45.2.4 float mitkConeBeamReconFromHelix::GetDetectorMoveZ () [inline]

??

8.45.2.5 double* mitkConeBeamReconFromHelix::GetFov () [inline]

Get field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.45.2.6 double mitkConeBeamReconFromHelix::GetSourceToCenterDistance () const [inline]

Get the distance between x-ray source and rotate center

8.45.2.7 double mitkConeBeamReconFromHelix::GetSourceToDetectorDistance () const [inline]

Get the distance between x-ray source and detector center

8.45.2.8 virtual void mitkConeBeamReconFromHelix::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkCTReconstructionFilter.

Reimplemented in mitkCBFDKReconCF, mitkCBFDKReconCFCPU, mitkCBTFDKReconCF, and mitkCBFDKReconCFGPU.

8.45.2.9 void mitkConeBeamReconFromHelix::SetDetectorLengthX (double *m*) [inline]

Set the length of detector of x axis

8.45.2.10 void mitkConeBeamReconFromHelix::SetDetectorLengthZ (double *m*) [inline]

Set the length of detector of z axis

8.45.2.11 void mitkConeBeamReconFromHelix::SetDetectorMoveX (float *m*) [inline]

??

8.45.2.12 void mitkConeBeamReconFromHelix::SetDetectorMoveZ (float *m*) [inline]

??

8.45.2.13 void mitkConeBeamReconFromHelix::SetFov (double * *m*) [inline]

Set field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.45.2.14 void mitkConeBeamReconFromHelix::SetSourceToCenterDistance (double *m*) [inline]

Set the distance between x-ray source and rotate center

8.45.2.15 void mitkConeBeamReconFromHelix::SetSourceToDetectorDistance (double *m*) [inline]

Set the distance between x-ray source and detector center

The documentation for this class was generated from the following file:

- mitkConeBeamReconFromHelix.h

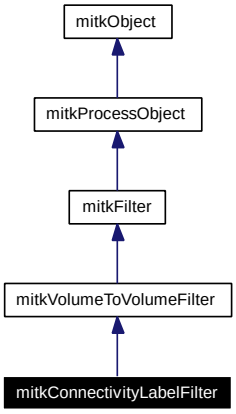
8.46 mitkConnectivityLabelFilter Class Reference

mitkConnectivityLabelFilter - A class that label each connected region with 1, 2, 3.....

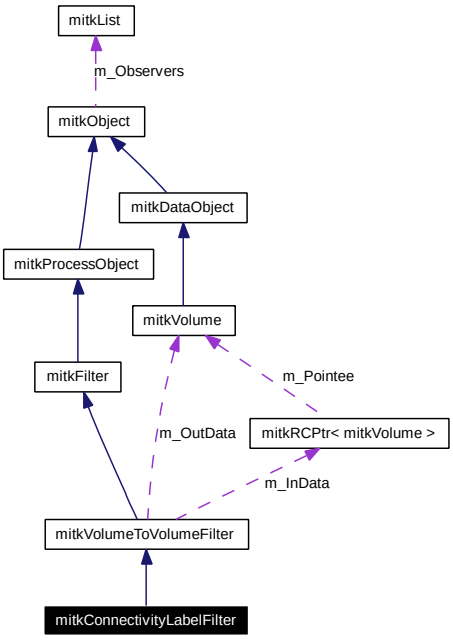
`#include <mitkConnectivityLabelFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkConnectivityLabelFilter:



Collaboration diagram for mitkConnectivityLabelFilter:



Public Member Functions

- mitkConnectivityLabelFilter ()
- virtual void PrintSelf (ostream &os)

- int GetConnectedRegionNum ()
- VectorIndexType GetVoxelNumOfRegion ()

8.46.1 Detailed Description

mitkConnectivityLabelFilter - A class that label each connected region with 1, 2, 3.....

mitkConnectivityLabelFilter is a class that label each connected region with 1, 2, 3..... Input: an image Output: an image of the same size as the input, containing labels for the connected components in the input.

8.46.2 Constructor & Destructor Documentation

8.46.2.1 mitkConnectivityLabelFilter::mitkConnectivityLabelFilter ()

Constructor of the class

8.46.3 Member Function Documentation

8.46.3.1 int mitkConnectivityLabelFilter::GetConnectedRegionNum () [inline]

Get the number of the connected region

Returns:
Return the connected region number

8.46.3.2 VectorIndexType mitkConnectivityLabelFilter::GetVoxelNumOfRegion () [inline]

Get the voxel number in each region

Returns:
Return a vector contains the voxel number list

8.46.3.3 virtual void mitkConnectivityLabelFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkConnectivityLabelFilter.h

8.47 mitkCTProjectionData Class Reference

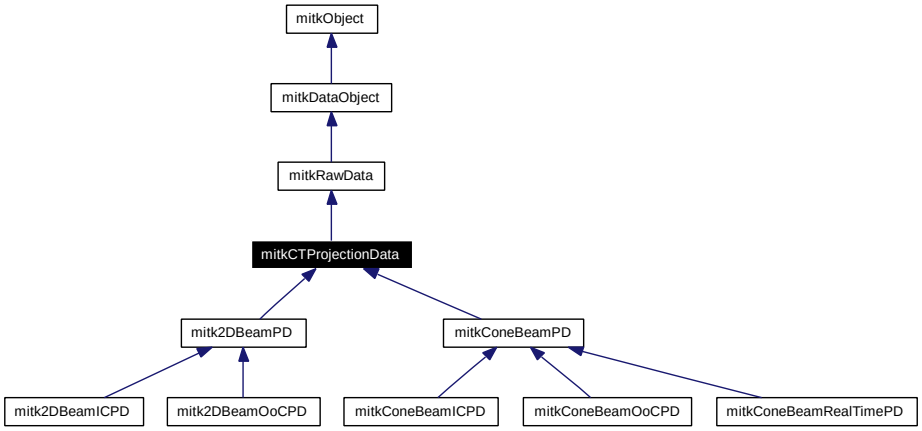
mitkCTProjectionData - abstract class to represent project data.

`#include <mitkCTProjectionData.h>`

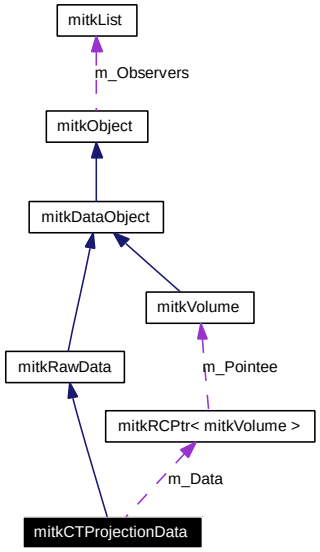
Inherits mitkRawData.

Inherited by mitk2DBeamPD, and mitkConeBeamPD.

Inheritance diagram for mitkCTProjectionData:



Collaboration diagram for mitkCTProjectionData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void const * GetVolumeData () const =0
- virtual void * GetVolumeData ()=0

- virtual void Initialize ()

8.47.1 Detailed Description

mitkCTProjectionData - abstract class to represent project data.

mitkCTProjectionData - abstract class to represent project data.

8.47.2 Member Function Documentation

8.47.2.1 virtual void* mitkCTProjectionData::GetVolumeData () [pure virtual]

Get data pointer of the project data (changeable).

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type

Implemented in mitk2DBeamPD, and mitkConeBeamPD.

8.47.2.2 virtual void const* mitkCTProjectionData::GetVolumeData () const [pure virtual]

Get data pointer of the project data (unchangeable).

Returns:
Return a void pointer to const data.

Note:
The returned type is void const *, it must be converted to some useful data type

Implemented in mitk2DBeamPD, and mitkConeBeamPD.

8.47.2.3 virtual void mitkCTProjectionData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Implements mitkDataObject.

Reimplemented in mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, and mitkConeBeamRealTimePD.

8.47.2.4 virtual void mitkCTProjectionData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkRawData.

Reimplemented in mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, and mitkConeBeamRealTimePD.

The documentation for this class was generated from the following file:

- mitkCTProjectionData.h

8.48 mitkCTReconstructionFilter Class Reference

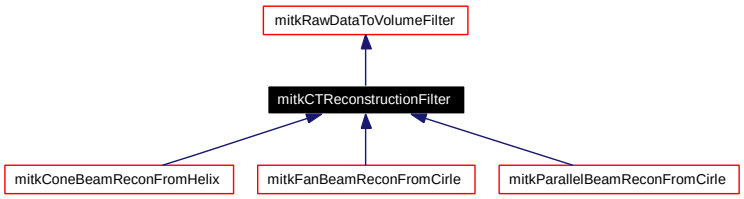
mitkCTReconstructionFilter - abstract class for forward construction

`#include <mitkCTReconstructionFilter.h>`

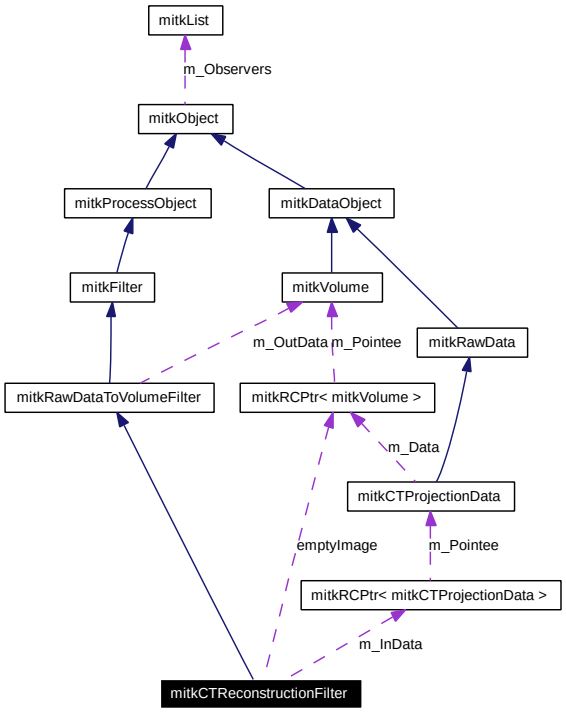
Inherits mitkRawDataToVolumeFilter.

Inherited by mitkConeBeamReconFromHelix, mitkFanBeamReconFromCirle, and mitkParallelBeam-ReconFromCirle.

Inheritance diagram for mitkCTReconstructionFilter:



Collaboration diagram for mitkCTReconstructionFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetBeginAngle (double angle)
- double GetBeginAngle () const
- void SetEndAngle (double angle)

- double GetEndAngle () const
- void SetInterpolationModel (int m)
- int GetInterpolationModel ()
- void SetInput (mitkCTProjectionData *inData)
- mitkCTProjectionData * GetInput ()
- mitkVolume * GetOutput ()
- void SetOutput (mitkVolume *m)
- void SetIsLogOperation (bool m)
- bool GetIsLogOperation ()
- void SetEmptyScanImage (mitkVolume *m)
- mitkVolume * GetEmptyScanImage ()

8.48.1 Detailed Description

mitkCTReconstructionFilter - abstract class for forward construction

mitkCTReconstructionFilter - abstract class for forward construction

8.48.2 Member Function Documentation

8.48.2.1 double mitkCTReconstructionFilter::GetBeginAngle () const [inline]

get beginning rotate angle which is the angle between detector and +x axis. unit are degree

8.48.2.2 mitkVolume* mitkCTReconstructionFilter::GetEmptyScanImage () [inline]

Get empty scan image usefull when isLogOperation is false

8.48.2.3 double mitkCTReconstructionFilter::GetEndAngle () const [inline]

get end rotate angle which is the angle between detector and +x axis. unit are degree

8.48.2.4 mitkCTProjectionData* mitkCTReconstructionFilter::GetInput () [inline]

Get the input ProjectData

Returns:
Return the input ProjectData

8.48.2.5 int mitkCTReconstructionFilter::GetInterpolationModel () [inline]

Get the interpolation model

8.48.2.6 bool mitkCTReconstructionFilter::GetIsLogOperation () [inline]

Get whether log operation is done

Returns:
Return the output volume

8.48.2.7 mitkVolume* mitkCTReconstructionFilter::GetOutput () [virtual]

Get the output volume

Returns:
Return the output volume

Implements mitkRawDataToVolumeFilter.

8.48.2.8 virtual void mitkCTReconstructionFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRawDataToVolumeFilter.

Reimplemented in mitkCBFDKReconCF, mitkCBFDKReconCFCPU, mitkCBTFDKReconCF, mitkConeBeamReconFromHelix, mitkFanBeamReconFromCirle, mitkFBFBPReconCF, mitkFBFBPReconCFCPU, mitkParallelBeamReconFromCirle, mitkPBFBPReconCF, mitkCBFDKReconCFGPU, and mitkFBFBPReconCFGPU.

8.48.2.9 void mitkCTReconstructionFilter::SetBeginAngle (double *angle*) [inline]

Set beginning rotate angle which is the angle between detector and +x axis. unit must be degree

8.48.2.10 void mitkCTReconstructionFilter::SetEmptyScanImage (mitkVolume * *m*) [inline]

Set empty scan image usefull when isLogOperation is false

8.48.2.11 void mitkCTReconstructionFilter::SetEndAngle (double *angle*) [inline]

Set end rotate angle which is the angle between detector and +x axis. unit must be degree

8.48.2.12 void mitkCTReconstructionFilter::SetInput (mitkCTProjectionData * *inData*) [inline]

Set the input ProjectData

Parameters:
inData Specify the input ProjectData

8.48.2.13 void mitkCTReconstructionFilter::SetInterpolationModel (int *m*) [inline]

Set the interpolation model

8.48.2.14 void mitkCTReconstructionFilter::SetIsLogOperation (bool *m*) [inline]

Set whether log operation is done

Returns:
Return the output volume

8.48.2.15 void mitkCTReconstructionFilter::SetOutput (mitkVolume * *m*) [inline]

Set the output volume

The documentation for this class was generated from the following file:

- mitkCTReconstructionFilter.h

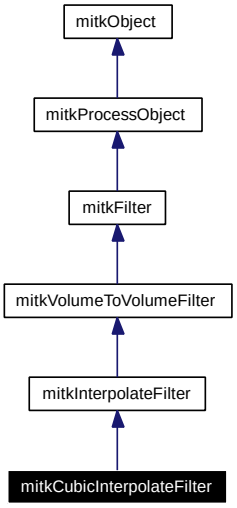
8.49 mitkCubicInterpolateFilter Class Reference

mitkCubicInterpolateFilter - a concrete class for cubic interpolation.

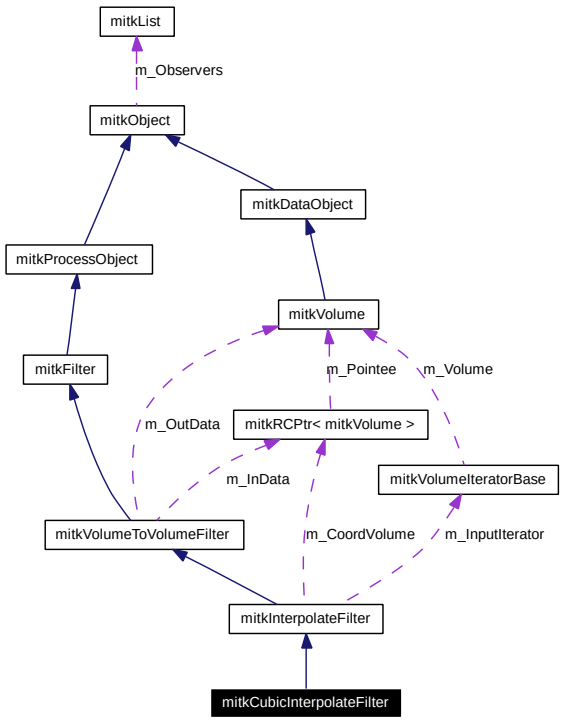
`#include <mitkCubicInterpolateFilter.h>`

Inherits mitkInterpolateFilter.

Inheritance diagram for mitkCubicInterpolateFilter:



Collaboration diagram for mitkCubicInterpolateFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkCubicInterpolateFilter ()

8.49.1 Detailed Description

mitkCubicInterpolateFilter - a concrete class for cubic interpolation.

mitkCubicInterpolateFilter is a concrete class for Catmull-Rom spline (Catmull & Rom 1974). A bi-cubic algorithem was implemented for 2D image which the intensity at a point is determined from the weighted sum of intensities at 16 pixels closest to it. Also a tri-cubic method was developed to perform 3d volume interpolation that the intensity at a point is estimated from 4x4x4 neighborhood.

This filter can perform point interpolation by using InterpolatePoint() and perform volume interpolation by using Run().

User should specify the input volume and run Update() first before interpolation.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 mitkCubicInterpolateFilter::mitkCubicInterpolateFilter ()

Constructor.

8.49.3 Member Function Documentation

8.49.3.1 virtual void mitkCubicInterpolateFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkInterpolateFilter.

The documentation for this class was generated from the following file:

- mitkCubicInterpolateFilter.h

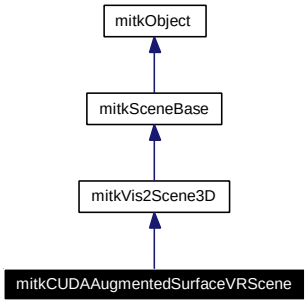
8.50 mitkCUDAAugmentedSurfaceVRScene Class Reference

mitkCUDAAugmentedSurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface.

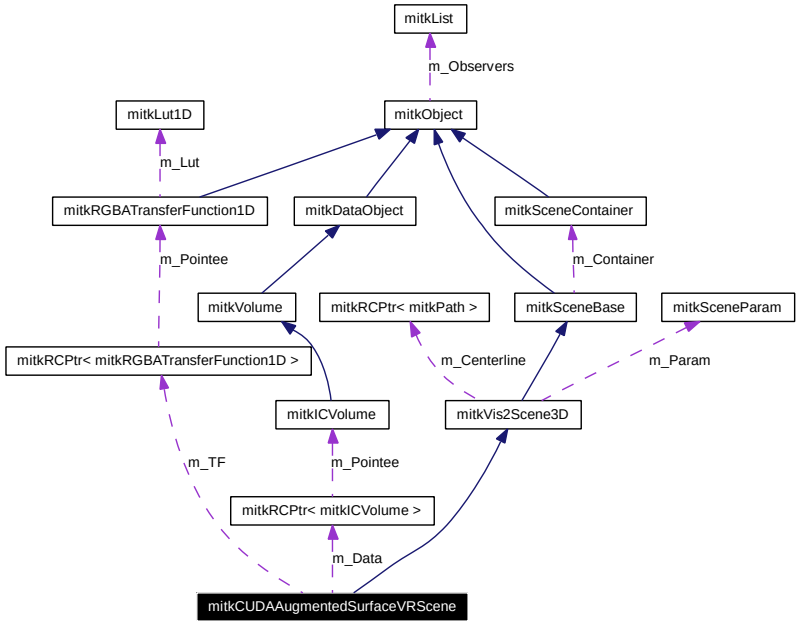
`#include <mitkCUDAAugmentedSurfaceVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAAugmentedSurfaceVRScene:



Collaboration diagram for mitkCUDAAugmentedSurfaceVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- void SetIsovalue (float isovalue)
- float GetIsovalue ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void SetDensityFactor (float densityFactor)
- float GetDensityFactor ()
- mitkRGBATransferFunction1D * GetTransferFunction ()
- void SetTransferFunction (mitkRGBATransferFunction1D *tf)
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void BeginAddPeelingWidget ()
- int GetCurrentWidgetLayerNumber ()
- void SetCurrentWidgetLayerNumber (int number)
- void RemoveCurrentPeelingWidget ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.50.1 Detailed Description

mitkCUDAAugmentedSurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface.

mitkCUDAAugmentedSurfaceVRScene is a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface. Comparing to mitkCUDAIsosurfaceVRScene, this isosurface has more abundunt color. mitkCUDAAugmentedSurfaceVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.50.2 Member Function Documentation

8.50.2.1 void mitkCUDAAugmentedSurfaceVRScene::BeginAddPeelingWidget ()

Add a peeling widget into the scene

8.50.2.2 float mitkCUDAAugmentedSurfaceVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.50.2.3 void mitkCUDAAugmentedSurfaceVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.50.2.4 void mitkCUDAAugmentedSurfaceVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.50.2.5 int mitkCUDAAugmentedSurfaceVRScene::GetCurrentWidgetLayerNumber ()

See how many layers does the current peeling widget peel.

8.50.2.6 float mitkCUDAAugmentedSurfaceVRScene::GetDensityFactor ()

Get the density factor used in the coloring of the isosurface

Returns:

Return The density factor used in the coloring of the isosurface

8.50.2.7 float mitkCUDAAugmentedSurfaceVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.50.2.8 float mitkCUDAAugmentedSurfaceVRScene::GetIsovalue ()

Get the isovalue of the isosurface to be displayed.

Returns:

Return the isovalue.

8.50.2.9 void mitkCUDAAugmentedSurfaceVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:

x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.50.2.10 float mitkCUDAAugmentedSurfaceVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:

Return The specular factor of the material of the isosurface

8.50.2.11 float mitkCUDAAugmentedSurfaceVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:

Return The specular power of the material of the isosurface

8.50.2.12 mitkRGBATransferFunction1D* mitkCUDAAugmentedSurfaceVRScene::GetTransferFunction ()

Get the transfer function

Returns:

Return the transfer function.

8.50.2.13 virtual void mitkCUDAAugmentedSurfaceVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.50.2.14 virtual void mitkCUDAAugmentedSurfaceVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.50.2.15 virtual void mitkCUDAAugmentedSurfaceVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.50.2.16 void mitkCUDAAugmentedSurfaceVRScene::RemoveCurrentPeelingWidget ()

Remove the current peeling widget

8.50.2.17 virtual void mitkCUDAAugmentedSurfaceVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.50.2.18 void mitkCUDAAugmentedSurfaceVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:

Whether turn the light rotation mode on.

8.50.2.19 void mitkCUDAAugmentedSurfaceVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:

value The ambient factor of the material of the isosurface

8.50.2.20 void mitkCUDAAugmentedSurfaceVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:

r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.50.2.21 void mitkCUDAAugmentedSurfaceVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.50.2.22 void mitkCUDAAugmentedSurfaceVRScene::SetCurrentWidgetLayerNumber (int *number*)

Set how many layers does the current peeling widget peel.

8.50.2.23 void mitkCUDAAugmentedSurfaceVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.50.2.24 void mitkCUDAAugmentedSurfaceVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.50.2.25 void mitkCUDAAugmentedSurfaceVRScene::SetDensityFactor (float *densityFactor*)

Set the density factor used in the coloring of the isosurface

Parameters:
densityFactor The density factor used in the coloring of the isosurface

8.50.2.26 void mitkCUDAAugmentedSurfaceVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.50.2.27 void mitkCUDAAugmentedSurfaceVRScene::SetIsovalue (float *isovalue*)

Set the isovalue of the isosurface to be displayed.

Parameters:
isovalue The isovalue.

8.50.2.28 void mitkCUDAAugmentedSurfaceVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.
g Specify the green component of the light source color.
b Specify the blue component of the light source color.

8.50.2.29 void mitkCUDAAugmentedSurfaceVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:

x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.50.2.30 void mitkCUDAAugmentedSurfaceVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:

intensity Light's intensity

8.50.2.31 void mitkCUDAAugmentedSurfaceVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:

value The specular factor of the material of the isosurface

8.50.2.32 void mitkCUDAAugmentedSurfaceVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:

value The specular power of the material of the isosurface

8.50.2.33 void mitkCUDAAugmentedSurfaceVRScene::SetTransferFunction (mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:

tf The transfer function.

The documentation for this class was generated from the following file:

- mitkCUDAAugmentedSurfaceVRScene.h

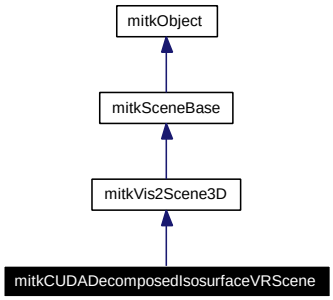
8.51 mitkCUDADecomposedIsosurfaceVRScene Class Reference

mitkCUDADecomposedIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets, for large data. Accelerated with the CUDA interface.

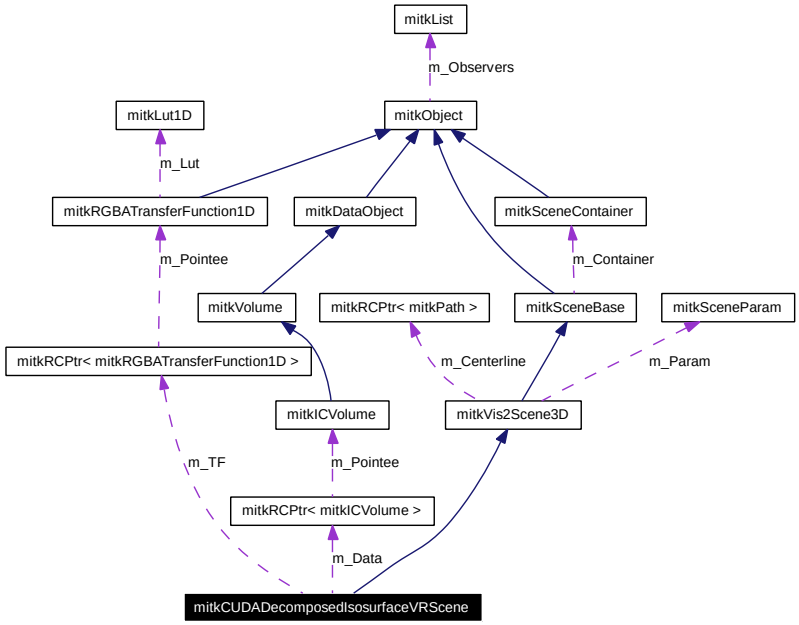
`#include <mitkCUDADecomposedIsosurfaceVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDADecomposedIsosurfaceVRScene:



Collaboration diagram for mitkCUDADecomposedIsosurfaceVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- void SetIsovalue (float isovalue)
- float GetIsovalue ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void SetDensityFactor (float densityFactor)
- float GetDensityFactor ()
- mitkRGBATransferFunction1D * GetTransferFunction ()
- void SetTransferFunction (mitkRGBATransferFunction1D *tf)
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void BeginAddPeelingWidget ()
- int GetCurrentWidgetLayerNumber ()
- void SetCurrentWidgetLayerNumber (int number)
- void RemoveCurrentPeelingWidget ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.51.1 Detailed Description

mitkCUDADecomposedIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets, for large data. Accelerated with the CUDA interface.

mitkCUDADecomposedIsosurfaceVRScene is a 3D scene to display isosurfaces from volume datasets, for large data. Accelerated with the CUDA interface.

8.51.2 Member Function Documentation

8.51.2.1 void mitkCUDADecomposedIsosurfaceVRScene::BeginAddPeelingWidget ()

Add a peeling widget into the scene

8.51.2.2 float mitkCUDADecomposedIsosurfaceVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.51.2.3 void mitkCUDADecomposedIsosurfaceVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.51.2.4 void mitkCUDADecomposedIsosurfaceVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.51.2.5 int mitkCUDADecomposedIsosurfaceVRScene::GetCurrentWidgetLayerNumber ()

See how many layers does the current peeling widget peel.

8.51.2.6 float mitkCUDADecomposedIsosurfaceVRScene::GetDensityFactor ()

Get the density factor used in the coloring of the isosurface

Returns:

Return The density factor used in the coloring of the isosurface

8.51.2.7 float mitkCUDADecomposedIsosurfaceVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.51.2.8 float mitkCUDADecomposedIsosurfaceVRScene::GetIsovalue ()

Get the isovalue of the isosurface to be displayed.

Returns:

Return the isovalue.

8.51.2.9 void mitkCUDADecomposedIsosurfaceVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:

x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.51.2.10 float mitkCUDADecomposedIsosurfaceVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:

Return The specular factor of the material of the isosurface

8.51.2.11 float mitkCUDADecomposedIsosurfaceVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:

Return The specular power of the material of the isosurface

8.51.2.12 mitkRGBATransferFunction1D* mitkCUDADecomposedIsosurfaceVRScene::GetTransferFunction ()

Get the transfer function

Returns:

Return the transfer function.

8.51.2.13 virtual void mitkCUDADecomposedIsosurfaceVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.51.2.14 virtual void mitkCUDADecomposedIsosurfaceVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.51.2.15 virtual void mitkCUDADecomposedIsosurfaceVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.51.2.16 void mitkCUDADecomposedIsosurfaceVRScene::RemoveCurrentPeelingWidget ()

Set how many layers does the current peeling widget peel.

8.51.2.17 virtual void mitkCUDADecomposedIsosurfaceVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.51.2.18 void mitkCUDADecomposedIsosurfaceVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:

Whether turn the light rotation mode on.

8.51.2.19 void mitkCUDADecomposedIsosurfaceVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:

value The ambient factor of the material of the isosurface

8.51.2.20 void mitkCUDADecomposedIsosurfaceVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:

r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.51.2.21 void mitkCUDADecomposedIsosurfaceVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.51.2.22 void mitkCUDADecomposedIsosurfaceVRScene::SetCurrentWidgetLayerNumber (int *number*)

Set how many layers does the current peeling widget peel.

8.51.2.23 void mitkCUDADecomposedIsosurfaceVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.51.2.24 void mitkCUDADecomposedIsosurfaceVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.51.2.25 void mitkCUDADecomposedIsosurfaceVRScene::SetDensityFactor (float *densityFactor*)

Set the density factor used in the coloring of the isosurface

Parameters:
densityFactor The density factor used in the coloring of the isosurface

8.51.2.26 void mitkCUDADecomposedIsosurfaceVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.51.2.27 void mitkCUDADecomposedIsosurfaceVRScene::SetIsovalue (float *isovalue*)

Set the isovalue of the isosurface to be displayed.

Parameters:
isovalue The isovalue.

8.51.2.28 void mitkCUDADecomposedIsosurfaceVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.
g Specify the green component of the light source color.
b Specify the blue component of the light source color.

8.51.2.29 void mitkCUDADecomposedIsosurfaceVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:

x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.51.2.30 void mitkCUDADecomposedIsosurfaceVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:

intensity Light's intensity

8.51.2.31 void mitkCUDADecomposedIsosurfaceVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:

value The specular factor of the material of the isosurface

8.51.2.32 void mitkCUDADecomposedIsosurfaceVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:

value The specular power of the material of the isosurface

8.51.2.33 void mitkCUDADecomposedIsosurfaceVRScene::SetTransferFunction (mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:

tf The transfer function.

The documentation for this class was generated from the following file:

- mitkCUDADecomposedIsosurfaceVRScene.h

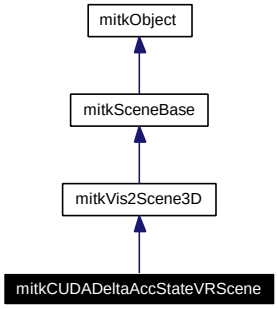
8.52 mitkCUDADeltaAccStateVRScene Class Reference

mitkCUDADeltaAccStateVRScene - a 3D scene for volume rendering with delta accumulation based ray feature analysis. Accelerated with the CUDA interface.

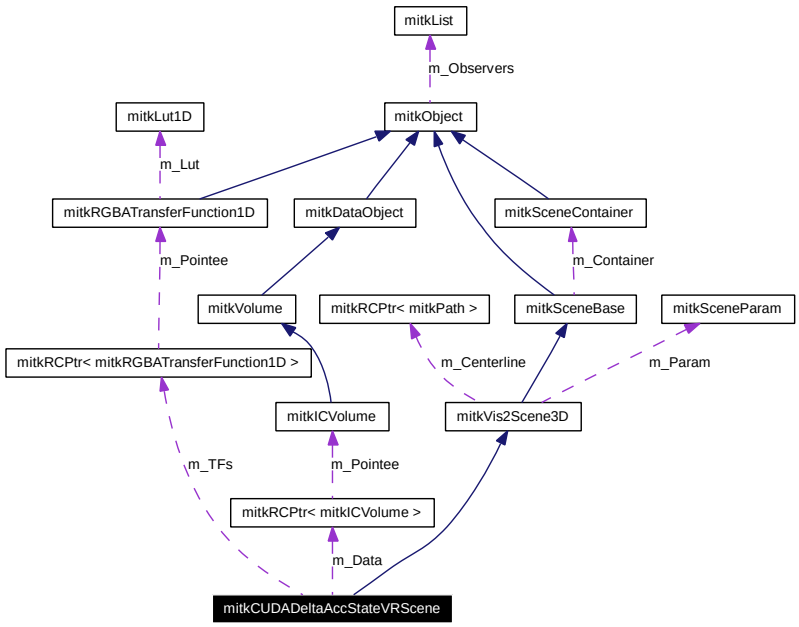
`#include <mitkCUDADeltaAccStateVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDADeltaAccStateVRScene:



Collaboration diagram for mitkCUDADeltaAccStateVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- mitkRGBATransferFunction1D * GetTransferFunction (int index)
- void SetTransferFunction (int index, mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetShade (bool shade)
- bool GetShade ()
- void ShadeOn ()
- void ShadeOff ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetLayerShadingMode (int layer, ShadingMode mode)
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])
- void SetBackwardsTest (bool test)
- bool GetBackwardsTest ()

8.52.1 Detailed Description

mitkCUDADeltaAccStateVRScene - a 3D scene for volume rendering with delta accumulation based ray feature analysis. Accelerated with the CUDA interface.

mitkCUDADeltaAccStateVRScene is a 3D scene for volume rendering with delta accumulation based ray feature analysis. Accelerated with the CUDA interface.

8.52.2 Member Function Documentation

8.52.2.1 float mitkCUDADeltaAccStateVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.52.2.2 void mitkCUDADeltaAccStateVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.52.2.3 bool mitkCUDADeltaAccStateVRScene::GetBackwardsTest () [inline]

Get the backwards test status

Returns:

: the backwards test status

8.52.2.4 void mitkCUDADeltaAccStateVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.52.2.5 float mitkCUDADeltaAccStateVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.52.2.6 bool mitkCUDADeltaAccStateVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:

: Return the illustrative shading status.

8.52.2.7 void mitkCUDADeltaAccStateVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:

x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.52.2.8 bool mitkCUDADeltaAccStateVRScene::GetShade ()

Get the shading status

Returns:
Return the shading status.

8.52.2.9 float mitkCUDADeltaAccStateVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.52.2.10 float mitkCUDADeltaAccStateVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.52.2.11 float mitkCUDADeltaAccStateVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.52.2.12 mitkRGBATransferFunction1D* mitkCUDADeltaAccStateVRScene::GetTransferFunction (int *index*)

Get the transfer function

Returns:
Return the transfer function.

8.52.2.13 virtual void mitkCUDADeltaAccStateVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.52.2.14 virtual void mitkCUDADeltaAccStateVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.52.2.15 virtual void mitkCUDADeltaAccStateVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.52.2.16 virtual void mitkCUDADeltaAccStateVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.52.2.17 void mitkCUDADeltaAccStateVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:

Whether turn the light rotation mode on.

8.52.2.18 void mitkCUDADeltaAccStateVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:

value The ambient factor of the material of the isosurface

8.52.2.19 void mitkCUDADeltaAccStateVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:

r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.52.2.20 void mitkCUDADeltaAccStateVRScene::SetBackwardsTest (bool *test*) [inline]

Set the backwards test status

Parameters:

test,: the backwards test status

8.52.2.21 void mitkCUDADeltaAccStateVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.52.2.22 void mitkCUDADeltaAccStateVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.52.2.23 void mitkCUDADeltaAccStateVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.52.2.24 void mitkCUDADeltaAccStateVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.52.2.25 void mitkCUDADeltaAccStateVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:
shade,: The illustrative shading status.

8.52.2.26 void mitkCUDADeltaAccStateVRScene::SetLayerShadingMode (int *layer*, ShadingMode *mode*)

Set shading mode of a layer

Parameters:
layer,: the id of the layer

mode,: the shading mode

8.52.2.27 void mitkCUDADeltaAccStateVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.

g Specify the green component of the light source color.

b Specify the blue component of the light source color.

8.52.2.28 void mitkCUDADeltaAccStateVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:

x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.52.2.29 void mitkCUDADeltaAccStateVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:

intensity Light's intensity

8.52.2.30 void mitkCUDADeltaAccStateVRScene::SetShade (bool *shade*)

Turn on or turn off shading.

Parameters:

shade The shading status.

8.52.2.31 void mitkCUDADeltaAccStateVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:

value The specular factor of the material of the isosurface

8.52.2.32 void mitkCUDADeltaAccStateVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:

value The specular power of the material of the isosurface

8.52.2.33 void mitkCUDADeltaAccStateVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:

stdSampleDistance The standard sample distance

8.52.2.34 void mitkCUDADeltaAccStateVRScene::SetTransferFunction (int *index*, mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

8.52.2.35 void mitkCUDADeltaAccStateVRScene::ShadeOff ()

Turn off shading.

8.52.2.36 void mitkCUDADeltaAccStateVRScene::ShadeOn ()

Turn on shading.

The documentation for this class was generated from the following file:

- mitkCUDADeltaAccStateVRScene.h

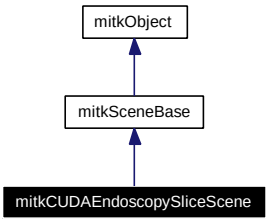
8.53 mitkCUDAEndoscopySliceScene Class Reference

mitkCUDAEndoscopySliceScene - A CUDA based slice preview scene for virtual endoscopy navigation

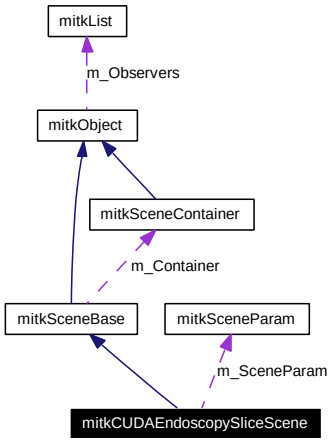
`#include <mitkCUDAEndoscopySliceScene.h>`

Inherits mitkSceneBase.

Inheritance diagram for mitkCUDAEndoscopySliceScene:



Collaboration diagram for mitkCUDAEndoscopySliceScene:



Public Member Functions

- virtual void RenderGL ()
- void SetVolumeTexture (mitkCUDAVolumeTexture *tex)
- mitkCUDAVolumeTexture * GetVolumeTexture ()
- void SetVolumeInfo (int dims[3], float spacings[3])
- void SetHorizontial ()
- void SetVertical ()
- bool IsHorizontial ()
- void SetGridSpacing (float spacing)
- float GetGridSpacing ()
- void SetSceneParam (mitkSceneParam *sceneParam)
- mitkSceneParam * GetSceneParam ()
- void SetWidthCenter (float windowWidth, float windowCenter)
- void SetWindowWidth (float windowWidth)
- void SetWindowCenter (float windowCenter)

- float GetWindowWidth ()
- float GetWindowCenter ()
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseWheel (bool ctrlDown, bool shiftDown, int xPos, int yPos, int delta)
- void SetRotateRadPerPixel (float radPerPixel)
- float GetRotateRadPerPixel () const

8.53.1 Detailed Description

mitkCUDAEndoscopySliceScene - A CUDA based slice preview scene for virtual endoscopy navigation

mitkCUDAEndoscopySliceScene is a CUDA based slice preview scene for virtual endoscopy navigation

8.53.2 Member Function Documentation

8.53.2.1 float mitkCUDAEndoscopySliceScene::GetGridSpacing () [inline]

Get the 2D grid spacing in physical unit.

Returns:
: 2D grid spacing in physical unit.

8.53.2.2 float mitkCUDAEndoscopySliceScene::GetRotateRadPerPixel () const [inline]

Get the interaction sensitivity of rotation.

Returns:
Return the rad the object is rotated when the mouse is moved 1 pixel

8.53.2.3 mitkSceneParam* mitkCUDAEndoscopySliceScene::GetSceneParam () [inline]

Get the internal scene param.

Returns:
: the internal scene param.

8.53.2.4 mitkCUDAVolumeTexture* mitkCUDAEndoscopySliceScene::GetVolumeTexture ()

Get the data object to be navigated

Returns:
The data object to be navigated.

8.53.2.5 float mitkCUDAEndoscopySliceScene::GetWindowCenter ()

Get the window center

Returns:
Return the window center

8.53.2.6 float mitkCUDAEndoscopySliceScene::GetWindowWidth ()

Get the window width

Returns:
Return the window width

8.53.2.7 bool mitkCUDAEndoscopySliceScene::IsHorizontial () [inline]

Get the display mode.

Returns:
: true=horizontal false=vertical

8.53.2.8 virtual void mitkCUDAEndoscopySliceScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkSceneBase.

8.53.2.9 virtual void mitkCUDAEndoscopySliceScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkSceneBase.

8.53.2.10 virtual void mitkCUDAEndoscopySliceScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkSceneBase.

8.53.2.11 virtual void mitkCUDAEndoscopySliceScene::OnMouseWheel (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *delta*) [virtual]

The Interface for handling the mouse wheel event pass by the container

Reimplemented from mitkSceneBase.

8.53.2.12 virtual void mitkCUDAEndoscopySliceScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkSceneBase.

8.53.2.13 void mitkCUDAEndoscopySliceScene::SetGridSpacing (float *spacing*) [inline]

Set the 2D grid spacing in physical unit.

Parameters:

spacings,: 2D grid spacing in physical unit.

8.53.2.14 void mitkCUDAEndoscopySliceScene::SetHorizontial () [inline]

Set to horizontal mode

8.53.2.15 void mitkCUDAEndoscopySliceScene::SetRotateRadPerPixel (float *radPerPixel*) [inline]

Set the interaction sensitivity of rotation.

Parameters:

radPerPixel. The rad the object is rotated when the mouse is moved 1 pixel

8.53.2.16 void mitkCUDAEndoscopySliceScene::SetSceneParam (mitkSceneParam * *sceneParam*) [inline]

Set the internal scene param.

Parameters:

sceneParam,: the internal scene param.

8.53.2.17 void mitkCUDAEndoscopySliceScene::SetVertical () [inline]

Set to vertical mode

8.53.2.18 void mitkCUDAEndoscopySliceScene::SetVolumeInfo (int *dims*[3], float *spacings*[3])

Set the volume dimensions and spacings then calculate the cropping boundaries

Parameters:

dims,: dimensions.

float,: spacings.

8.53.2.19 void mitkCUDAEndoscopySliceScene::SetVolumeTexture (mitkCUDAVolumeTexture * *tex*)

Set the data object to be navigated

Parameters:
tex,: The data object to be navigated.

8.53.2.20 void mitkCUDAEndoscopySliceScene::SetWidthCenter (float *windowWidth*, float *windowCenter*)

Set the window width and windows center

Parameters:
windowWidth The window width

windowCenter The window center

8.53.2.21 void mitkCUDAEndoscopySliceScene::SetWindowCenter (float *windowCenter*)

Set the window center

Parameters:
windowCenter The window center

8.53.2.22 void mitkCUDAEndoscopySliceScene::SetWindowWidth (float *windowWidth*)

Set the window width

Parameters:
windowWidth The window width

The documentation for this class was generated from the following file:

- mitkCUDAEndoscopySliceScene.h

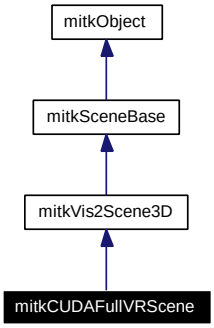
8.54 mitkCUDAFullVRScene Class Reference

mitkCUDAFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface.

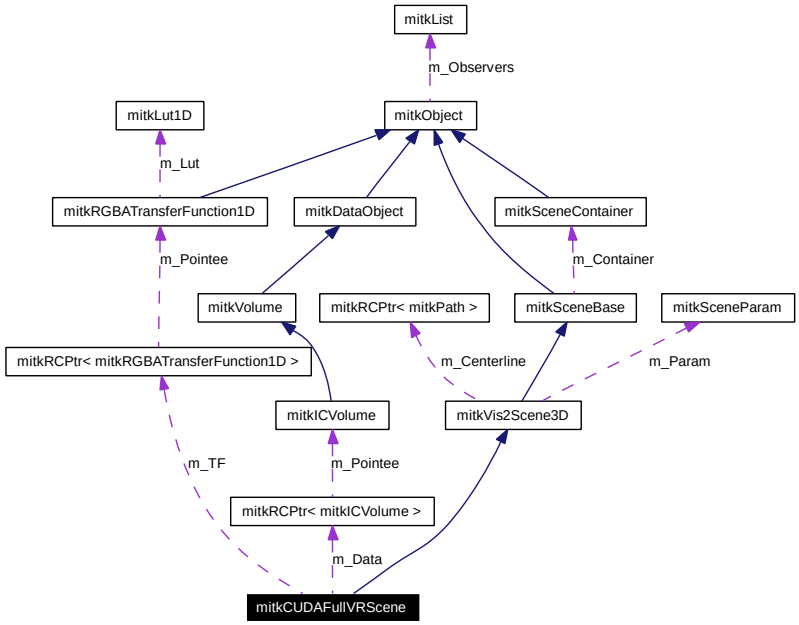
`#include <mitkCUDAFullVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAFullVRScene:



Collaboration diagram for mitkCUDAFullVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)

- void GetBackGroundColor (float &r, float &g, float &b)
- mitkRGBATransferFunction1D * GetTransferFunction ()
- void SetTransferFunction (mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetShade (bool shade)
- bool GetShade ()
- void ShadeOn ()
- void ShadeOff ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])
- void SetPreintegration (bool preInt)
- bool GetPreintegration ()

8.54.1 Detailed Description

mitkCUDAFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface.

mitkCUDAFullVRScene is a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface. mitkCUDAFullVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.54.2 Member Function Documentation

8.54.2.1 float mitkCUDAFullVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.54.2.2 void mitkCUDAFullVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.54.2.3 void mitkCUDAFullVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.54.2.4 float mitkCUDAFullVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.54.2.5 bool mitkCUDAFullVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:

: Return the illustrative shading status.

8.54.2.6 void mitkCUDAFullVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:

x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.54.2.7 bool mitkCUDAFullVRScene::GetPreintegration ()

Get the preintegration status

Returns:

: the preintegration status

8.54.2.8 bool mitkCUDAFullVRScene::GetShade ()

Get the shading status

Returns:
Return the shading status.

8.54.2.9 float mitkCUDAFullVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.54.2.10 float mitkCUDAFullVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.54.2.11 float mitkCUDAFullVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.54.2.12 mitkRGBATransferFunction1D* mitkCUDAFullVRScene::GetTransferFunction ()

Get the transfer function

Returns:
Return the transfer function.

8.54.2.13 virtual void mitkCUDAFullVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.54.2.14 virtual void mitkCUDAFullVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.54.2.15 virtual void mitkCUDAFullVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.54.2.16 virtual void mitkCUDAFullVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.54.2.17 void mitkCUDAFullVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:

Whether turn the light rotation mode on.

8.54.2.18 void mitkCUDAFullVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:

value The ambient factor of the material of the isosurface

8.54.2.19 void mitkCUDAFullVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:

r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.54.2.20 void mitkCUDAFullVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.54.2.21 void mitkCUDAFullVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:

data The data object to be rendered.

8.54.2.22 void mitkCUDAFullVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.54.2.23 void mitkCUDAFullVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.54.2.24 void mitkCUDAFullVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:
shade,: The illustrative shading status.

8.54.2.25 void mitkCUDAFullVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.

g Specify the green component of the light source color.

b Specify the blue component of the light source color.

8.54.2.26 void mitkCUDAFullVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.54.2.27 void mitkCUDAFullVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.54.2.28 void mitkCUDAFullVRScene::SetPreintegration (bool *preInt*)

Set the preintegration status

Parameters:
preInt,: the preintegration status

8.54.2.29 void mitkCUDAFullVRScene::SetShade (bool *shade*)

Turn on or turn off shading.

Parameters:
shade The shading status.

8.54.2.30 void mitkCUDAFullVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.54.2.31 void mitkCUDAFullVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.54.2.32 void mitkCUDAFullVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:
stdSampleDistance The standard sample distance

8.54.2.33 void mitkCUDAFullVRScene::SetTransferFunction (mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

8.54.2.34 void mitkCUDAFullVRScene::ShadeOff ()

Turn off shading.

8.54.2.35 void mitkCUDAFullVRScene::ShadeOn ()

Turn on shading.

The documentation for this class was generated from the following file:

- mitkCUDAFullVRScene.h

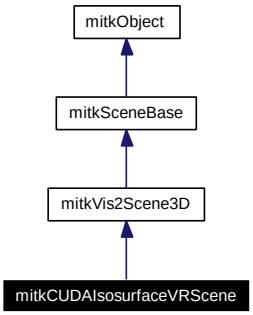
8.55 mitkCUDAIsosurfaceVRScene Class Reference

mitkCUDAIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface.

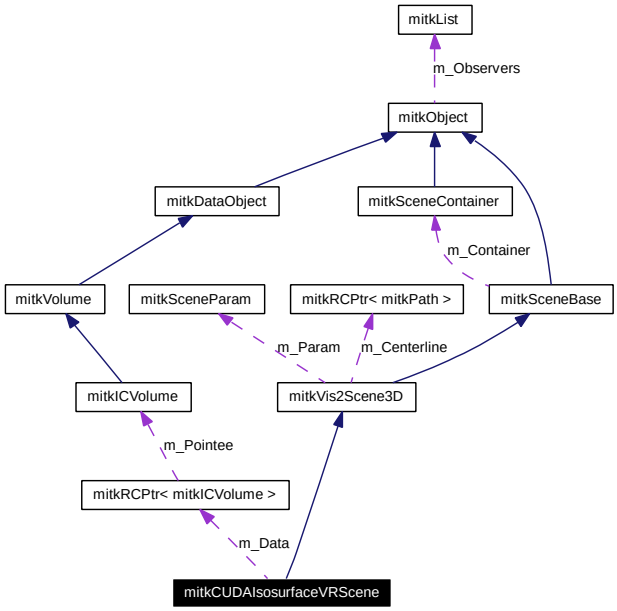
`#include <mitkCUDAIsosurfaceVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAIsosurfaceVRScene:



Collaboration diagram for mitkCUDAIsosurfaceVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- void SetIsovalue (float isovalue)
- float GetIsovalue ()
- void SetSurfaceColor (float r, float g, float b)
- void GetSurfaceColor (float &r, float &g, float &b)
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.55.1 Detailed Description

mitkCUDAIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface.

mitkCUDAIsosurfaceVRScene is a 3D scene to display isosurfaces from volume datasets. Accelerated with the CUDA interface. mitkCUDAIsosurfaceVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.55.2 Member Function Documentation

8.55.2.1 float mitkCUDAIsosurfaceVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.55.2.2 void mitkCUDAIsosurfaceVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:
r Return the red component of the background color.
g Return the green component of the background color.
b Return the blue component of the background color.

8.55.2.3 void mitkCUDAIsosurfaceVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.55.2.4 float mitkCUDAIsosurfaceVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:
Return The diffuse factor of the material of the isosurface

8.55.2.5 bool mitkCUDAIsosurfaceVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:
: Return the illustrative shading status.

8.55.2.6 float mitkCUDAIsosurfaceVRScene::GetIsovalue ()

Get the isovalue of the isosurface to be displayed.

Returns:
Return the isovalue.

8.55.2.7 void mitkCUDAIsosurfaceVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:
x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.55.2.8 float mitkCUDAIsosurfaceVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.55.2.9 float mitkCUDAIsosurfaceVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.55.2.10 void mitkCUDAIsosurfaceVRScene::GetSurfaceColor (float & *r*, float & *g*, float & *b*)

Get the color of the isosurface to be displayed

Parameters:
r Return the red component of the surface color.
g Return the green component of the surface color.
b Return the blue component of the surface color.

8.55.2.11 virtual void mitkCUDAIsosurfaceVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.55.2.12 virtual void mitkCUDAIsosurfaceVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.55.2.13 virtual void mitkCUDAIsosurfaceVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.55.2.14 virtual void mitkCUDAIsosurfaceVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.55.2.15 void mitkCUDAIsosurfaceVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:
Whether turn the light rotation mode on.

8.55.2.16 void mitkCUDAIsosurfaceVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:
value The ambient factor of the material of the isosurface

8.55.2.17 void mitkCUDAIsosurfaceVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:
r Specify the red component of the background color.
g Specify the green component of the background color.
b Specify the blue component of the background color.

8.55.2.18 void mitkCUDAIsosurfaceVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.55.2.19 void mitkCUDAIsosurfaceVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.55.2.20 void mitkCUDAIsosurfaceVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.55.2.21 void mitkCUDAIsosurfaceVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.55.2.22 void mitkCUDAIsosurfaceVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:
shade,: The illustrative shading status.

8.55.2.23 void mitkCUDAIsosurfaceVRScene::SetIsovalue (float *isovalue*)

Set the isovalue of the isosurface to be displayed.

Parameters:
isovalue The isovalue.

8.55.2.24 void mitkCUDAIsosurfaceVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.
g Specify the green component of the light source color.
b Specify the blue component of the light source color.

8.55.2.25 void mitkCUDAIsosurfaceVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
x Specify the x component of the light direction.
y Specify the y component of the light direction.
z Specify the z component of the light direction.

8.55.2.26 void mitkCUDAIsosurfaceVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.55.2.27 void mitkCUDAIsosurfaceVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.55.2.28 void mitkCUDAIsosurfaceVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.55.2.29 void mitkCUDAIsosurfaceVRScene::SetSurfaceColor (float *r*, float *g*, float *b*)

Set the color of the isosurface to be displayed

Parameters:

r Specify the red component of the surface color.

g Specify the green component of the surface color.

b Specify the blue component of the surface color.

The documentation for this class was generated from the following file:

- mitkCUDAIsosurfaceVRScene.h

8.56 mitkCUDALut1D Class Reference

mitkGLSLLut1D - A data class for 1D-LookUPTable,

`#include <mitkCUDALut1D.h>`

8.56.1 Detailed Description

mitkGLSLLut1D - A data class for 1D-LookUPTable,

mitkGLSLLut1D is a data class for 1D-LookUPTable

The documentation for this class was generated from the following file:

- mitkCUDALut1D.h

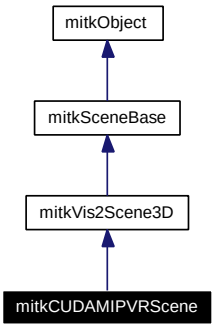
8.57 mitkCUDAMIPVRScene Class Reference

mitkCUDAMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Accelerated with the CUDA interface.

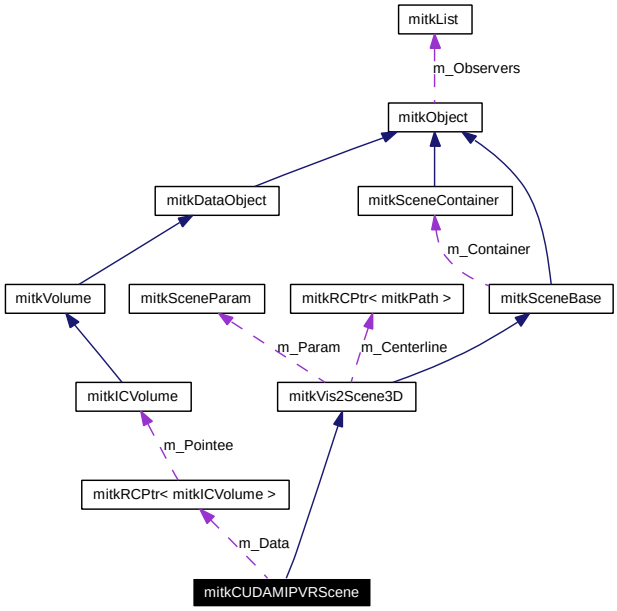
`#include <mitkCUDAMIPVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAMIPVRScene:



Collaboration diagram for mitkCUDAMIPVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetWidthCenter (float windowWidth, float windowCenter)
- void SetWindowWidth (float windowWidth)

- void SetWindowCenter (float windowCenter)
- float GetWindowWidth ()
- float GetWindowCenter ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.57.1 Detailed Description

mitkCUDAMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Accelerated with the CUDA interface.

mitkCUDAMIPVRScene is a 3D scene for MIP Rendering from volume datasets. Accelerated with the CUDA interface. mitkCUDAMIPVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.57.2 Member Function Documentation

8.57.2.1 void mitkCUDAMIPVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.57.2.2 float mitkCUDAMIPVRScene::GetWindowCenter ()

Get the window center for the MIP

Returns:
Return the window center of the MIP

8.57.2.3 float mitkCUDAMIPVRScene::GetWindowWidth ()

Get the window width for the MIP

Returns:
Return the window width of the MIP

8.57.2.4 virtual void mitkCUDAMIPVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.57.2.5 void mitkCUDAMIPVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.57.2.6 void mitkCUDAMIPVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.57.2.7 void mitkCUDAMIPVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.57.2.8 void mitkCUDAMIPVRScene::SetWidthCenter (float *windowWidth*, float *windowCenter*)

Set the window width and windows center for the MIP

Parameters:
windowWidth The window width of the MIP
windowCenter The window center of the MIP

8.57.2.9 void mitkCUDAMIPVRScene::SetWindowCenter (float *windowCenter*)

Set the window center for the MIP

Parameters:
windowCenter The window center of the MIP

8.57.2.10 void mitkCUDAMIPVRScene::SetWindowWidth (float *windowWidth*)

Set the window width for the MIP

Parameters:
windowWidth The window width of the MIP

The documentation for this class was generated from the following file:

- mitkCUDAMIPVRScene.h

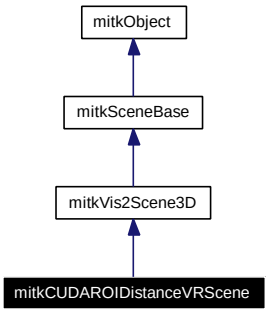
8.58 mitkCUDAROIDistanceVRScene Class Reference

mitkCUDAROIDistanceVRScene - a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface.

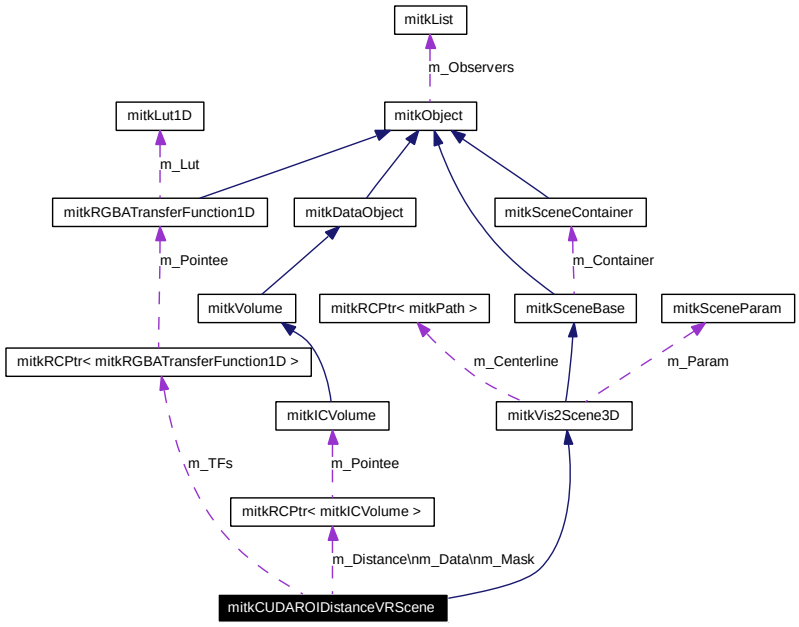
`#include <mitkCUDAROIDistanceVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAROIDistanceVRScene:



Collaboration diagram for mitkCUDAROIDistanceVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetMaskVolume (mitkICVolume *data)
- mitkICVolume * GetMaskVolume ()
- void SetDataModified ()

- void SetMaskModified ()
- void SetCurrentROI (int index)
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)
- void SetBackGroundIsovalue (float isovalue_bg)
- float GetBackGroundIsovalue ()
- void SetDistanceIsovalue (float isoDistance)
- float GetDistanceIsovalue ()
- void SetDistanceSurfaceColor (float r, float g, float b, float a)
- void GetDistanceSurfaceColor (float &r, float &g, float &b, float &a)
- mitkRGBATransferFunction1D * GetTransferFunction (int index)
- void SetTransferFunction (int index, mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.58.1 Detailed Description

mitkCUDAROIDistanceVRScene - a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface.

mitkCUDAROIDistanceVRScene is a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface. mitkCUDAROIDistanceVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.58.2 Member Function Documentation

8.58.2.1 float mitkCUDAROIDistanceVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.58.2.2 void mitkCUDAROIDistanceVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.58.2.3 float mitkCUDAROIDistanceVRScene::GetBackGroundIsovalue ()

Get the background isovalue

Returns:

: the background isovalue

8.58.2.4 void mitkCUDAROIDistanceVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.58.2.5 float mitkCUDAROIDistanceVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.58.2.6 float mitkCUDAROIDistanceVRScene::GetDistanceIsovalue ()

Get the bdistance isovalue

Returns:

: the distance isovalue

8.58.2.7 void mitkCUDAROIDistanceVRScene::GetDistanceSurfaceColor (float & *r*, float & *g*, float & *b*, float & *a*)

Return the distance surface color

Parameters:

r Return the red component of the distance surface color.

g Return the green component of the distance surface color.

b Return the blue component of the distance surface color.

a Return the alpha component of the distance surface color.

8.58.2.8 bool mitkCUDAROIDistanceVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:
: Return the illustrative shading status.

8.58.2.9 void mitkCUDAROIDistanceVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:
x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.58.2.10 mitkICVolume* mitkCUDAROIDistanceVRScene::GetMaskVolume ()

Get the mask of the segmentation of the dataset

Returns:
Return the mask of the segmentation of the dataset

8.58.2.11 float mitkCUDAROIDistanceVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.58.2.12 float mitkCUDAROIDistanceVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.58.2.13 float mitkCUDAROIDistanceVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.58.2.14 mitkRGBATransferFunction1D* mitkCUDAROIDistanceVRScene::GetTransferFunction (int *index*)

Get the transfer function

Returns:
Return the transfer function.

8.58.2.15 virtual void mitkCUDAROIDistanceVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.58.2.16 virtual void mitkCUDAROIDistanceVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.58.2.17 virtual void mitkCUDAROIDistanceVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.58.2.18 virtual void mitkCUDAROIDistanceVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.58.2.19 void mitkCUDAROIDistanceVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:
Whether turn the light rotation mode on.

8.58.2.20 void mitkCUDAROIDistanceVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:
value The ambient factor of the material of the isosurface

8.58.2.21 void mitkCUDAROIDistanceVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:
- ***r*** Specify the red component of the background color.
- ***g*** Specify the green component of the background color.
- ***b*** Specify the blue component of the background color.

8.58.2.22 void mitkCUDAROIDistanceVRScene::SetBackGroundIsovalue (float *isovalue_bg*)

Set the background isovalue

Parameters:
isovalue_bg,: the background isovalue

8.58.2.23 void mitkCUDAROIDistanceVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.58.2.24 void mitkCUDAROIDistanceVRScene::SetCurrentROI (int *index*)

Set the id of current ROI

Parameters:
index,: the id of current ROI

8.58.2.25 void mitkCUDAROIDistanceVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.58.2.26 void mitkCUDAROIDistanceVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.58.2.27 void mitkCUDAROIDistanceVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.58.2.28 void mitkCUDAROIDistanceVRScene::SetDistanceIsovalue (float *isoDistance*)

Set the distance isovalue

Parameters:
isoDistance,: the distance isovalue

8.58.2.29 void mitkCUDAROIDistanceVRScene::SetDistanceSurfaceColor (float *r*, float *g*, float *b*, float *a*)

Set the distance surface color

Parameters:
r Specify the red component of the distance surface color.

g Specify the green component of the distance surface color.

b Specify the blue component of the distance surface color.

a Specify the alpha component of the distance surface color.

8.58.2.30 void mitkCUDAROIDistanceVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:
shade,: The illustrative shading status.

8.58.2.31 void mitkCUDAROIDistanceVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.

g Specify the green component of the light source color.

b Specify the blue component of the light source color.

8.58.2.32 void mitkCUDAROIDistanceVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.58.2.33 void mitkCUDAROIDistanceVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.58.2.34 void mitkCUDAROIDistanceVRScene::SetMaskModified ()

Tell the scene the mask is modified when any modification is done to the mask

8.58.2.35 void mitkCUDAROIDistanceVRScene::SetMaskVolume (mitkICVolume * *data*)

Set the mask of the segmentation of the dataset

Parameters:
mask The mask of the segmentation of the dataset

8.58.2.36 void mitkCUDAROIDistanceVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.58.2.37 void mitkCUDAROIDistanceVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.58.2.38 void mitkCUDAROIDistanceVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:
stdSampleDistance The standard sample distance

8.58.2.39 void mitkCUDAROIDistanceVRScene::SetTransferFunction (int *index*, mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

The documentation for this class was generated from the following file:

- mitkCUDAROIDistanceVRScene.h

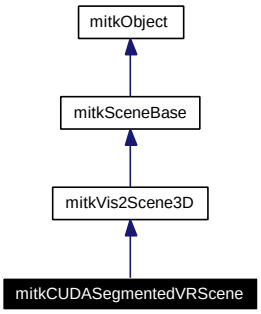
8.59 mitkCUDASegmentedVRScene Class Reference

mitkCUDASegmentedVRScene - a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface.

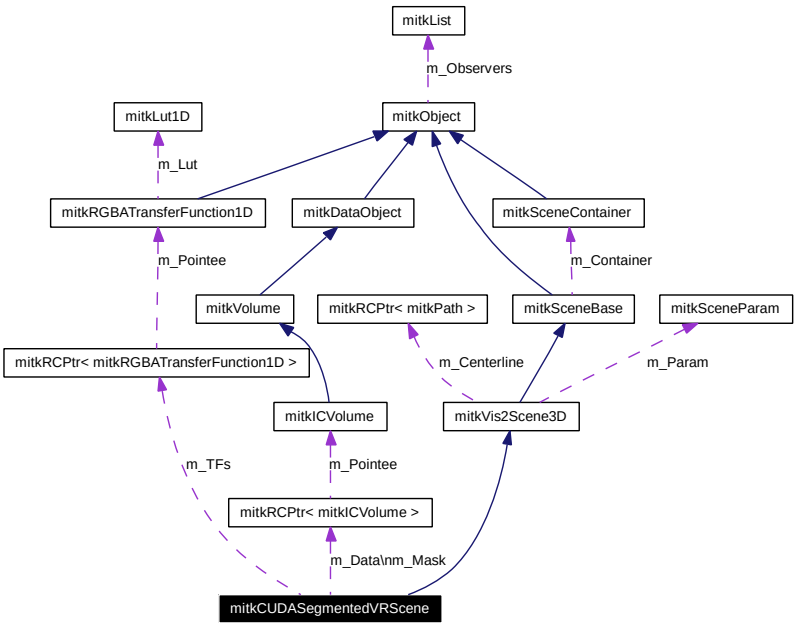
`#include <mitkCUDASegmentedVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDASegmentedVRScene:



Collaboration diagram for mitkCUDASegmentedVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetMaskVolume (mitkICVolume *data)
- mitkICVolume * GetMaskVolume ()
- void SetDataModified ()

- void SetMaskModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)
- mitkRGBATransferFunction1D * GetTransferFunction (int index)
- void SetTransferFunction (int index, mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetShade (bool shade)
- bool GetShade ()
- void ShadeOn ()
- void ShadeOff ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.59.1 Detailed Description

mitkCUDASegmentedVRScene - a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface.

mitkCUDASegmentedVRScene is a 3D scene for shaded semi-transparent rendering from segmented volume datasets. Accelerated with the CUDA interface. mitkCUDASegmentedVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.59.2 Member Function Documentation

8.59.2.1 float mitkCUDASegmentedVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.59.2.2 void mitkCUDASegmentedVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.59.2.3 void mitkCUDASegmentedVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.59.2.4 float mitkCUDASegmentedVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.59.2.5 bool mitkCUDASegmentedVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:

: Return the illustrative shading status.

8.59.2.6 void mitkCUDASegmentedVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:

x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.59.2.7 mitkICVolume* mitkCUDASegmentedVRScene::GetMaskVolume ()

Get the mask of the segmentation of the dataset

Returns:

Return the mask of the segmentation of the dataset

8.59.2.8 bool mitkCUDASegmentedVRScene::GetShade ()

Get the shading status

Returns:
Return the shading status.

8.59.2.9 float mitkCUDASegmentedVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.59.2.10 float mitkCUDASegmentedVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.59.2.11 float mitkCUDASegmentedVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.59.2.12 mitkRGBATransferFunction1D* mitkCUDASegmentedVRScene::GetTransferFunction (int *index*)

Get the transfer function

Returns:
Return the transfer function.

8.59.2.13 virtual void mitkCUDASegmentedVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.59.2.14 virtual void mitkCUDASegmentedVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.59.2.15 virtual void mitkCUDASegmentedVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.59.2.16 virtual void mitkCUDASegmentedVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.59.2.17 void mitkCUDASegmentedVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:

Whether turn the light rotation mode on.

8.59.2.18 void mitkCUDASegmentedVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:

value The ambient factor of the material of the isosurface

8.59.2.19 void mitkCUDASegmentedVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:

r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.59.2.20 void mitkCUDASegmentedVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.59.2.21 void mitkCUDASegmentedVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:

data The data object to be rendered.

8.59.2.22 void mitkCUDASegmentedVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.59.2.23 void mitkCUDASegmentedVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.59.2.24 void mitkCUDASegmentedVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:
shade,: The illustrative shading status.

8.59.2.25 void mitkCUDASegmentedVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.
g Specify the green component of the light source color.
b Specify the blue component of the light source color.

8.59.2.26 void mitkCUDASegmentedVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
x Specify the x component of the light direction.
y Specify the y component of the light direction.
z Specify the z component of the light direction.

8.59.2.27 void mitkCUDASegmentedVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.59.2.28 void mitkCUDASegmentedVRScene::SetMaskModified ()

Tell the scene the mask is modified when any modification is done to the mask

8.59.2.29 void mitkCUDASegmentedVRScene::SetMaskVolume (mitkICVolume * *data*)

Set the mask of the segmentation of the dataset

Parameters:
mask The mask of the segmentation of the dataset

8.59.2.30 void mitkCUDASegmentedVRScene::SetShade (bool *shade*)

Turn on or turn off shading.

Parameters:
shade The shading status.

8.59.2.31 void mitkCUDASegmentedVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.59.2.32 void mitkCUDASegmentedVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.59.2.33 void mitkCUDASegmentedVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:
stdSampleDistance The standard sample distance

8.59.2.34 void mitkCUDASegmentedVRScene::SetTransferFunction (int *index*, mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

8.59.2.35 void mitkCUDASegmentedVRScene::ShadeOff ()

Turn off shading.

8.59.2.36 void mitkCUDASegmentedVRScene::ShadeOn ()

Turn on shading.

The documentation for this class was generated from the following file:

- mitkCUDASegmentedVRScene.h

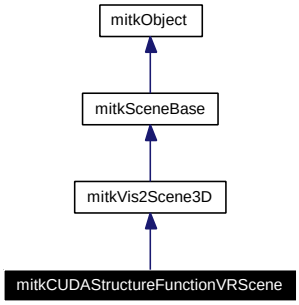
8.60 mitkCUDAStructureFunctionVRScene Class Reference

mitkCUDAStructureFunctionVRScene - a 3D scene for rendering a structure volume together with a function volume. Accelerated with the CUDA interface.

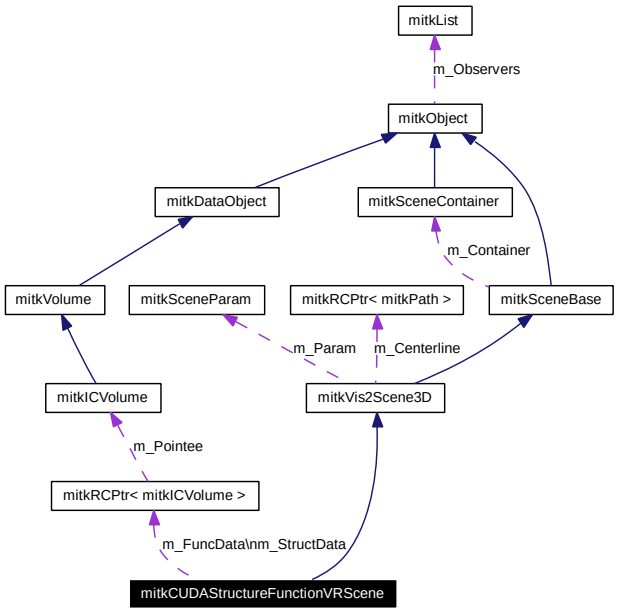
`#include <mitkCUDAStructureFunctionVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAStructureFunctionVRScene:



Collaboration diagram for mitkCUDAStructureFunctionVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetStructData (mitkICVolume *data)
- void SetStructDataModified ()
- void SetFuncData (mitkICVolume *data)
- void SetFuncDataModified ()

- void SetTransformFunc (bool trans)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetStruct2FuncMatrix (const mitkVis2Matrix4x4 &matrix)
- const mitkVis2Matrix4x4 & GetStruct2FuncMatrix () const
- void SetStructRange (float StructMin, float StructMax)
- void GetStructRange (float &StructMin, float &StructMax)
- void SetFuncRange (float FuncMin, float FuncMax)
- void GetFuncRange (float &FuncMin, float &FuncMax)
- void SetStructOpacity (float StructOpacity)
- float GetStructOpacity ()
- void SetFuncIntensity (float FuncIntensity)
- float GetFuncIntensity ()
- void SetSampleDistance (float sd)
- float GetSampleDistance ()

8.60.1 Detailed Description

mitkCUDAStructureFunctionVRScene - a 3D scene for rendering a structure volume together with a function volume. Accelerated with the CUDA interface.

mitkCUDAStructureFunctionVRScene is a 3D scene for rendering a structure volume together with a function volume. Accelerated with the CUDA interface. mitkCUDAStructureFunctionVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.60.2 Member Function Documentation

8.60.2.1 float mitkCUDAStructureFunctionVRScene::GetFuncIntensity ()

Get the intensity for function rendering

Returns:
: the intensity for function rendering

8.60.2.2 void mitkCUDAStructureFunctionVRScene::GetFuncRange (float & *FuncMin*, float & *FuncMax*)

Get the data range of function data

Parameters:
FuncMin,: minimum value of function data
FuncMax,: maximum value of function data

8.60.2.3 float mitkCUDAStructureFunctionVRScene::GetSampleDistance ()

Get the sampling distance of ray traversal.

Returns:
Return the sampling distance of ray traversal.

8.60.2.4 const mitkVis2Matrix4x4& mitkCUDAStructureFunctionVRScene::GetStruct2FuncMatrix () const

Get the transform matrix for function matrix registration

Parameters:
matrix,: the transform matrix for function matrix registration

8.60.2.5 float mitkCUDAStructureFunctionVRScene::GetStructOpacity ()

Get the opacity for structure rendering

Returns:
: the opacity for structure rendering

8.60.2.6 void mitkCUDAStructureFunctionVRScene::GetStructRange (float & *StructMin*, float & *StructMax*)

Get the data range of structure data

Parameters:
StructMin,: minimum value of structure data
StructMax,: maximum value of structure data

8.60.2.7 virtual void mitkCUDAStructureFunctionVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.60.2.8 virtual void mitkCUDAStructureFunctionVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.60.2.9 virtual void mitkCUDAStructureFunctionVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.60.2.10 virtual void mitkCUDAStructureFunctionVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.60.2.11 void mitkCUDAStructureFunctionVRScene::SetFuncData (mitkICVolume * *data*)

Set the function data object to be rendered

Parameters:
data The function data object to be rendered.

8.60.2.12 void mitkCUDAStructureFunctionVRScene::SetFuncDataModified ()

Tell the scene the function data is modified when any modification is done to the data object

8.60.2.13 void mitkCUDAStructureFunctionVRScene::SetFuncIntensity (float *FuncIntensity*)

Set the intensity for function rendering

Parameters:
FuncIntensity,: the intensity for function rendering

8.60.2.14 void mitkCUDAStructureFunctionVRScene::SetFuncRange (float *FuncMin*, float *FuncMax*)

Set the data range of function data

Parameters:
FuncMin,: minimum value of function data
FuncMax,: maximum value of function data

8.60.2.15 void mitkCUDAStructureFunctionVRScene::SetSampleDistance (float *sd*)

Set the sampling distance of ray traversal.

Parameters:
sd The sampling distance of ray traversal.

8.60.2.16 void mitkCUDAStructureFunctionVRScene::SetStruct2FuncMatrix (const mitkVis2Matrix4x4 & *matrix*)

Set the transform matrix for function matrix registration

Parameters:
matrix,: the transform matrix for function matrix registration

8.60.2.17 void mitkCUDAStructureFunctionVRScene::SetStructData (mitkICVolume * *data*)

Set the structure data object to be rendered

Parameters:
data The structure data object to be rendered.

8.60.2.18 void mitkCUDAStructureFunctionVRScene::SetStructDataModified ()

Tell the scene the structure data is modified when any modification is done to the data object

8.60.2.19 void mitkCUDAStructureFunctionVRScene::SetStructOpacity (float *StructOpacity*)

Set the opacity for structure rendering

Parameters:

StructOpacity,: the opacity for structure rendering

8.60.2.20 void mitkCUDAStructureFunctionVRScene::SetStructRange (float *StructMin*, float *StructMax*)

Set the data range of structure data

Parameters:

StructMin,: minimum value of structure data

StructMax,: maximum value of structure data

8.60.2.21 void mitkCUDAStructureFunctionVRScene::SetTransformFunc (bool *trans*) [inline]

Transform the function data for registration

Parameters:

trans The function data

The documentation for this class was generated from the following file:

- mitkCUDAStructureFunctionVRScene.h

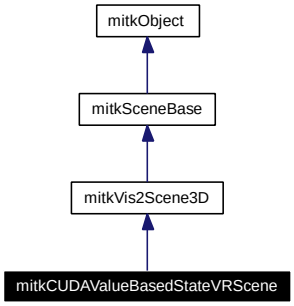
8.61 mitkCUDAValueBasedStateVRScene Class Reference

mitkCUDADeltaAccStateVRScene - a 3D scene for volume rendering with value based ray feature analysis. Accelerated with the CUDA interface.

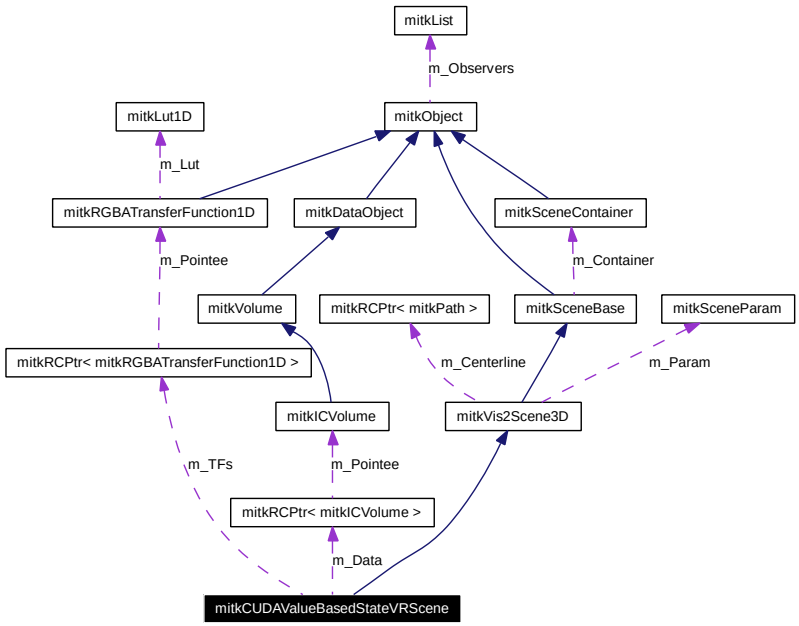
`#include <mitkCUDAValueBasedStateVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAValueBasedStateVRScene:



Collaboration diagram for mitkCUDAValueBasedStateVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- void SetMaxLayer (int maxLayer)
- int GetMaxLayer ()
- void SetStateController (int id, int less_more, float value)
- void GetStateController (int id, int &less_more, float &value)
- void SetThicknessThreshold (float thincknessThresh)
- float GetThicknessThreshold ()
- mitkRGBATransferFunction1D * GetTransferFunction (int index)
- void SetTransferFunction (int index, mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetShade (bool shade)
- bool GetShade ()
- void ShadeOn ()
- void ShadeOff ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetLayerShadingMode (int layer, ShadingMode mode)
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])
- void SetBackwardsTest (bool test)
- bool GetBackwardsTest ()

8.61.1 Detailed Description

mitkCUDADeltaAccStateVRScene - a 3D scene for volume rendering with value based ray feature analysis. Accelerated with the CUDA interface.

mitkCUDADeltaAccStateVRScene is a 3D scene for volume rendering with value based ray feature analysis. Accelerated with the CUDA interface.

8.61.2 Member Function Documentation

8.61.2.1 float mitkCUDAValueBasedStateVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.61.2.2 void mitkCUDAValueBasedStateVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:
r Return the red component of the background color.
g Return the green component of the background color.
b Return the blue component of the background color.

8.61.2.3 bool mitkCUDAValueBasedStateVRScene::GetBackwardsTest () [inline]

Get the backwards test status

Returns:
: the backwards test status

8.61.2.4 void mitkCUDAValueBasedStateVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.61.2.5 float mitkCUDAValueBasedStateVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:
Return The diffuse factor of the material of the isosurface

8.61.2.6 bool mitkCUDAValueBasedStateVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:
: Return the illustrative shading status.

8.61.2.7 void mitkCUDAValueBasedStateVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:

x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.61.2.8 int mitkCUDAValueBasedStateVRScene::GetMaxLayer ()

Get the maximum layer number

Returns:

: the maximum layer number

8.61.2.9 bool mitkCUDAValueBasedStateVRScene::GetShade ()

Get the shading status

Returns:

Return the shading status.

8.61.2.10 float mitkCUDAValueBasedStateVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:

Return The specular factor of the material of the isosurface

8.61.2.11 float mitkCUDAValueBasedStateVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:

Return The specular power of the material of the isosurface

8.61.2.12 void mitkCUDAValueBasedStateVRScene::GetStateController (int *id*, int & *less_more*, float & *value*)

Get the ith state controller

Parameters:

id,: layer id

less_more,: direction of inequality

value,: threshold value

8.61.2.13 float mitkCUDAValueBasedStateVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.61.2.14 float mitkCUDAValueBasedStateVRScene::GetThicknessThreshold ()

Get the thickness threshold value

Returns:
: the thickness threshold value

8.61.2.15 mitkRGBATransferFunction1D* mitkCUDAValueBasedStateVRScene::GetTransferFunction (int *index*)

Get the transfer function

Returns:
Return the transfer function.

8.61.2.16 virtual void mitkCUDAValueBasedStateVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.61.2.17 virtual void mitkCUDAValueBasedStateVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.61.2.18 virtual void mitkCUDAValueBasedStateVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.61.2.19 virtual void mitkCUDAValueBasedStateVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.61.2.20 void mitkCUDAValueBasedStateVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:
Whether turn the light rotation mode on.

8.61.2.21 void mitkCUDAValueBasedStateVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:
value The ambient factor of the material of the isosurface

8.61.2.22 void mitkCUDAValueBasedStateVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:
r Specify the red component of the background color.
g Specify the green component of the background color.
b Specify the blue component of the background color.

8.61.2.23 void mitkCUDAValueBasedStateVRScene::SetBackwardsTest (bool *test*) [inline]

Set the backwards test status

Parameters:
test,: the backwards test status

8.61.2.24 void mitkCUDAValueBasedStateVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.61.2.25 void mitkCUDAValueBasedStateVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.61.2.26 void mitkCUDAValueBasedStateVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.61.2.27 void mitkCUDAValueBasedStateVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:

value The diffuse factor of the material of the isosurface

8.61.2.28 void mitkCUDAValueBasedStateVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:

shade,: The illustrative shading status.

8.61.2.29 void mitkCUDAValueBasedStateVRScene::SetLayerShadingMode (int *layer*, ShadingMode *mode*)

Set shading mode of a layer

Parameters:

layer,: the id of the layer

mode,: the shading mode

8.61.2.30 void mitkCUDAValueBasedStateVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:

r Specify the red component of the light source color.

g Specify the green component of the light source color.

b Specify the blue component of the light source color.

8.61.2.31 void mitkCUDAValueBasedStateVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:

x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.61.2.32 void mitkCUDAValueBasedStateVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.61.2.33 void mitkCUDAValueBasedStateVRScene::SetMaxLayer (int *maxLayer*)

Set the maximum layer number

Parameters:
maxLayer,: the maximum layer number

8.61.2.34 void mitkCUDAValueBasedStateVRScene::SetShade (bool *shade*)

Turn on or turn off shading.

Parameters:
shade The shading status.

8.61.2.35 void mitkCUDAValueBasedStateVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.61.2.36 void mitkCUDAValueBasedStateVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.61.2.37 void mitkCUDAValueBasedStateVRScene::SetStateController (int *id*, int *less_more*, float *value*)

Set the ith state controller

Parameters:
id,: layer id

less_more,: direction of inequality

value,: threshold value

8.61.2.38 void mitkCUDAValueBasedStateVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:
stdSampleDistance The standard sample distance

8.61.2.39 void mitkCUDAValueBasedStateVRScene::SetThicknessThreshold (float *thincknessThresh*)

Set the thickness threshold value

Parameters:
thincknessThresh,: the thickness threshold value

8.61.2.40 void mitkCUDAValueBasedStateVRScene::SetTransferFunction (int *index*, mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

8.61.2.41 void mitkCUDAValueBasedStateVRScene::ShadeOff ()

Turn off shading.

8.61.2.42 void mitkCUDAValueBasedStateVRScene::ShadeOn ()

Turn on shading.

The documentation for this class was generated from the following file:

- mitkCUDAValueBasedStateVRScene.h

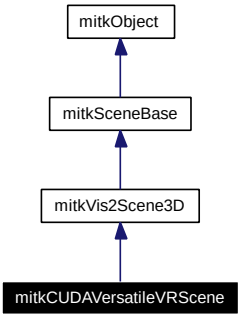
8.62 mitkCUDAVersatileVRScene Class Reference

mitkCUDAVersatileVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface.

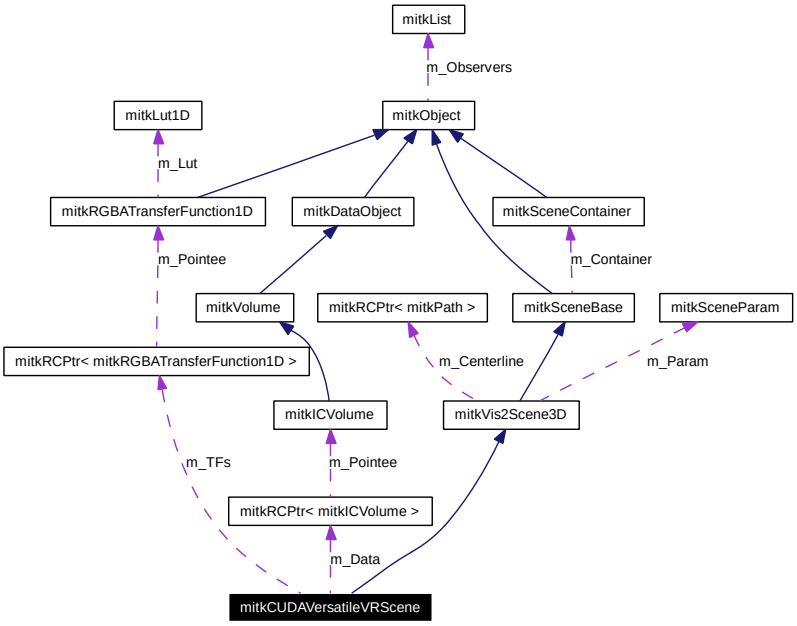
`#include <mitkCUDAVersatileVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkCUDAVersatileVRScene:



Collaboration diagram for mitkCUDAVersatileVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- mitkRGBATransferFunction1D * GetTransferFunction (int index)
- void SetTransferFunction (int index, mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])
- void SetPreintegration (bool preInt)
- bool GetPreintegration ()

8.62.1 Detailed Description

mitkCUDAVersatileVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface.

mitkCUDAVersatileVRScene is a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the CUDA interface. mitkCUDAVersatileVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.62.2 Member Function Documentation

8.62.2.1 float mitkCUDAVersatileVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.62.2.2 void mitkCUDAVersatileVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:
r Return the red component of the background color.
g Return the green component of the background color.
b Return the blue component of the background color.

8.62.2.3 void mitkCUDAVersatileVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.62.2.4 float mitkCUDAVersatileVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:
Return The diffuse factor of the material of the isosurface

8.62.2.5 void mitkCUDAVersatileVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:
x Return the x component of the light direction.
y Return the y component of the light direction.
z Return the z component of the light direction.

8.62.2.6 bool mitkCUDAVersatileVRScene::GetPreintegration ()

Get the preintegration status

Returns:
: the preintegration status

8.62.2.7 float mitkCUDAVersatileVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.62.2.8 float mitkCUDAVersatileVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.62.2.9 float mitkCUDAVersatileVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.62.2.10 mitkRGBATransferFunction1D* mitkCUDAVersatileVRScene::GetTransferFunction (int *index*)

Get the transfer function

Returns:
Return the transfer function.

8.62.2.11 virtual void mitkCUDAVersatileVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.62.2.12 virtual void mitkCUDAVersatileVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.62.2.13 virtual void mitkCUDAVersatileVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.62.2.14 virtual void mitkCUDAVersatileVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.62.2.15 void mitkCUDAVersatileVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:
Whether turn the light rotation mode on.

8.62.2.16 void mitkCUDAVersatileVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:
value The ambient factor of the material of the isosurface

8.62.2.17 void mitkCUDAVersatileVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:
r Specify the red component of the background color.
g Specify the green component of the background color.
b Specify the blue component of the background color.

8.62.2.18 void mitkCUDAVersatileVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.62.2.19 void mitkCUDAVersatileVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.62.2.20 void mitkCUDAVersatileVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.62.2.21 void mitkCUDAVersatileVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.62.2.22 void mitkCUDAVersatileVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.
g Specify the green component of the light source color.
b Specify the blue component of the light source color.

8.62.2.23 void mitkCUDAVersatileVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:

x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.62.2.24 void mitkCUDAVersatileVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:

intensity Light's intensity

8.62.2.25 void mitkCUDAVersatileVRScene::SetPreintegration (bool *preInt*)

Set the preintegration status

Parameters:

preInt,: the preintegration status

8.62.2.26 void mitkCUDAVersatileVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:

value The specular factor of the material of the isosurface

8.62.2.27 void mitkCUDAVersatileVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:

value The specular power of the material of the isosurface

8.62.2.28 void mitkCUDAVersatileVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:

stdSampleDistance The standard sample distance

8.62.2.29 void mitkCUDAVersatileVRScene::SetTransferFunction (int *index*, mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

The documentation for this class was generated from the following file:

- mitkCUDAVersatileVRScene.h

8.63 mitkDataModel Class Reference

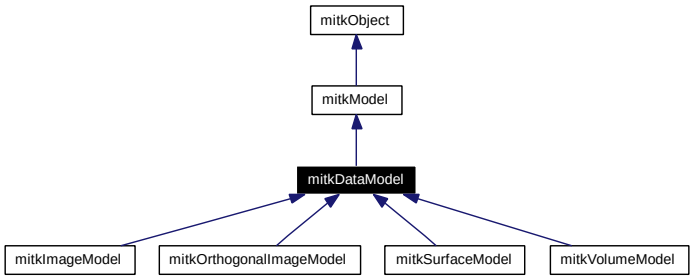
mitkDataModel - abstract class used to represent an data entity in a rendering scene

`#include <mitkDataModel.h>`

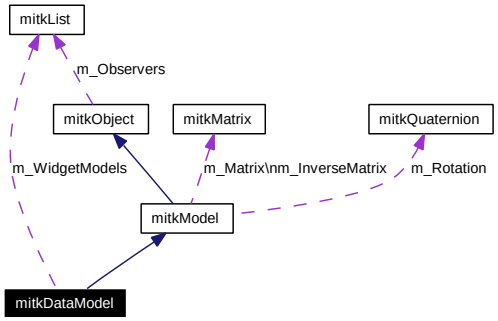
Inherits mitkModel.

Inherited by mitkImageModel, mitkOrthogonalImageModel, mitkSurfaceModel, and mitkVolumeModel.

Inheritance diagram for mitkDataModel:



Collaboration diagram for mitkDataModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddWidget (mitkWidgetModel *widget)
- void UpdateWidgets ()
- void RemoveWidget (mitkWidgetModel *widget)
- void RemoveAllWidgets ()
- int GetWidgetCount ()
- mitkWidgetModel * GetWidgetModel (int index)
- bool IsEmpty ()

8.63.1 Detailed Description

mitkDataModel - abstract class used to represent an data entity in a rendering scene

mitkDataModel is an abstract class used to represent an entity (e.g. a surface or a volume) in a rendering scene.

8.63.2 Member Function Documentation

8.63.2.1 void mitkDataModel::AddWidget (mitkWidgetModel * *widget*)

Attach a widget to this data model.

Parameters:
widget pointer to an mitkWidgetModel to attach

8.63.2.2 int mitkDataModel::GetWidgetCount ()

Get the count of the widget models attached to this data model.

Returns:
Return the count of the widget models attached to this data model.

8.63.2.3 mitkWidgetModel* mitkDataModel::GetWidgetModel (int *index*)

Get the pointer to the widget model object indicated by index.

Returns:
Return the pointer to the widget model object, NULL if not found.

8.63.2.4 bool mitkDataModel::IsEmpty () `[inline]`

Whether is this data model empty (i.e. no data has been set to this model).

Returns:
Return true if no data has been set to this model.

8.63.2.5 virtual void mitkDataModel::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkModel.

Reimplemented in mitkImageModel, mitkOrthogonalImageModel, mitkSurfaceModel, and mitkVolumeModel.

8.63.2.6 void mitkDataModel::RemoveAllWidgets ()

Detach all widgets from this data model.

8.63.2.7 void mitkDataModel::RemoveWidget (mitkWidgetModel * *widget*)

Detach a widget from this data model.

Parameters:
widget pointer to a mitkWidgetModel to detach

8.63.2.8 void mitkDataModel::UpdateWidgets ()

Update the parameters of the widgets attached to this data model when the data has been changed.

The documentation for this class was generated from the following file:

- mitkDataModel.h

8.64 mitkDataObject Class Reference

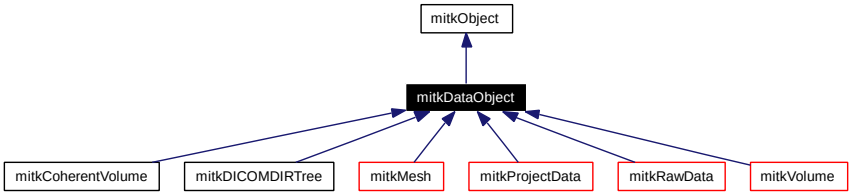
mitkDataObject - an abstract class to represents a data object in MITK

`#include <mitkDataObject.h>`

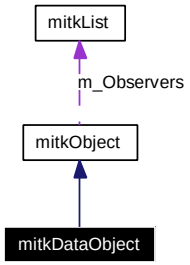
Inherits mitkObject.

Inherited by mitkCoherentVolume, mitkDICOMDIRTree, mitkMesh, mitkProjectData, mitkRawData, mitkTetra, and mitkVolume.

Inheritance diagram for mitkDataObject:



Collaboration diagram for mitkDataObject:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()=0
- virtual int GetDataObjectType () const
- virtual unsigned long long GetActualMemorySize () const =0
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0

8.64.1 Detailed Description

mitkDataObject - an abstract class to represents a data object in MITK

mitkDataObject is an abstract class to represents a data object in MITK. And in MITK, you have only two kinds of data object to deal with.

See also:
mitkVolume
mitkMesh

8.64.2 Member Function Documentation

8.64.2.1 virtual void mitkDataObject::DeepCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implemented in mitkDICOMDIRTree, mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, mitkVolume, mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, mitkConeBeamProjectData, mitkConeBeamRealTimePD, mitkFanBeamProjectData, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkParallelProjectData, mitkRawConeBeamProjectData, mitkTrackedBscanData, mitkTrackedBscanICData, mitkTrackedBscanOoCData, and mitkCoherentVolume.

8.64.2.2 virtual unsigned long long mitkDataObject::GetActualMemorySize () const [pure virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Note:
Pure virtual function. Its concrete subclass must implement this function and return its memory size.

Implemented in mitkDICOMDIRTree, mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkOoCTriangleMesh, mitkOoCVolume, mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, mitkConeBeamProjectData, mitkConeBeamRealTimePD, mitkFanBeamProjectData, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkRawConeBeamProjectData, mitkTrackedBscanData, mitkTrackedBscanICData, mitkTrackedBscanOoCData, and mitkCoherentVolume.

8.64.2.3 virtual int mitkDataObject::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_DATA_OBJECT

Note:
Pure virtual function. Its concrete subclass must implement this function and return its data object type.

Reimplemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, mitkVolume, mitkOoCParallelProjectData, mitkParallelProjectData, mitkRawData, mitkTrackedBscanData, mitkTrackedBscanICData, mitkTrackedBscanOoCData, and mitkCoherentVolume.

8.64.2.4 virtual void mitkDataObject::Initialize () [pure virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Implemented in mitkDICOMDIRTree, mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkHETriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, mitkVolume, mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, mitkConeBeamProjectData, mitkConeBeamRealTimePD, mitkCTProjectionData, mitkFanBeamProjectData, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkParallelProjectData, mitkProjectData, mitkRawConeBeamProjectData, mitkTrackedBscanData, mitkTrackedBscanICData, mitkTrackedBscanOoCData, and mitkCoherentVolume.

8.64.2.5 virtual void mitkDataObject::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkHETriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, mitkVolume, mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, mitkConeBeamProjectData, mitkConeBeamRealTimePD, mitkCTProjectionData, mitkFanBeamProjectData, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkParallelProjectData, mitkProjectData, mitkRawConeBeamProjectData, mitkRawData, mitkTrackedBscanData, mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.64.2.6 virtual void mitkDataObject::ShallowCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implemented in mitkDICOMDIRTree, mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, mitkVolume, mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, mitkConeBeamProjectData, mitkConeBeamRealTimePD, mitkFanBeamProjectData, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkParallelProjectData, mitkRawConeBeamProjectData, mitkTrackedBscanData, mitkTrackedBscanICData, mitkTrackedBscanOoCData, and mitkCoherentVolume.

The documentation for this class was generated from the following file:

- mitkDataObject.h

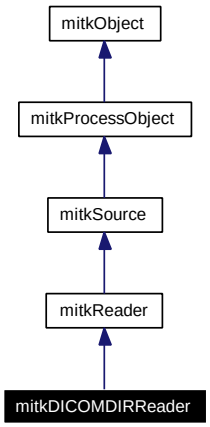
8.65 mitkDICOMDIRReader Class Reference

mitkDICOMDIRReader - A reader class that reads a DICOMDIR file from disk

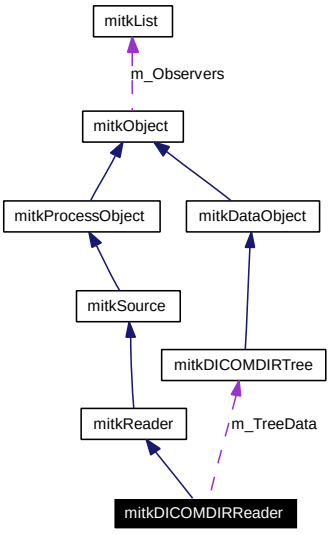
`#include <mitkDICOMDIRReader.h>`

Inherits mitkReader.

Inheritance diagram for mitkDICOMDIRReader:



Collaboration diagram for mitkDICOMDIRReader:



Public Member Functions

- virtual mitkDICOMDIRTree * GetOutput ()

8.65.1 Detailed Description

mitkDICOMDIRReader - A reader class that reads a DICOMDIR file from disk

mitkDICOMDIRReader is a reader class that reads a DICOMDIR file from disk. The result is stored as mitkDICOMDIRTree

8.65.2 Member Function Documentation

8.65.2.1 virtual mitkDICOMDIRTree* mitkDICOMDIRReader::GetOutput () [virtual]

Get the output data

Returns:
Return the output data

The documentation for this class was generated from the following file:

- mitkDICOMDIRReader.h

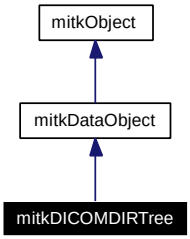
8.66 mitkDICOMDIRTree Class Reference

mitkDICOMDIRTree - Data Object that stores the DICOMDIR structure of a data base

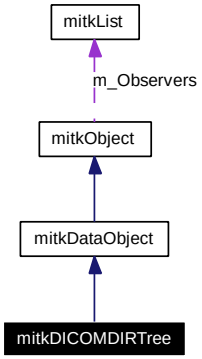
`#include <mitkDICOMDIRTree.h>`

Inherits mitkDataObject.

Inheritance diagram for mitkDICOMDIRTree:



Collaboration diagram for mitkDICOMDIRTree:



Public Member Functions

- virtual void Initialize ()

  Delete the allocated memory (if any) and initialize to default status.

- virtual void DeepCopy (mitkDataObject *src)
- virtual void ShallowCopy (mitkDataObject *src)
- virtual unsigned long long GetActualMemorySize () const
- void SetTreeData (DcmDirectoryModel *data)

8.66.1 Detailed Description

mitkDICOMDIRTree - Data Object that stores the DICOMDIR structure of a data base

mitkDICOMDIRTree is a data Object class that stores the DICOMDIR structure of a data base

8.66.2 Member Function Documentation

8.66.2.1 virtual void mitkDICOMDIRTree::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Implements mitkDataObject.

8.66.2.2 virtual unsigned long long mitkDICOMDIRTree::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by the volume data. The unit is BYTE.

Returns:
Return the actual memory size occupied by this volume. The unit is BYTE.

Implements mitkDataObject.

8.66.2.3 void mitkDICOMDIRTree::SetTreeData (DcmDirectoryModel * *data*)

Called internally by the reader to setup the data

8.66.2.4 virtual void mitkDICOMDIRTree::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Implements mitkDataObject.

The documentation for this class was generated from the following file:

- mitkDICOMDIRTree.h

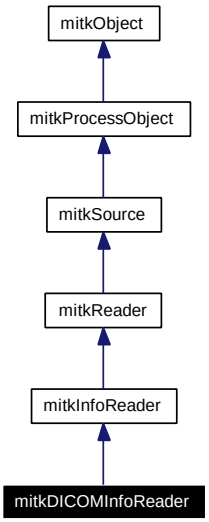
8.67 mitkDICOMInfoReader Class Reference

mitkDICOMInfoReader - a concrete reader for reading element information in DICOM files

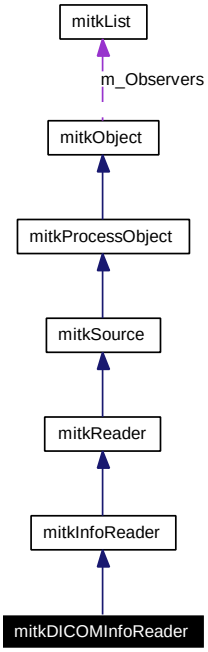
`#include <mitkDICOMInfoReader.h>`

Inherits mitkInfoReader.

Inheritance diagram for mitkDICOMInfoReader:



Collaboration diagram for mitkDICOMInfoReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- bool GetDataElement (unsigned long tag, DICOMELEMENT &element)

Static Public Member Functions

- static const char * GetDescription (unsigned long tag)

8.67.1 Detailed Description

mitkDICOMInfoReader - a concrete reader for reading element information in DICOM files

mitkDICOMInfoReader is a concrete reader for reading element information in DICOM files To use this reader, the code snippet is:

```
mitkDICOMInfoReader *aReader = new mitkDICOMInfoReader;
aReader->AddFileName(filename); //Only one file will be processed at a time
if (aReader->Run())
{
    DICOMELEMENT aElement;
    if (aReader->GetDataElement(somedicomtag, aElement)
    {
        Using aElement
    }
}
```

8.67.2 Member Function Documentation

8.67.2.1 bool mitkDICOMInfoReader::GetDataElement (unsigned long *tag*, DICOMELEMENT & *element*)

Get data element in the DICOM file.

Parameters:

tag the tag of the data element to get from file

element the DICOMELEMENT structure to contain the data element found from the file (if the element has not been found, the value is unchanged)

Returns:

Return true if the data element is found in the file.

8.67.2.2 static const char* mitkDICOMInfoReader::GetDescription (unsigned long *tag*) [static]

Get the description of the tag.

Parameters:

tag the tag value

Returns:

Return the string of description of the tag. Return NULL if the description of the tag has not been found.

8.67.2.3 virtual void mitkDICOMInfoReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkInfoReader.

The documentation for this class was generated from the following file:

- mitkDICOMInfoReader.h

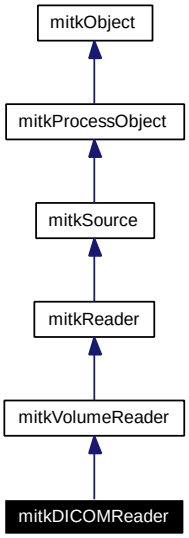
8.68 mitkDICOMReader Class Reference

mitkDICOMReader - a concrete reader for reading DICOM image files

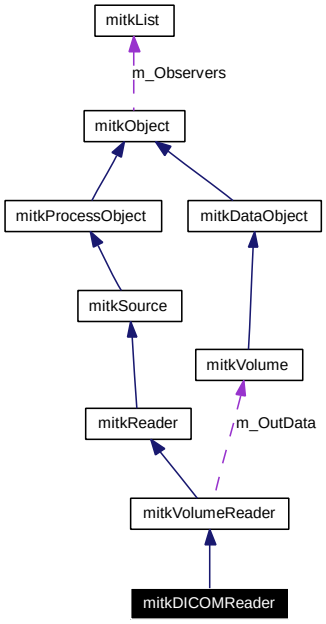
`#include <mitkDICOMReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkDICOMReader:



Collaboration diagram for mitkDICOMReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.68.1 Detailed Description

mitkDICOMReader - a concrete reader for reading DICOM image files

mitkDICOMReader reads a set of DICOM image files to a volume. To use this reader, the code snippet is:

```
mitkDICOMReader *aReader = new mitkDICOMReader;
aReader->AddFileName(file1);
aReader->AddFileName(file2);
...  ...  ...
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

Warning:
All of the images must have equal width and height. Otherwise they can't form a volume. The files which have different width or height will be discarded. Now MITK only partly supports the DICOM 3.0 standard.

8.68.2 Member Function Documentation

8.68.2.1 virtual void mitkDICOMReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

The documentation for this class was generated from the following file:

- mitkDICOMReader.h

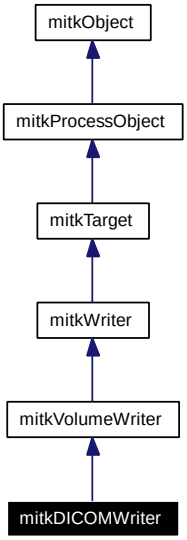
8.69 mitkDICOMWriter Class Reference

mitkDICOMWriter - a concrete writer for writing a volume to DICOM image files

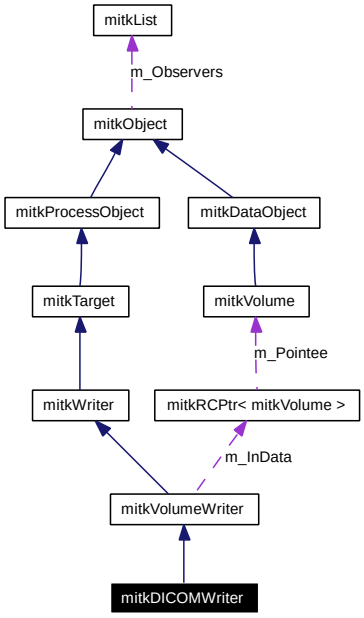
`#include <mitkDICOMWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkDICOMWriter:



Collaboration diagram for mitkDICOMWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetStudyUID (const string &uid)
- void SetSeriesUID (const string &uid)

8.69.1 Detailed Description

mitkDICOMWriter - a concrete writer for writing a volume to DICOM image files

mitkDICOMWriter writes a volume to a set of DICOM image files. Because the volume is a 3D dataset, it may contain many slices. So the file names must be generated and passed to writer properly.

DICOM files to be writen will take format as follows:

Media Storage SOP Class : Not Specified (UID = "");

Media Storage SOP Instance : Not Specified (UID = "");

Transfer Syntax : Explicit VR Little Endian (UID = "1.2.840.10008.1.2.1");

Implementation Class : Not Specified (UID = "");

Implementation Version Name : Ignored;

Source Application Entity Title : Ignored;

Private Information Creator UID and Private Information : Ignored;

Each file will contain one single image of an image series.

To use this writer, the code snippet is:

```
mitkDICOMWriter *aWriter = new mitkDICOMWriter;
aWriter->SetInput(aVolume);
int imageNum = aVolume->GetImageNum();
Gerenate file names into files[imageNum];
for(int i = 0; i < imageNum; i++)
    aWriter->AddFileName(files[i]);
aWriter->Run();
```

8.69.2 Member Function Documentation

8.69.2.1 virtual void mitkDICOMWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

8.69.2.2 void mitkDICOMWriter::SetSeriesUID (const string & *uid*)

Set Series UID.

Parameters:
uid a const reference to a string contains UID

8.69.2.3 void mitkDICOMWriter::SetStudyUID (const string & *uid*)

Set Study UID.

Parameters:
uid a const reference to a string contains UID

The documentation for this class was generated from the following file:

- mitkDICOMWriter.h

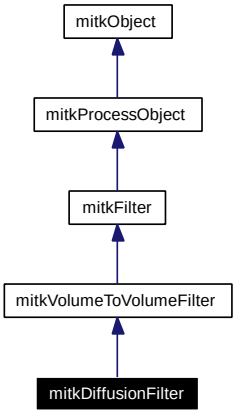
8.70 mitkDiffusionFilter Class Reference

mitkDiffusionFilter - a class used to diffuse the mitkVolume data (2D or 3D)

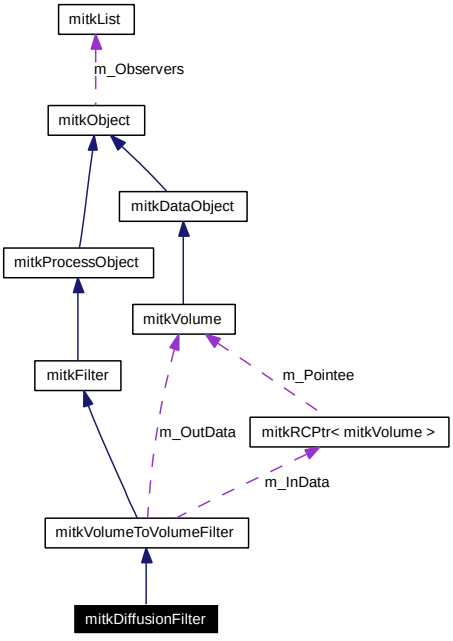
`#include <mitkDiffusionFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkDiffusionFilter:



Collaboration diagram for mitkDiffusionFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void Setm_K (double param=200)

- void SetTimeInterval (float dt=0.20)
- void SetDiffusionType (bool Is3D=false)

8.70.1 Detailed Description

mitkDiffusionFilter - a class used to diffuse the mitkVolume data (2D or 3D)

mitkDiffusionFilter- a filter class derived from mitkVolumeToVolumeFilter,we use this filter to smooth an image(2D) or a mitkVolume object(3D), the process is quite efficient, and the result is quite good contrast to the simplicity of the algorithm if the parameters are appropriately set. The algorithm is based on the following article:

- Pietro PPerona and Jitendra Malik, "Scale-Space and Edge Detection Using Anisotropic Diffusion," *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 12, No. 7, pp. 629-639, 1990

8.70.2 Member Function Documentation

8.70.2.1 virtual void mitkDiffusionFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.70.2.2 void mitkDiffusionFilter::SetDiffusionType (bool *Is3D* = false) [inline]

Set the class member variable m_Enable3D,determine the diffusion type(2D or 3D).

Parameters:

Is3D represents the diffusion type(true:3D false:2D).

8.70.2.3 void mitkDiffusionFilter::Setm_K (double *param* = 200)

Set the class member variable m_K.

Parameters:

param represents the extend of the intensity change of the data. a larger param usually means a more impressive diffusion while at the same time cause more blur in the edge.

Note:

if you want to get a better result, we recommend you choose a small param between 10 and 20 (this is not always true, you need to try yourself), and after several iterations you will get what you want.

8.70.2.4 void mitkDiffusionFilter::SetTimeInterval (float *dt* = 0.20)

Set the class member variable m_TimeInterval.

Parameters:

dt represents the time between the neighboring iterations.

Note:

A larger dt will cause a more obvious diffusion while the edge area is also greatly blured. If dt is too large (usually: dt>1), the iterations will not converge.

The documentation for this class was generated from the following file:

- mitkDiffusionFilter.h

8.71 mitkDirectionEncoder Class Reference

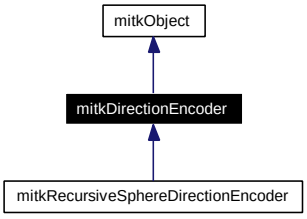
mitkDirectionEncoder - an abstract class to encode a direction into a one or two byte value

`#include <mitkDirectionEncoder.h>`

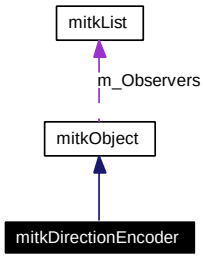
Inherits mitkObject.

Inherited by mitkRecursiveSphereDirectionEncoder.

Inheritance diagram for mitkDirectionEncoder:



Collaboration diagram for mitkDirectionEncoder:



Public Member Functions

- virtual int GetEncodedDirection (float n[3])=0
- virtual float * GetDecodedGradient (int value)=0
- virtual int GetNumberOfEncodedDirections ()=0
- virtual float * GetDecodedGradientTable ()=0

8.71.1 Detailed Description

mitkDirectionEncoder - an abstract class to encode a direction into a one or two byte value

mitkDirectionEncoder is an abstract class to encode a direction into a one or two byte value. Some codes are borrowed from VTK, and please see the copyright at end.

8.71.2 Member Function Documentation

8.71.2.1 virtual float* mitkDirectionEncoder::GetDecodedGradient (int *value*) [pure virtual]

Given an encoded value, return a pointer to the normal vector

Parameters:
value Represent the encoded value

Implemented in mitkRecursiveSphereDirectionEncoder.

8.71.2.2 virtual float* mitkDirectionEncoder::GetDecodedGradientTable () [pure virtual]

Get the decoded gradient table. There are this->GetNumberOfEncodedDirections() entries in the table, each containing a normal (direction) vector. This is a flat structure - 3 times the number of directions floats in an array.

Returns:
Return the decoded gradient table

Implemented in mitkRecursiveSphereDirectionEncoder.

8.71.2.3 virtual int mitkDirectionEncoder::GetEncodedDirection (float *n*[3]) [pure virtual]

Given a normal vector n, return the encoded direction

Parameters:
n Represent the normal vector

Implemented in mitkRecursiveSphereDirectionEncoder.

8.71.2.4 virtual int mitkDirectionEncoder::GetNumberOfEncodedDirections () [pure virtual]

Get the number of encoded directions

Returns:
Return the number of encoded directions

Implemented in mitkRecursiveSphereDirectionEncoder.

The documentation for this class was generated from the following file:

- mitkDirectionEncoder.h

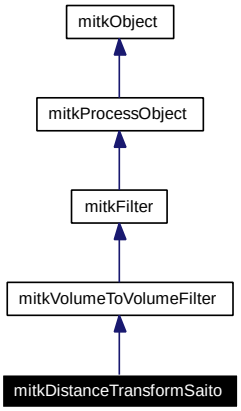
8.72 mitkDistanceTransformSaito Class Reference

mitkDistanceTransformSaito - computes 3D Euclidean DT

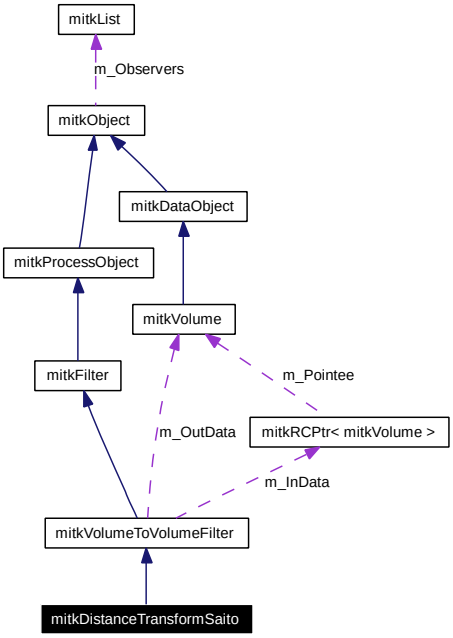
`#include <mitkDistanceTransformSaito.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkDistanceTransformSaito:



Collaboration diagram for mitkDistanceTransformSaito:



Public Member Functions

- mitkDistanceTransformSaito ()
- void SetDimension (int d)

- void SetImagePart (int nIp=3)
- int GetImagePart ()

8.72.1 Detailed Description

mitkDistanceTransformSaito - computes 3D Euclidean DT

mitkDistanceTransformSaito implements the Euclidean DT using Saito's algorithm. The distance map produced contains the square of the Euclidean distance values to the nearest zero pixel. Input of this class is a black and white image.

8.72.2 Constructor & Destructor Documentation

8.72.2.1 mitkDistanceTransformSaito::mitkDistanceTransformSaito ()

Constructor of the class

8.72.3 Member Function Documentation

8.72.3.1 int mitkDistanceTransformSaito::GetImagePart () [inline]

Get the part of the image to compute EDT

Returns:
1: the white part of the image is computed 2: the black part of the image is computed 3: both the black and white parts of the image is computed.

8.72.3.2 void mitkDistanceTransformSaito::SetDimension (int *d*)

Set the number of dimensions that you want to compute EDT

Parameters:
d Represent the dimention of the volume, 2 or 3

8.72.3.3 void mitkDistanceTransformSaito::SetImagePart (int *nIp* = 3)

Set the part of the image to compute EDT

Parameters:
nIp 1: compute the EDT of the white part 2: compute the EDT of the black part 3: compute the EDT of both the black and white part. distance of the black part is negtive, those of the white part is positive

The documentation for this class was generated from the following file:

- mitkDistanceTransformSaito.h

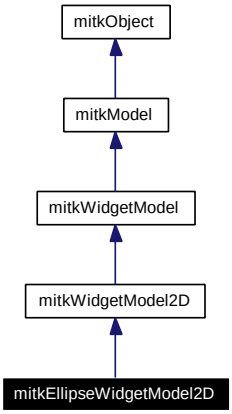
8.73 mitkEllipseWidgetModel2D Class Reference

mitkEllipseWidgetModel2D - an ellipse widget used in 2D scenes

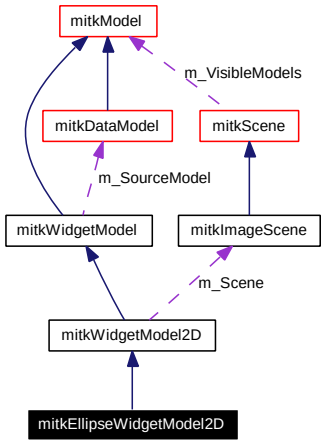
`#include <mitkEllipseWidgetModel2D.h>`

Inherits mitkWidgetModel2D.

Inheritance diagram for mitkEllipseWidgetModel2D:



Collaboration diagram for mitkEllipseWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- void SetCenterPoint (int scx, int scy)
- void SetCenterPoint (float cx, float cy)
- void SetCenterPoint (float p[2])
- void SetRadius (float xr, float yr)

- void SetRadius (float r)
- void SetStartPoint (float point[2])
- void SetStartPoint (float x, float y)
- void SetStartPoint (int sx, int sy)
- void SetMovePoint (float point[2])
- void SetMovePoint (float x, float y)
- void SetMovePoint (int sx, int sy)
- void SetEndPoint (float point[2])
- void SetEndPoint (float x, float y)
- void SetEndPoint (int sx, int sy)
- void SetUnitName (const string &name)
- const string & GetUnitName ()
- float GetArea ()
- float GetPerimeter ()
- void GetCenterPoint (float &tx, float &ty)
- void GetCenterPoint (int &ix, int &iy)
- void GetSemiMajorMinorAxis (float &ta, float &tb)
- void GetSemiMajorMinorAxis (int &a, int &b)
- virtual mitkVolume * GetRegionMask ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.73.1 Detailed Description

mitkEllipseWidgetModel2D - an ellipse widget used in 2D scenes

mitkEllipseWidgetModel2D is an ellipse widget used in 2D scenes which can respond the mouse events to change its statuses and return the information about itself (e.g. center coordinates, area ...). It is supposed to be attached to a 2D data model (e.g. mitkImageModel) and add to a 2D scene (e.g. mitkImageScene), or the display could be improper.

8.73.2 Member Function Documentation

8.73.2.1 virtual void mitkEllipseWidgetModel2D::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.73.2.2 virtual void mitkEllipseWidgetModel2D::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.73.2.3 virtual void mitkEllipseWidgetModel2D::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.73.2.4 float mitkEllipseWidgetModel2D::GetArea ()

Get area of this ellipse.

Returns:

Return the area of this ellipse.

8.73.2.5 void mitkEllipseWidgetModel2D::GetCenterPoint (int & *ix*, int & *iy*)

Get the integral coordinates of the center point in the original image.

Parameters:

ix return the x-coordinate of the center point

iy return the y-coordinate of the center point

8.73.2.6 void mitkEllipseWidgetModel2D::GetCenterPoint (float & *tx*, float & *ty*)

Get the physical coordinates in the object space of the center point.

Parameters:

tx return the x-coordinate of the center point

ty return the y-coordinate of the center point

8.73.2.7 float mitkEllipseWidgetModel2D::GetPerimeter ()

Get the perimeter of the ellipse.

Returns:

Return the perimeter of the ellipse.

Note:

The return value is an approximation. It is calculated by the following formula:

$$\pi(a+b)\frac{64-3\lambda^4}{64-16\lambda^2}$$

where

$$\lambda = \frac{a-b}{a+b}$$

a is the semimajor axis, b is the semiminor axis.

8.73.2.8 virtual mitkVolume* mitkEllipseWidgetModel2D::GetRegionMask () [virtual]

Get mask image where the value of widget region is 255 and the rest is 0.

Returns:

Return a pointer of mitkVolume object contains the mask image.

Note:

The returned object pointer should be deleted properly by yourself.

Reimplemented from mitkWidgetModel2D.

8.73.2.9 void mitkEllipseWidgetModel2D::GetSemiMajorMinorAxis (int & *a*, int & *b*)

Get the integral length of the semimajor and minor axis in the original image.

Parameters:

a return the semimajor axis

b return the semiminor axis

Note:

a is always greater than b.

8.73.2.10 void mitkEllipseWidgetModel2D::GetSemiMajorMinorAxis (float & *ta*, float & *tb*)

Get the physical length of the semimajor and minor axis in the object space.

Parameters:

ta return the semimajor axis

tb return the semiminor axis

Note:

ta is always greater than tb.

8.73.2.11 const string& mitkEllipseWidgetModel2D::GetUnitName () [inline]

Get name string of unit.

Returns:

Return a constant reference to a string contains the name of the length unit this line uses

8.73.2.12 virtual void mitkEllipseWidgetModel2D::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:

names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.73.2.13 virtual void mitkEllipseWidgetModel2D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkWidgetModel2D.

8.73.2.14 virtual void mitkEllipseWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.73.2.15 virtual int mitkEllipseWidgetModel2D::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:

scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.73.2.16 void mitkEllipseWidgetModel2D::SetCenterPoint (float *p*[2]) [inline]

Set the center point of this ellipse in object space.

Parameters:
p[0] x-coordinate of the center point

p[1] y-coordinate of the center point

8.73.2.17 void mitkEllipseWidgetModel2D::SetCenterPoint (float *cx*, float *cy*) [inline]

Set the center point of this ellipse in object space.

Parameters:
cx x-coordinate of the center point

cy y-coordinate of the center point

8.73.2.18 void mitkEllipseWidgetModel2D::SetCenterPoint (int *scx*, int *scy*) [inline]

Set the center point of this ellipse.

Parameters:
scx x-coordinate of this point on screen

scy y-coordinate of this point on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.73.2.19 void mitkEllipseWidgetModel2D::SetEndPoint (int *sx*, int *sy*) [inline]

Set position of the end point.

Parameters:
sx x-coordinate of the end point of mouse dragging on screen

sy y-coordinate of the end point of mouse dragging on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.73.2.20 void mitkEllipseWidgetModel2D::SetEndPoint (float *x*, float *y*) [inline]

Set position of the end point in the object space.

Parameters:
x x-coordinate of the end point of mouse dragging
y y-coordinate of the end point of mouse dragging

Note:
The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.73.2.21 void mitkEllipseWidgetModel2D::SetEndPoint (float *point*[2]) [inline]

Set position of the end point in the object space.

Parameters:
point[0] x-coordinate of the end point of mouse dragging
point[1] y-coordinate of the end point of mouse dragging

Note:
The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.73.2.22 void mitkEllipseWidgetModel2D::SetMovePoint (int *sx*, int *sy*) [inline]

Set position of the moving end point.

Parameters:
sx x-coordinate of the moving end point of mouse dragging on screen
sy y-coordinate of the moving end point of mouse dragging on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.73.2.23 void mitkEllipseWidgetModel2D::SetMovePoint (float *x*, float *y*) [inline]

Set position of the moving end point in the object space.

Parameters:

x x-coordinate of the moving end point of mouse dragging

y y-coordinate of the moving end point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.73.2.24 void mitkEllipseWidgetModel2D::SetMovePoint (float *point*[2]) [inline]

Set position of the moving end point in the object space.

Parameters:

point[0] x-coordinate of the moving end point of mouse dragging

point[1] y-coordinate of the moving end point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.73.2.25 void mitkEllipseWidgetModel2D::SetRadius (float *r*) [inline]

Set the radiuses of the ellipse to the same value (i.e the ellipse is a circle).

Parameters:

r radius of the circle

8.73.2.26 void mitkEllipseWidgetModel2D::SetRadius (float *xr*, float *yr*) [inline]

Set the radiuses of the ellipse.

Parameters:

xr radius along x-axis

yr radius along y-axis

8.73.2.27 void mitkEllipseWidgetModel2D::SetStartPoint (int *sx*, int *sy*) [inline]

Set position of the start point.

Parameters:

sx x-coordinate of the start point of mouse dragging on screen

sy y-coordinate of the start point of mouse dragging on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.73.2.28 void mitkEllipseWidgetModel2D::SetStartPoint (float *x*, float *y*) [inline]

Set position of the start point in the object space.

Parameters:
x x-coordinate of the start point of mouse dragging

y y-coordinate of the start point of mouse dragging

Note:
The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.73.2.29 void mitkEllipseWidgetModel2D::SetStartPoint (float *point*[2]) [inline]

Set position of the start point in the object space.

Parameters:
point[0] x-coordinate of the start point of mouse dragging

point[1] y-coordinate of the start point of mouse dragging

Note:
The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.73.2.30 void mitkEllipseWidgetModel2D::SetUnitName (const string & *name*) [inline]

Set name string of unit.

Parameters:
name a constant reference to a string contains the name of the length unit (e.g. mm) this line uses

The documentation for this class was generated from the following file:

- mitkEllipseWidgetModel2D.h

8.74 mitkEncodedGradientEstimator Class Reference

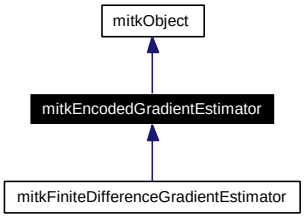
mitkEncodedGradientEstimator - an abstract class for gradient estimation

`#include <mitkEncodedGradientEstimator.h>`

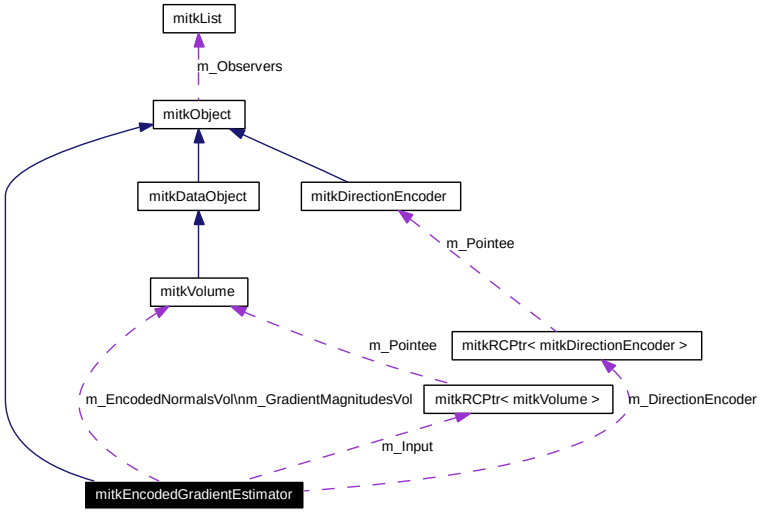
Inherits mitkObject.

Inherited by mitkFiniteDifferenceGradientEstimator.

Inheritance diagram for mitkEncodedGradientEstimator:



Collaboration diagram for mitkEncodedGradientEstimator:



Public Member Functions

- void SetInput (mitkVolume *pInVolume)
- mitkVolume * GetInput ()
- void SetGradientMagnitudeScale (float fGMS)
- float GetGradientMagnitudeScale ()
- void SetGradientMagnitudeBias (float fGMB)
- float GetGradientMagnitudeBias ()
- virtual void SetBoundsClip (int nBoundsClip)
- virtual int GetBoundsClipMinValue ()
- virtual int GetBoundsClipMaxValue ()
- virtual int GetBoundsClip ()

- virtual void SetBoundsClipOn ()
- virtual void SetBoundsClipOff ()
- virtual void SetBounds (int arg1, int arg2, int arg3, int arg4, int arg5, int arg6)
- virtual void SetBounds (int arg[6])
- virtual void GetBounds (int &arg1, int &arg2, int &arg3, int &arg4, int &arg5, int &arg6)
- void GetBounds (int arg[6])
- int * GetBounds ()
- void SetComputeGradientMagnitudes (int nCGM)
- int GetComputeGradientMagnitudes ()
- virtual void SetComputeGradientMagnitudesOn ()
- virtual void SetComputeGradientMagnitudesOff ()
- void SetCylinderClip (int nCC)
- int GetCylinderClip ()
- virtual void SetCylinderClipOn ()
- virtual void SetCylinderClipOff ()
- int GetUseCylinderClip ()
- int * GetCircleLimits ()
- void SetZeroNormalThreshold (float v)
- float GetZeroNormalThreshold ()
- virtual void SetZeroPad (int nZeroPad)
- virtual int GetZeroPadMinValue ()
- virtual int GetZeroPadMaxValue ()
- virtual int GetZeroPad ()
- virtual void SetZeroPadOn ()
- virtual void SetZeroPadOff ()
- void Update ()
- unsigned short * GetEncodedNormals ()
- mitkVolume * GetEncodedNormalsVol ()
- unsigned char * GetGradientMagnitudes ()
- mitkVolume * GetGradientMagnitudesVol ()
- void SetDirectionEncoder (mitkDirectionEncoder *direnc)
- mitkDirectionEncoder * GetDirectionEncoder ()

8.74.1 Detailed Description

mitkEncodedGradientEstimator - an abstract class for gradient estimation

mitkEncodedGradientEstimator is an abstract class for gradient estimation. Some codes are borrowed from VTK, and please see the copyright at end.

8.74.2 Member Function Documentation

8.74.2.1 int* mitkEncodedGradientEstimator::GetBounds ()

Get the bounds of the computation (used if this->ComputationBounds is 1.) The bounds are specified xmin, xmax, ymin, ymax, zmin, zmax.

Returns:
Return the bounds value

8.74.2.2 void mitkEncodedGradientEstimator::GetBounds (int *arg*[6])

Get the bounds of the computation (used if this->ComputationBounds is 1.) The bounds are specified xmin, xmax, ymin, ymax, zmin, zmax.

Parameters:

arg[0] Represent the first bounds value

arg[1] Represent the second bounds value

arg[2] Represent the third bounds value

arg[3] Represent the forth bounds value

arg[4] Represent the fifth bounds value

arg[5] Represent the sixth bounds value

8.74.2.3 virtual void mitkEncodedGradientEstimator::GetBounds (int & *arg1*, int & *arg2*, int & *arg3*, int & *arg4*, int & *arg5*, int & *arg6*) [virtual]

Get the bounds of the computation (used if this->ComputationBounds is 1.) The bounds are specified xmin, xmax, ymin, ymax, zmin, zmax.

Parameters:

arg1 Represent the first bounds value

arg2 Represent the second bounds value

arg3 Represent the third bounds value

arg4 Represent the forth bounds value

arg5 Represent the fifth bounds value

arg6 Represent the sixth bounds value

8.74.2.4 virtual int mitkEncodedGradientEstimator::GetBoundsClip () [inline, virtual]

Get the bounds clip value

Returns:

Return the bounds clip value

8.74.2.5 virtual int mitkEncodedGradientEstimator::GetBoundsClipMaxValue () [inline, virtual]

Get the max bounds clip value

Returns:

Return the max bounds clip value

8.74.2.6 virtual int mitkEncodedGradientEstimator::GetBoundsClipMinValue () [inline, virtual]

Get the min bounds clip value

Returns:

Return the min bounds clip value

8.74.2.7 int* mitkEncodedGradientEstimator::GetCircleLimits () [inline]

Get the circle limits value

Returns:
Return the circle limits value

8.74.2.8 int mitkEncodedGradientEstimator::GetComputeGradientMagnitudes () [inline]

Get the compute gradient magnitude value

Returns:
Return the compute gradient magnitude value

8.74.2.9 int mitkEncodedGradientEstimator::GetCylinderClip () [inline]

Get the cylinder clip value

Returns:
Return the cylinder clip value

8.74.2.10 mitkDirectionEncoder* mitkEncodedGradientEstimator::GetDirectionEncoder () [inline]

Get the direction encoder used to encode normal directions to fit within two bytes

Returns:
Return the direction encoder value

8.74.2.11 unsigned short* mitkEncodedGradientEstimator::GetEncodedNormals ()

Get the encoded normals.

Returns:
Return the encoded normal

Warning:
The return value could be NULL, if the original data is out-of-core. Under this circumstance, please use GetEncodedNormalsVol() instead to get the volume of encoded normals and use the interfaces of mitkVolume (such as GetSliceData(), ReadSliceData() etc.) to get actual encoded normals.

8.74.2.12 mitkVolume* mitkEncodedGradientEstimator::GetEncodedNormalsVol ()

Get the encoded normals volume (in case of out-of-core data).

Returns:
Return the volume contains the encoded normals.

8.74.2.13 float mitkEncodedGradientEstimator::GetGradientMagnitudeBias () [inline]

Get the scale and bias for the gradient magnitude

Returns:
Return the gradient magnitude

8.74.2.14 unsigned char* mitkEncodedGradientEstimator::GetGradientMagnitudes ()

Get the gradient magnitudes

Returns:
Return the gradient magnitude value

Warning:
The return value could be NULL, if the original data is out-of-core. Under this circumstance, please use GetGradientMagnitudesVol() instead to get the volume of gradient magnitudes and use the interfaces of mitkVolume (such as GetSliceData(), ReadSliceData() etc.) to get actual gradient magnitude values.

8.74.2.15 float mitkEncodedGradientEstimator::GetGradientMagnitudeScale () [inline]

Get the scale and bias for the gradient magnitude

Returns:
Return the scale and bias for the gradient magnitude

8.74.2.16 mitkVolume* mitkEncodedGradientEstimator::GetGradientMagnitudesVol ()

Get the gradient magnitudes volume (in case of out-of-core data).

Returns:
Return the volume contains the gradient magnitude value.

8.74.2.17 mitkVolume* mitkEncodedGradientEstimator::GetInput () [inline]

Get the scalar input for which the normals will be calculated

Returns:
Return the scalar input for which the normals will be calculated

8.74.2.18 int mitkEncodedGradientEstimator::GetUseCylinderClip () [inline]

Get the use cylinder clip value

Returns:
Return the use cylinder clip value

8.74.2.19 float mitkEncodedGradientEstimator::GetZeroNormalThreshold () [inline]

Get teh zero normal threshold value

Returns:
Return the zero normal threshold value

8.74.2.20 virtual int mitkEncodedGradientEstimator::GetZeroPad () [inline, virtual]

Get the zero pad value

Returns:
Return the zero pad value

8.74.2.21 virtual int mitkEncodedGradientEstimator::GetZeroPadMaxValue () [inline, virtual]

Get the max zero pad value

Returns:
Return the min zero pad value

8.74.2.22 virtual int mitkEncodedGradientEstimator::GetZeroPadMinValue () [inline, virtual]

Get the min zero pad value

Returns:
Return the min zero pad value

8.74.2.23 virtual void mitkEncodedGradientEstimator::SetBounds (int *arg*[6]) [virtual]

Set the bounds of the computation (used if this->ComputationBounds is 1.) The bounds are specified xmin, xmax, ymin, ymax, zmin, zmax.

Parameters:
arg[0] Represent the first bounds value

arg[1] Represent the second bounds value

arg[2] Represent the third bounds value

arg[3] Represent the forth bounds value

arg[4] Represent the fifth bounds value

arg[5] Represent the sixth bounds value

8.74.2.24 virtual void mitkEncodedGradientEstimator::SetBounds (int *arg1*, int *arg2*, int *arg3*, int *arg4*, int *arg5*, int *arg6*) [virtual]

Set the bounds of the computation (used if this->ComputationBounds is 1.) The bounds are specified xmin, xmax, ymin, ymax, zmin, zmax.

Parameters:

arg1 Represent the first bounds value

arg2 Represent the second bounds value

arg3 Represent the third bounds value

arg4 Represent the forth bounds value

arg5 Represent the fifth bounds value

arg6 Represent the sixth bounds value

8.74.2.25 virtual void mitkEncodedGradientEstimator::SetBoundsClip (int *nBoundsClip*) [inline, virtual]

Turn on the bounding of the normal computation by the this->Bounds bounding box

Parameters:

nBoundsClip Represent the on or off value

8.74.2.26 virtual void mitkEncodedGradientEstimator::SetBoundsClipOff () [inline, virtual]

Turn off the bounding of the normal computation by the this->Bounds bounding box

8.74.2.27 virtual void mitkEncodedGradientEstimator::SetBoundsClipOn () [inline, virtual]

Turn on the bounding of the normal computation by the this->Bounds bounding box

8.74.2.28 void mitkEncodedGradientEstimator::SetComputeGradientMagnitudes (int *nCGM*) [inline]

If you don't want to compute gradient magnitudes (but you do want normals for shading) this can be used. Be careful - if if you a non-constant gradient magnitude transfer function and you turn this on, it may crash

Parameters:

nCGM Represent the compute gradient magnitude value

8.74.2.29 virtual void mitkEncodedGradientEstimator::SetComputeGradientMagnitudesOff () [inline, virtual]

If you don't want to compute gradient magnitudes (but you do want normals for shading) this can be used. Be careful - if if you a non-constant gradient magnitude transfer function and you turn this on, it may crash

8.74.2.30 virtual void mitkEncodedGradientEstimator::SetComputeGradientMagnitudesOn () [inline, virtual]

If you don't want to compute gradient magnitudes (but you do want normals for shading) this can be used. Be careful - if if you a non-constant gradient magnitude transfer function and you turn this on, it may crash

8.74.2.31 void mitkEncodedGradientEstimator::SetCylinderClip (int *nCC*) [inline]

If the data in each slice is only contained within a circle circumscribed within the slice, and the slice is square, then don't compute anything outside the circle. This circle through the slices forms a cylinder.

Parameters:
nCC Represent the cylinder clip value

8.74.2.32 virtual void mitkEncodedGradientEstimator::SetCylinderClipOff () [inline, virtual]

If the data in each slice is only contained within a circle circumscribed within the slice, and the slice is square, then don't compute anything outside the circle. This circle through the slices forms a cylinder.

8.74.2.33 virtual void mitkEncodedGradientEstimator::SetCylinderClipOn () [inline, virtual]

If the data in each slice is only contained within a circle circumscribed within the slice, and the slice is square, then don't compute anything outside the circle. This circle through the slices forms a cylinder.

8.74.2.34 void mitkEncodedGradientEstimator::SetDirectionEncoder (mitkDirectionEncoder * *direnc*)

Set the direction encoder used to encode normal directions to fit within two bytes

Parameters:
direnc Represent the direction encoder value

8.74.2.35 void mitkEncodedGradientEstimator::SetGradientMagnitudeBias (float *fGMB*) [inline]

Set the scale and bias for the gradient magnitude

Parameters:
fGMB Represent the gradient magnitude

8.74.2.36 void mitkEncodedGradientEstimator::SetGradientMagnitudeScale (float *fGMS*) [inline]

Set the scale and bias for the gradient magnitude

Parameters:
fGMS Represent the scale and bias for the gradient magnitude

8.74.2.37 void mitkEncodedGradientEstimator::SetInput (mitkVolume * *pInVolume*)

Set the scalar input for which the normals will be calculated

Parameters:
pInVolume Represent the scalar input for which the normals will be calculated

8.74.2.38 void mitkEncodedGradientEstimator::SetZeroNormalThreshold (float *v*)

Set the ZeroNormalThreshold - this defines the minimum magnitude of a gradient that is considered sufficient to define a direction. Gradients with magnitudes at or less than this value are given a "zero normal" index. These are handled specially in the shader, and you can set the intensity of light for these zero normals in the gradient shader.

Parameters:
v Represent the zero normal threshold value

8.74.2.39 virtual void mitkEncodedGradientEstimator::SetZeroPad (int *nZeroPad*) [inline, virtual]

Assume that the data value outside the volume is zero when computing normals.

Parameters:
nZeroPad Represent teh zero pad value

8.74.2.40 virtual void mitkEncodedGradientEstimator::SetZeroPadOff () [inline, virtual]

Assume that the data value outside the volume is zero when computing normals.

8.74.2.41 virtual void mitkEncodedGradientEstimator::SetZeroPadOn () [inline, virtual]

Assume that the data value outside the volume is zero when computing normals.

8.74.2.42 void mitkEncodedGradientEstimator::Update ()

Recompute the encoded normals and gradient magnitudes.

The documentation for this class was generated from the following file:

- mitkEncodedGradientEstimator.h

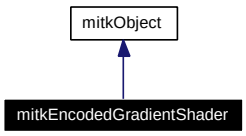
8.75 mitkEncodedGradientShader Class Reference

mitkEncodedGradientShader - Computes shading tables for encoded normals

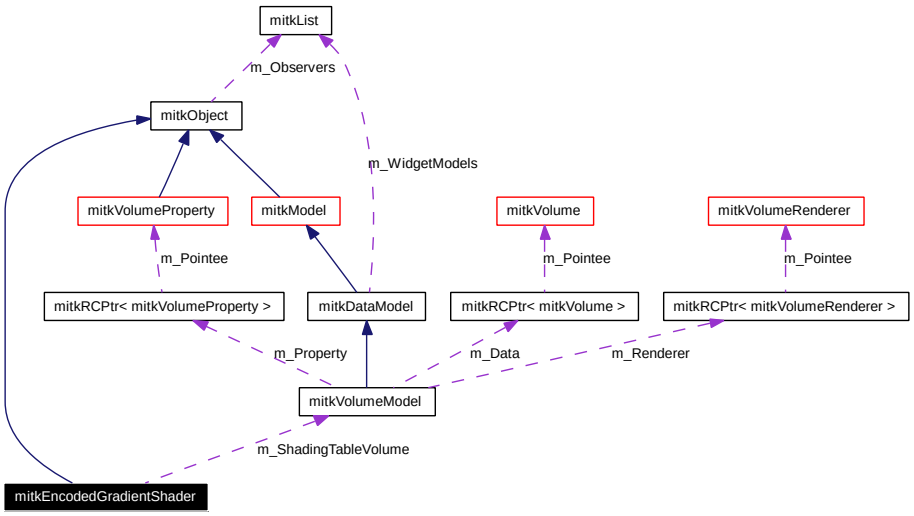
`#include <mitkEncodedGradientShader.h>`

Inherits mitkObject.

Inheritance diagram for mitkEncodedGradientShader:



Collaboration diagram for mitkEncodedGradientShader:



8.75.1 Detailed Description

mitkEncodedGradientShader - Computes shading tables for encoded normals

mitkEncodedGradientShader computes shading tables for encoded normals. Some codes are borrowed from VTK, and please see the copyright at end.

The documentation for this class was generated from the following file:

- mitkEncodedGradientShader.h

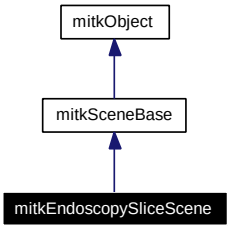
8.76 mitkEndoscopySliceScene Class Reference

mitkEndoscopySliceScene - A slice preview scene for virtual endoscopy navigation

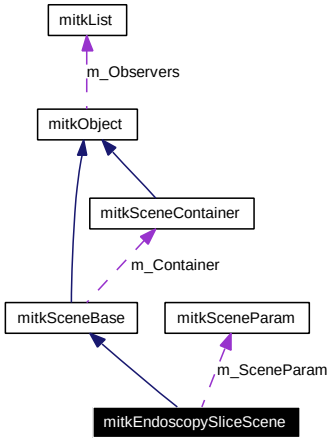
`#include <mitkEndoscopySliceScene.h>`

Inherits mitkSceneBase.

Inheritance diagram for mitkEndoscopySliceScene:



Collaboration diagram for mitkEndoscopySliceScene:



Public Member Functions

- virtual void RenderGL ()
- void SetVolume (mitkCoherentVolume *vol)
- mitkCoherentVolume * GetVolume ()
- void SetVolumeInfo (int dims[3], float spacings[3])
- void SetHorizontial ()
- void SetVertical ()
- bool IsHorizontial ()
- void SetGridSpacing (float spacing)
- float GetGridSpacing ()
- void SetSceneParam (mitkSceneParam *sceneParam)
- mitkSceneParam * GetSceneParam ()
- void SetWidthCenter (float windowWidth, float windowCenter)
- void SetWindowWidth (float windowWidth)

- void SetWindowCenter (float windowCenter)
- float GetWindowWidth ()
- float GetWindowCenter ()
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseWheel (bool ctrlDown, bool shiftDown, int xPos, int yPos, int delta)
- void SetRotateRadPerPixel (float radPerPixel)
- float GetRotateRadPerPixel () const

8.76.1 Detailed Description

mitkEndoscopySliceScene - A slice preview scene for virtual endoscopy navigation

mitkEndoscopySliceScene is a slice preview scene for virtual endoscopy navigation

8.76.2 Member Function Documentation

8.76.2.1 float mitkEndoscopySliceScene::GetGridSpacing () [inline]

Get the 2D grid spacing in physical unit.

Returns:
: 2D grid spacing in physical unit.

8.76.2.2 float mitkEndoscopySliceScene::GetRotateRadPerPixel () const [inline]

Get the interaction sensitivity of rotation.

Returns:
Return the rad the object is rotated when the mouse is moved 1 pixel

8.76.2.3 mitkSceneParam* mitkEndoscopySliceScene::GetSceneParam () [inline]

Get the internal scene param.

Returns:
: the internal scene param.

8.76.2.4 mitkCoherentVolume* mitkEndoscopySliceScene::GetVolume ()

Get the data object to be navigated

Returns:
The data object to be navigated.

8.76.2.5 float mitkEndoscopySliceScene::GetWindowCenter ()

Get the window center

Returns:
Return the window center

8.76.2.6 float mitkEndoscopySliceScene::GetWindowWidth ()

Get the window width

Returns:
Return the window width

8.76.2.7 bool mitkEndoscopySliceScene::IsHorizontial () [inline]

Get the display mode.

Returns:
: true=horizontal false=vertical

8.76.2.8 virtual void mitkEndoscopySliceScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkSceneBase.

8.76.2.9 virtual void mitkEndoscopySliceScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkSceneBase.

8.76.2.10 virtual void mitkEndoscopySliceScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkSceneBase.

8.76.2.11 virtual void mitkEndoscopySliceScene::OnMouseWheel (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *delta*) [virtual]

The Interface for handling the mouse wheel event pass by the container

Reimplemented from mitkSceneBase.

8.76.2.12 virtual void mitkEndoscopySliceScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkSceneBase.

8.76.2.13 void mitkEndoscopySliceScene::SetGridSpacing (float *spacing*) [inline]

Set the 2D grid spacing in physical unit.

Parameters:
spacings,: 2D grid spacing in physical unit.

8.76.2.14 void mitkEndoscopySliceScene::SetHorizontial () [inline]

Set to horizontal mode

8.76.2.15 void mitkEndoscopySliceScene::SetRotateRadPerPixel (float *radPerPixel*) [inline]

Set the interaction sensitivity of rotation.

Parameters:
radPerPixel. The rad the object is rotated when the mouse is moved 1 pixel

8.76.2.16 void mitkEndoscopySliceScene::SetSceneParam (mitkSceneParam * *sceneParam*) [inline]

Set the internal scene param.

Parameters:
sceneParam,: the internal scene param.

8.76.2.17 void mitkEndoscopySliceScene::SetVertical () [inline]

Set to vertical mode

8.76.2.18 void mitkEndoscopySliceScene::SetVolume (mitkCoherentVolume * *vol*)

Set the data object to be navigated

Parameters:
vol,: The data object to be navigated.

8.76.2.19 void mitkEndoscopySliceScene::SetVolumeInfo (int *dims*[3], float *spacings*[3])

Set the volume dimensions and spacings then calculate the cropping boundaries

Parameters:

dims,: dimensions.

float,: spacings.

8.76.2.20 void mitkEndoscopySliceScene::SetWidthCenter (float *windowWidth*, float *windowCenter*)

Set the window width and windows center

Parameters:

windowWidth The window width

windowCenter The window center

8.76.2.21 void mitkEndoscopySliceScene::SetWindowCenter (float *windowCenter*)

Set the window center

Parameters:

windowCenter The window center

8.76.2.22 void mitkEndoscopySliceScene::SetWindowWidth (float *windowWidth*)

Set the window width

Parameters:

windowWidth The window width

The documentation for this class was generated from the following file:

- mitkEndoscopySliceScene.h

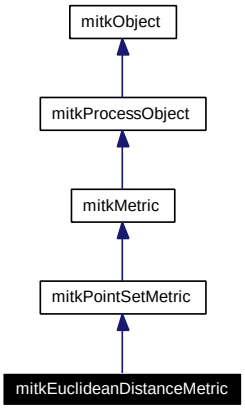
8.77 mitkEuclideanDistanceMetric Class Reference

mitkEuclideanDistanceMetric - a concrete class computing similarity between two point sets

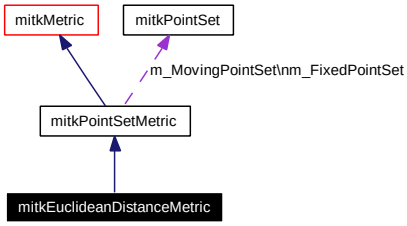
`#include <mitkEuclideanDistanceMetric.h>`

Inherits mitkPointSetMetric.

Inheritance diagram for mitkEuclideanDistanceMetric:



Collaboration diagram for mitkEuclideanDistanceMetric:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkEuclideanDistanceMetric ()
- void GetClosestPointPairs (vector< int > *closestPointMask)
- double GetPointSetSimilarity ()

8.77.1 Detailed Description

mitkEuclideanDistanceMetric - a concrete class computing similarity between two point sets

mitkEuclideanDistanceMetric is a concrete class computing similarity between two point sets using euclidean distance.

8.77.2 Constructor & Destructor Documentation

8.77.2.1 mitkEuclideanDistanceMetric::mitkEuclideanDistanceMetric ()

Constructor.

8.77.3 Member Function Documentation

8.77.3.1 void mitkEuclideanDistanceMetric::GetClosestPointPairs (vector< int > * *closestPointMask*) [virtual]

Find correspondent point matches from moving point set to fixed point set.

Parameters:
closestPointMask A mask contains matched points' index from moving point set

Reimplemented from mitkPointSetMetric.

8.77.3.2 double mitkEuclideanDistanceMetric::GetPointSetSimilarity () [virtual]

Get the similarity measure between fixed point set and moving point set.

Returns:
Return the similarity measure value.

Reimplemented from mitkPointSetMetric.

8.77.3.3 virtual void mitkEuclideanDistanceMetric::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkPointSetMetric.

The documentation for this class was generated from the following file:

- mitkEuclideanDistanceMetric.h

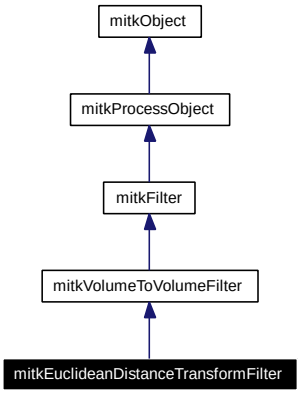
8.78 mitkEuclideanDistanceTransformFilter Class Reference

mitkEuclideanDistanceTransformFilter - computes 3D Euclidean DT using Meijster's Linear Time Algorithm

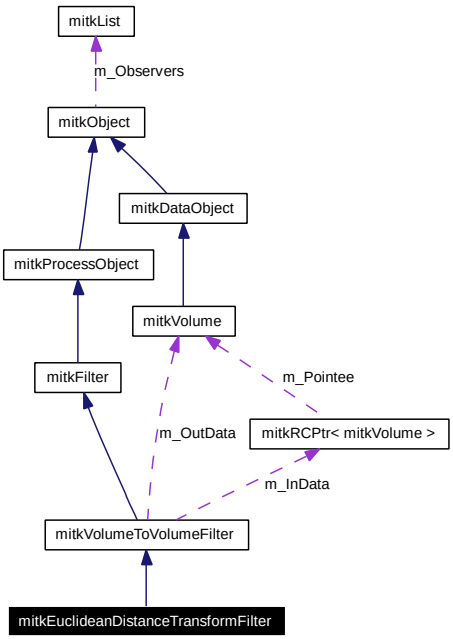
`#include <mitkEuclideanDistanceTransformFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkEuclideanDistanceTransformFilter:



Collaboration diagram for mitkEuclideanDistanceTransformFilter:



Public Member Functions

- void SetLowThreshValue (double d)
- int GetLowThreshValue ()

- void SetHighThreshValue (double d)
- int GetHighThreshValue ()
- void SetSquareDistance (bool sd)
- bool UsingSquareDistance ()
- void SetBackgroundMode (bool bm=true)

8.78.1 Detailed Description

mitkEuclideanDistanceTransformFilter - computes 3D Euclidean DT using Meijster's Linear Time Algorithm

mitkDistanceTransformSaito implements the Euclidean DT using Meijster's Linear Time Algorithm.

8.78.2 Member Function Documentation

8.78.2.1 int mitkEuclideanDistanceTransformFilter::GetHighThreshValue () [inline]

Return the high threshold of the target area

Returns:
the high threshold of the target area

8.78.2.2 int mitkEuclideanDistanceTransformFilter::GetLowThreshValue () [inline]

Return the low threshold of the target area

Returns:
the low threshold of the target area

8.78.2.3 void mitkEuclideanDistanceTransformFilter::SetBackgroundMode (bool *bm* = true) [inline]

Set whether use background mode. In background mode, the areas where value is outside the threshold range is consider to be the target.

Parameters:
bm true for background mode; false for foreground mode.

8.78.2.4 void mitkEuclideanDistanceTransformFilter::SetHighThreshValue (double *d*) [inline]

Set the high threshold of the target area

Parameters:
d the high threshold of the target area

8.78.2.5 void mitkEuclideanDistanceTransformFilter::SetLowThreshValue (double *d*) [inline]

Set the low threshold of the target area

Parameters:
d the low threshold of the target area

8.78.2.6 void mitkEuclideanDistanceTransformFilter::SetSquareDistance (bool *sd*) [inline]

Set whether use square distance to represent the result

Parameters:
sd true for square distance; false for linear distance.

8.78.2.7 bool mitkEuclideanDistanceTransformFilter::UsingSquareDistance () [inline]

See whether the filter is using square distance to represent the result

Returns:
true for square distance; false for linear distance.

The documentation for this class was generated from the following file:

- mitkEuclideanDistanceTransformFilter.h

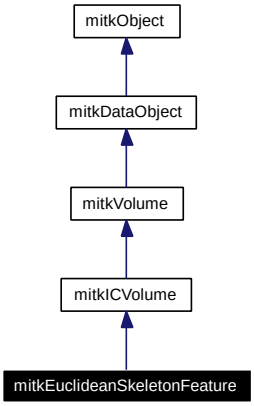
8.79 mitkEuclideanSkeletonFeature Class Reference

mitkEuclideanSkeletonFeature - Data structure representing a decomposed volume

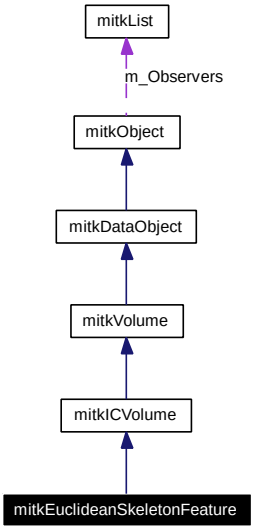
`#include <mitkEuclideanSkeletonFeatureExtraction.h>`

Inherits mitkICVolume.

Inheritance diagram for mitkEuclideanSkeletonFeature:



Collaboration diagram for mitkEuclideanSkeletonFeature:



Public Member Functions

- void GetSkeletonPosition (unsigned skeletonIndex, int &posX, int &posY, int &posZ)
- void GetSkeletonPosition (unsigned skeletonIndex, int pos[3])
- unsigned AddDescriptor (const Descriptor &descriptor)
- unsigned GetNumberOfDescriptors ()
- Descriptor & GetDescriptor (unsigned index)
- void RemoveAllDescriptors ()

8.79.1 Detailed Description

mitkEuclideanSkeletonFeature - Data structure representing a decomposed volume

mitkEuclideanSkeletonFeature is a data structure representing a decomposed volume

8.79.2 Member Function Documentation

8.79.2.1 unsigned mitkEuclideanSkeletonFeature::AddDescriptor (const Descriptor & *descriptor*) [inline]

Add a structure descriptor into the data structure

8.79.2.2 Descriptor& mitkEuclideanSkeletonFeature::GetDescriptor (unsigned *index*) [inline]

Return a specific structure descriptor

Returns:
the structure descriptor

8.79.2.3 unsigned mitkEuclideanSkeletonFeature::GetNumberOfDescriptors () [inline]

Return the number of descriptors in the data structure

Returns:
the number of descriptors in the data structure

8.79.2.4 void mitkEuclideanSkeletonFeature::GetSkeletonPosition (unsigned *skeletonIndex*, int *pos*[3])

Return the position of the skeleton center of a area , returns the position of the skeleton center of a area

8.79.2.5 void mitkEuclideanSkeletonFeature::GetSkeletonPosition (unsigned *skeletonIndex*, int & *posX*, int & *posY*, int & *posZ*)

Return the position of the skeleton center of a area , posY, posZ, return the position of the skeleton center of a area

8.79.2.6 void mitkEuclideanSkeletonFeature::RemoveAllDescriptors () [inline]

Remove all the descriptors from the data structure

The documentation for this class was generated from the following file:

- mitkEuclideanSkeletonFeatureExtraction.h

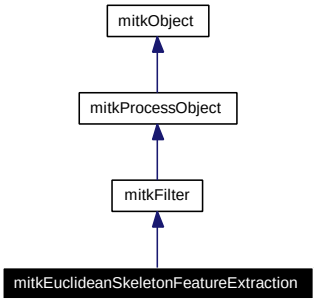
8.80 mitkEuclideanSkeletonFeatureExtraction Class Reference

mitkEuclideanSkeletonFeatureExtraction - Decompose a volume based on Euclidean DT and Reverse DT

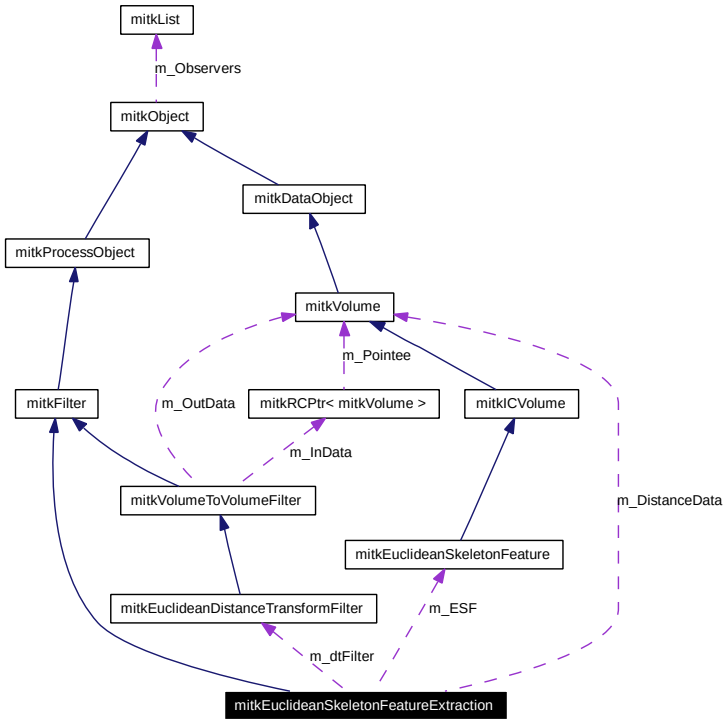
`#include <mitkEuclideanSkeletonFeatureExtraction.h>`

Inherits mitkFilter.

Inheritance diagram for mitkEuclideanSkeletonFeatureExtraction:



Collaboration diagram for mitkEuclideanSkeletonFeatureExtraction:



Public Member Functions

- void SetLowThreshValue (double d)
- int GetLowThreshValue ()
- void SetHighThreshValue (double d)

- int GetHighThreshValue ()
- void SetInput (mitkVolume *inData)
- mitkVolume * GetInput ()
- mitkEuclideanSkeletonFeature * GetEuclideanSkeletonFeature ()

8.80.1 Detailed Description

mitkEuclideanSkeletonFeatureExtraction - Decompose a volume based on Euclidean DT and Reverse DT

mitkEuclideanSkeletonFeatureExtraction decomposes a volume based on Euclidean DT and Reverse DT, and calculates some features like radius, size

8.80.2 Member Function Documentation

8.80.2.1 mitkEuclideanSkeletonFeature* mitkEuclideanSkeletonFeatureExtraction::GetEuclideanSkeletonFeature ()

Return the decomposed volume

Returns:
the decomposed volume

8.80.2.2 int mitkEuclideanSkeletonFeatureExtraction::GetHighThreshValue () [inline]

Return the high threshold of the target area

Returns:
the high threshold of the target area

8.80.2.3 mitkVolume* mitkEuclideanSkeletonFeatureExtraction::GetInput () [inline]

Return the input data

Returns:
the input data

8.80.2.4 int mitkEuclideanSkeletonFeatureExtraction::GetLowThreshValue () [inline]

Return the low threshold of the target area

Returns:
the low threshold of the target area

8.80.2.5 void mitkEuclideanSkeletonFeatureExtraction::SetHighThreshValue (double *d*) [inline]

Set the high threshold of the target area

Parameters:
d the high threshold of the target area

8.80.2.6 void mitkEuclideanSkeletonFeatureExtraction::SetInput (mitkVolume * *inData*) [inline]

Set the input data

Parameters:
inData the input data

8.80.2.7 void mitkEuclideanSkeletonFeatureExtraction::SetLowThreshValue (double *d*) [inline]

Set the low threshold of the target area

Parameters:
d the low threshold of the target area

The documentation for this class was generated from the following file:

- mitkEuclideanSkeletonFeatureExtraction.h

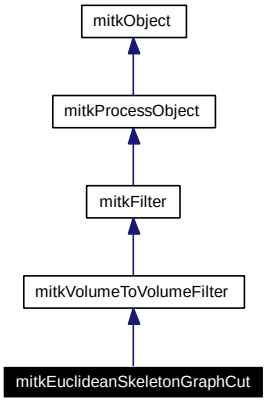
8.81 mitkEuclideanSkeletonGraphCut Class Reference

mitkEuclideanSkeletonGraphCut - A segmentation class implementing the Skeleton-Cut algorithm

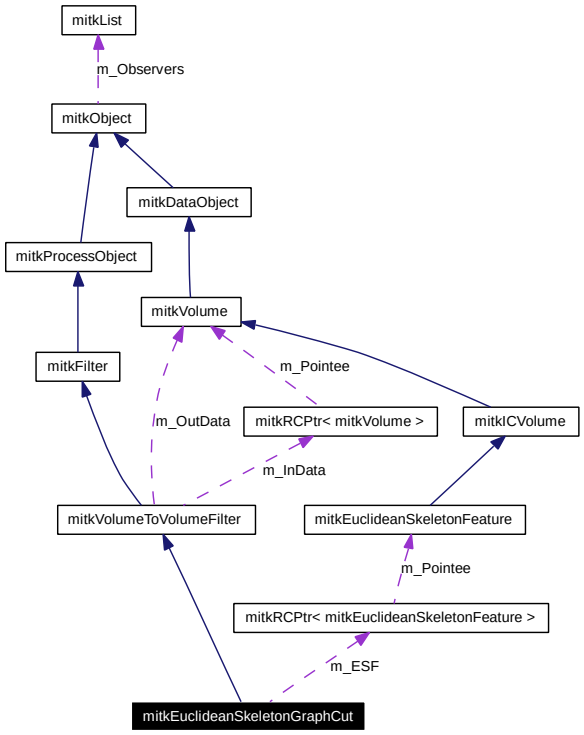
`#include <mitkEuclideanSkeletonGraphCut.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkEuclideanSkeletonGraphCut:



Collaboration diagram for mitkEuclideanSkeletonGraphCut:



Public Member Functions

- void SetEuclideanSkeletonFeature (mitkEuclideanSkeletonFeature *inData)
- mitkEuclideanSkeletonFeature * GetEuclideanSkeletonFeature ()
- void SetUserSelection (unsigned char *userSelection)
- void SetIntensityFactor (float intFac)
- void SetRadiusFactor (float radFac)
- void SetOutputMode (OutputMode om)
- void GetCutResult (unsigned char *cr)
- bool GenerateOutput (OutputMode om)

8.81.1 Detailed Description

mitkEuclideanSkeletonGraphCut - A segmentation class implementing the Skeleton-Cut algorithm

mitkEuclideanSkeletonGraphCut is a segmentation class implementing the Skeleton-Cut algorithm which is described in:Xiang, D. and Tian, J. and Yang, F. and Yang, Q. and Zhang, X. and Li, Q. and Liu, X. Skeleton Cuts - An Efficient Segmentation Method for Volume Rendering. IEEE Transactions on Visualization and Computer Graphics.

8.81.2 Member Function Documentation

8.81.2.1 bool mitkEuclideanSkeletonGraphCut::GenerateOutput (OutputMode *om*)

Generate a specific output

Parameters:
om the mode of output representation

8.81.2.2 void mitkEuclideanSkeletonGraphCut::GetCutResult (unsigned char * *cr*)

Return the cut result

Returns:
the cut result represented by a array, each element corresponding to a skeleton feature.

8.81.2.3 mitkEuclideanSkeletonFeature* mitkEuclideanSkeletonGraphCut::GetEuclideanSkeletonFeature () [inline]

Set the extracted skeleton features

Parameters:
inData the extracted skeleton features

8.81.2.4 void mitkEuclideanSkeletonGraphCut::SetEuclideanSkeletonFeature (mitkEuclideanSkeletonFeature * *inData*) [inline]

Set the extracted skeleton features

Parameters:
inData the extracted skeleton features

8.81.2.5 void mitkEuclideanSkeletonGraphCut::SetIntensityFactor (float *intFac*) [inline]

Set the Intensity Factor

Parameters:
intFac the Intensity Factor

8.81.2.6 void mitkEuclideanSkeletonGraphCut::SetOutputMode (OutputMode *om*) [inline]

Set the mode of output representation

Parameters:
om the mode of output representation

8.81.2.7 void mitkEuclideanSkeletonGraphCut::SetRadiusFactor (float *radFac*) [inline]

Set the Radius Factor

Parameters:
radFac the Radius Factor

8.81.2.8 void mitkEuclideanSkeletonGraphCut::SetUserSelection (unsigned char * *userSelection*)

Set the user selection represented by a array. Each element is corresponding to a skeleton feature. 0 for unselected, 1 for foreground selection, 2 for background selection

Parameters:
inData the extracted skeleton features

The documentation for this class was generated from the following file:

- mitkEuclideanSkeletonGraphCut.h

8.82 mitkFanBeamProjectData Class Reference

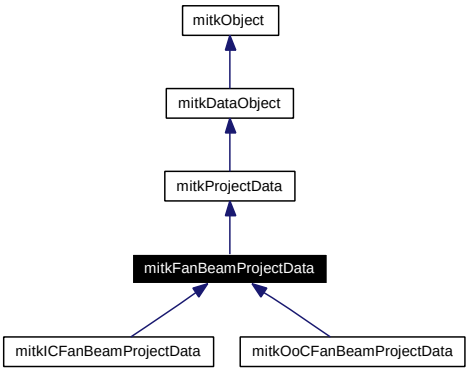
mitkFanBeamProjectData - abstract class to represent fan-beam project data.

`#include <mitkFanBeamProjectData.h>`

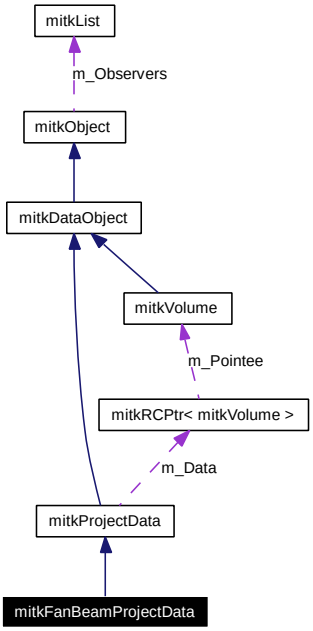
Inherits mitkProjectData.

Inherited by mitkICFanBeamProjectData, and mitkOoCFanBeamProjectData.

Inheritance diagram for mitkFanBeamProjectData:



Collaboration diagram for mitkFanBeamProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()
- virtual void const * GetData () const

- virtual void * GetData ()
- void SetData (mitkVolume *m)
- float GetDetectorMove ()
- void SetDetectorMove (float m)
- void SetDetectorLength (double m)
- double GetDetectorLength ()
- void SetFov (double *m)
- double * GetFov ()
- void SetXrayNum (int m)
- int GetXrayNum () const
- void SetSourceToCenterDistance (double m)
- double GetSourceToCenterDistance () const
- void SetSourceToDetectorDistance (double m)
- double GetSourceToDetectorDistance () const
- void SetSliceNum (int m)
- int GetSliceNum () const
- virtual unsigned long long GetActualMemorySize () const =0
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0
- virtual void const * GetSliceForRead (int sliceIdx)=0

8.82.1 Detailed Description

mitkFanBeamProjectData - abstract class to represent fan-beam project data.

mitkFanBeamProjectData - abstract class to represent fan-beam project data. . ... sod . . . . . o . ..... sdd . . . .

8.82.2 Member Function Documentation

8.82.2.1 virtual void mitkFanBeamProjectData::DeepCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkICFanBeamProjectData, and mitkOoCFanBeamProjectData.

8.82.2.2 virtual unsigned long long mitkFanBeamProjectData::GetActualMemorySize () const [pure virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Note:
Pure virtual function. Its concrete subclass must implement this function and return its memory size.

Implements mitkDataObject.

Implemented in mitkICFanBeamProjectData, and mitkOoCFanBeamProjectData.

8.82.2.3 virtual void* mitkFanBeamProjectData::GetData () [inline, virtual]

get volume pointer

Implements mitkProjectData.

8.82.2.4 virtual void const* mitkFanBeamProjectData::GetData () const [inline, virtual]

get a const volume pointer

Implements mitkProjectData.

8.82.2.5 double mitkFanBeamProjectData::GetDetectorLength () [inline]

// Get the length of detector

8.82.2.6 float mitkFanBeamProjectData::GetDetectorMove () [inline]

??

8.82.2.7 double* mitkFanBeamProjectData::GetFov () [inline]

Get field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.82.2.8 virtual void const* mitkFanBeamProjectData::GetSliceForRead (int *sliceIdx*) [pure virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implemented in mitkICFanBeamProjectData, and mitkOoCFanBeamProjectData.

8.82.2.9 int mitkFanBeamProjectData::GetSliceNum () const [inline]

Get Slice Number

8.82.2.10 double mitkFanBeamProjectData::GetSourceToCenterDistance () const [inline]

Get the distance between x-ray source and rotate center

8.82.2.11 double mitkFanBeamProjectData::GetSourceToDetectorDistance () const [inline]

Get the distance between x-ray source and detector center

8.82.2.12 int mitkFanBeamProjectData::GetXrayNum () const [inline]

Get number of Xray

8.82.2.13 virtual void mitkFanBeamProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkProjectData.

Reimplemented in mitkICFanBeamProjectData, and mitkOoCFanBeamProjectData.

8.82.2.14 virtual void mitkFanBeamProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkProjectData.

Reimplemented in mitkICFanBeamProjectData, and mitkOoCFanBeamProjectData.

8.82.2.15 void mitkFanBeamProjectData::SetData (mitkVolume * *m*)

Set volume pointer

8.82.2.16 void mitkFanBeamProjectData::SetDetectorLength (double *m*) [inline]

Set the length of detector

8.82.2.17 void mitkFanBeamProjectData::SetDetectorMove (float *m*) [inline]

??

8.82.2.18 void mitkFanBeamProjectData::SetFov (double * *m*) [inline]

Set field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.82.2.19 void mitkFanBeamProjectData::SetSliceNum (int *m*) [inline]

Set Slice Number

8.82.2.20 void mitkFanBeamProjectData::SetSourceToCenterDistance (double *m*) [inline]

Set the distance between x-ray source and rotate center

8.82.2.21 void mitkFanBeamProjectData::SetSourceToDetectorDistance (double *m*) [inline]

Set the distance between x-ray source and detector center

8.82.2.22 void mitkFanBeamProjectData::SetXrayNum (int *m*) [inline]

Set number of Xray

8.82.2.23 virtual void mitkFanBeamProjectData::ShallowCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkICFanBeamProjectData, and mitkOoCFanBeamProjectData.

The documentation for this class was generated from the following file:

- mitkFanBeamProjectData.h

8.83 mitkFanBeamReconFromCirle Class Reference

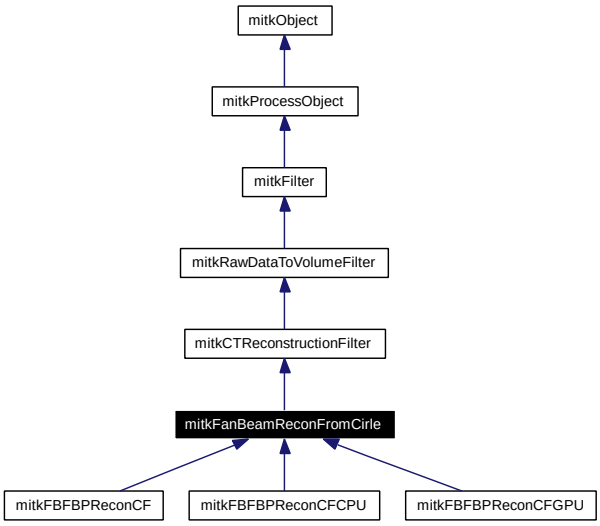
mitkFanBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered back-projection Algorithm.

`#include <mitkFanBeamReconFromCirle.h>`

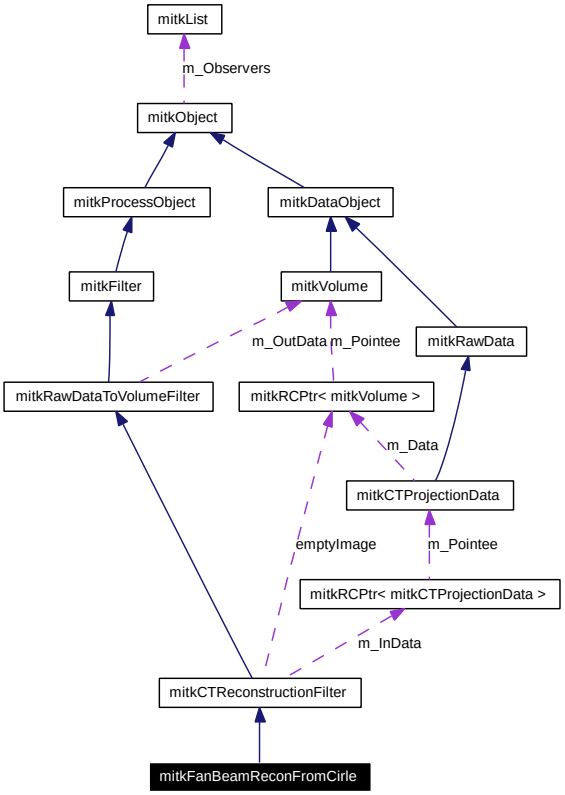
Inherits mitkCTReconstructionFilter.

Inherited by mitkFBFBPReconCF, mitkFBFBPReconCFCPU, and mitkFBFBPReconCFGPU.

Inheritance diagram for mitkFanBeamReconFromCirle:



Collaboration diagram for mitkFanBeamReconFromCirle:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- float GetDetectorMove ()
- void SetDetectorMove (float m)
- void SetDetectorLength (double m)
- double GetDetectorLength ()
- void SetFov (double *m)
- double * GetFov ()
- void SetSourceToCenterDistance (double m)
- double GetSourceToCenterDistance () const
- void SetSourceToDetectorDistance (double m)
- double GetSourceToDetectorDistance () const

8.83.1 Detailed Description

mitkFanBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered back-projection Algorithm.

mitkFanBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered back-projection Algorithm.

8.83.2 Member Function Documentation

8.83.2.1 double mitkFanBeamReconFromCirle::GetDetectorLength () [inline]

// Get the length of detector

8.83.2.2 float mitkFanBeamReconFromCirle::GetDetectorMove () [inline]

??

8.83.2.3 double* mitkFanBeamReconFromCirle::GetFov () [inline]

Get field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.83.2.4 double mitkFanBeamReconFromCirle::GetSourceToCenterDistance () const [inline]

Get the distance between x-ray source and rotate center

8.83.2.5 double mitkFanBeamReconFromCirle::GetSourceToDetectorDistance () const [inline]

Get the distance between x-ray source and detector center

8.83.2.6 virtual void mitkFanBeamReconFromCirle::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkCTReconstructionFilter.

Reimplemented in mitkFBFBPReconCF, mitkFBFBPReconCFCPU, and mitkFBFBPReconCFGPU.

8.83.2.7 void mitkFanBeamReconFromCirle::SetDetectorLength (double *m*) [inline]

Set the length of detector

8.83.2.8 void mitkFanBeamReconFromCirle::SetDetectorMove (float *m*) [inline]

??

8.83.2.9 void mitkFanBeamReconFromCirle::SetFov (double * *m*) [inline]

Set field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.83.2.10 void mitkFanBeamReconFromCirle::SetSourceToCenterDistance (double *m*) [inline]

Set the distance between x-ray source and rotate center

8.83.2.11 void mitkFanBeamReconFromCirle::SetSourceToDetectorDistance (double *m*) [inline]

Set the distance between x-ray source and detector center

The documentation for this class was generated from the following file:

- mitkFanBeamReconFromCirle.h

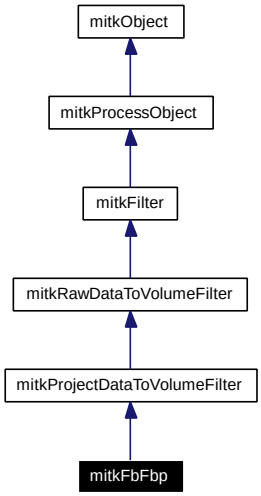
8.84 mitkFbFbp Class Reference

mitkFbFbp - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

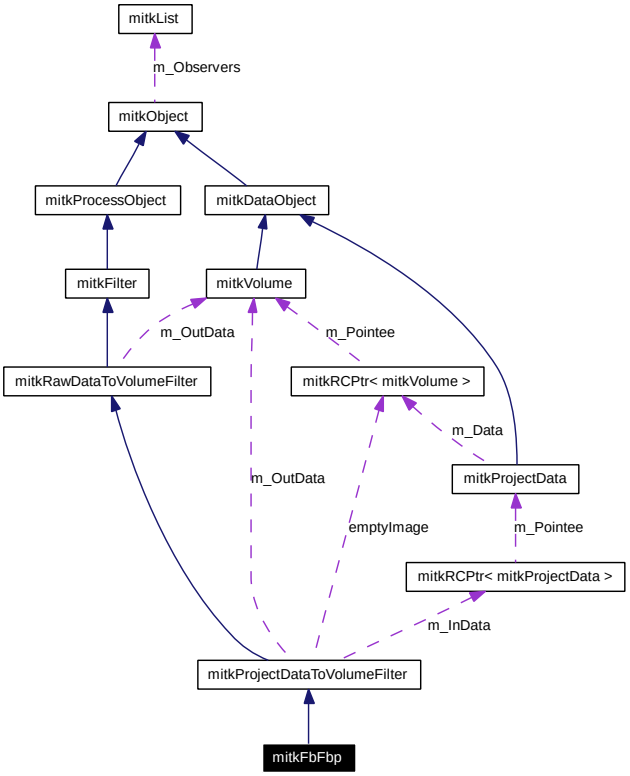
`#include <mitkFbFbp.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkFbFbp:



Collaboration diagram for mitkFbFbp:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.84.1 Detailed Description

mitkFbFbp - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

mitkFbFbp - concret class implement fan-beam reconstruction using filtered backprojection Algorithm, this implemention just using C++ without any acceleration

8.84.2 Member Function Documentation

8.84.2.1 virtual void mitkFbFbp::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkFbFbp.h

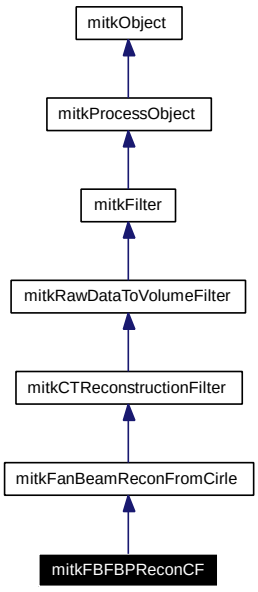
8.85 mitkFBFBPReconCF Class Reference

mitkFBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

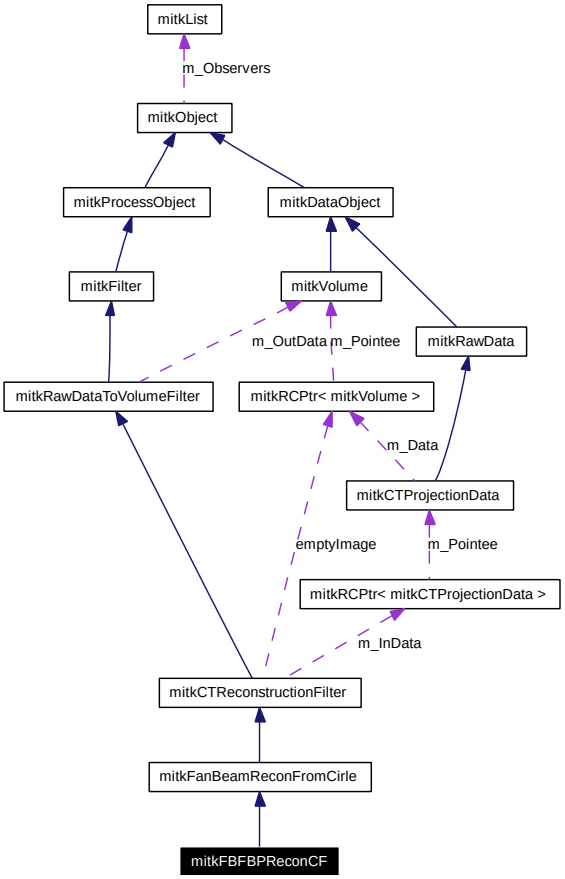
`#include <mitkFBFBPReconCF.h>`

Inherits mitkFanBeamReconFromCirle.

Inheritance diagram for mitkFBFBPReconCF:



Collaboration diagram for mitkFBFBPReconCF:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.85.1 Detailed Description

mitkFBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

mitkFBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm, this implemention just using C++ without any acceleration

8.85.2 Member Function Documentation

8.85.2.1 virtual void mitkFBFBPReconCF::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkFanBeamReconFromCirle.

The documentation for this class was generated from the following file:

- mitkFBFBPReconCF.h

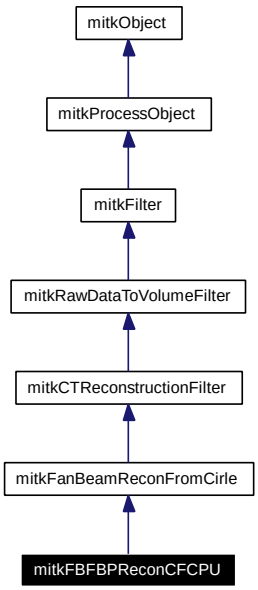
8.86 mitkFBFBPReconCFCPU Class Reference

mitkFBFBPReconCFCPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

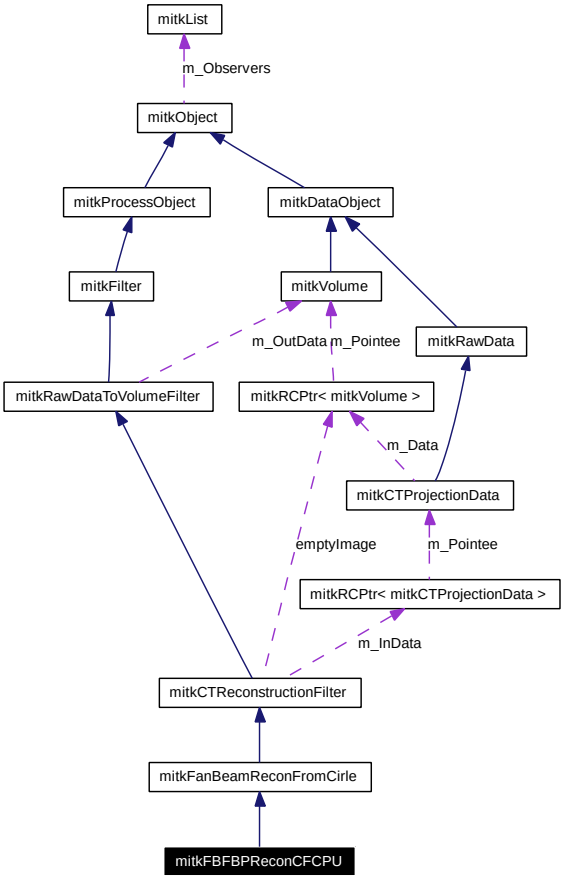
`#include <mitkFBFBPReconCFCPU.h>`

Inherits mitkFanBeamReconFromCirle.

Inheritance diagram for mitkFBFBPReconCFCPU:



Collaboration diagram for mitkFBFBPReconCFCPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.86.1 Detailed Description

mitkFBFBPReconCFCPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

mitkFBFBPReconCFCPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm. This implementation using SIMD for accelerating

8.86.2 Member Function Documentation

8.86.2.1 virtual void mitkFBFBPReconCFCPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkFanBeamReconFromCirle.

The documentation for this class was generated from the following file:

- mitkFBFBPReconCFCPU.h

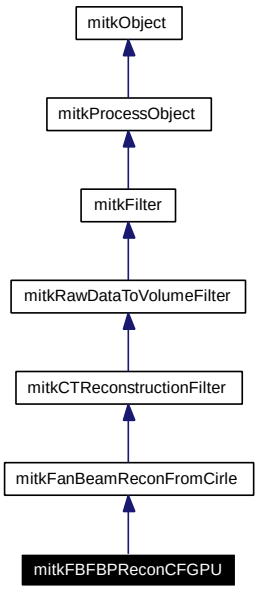
8.87 mitkFBFBPReconCFGPU Class Reference

mitkFBFBPReconCFGPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

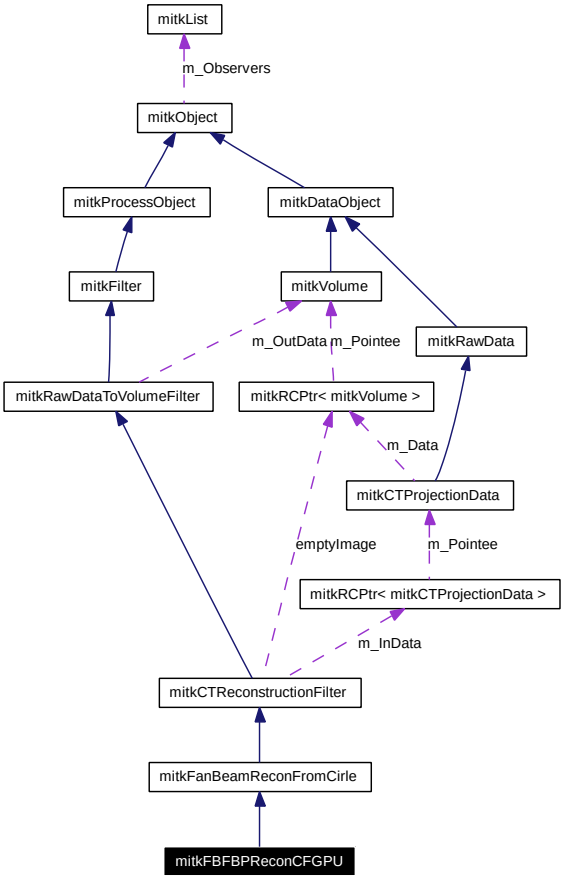
`#include <mitkFBFBPReconCFGPU.h>`

Inherits mitkFanBeamReconFromCirle.

Inheritance diagram for mitkFBFBPReconCFGPU:



Collaboration diagram for mitkFBFBPReconCFGPU:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.87.1 Detailed Description

mitkFBFBPReconCFGPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

mitkFBFBPReconCFGPU - concret class implement fan-beam reconstruction using filtered backprojection Algorithm. This implementation using SIMD for accelerating

8.87.2 Member Function Documentation

8.87.2.1 virtual void mitkFBFBPReconCFGPU::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkFanBeamReconFromCirle.

The documentation for this class was generated from the following file:

- mitkFBFBPReconCFGPU.h

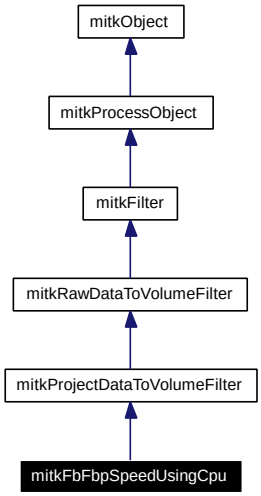
8.88 mitkFbFbpSpeedUsingCpu Class Reference

mitkFbFbpSpeedUsingCpu - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

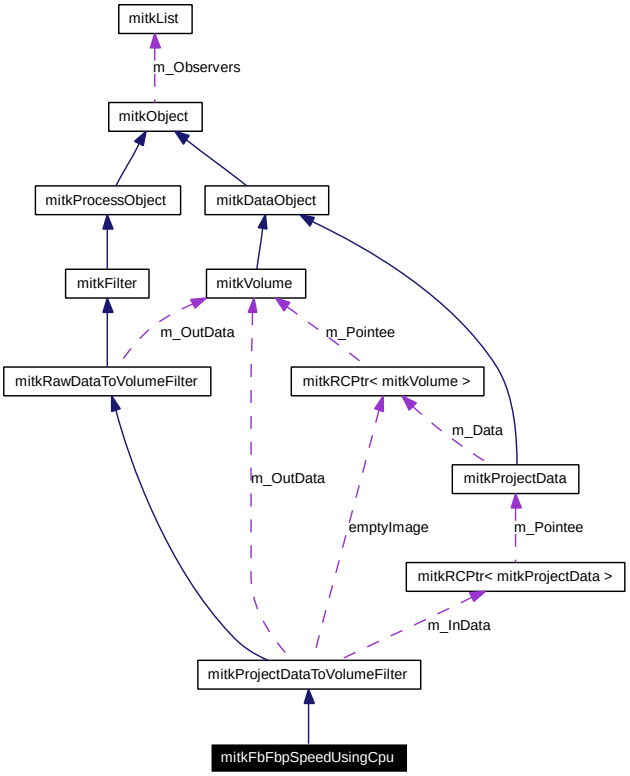
`#include <mitkFbFbpSpeedUsingCpu.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkFbFbpSpeedUsingCpu:



Collaboration diagram for mitkFbFbpSpeedUsingCpu:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.88.1 Detailed Description

mitkFbFbpSpeedUsingCpu - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

mitkFbFbpSpeedUsingCpu - concret class implement fan-beam reconstruction using filtered backprojection Algorithm. This implementation using SIMD for accelerating

8.88.2 Member Function Documentation

8.88.2.1 virtual void mitkFbFbpSpeedUsingCpu::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkFbFbpSpeedUsingCpu.h

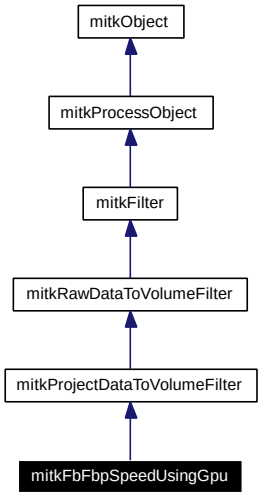
8.89 mitkFbFbpSpeedUsingGpu Class Reference

mitkFbFbpSpeedUsingGpu- concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

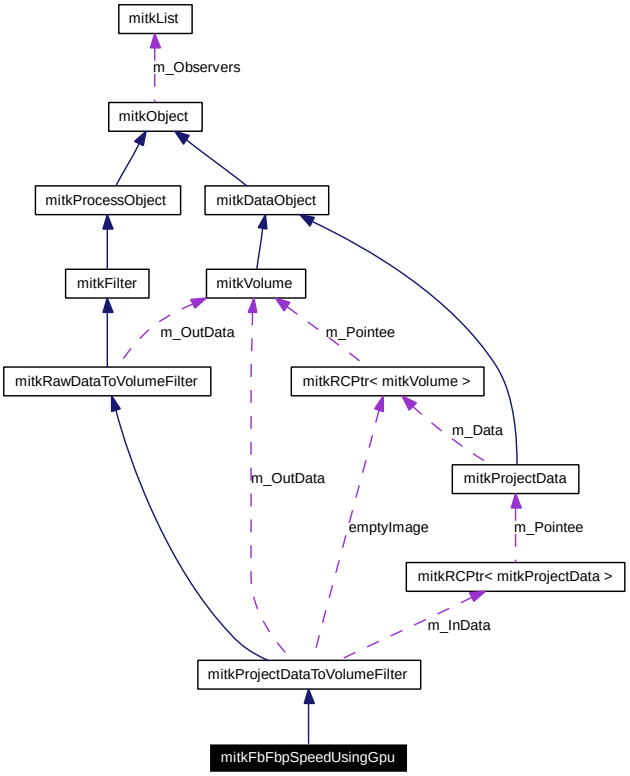
`#include <mitkFbfbpSpeedUsingGpu.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkFbFbpSpeedUsingGpu:



Collaboration diagram for mitkFbFbpSpeedUsingGpu:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.89.1 Detailed Description

mitkFbFbpSpeedUsingGpu- concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

mitkFbFbpSpeedUsingGpu- concret class implement fan-beam reconstruction using filtered backprojection Algorithm. This implementation using GPU for acceleration, and NVIDIA card is required, and 8800series or up is better.

8.89.2 Member Function Documentation

8.89.2.1 virtual void mitkFbFbpSpeedUsingGpu::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkFbfbpSpeedUsingGpu.h

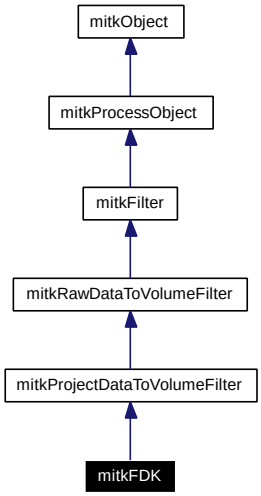
8.90 mitkFDK Class Reference

mitkFDK - concret class implement cone-beam reconstruction using FDK Algorithm.

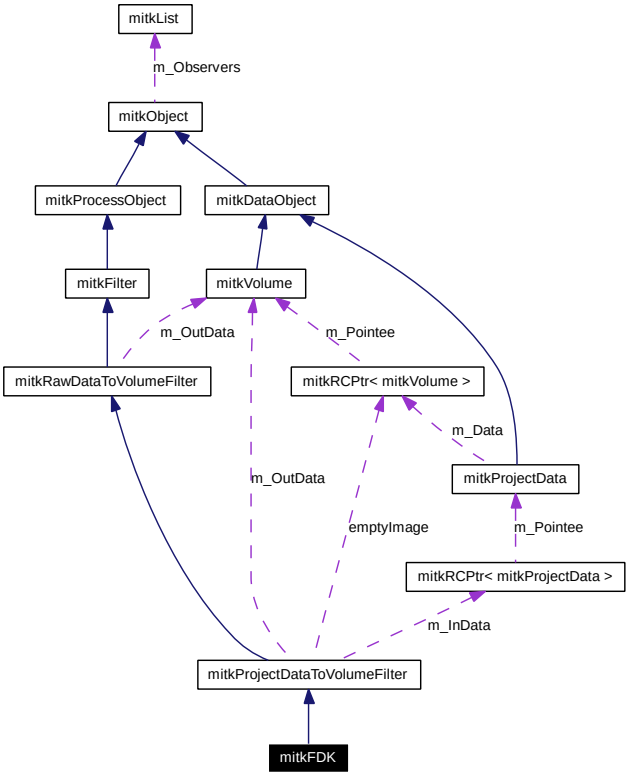
`#include <mitkFDK.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkFDK:



Collaboration diagram for mitkFDK:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.90.1 Detailed Description

mitkFDK - concret class implement cone-beam reconstruction using FDK Algorithm.

mitkFDK - concret class implement cone-beam reconstruction FDK, this implemention just using C++ without any acceleration

8.90.2 Member Function Documentation

8.90.2.1 virtual void mitkFDK::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkFDK.h

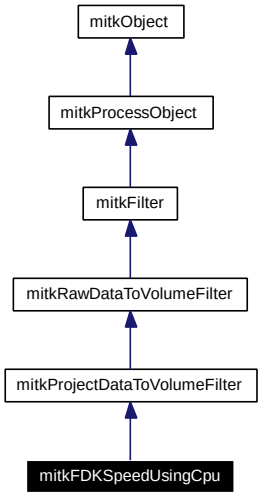
8.91 mitkFDKSpeedUsingCpu Class Reference

mitkFDKSpeedUsingCpu - concret class implement cone-beam reconstruction using FDK Algorithm.

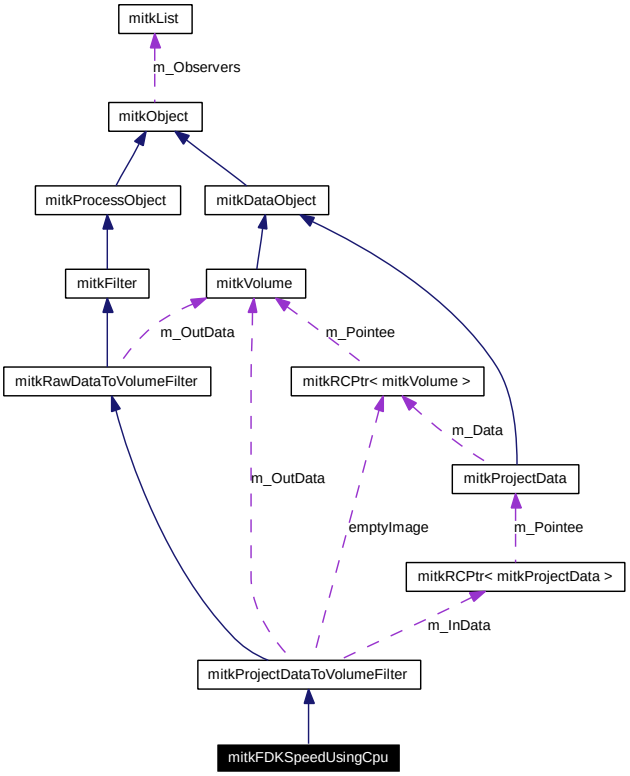
`#include <mitkFDKSpeedUsingCpu.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkFDKSpeedUsingCpu:



Collaboration diagram for mitkFDKSpeedUsingCpu:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.91.1 Detailed Description

mitkFDKSpeedUsingCpu - concret class implement cone-beam reconstruction using FDK Algorithm.

mitkFDKSpeedUsingCpu - concret class implement cone-beam reconstruction FDK, This implementation using SIMD for accelerating

8.91.2 Member Function Documentation

8.91.2.1 virtual void mitkFDKSpeedUsingCpu::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkFDKSpeedUsingCpu.h

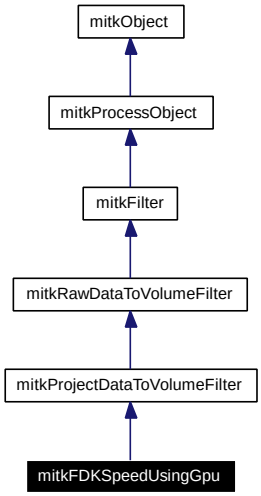
8.92 mitkFDKSpeedUsingGpu Class Reference

mitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction using FDK Algorithm.

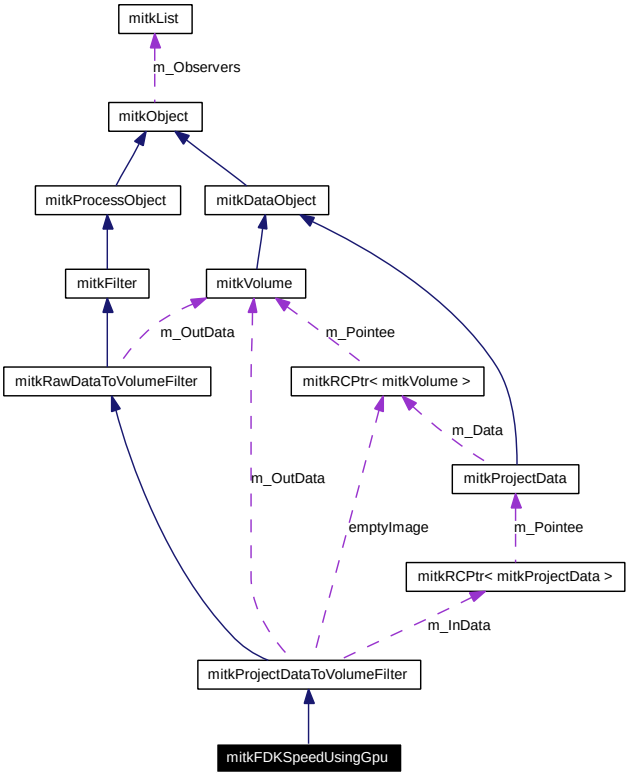
`#include <mitkFDKSpeedUsingGpu.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkFDKSpeedUsingGpu:



Collaboration diagram for mitkFDKSpeedUsingGpu:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.92.1 Detailed Description

mitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction using FDK Algorithm.

mitkFDKSpeedUsingGpu - concrete class implement cone-beam reconstruction FDK,This implementation

8.92.2 Member Function Documentation

8.92.2.1 virtual void mitkFDKSpeedUsingGpu::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkFDKSpeedUsingGpu.h

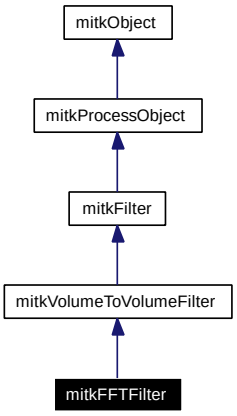
8.93 mitkFFTFilter Class Reference

mitkFFTFilter - a concrete class for implementation of 3D-FFT Transform

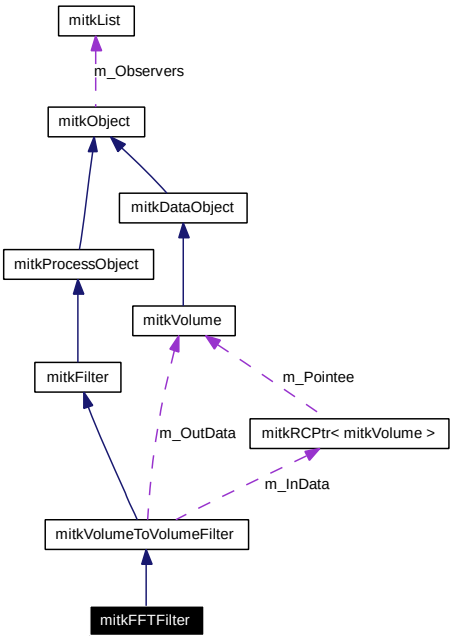
`#include <mitkFFTFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkFFTFilter:



Collaboration diagram for mitkFFTFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkVolume * GetMagnitude ()

8.93.1 Detailed Description

mitkFFTFilter - a concrete class for implementation of 3D-FFT Transform

mitkFFTFilter - a concrete class for implementation of 3D-FFT Transform

8.93.2 Member Function Documentation

8.93.2.1 mitkVolume* mitkFFTFilter::GetMagnitude ()

Retrive the magnitude volume of the FFT result

Returns:
the magnitude volume of the FFT result

8.93.2.2 virtual void mitkFFTFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkFFTFilter.h

8.94 mitkFilter Class Reference

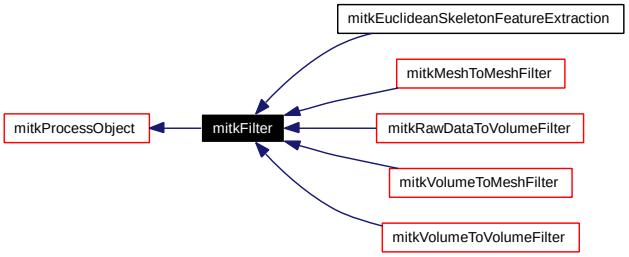
mitkFilter - abstract class specifies interface for filter object

`#include <mitkFilter.h>`

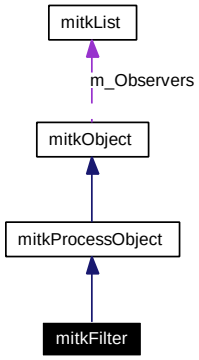
Inherits mitkProcessObject.

Inherited by mitkEuclideanSkeletonFeatureExtraction, mitkMeshToMeshFilter, mitkPhantomToVolumeFilter, mitkRawDataToVolumeFilter, mitkVolumeToMeshFilter, and mitkVolumeToVolumeFilter.

Inheritance diagram for mitkFilter:



Collaboration diagram for mitkFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.94.1 Detailed Description

mitkFilter - abstract class specifies interface for filter object

mitkFilter is an abstract object that specifies behavior and interface of filter objects. Filter object representation a algorithm that process input data and return the output data.

8.94.2 Member Function Documentation

8.94.2.1 virtual void mitkFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProcessObject.

Reimplemented in mitkBinaryFilter, mitkDiffusionFilter, mitkFFTFilter, mitkGaussianDerivativeImageFilter, mitkGaussianFilter, mitkMeshToMeshFilter, mitkNeumannBoundary, mitkQEMSimplification, mitkRGBToGrayFilter, mitkSheetFilter, mitkSobelEdgeDetectFilter, mitkTriangleMeshSimplification, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeResizeFilter, mitkVolumeResliceFilter, mitkVolumeToMeshFilter, mitkVolumeToVolumeFilter, mitk3DUSPixelBasedMT, mitk3DUSReconstructionFilter, mitkCBFDKReconCF, mitkCBFDKReconCFCPU, mitkCBTFDKReconCF, mitkConeBeamReconFromHelix, mitkCTReconstructionFilter, mitkFanBeamReconFromCirle, mitkFbFbp, mitkFBFBPReconCF, mitkFBFBPReconCFCPU, mitkFbFbpSpeedUsingCpu, mitkFDK, mitkFDKSpeedUsingCpu, mitkParallelBeamReconFromCirle, mitkPbFbp, mitkPBFBPReconCF, mitkProjectDataToVolumeFilter, mitkRawDataToVolumeFilter, mitkTFDK, mitkBSplineInterpolateFilter, mitkCubicInterpolateFilter, mitkInterpolateFilter, mitkLandmarkRegistrationFilter, mitkLinearInterpolateFilter, mitkNearestNeighborInterpolateFilter, mitkPointSetRegistrationFilter, mitkRegistrationFilter, mitkSubtractImageFilter, mitkConnectivityLabelFilter, mitkLiveWireImageFilter, mitkOtsuSegmentationFilter, mitkRegionGrowImageFilter, mitkSeedFillFilter, mitkThresholdSegmentationFilter, mitkBinMarchingCubes, mitkMarchingCubes, mitkCBFDKReconCFGPU, mitkFBFBPReconCFGPU, mitkFbFbpSpeedUsingGpu, mitkFDKSpeedUsingGpu, and mitk3DUSPixelBasedGPU.

The documentation for this class was generated from the following file:

- mitkFilter.h

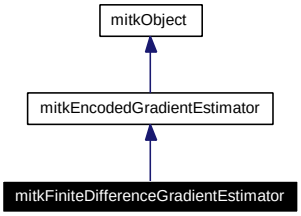
8.95 mitkFiniteDifferenceGradientEstimator Class Reference

mitkFiniteDifferenceGradientEstimator - a concrete class to use finite differences to estimate gradient

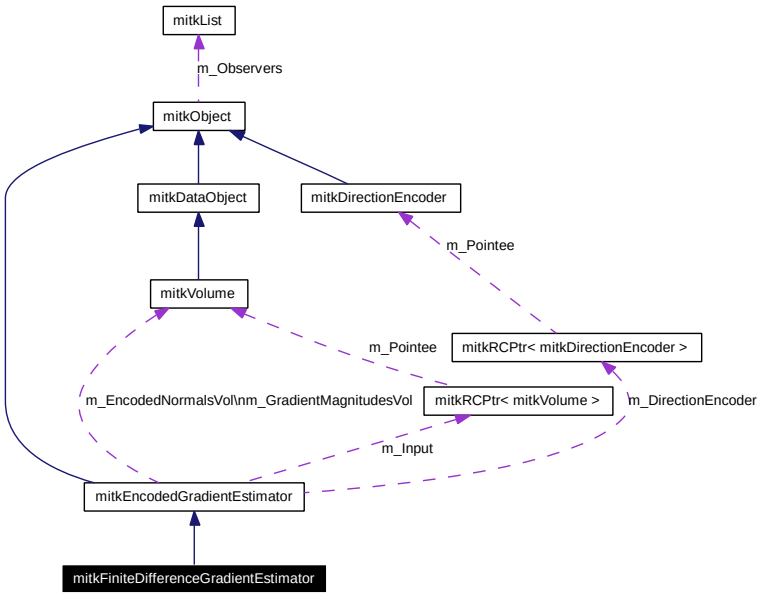
`#include <mitkFiniteDifferenceGradientEstimator.h>`

Inherits mitkEncodedGradientEstimator.

Inheritance diagram for mitkFiniteDifferenceGradientEstimator:



Collaboration diagram for mitkFiniteDifferenceGradientEstimator:



Public Member Functions

- void SetSampleSpacingInVoxels (int nVal)
- int GetSampleSpacingInVoxels ()

8.95.1 Detailed Description

mitkFiniteDifferenceGradientEstimator - a concrete class to use finite differences to estimate gradient

mitkFiniteDifferenceGradientEstimator is a concrete class to use finite differences to estimate gradient. Some codes are borrowed from VTK, and please see the copyright at end.

8.95.2 Member Function Documentation

8.95.2.1 int mitkFiniteDifferenceGradientEstimator::GetSampleSpacingInVoxels () [inline]

Returns:
Return the spacing between samples for the finite differences method used to compute the normal

8.95.2.2 void mitkFiniteDifferenceGradientEstimator::SetSampleSpacingInVoxels (int *nVal*) [inline]

Parameters:
nVal Represent the spacing between samples for the finite differences method used to compute the normal

The documentation for this class was generated from the following file:

- mitkFiniteDifferenceGradientEstimator.h

8.96 mitkFootprint Class Reference

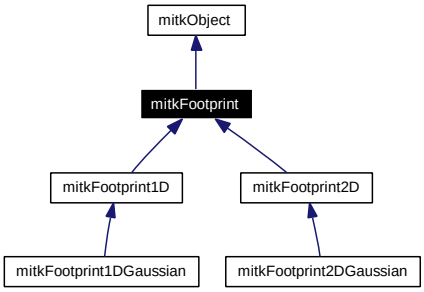
mitkFootprint - abstract class defines common interface for footprint

`#include <mitkFootprint.h>`

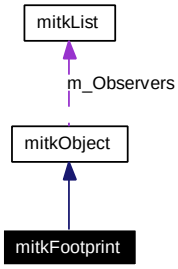
Inherits mitkObject.

Inherited by mitkFootprint1D, and mitkFootprint2D.

Inheritance diagram for mitkFootprint:



Collaboration diagram for mitkFootprint:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetRebuild (bool build)
- bool GetRebuild ()
- virtual float * EncodeFootprintTable ()
- virtual float DecodeFootprintTable (float x, float y)

8.96.1 Detailed Description

mitkFootprint - abstract class defines common interface for footprint

mitkFootprint is an abstract class that defines common interface for encoding and decoding of footprint table.

8.96.2 Member Function Documentation

8.96.2.1 virtual float mitkFootprint::DecodeFootprintTable (float *x*, float *y*) [inline, virtual]

Decode footprint table (each concrete class must implement this function).

Parameters:

x Coordinate value in horizon axis.

y Coordinate value in vertical axis.

Returns:

Return the corresponding value in the footprint table.

Reimplemented in mitkFootprint1DGaussian, and mitkFootprint2DGaussian.

8.96.2.2 virtual float* mitkFootprint::EncodeFootprintTable () [inline, virtual]

Encode footprint table (each concrete class must implement this function).

Returns:

Return the entry of the footprint table.

Reimplemented in mitkFootprint1DGaussian, and mitkFootprint2DGaussian.

8.96.2.3 bool mitkFootprint::GetRebuild () [inline]

Get the status of building.

Returns:

Return true, the building is enabled. Return false, the building is disabled.

8.96.2.4 virtual void mitkFootprint::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkFootprint1D, mitkFootprint1DGaussian, mitkFootprint2D, and mitkFootprint2DGaussian.

8.96.2.5 void mitkFootprint::SetRebuild (bool *build*) [inline]

Set the status of building.

Parameters:

build build = trueenable the rebuilding. build = falsedisable the rebuilding.

The documentation for this class was generated from the following file:

- mitkFootprint.h

8.97 mitkFootprint1D Class Reference

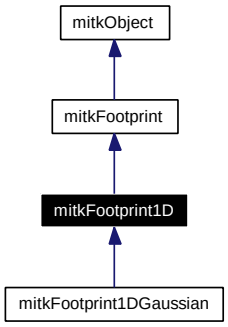
mitkFootprint1D - abstract class specifies interface for one dimensional footprint

`#include <mitkFootprint1D.h>`

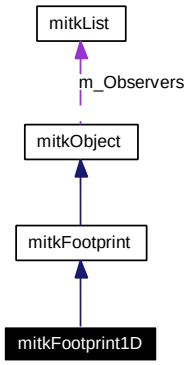
Inherits mitkFootprint.

Inherited by mitkFootprint1DGaussian.

Inheritance diagram for mitkFootprint1D:



Collaboration diagram for mitkFootprint1D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetRadiiSquare (float rs)
- void SetSampleNumber (int num)
- float GetRadiiSquare ()
- int GetSampleNumber ()

8.97.1 Detailed Description

mitkFootprint1D - abstract class specifies interface for one dimensional footprint

mitkFootprint1D is an abstract class that specifies interface for encoding and decoding of one dimensional footprint table.

8.97.2 Member Function Documentation

8.97.2.1 float mitkFootprint1D::GetRadiiSquare () [inline]

Get the footprint extent (radii square) in radial direction.

Returns:
Return the Radii square in radial direction.

8.97.2.2 int mitkFootprint1D::GetSampleNumber () [inline]

Get the sample number in radial direction.

Returns:
Return the sample number in radial direction.

8.97.2.3 virtual void mitkFootprint1D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFootprint.

Reimplemented in mitkFootprint1DGaussian.

8.97.2.4 void mitkFootprint1D::SetRadiiSquare (float *rs*) [inline]

Set the footprint extent (radii square) in radial direction.

Parameters:
rs Radii square in radial direction.

8.97.2.5 void mitkFootprint1D::SetSampleNumber (int *num*) [inline]

Set the sample number in radial direction.

Parameters:
num Sample number in radial direction.

The documentation for this class was generated from the following file:

- mitkFootprint1D.h

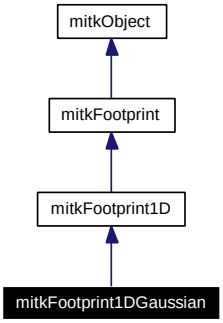
8.98 mitkFootprint1DGaussian Class Reference

mitkFootprint1DGaussian - concrete class for one dimensional footprint with Gaussian kernel

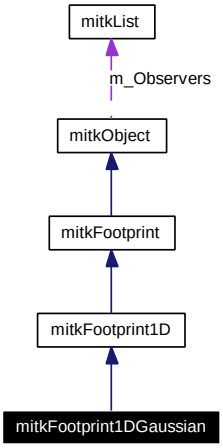
`#include <mitkFootprint1DGaussian.h>`

Inherits mitkFootprint1D.

Inheritance diagram for mitkFootprint1DGaussian:



Collaboration diagram for mitkFootprint1DGaussian:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkFootprint1DGaussian ()
- void SetCoefficient (float coeff)
- void SetAdjustRadii (float adjustradii)
- void SetVarianceMatrix (float vmatrix[4])
- float GetCoefficient ()
- float GetAdjustRadii ()
- void GetVarianceMatrix (float vmatrix[4])
- virtual float * EncodeFootprintTable ()
- virtual float DecodeFootprintTable (float x, float y)
- float DecodeFootprintTable (float rs)

8.98.1 Detailed Description

mitkFootprint1DGaussian - concrete class for one dimensional footprint with Gaussian kernel

mitkFootprint1DGaussian is a concrete class to encode and decode one dimensional footprint table using Gaussian kernel. The Gaussian footprint function is as follows: footprint(x,y) = coeff ∗ exp{-[(x,y) ∗ F^(-1) ∗ (x,y)']/adjustradii} , where "coeff" is the weight coefficient, adjustradii is the adjust radii coefficient, _ _ | vmatrix[0] vmatrix[2] | and "F" is the variance matrix = | | |_ vmatrix[1] vmatrix[3] _| the code snippet to use this class is:

```
mitkFootprint1DGaussian *footprint = new mitkFootprint1DGaussian;
footprint->SetCoefficient(coeff);
footprint->SetAdjustRadii(adjustradii);
footprint->SetVarianceMatrix(vmatrix);
footprint->SetRadiiSquare(rs);
footprint->SetSampleNumber(num);
table = footprint->EncodeFootprintTable();
.....
if(table)
  { value = footprint->DecodeFootprintTable(x, y); }
```

8.98.2 Constructor & Destructor Documentation

8.98.2.1 mitkFootprint1DGaussian::mitkFootprint1DGaussian ()

Constructor of the class

8.98.3 Member Function Documentation

8.98.3.1 float mitkFootprint1DGaussian::DecodeFootprintTable (float *rs*)

Decode the footprint table.

Parameters:

rs Radii square in radial direction.

Returns:

Return the corresponding value in the footprint table.

8.98.3.2 virtual float mitkFootprint1DGaussian::DecodeFootprintTable (float *x*, float *y*) [virtual]

Decode the footprint table.

Parameters:

x Coordinate value in horizon axis.

y Coordinate value in vertical axis.

Returns:

Return the corresponding value in the footprint table.

Reimplemented from mitkFootprint.

8.98.3.3 virtual float* mitkFootprint1DGaussian::EncodeFootprintTable () [virtual]

Encode the footprint table.

Returns:
Return the entry of the footprint table.

Reimplemented from mitkFootprint.

8.98.3.4 float mitkFootprint1DGaussian::GetAdjustRadii () [inline]

Get the adjust radii coefficient of Gaussian kernel.

Returns:
Return the adjust radii coefficient of Gaussian kernel.

8.98.3.5 float mitkFootprint1DGaussian::GetCoefficient () [inline]

Get the weight coefficient of Gaussian kernel.

Returns:
Return the weight coefficient of Gaussian kernel.

8.98.3.6 void mitkFootprint1DGaussian::GetVarianceMatrix (float *vmatrix*[4])

Get the variance matrix of Gaussian kernel.

Parameters:
vmatrix[0] Element of variance matrix at (row, column) = (0, 0).
vmatrix[1] Element of variance matrix at (row, column) = (1, 0).
vmatrix[2] Element of variance matrix at (row, column) = (0, 1).
vmatrix[3] Element of variance matrix at (row, column) = (1, 1).

8.98.3.7 virtual void mitkFootprint1DGaussian::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFootprint1D.

8.98.3.8 void mitkFootprint1DGaussian::SetAdjustRadii (float *adjustradii*) [inline]

Set the adjust radii coefficient of Gaussian kernel.

Parameters:
adjustradii The adjust radii coefficient of Gaussian kernel.

8.98.3.9 void mitkFootprint1DGaussian::SetCoefficient (float *coeff*) [inline]

Set the weight coefficient of Gaussian kernel.

Parameters:

coeff The weight coefficient of Gaussian kernel.

8.98.3.10 void mitkFootprint1DGaussian::SetVarianceMatrix (float *vmatrix*[4])

Set the variance matrix of Gaussian kernel.

Parameters:

vmatrix[0] Element of variance matrix at (row, column) = (0, 0).

vmatrix[1] Element of variance matrix at (row, column) = (1, 0).

vmatrix[2] Element of variance matrix at (row, column) = (0, 1).

vmatrix[3] Element of variance matrix at (row, column) = (1, 1).

The documentation for this class was generated from the following file:

- mitkFootprint1DGaussian.h

8.99 mitkFootprint2D Class Reference

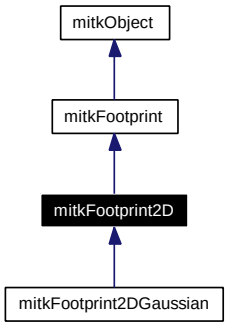
mitkFootprint2D - abstract class specifies interface for two dimensional footprint

`#include <mitkFootprint2D.h>`

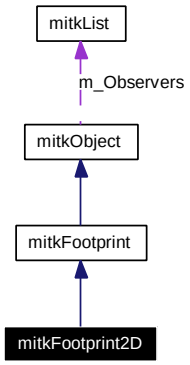
Inherits mitkFootprint.

Inherited by mitkFootprint2DGaussian.

Inheritance diagram for mitkFootprint2D:



Collaboration diagram for mitkFootprint2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetExtent (float ex, float ey)
- void SetSampleNumber (int numx, int numy)
- void GetExtent (float &ex, float &ey)
- void GetSampleNumber (int &numx, int &numy)

8.99.1 Detailed Description

mitkFootprint2D - abstract class specifies interface for two dimensional footprint

mitkFootprint2D is an abstract class that specifies interface for encoding and decoding of two dimensional footprint table.

8.99.2 Member Function Documentation

8.99.2.1 void mitkFootprint2D::GetExtent (float & *ex*, float & *ey*) [inline]

Get the positive footprint extent in horizon and vertical direction respectively.

Parameters:

ex The positive extent (half of the whole length) in horizon direction.

ey The positive extent (half of the whole length) in vertical direction.

8.99.2.2 void mitkFootprint2D::GetSampleNumber (int & *numx*, int & *numy*) [inline]

Get the sample number in horizon and vertical direction respectively.

Parameters:

numx The sample number in the positive horizon direction.

numy The sample number in the positive vertical direction.

8.99.2.3 virtual void mitkFootprint2D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkFootprint.

Reimplemented in mitkFootprint2DGaussian.

8.99.2.4 void mitkFootprint2D::SetExtent (float *ex*, float *ey*) [inline]

Set the positive footprint extent in horizon and vertical direction respectively.

Parameters:

ex The positive extent (half of the whole length) in horizon direction.

ey The positive extent (half of the whole length) in vertical direction.

8.99.2.5 void mitkFootprint2D::SetSampleNumber (int *numx*, int *numy*) [inline]

Set the sample number in horizon and vertical direction respectively.

Parameters:

numx The sample number in the positive horizon direction.

numy The sample number in the positive vertical direction.

The documentation for this class was generated from the following file:

- mitkFootprint2D.h

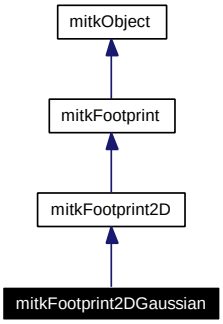
8.100 mitkFootprint2DGaussian Class Reference

mitkFootprint2DGaussian - concrete class for two dimensional footprint with Gaussian kernel

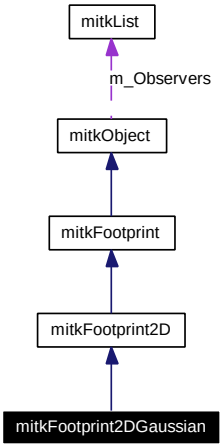
`#include <mitkFootprint2DGaussian.h>`

Inherits mitkFootprint2D.

Inheritance diagram for mitkFootprint2DGaussian:



Collaboration diagram for mitkFootprint2DGaussian:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkFootprint2DGaussian ()
- void SetCoefficient (float coeff)
- void SetVarianceMatrix (float vmatrix[4])
- void SetAdjustRadii (float adjustradii)
- float GetCoefficient ()
- void GetVarianceMatrix (float vmatrix[4])
- float GetAdjustRadii ()
- virtual float * EncodeFootprintTable ()
- virtual float DecodeFootprintTable (float x, float y)

8.100.1 Detailed Description

mitkFootprint2DGaussian - concrete class for two dimensional footprint with Gaussian kernel

mitkFootprint2DGaussian is a concrete class to encode and decode two dimensional footprint table using Gaussian kernel. The Gaussian footprint function is as follows: footprint(x,y) = coeff ∗ exp{-[(x,y) ∗ F^(-1) ∗ (x,y)']/adjustradii} , where "coeff" is the weight coefficient, adjustradii is the adjust radii coefficient, _ _ | vmatrix[0] vmatrix[2] | and "F" is the variance matrix = | | |_ vmatrix[1] vmatrix[3] _| the code snippet to use this class is:

```
mitkFootprint2DGaussian *footprint = new mitkFootprint2DGaussian;
footprint->SetCoefficient(coeff);
footprint->SetAdjustRadii(adjustradii);
footprint->SetVarianceMatrix(vmatrix);
footprint->SetExtent(ex, ey);
footprint->SetSampleNumber(numx, numy);
table = footprint->EncodeFootprintTable();
.....
if(table)
  { value = footprint->DecodeFootprintTable(x, y); }
```

8.100.2 Constructor & Destructor Documentation

8.100.2.1 mitkFootprint2DGaussian::mitkFootprint2DGaussian ()

Constructor of the class

8.100.3 Member Function Documentation

8.100.3.1 virtual float mitkFootprint2DGaussian::DecodeFootprintTable (float *x*, float *y*) [virtual]

Decode the footprint table.

Parameters:

x Coordinate value in horizon axis.

y Coordinate value in vertical axis.

Returns:

Return the corresponding value in the footprint table.

Reimplemented from mitkFootprint.

8.100.3.2 virtual float* mitkFootprint2DGaussian::EncodeFootprintTable () [virtual]

Encode the footprint table.

Returns:

Return the entry of the footprint table.

Reimplemented from mitkFootprint.

8.100.3.3 float mitkFootprint2DGaussian::GetAdjustRadii () [inline]

Get the adjust radii coefficient of Gaussian kernel.

Returns:
Return the adjust radii coefficient of Gaussian kernel.

8.100.3.4 float mitkFootprint2DGaussian::GetCoefficient () [inline]

Get the weight coefficient of Gaussian kernel.

Returns:
Return the weight coefficient of Gaussian kernel.

8.100.3.5 void mitkFootprint2DGaussian::GetVarianceMatrix (float *vmatrix*[4])

Get the variance matrix of Gaussian kernel.

Parameters:
vmatrix[0] Element of variance matrix at (row, column) = (0, 0).
vmatrix[1] Element of variance matrix at (row, column) = (1, 0).
vmatrix[2] Element of variance matrix at (row, column) = (0, 1).
vmatrix[3] Element of variance matrix at (row, column) = (1, 1).

8.100.3.6 virtual void mitkFootprint2DGaussian::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFootprint2D.

8.100.3.7 void mitkFootprint2DGaussian::SetAdjustRadii (float *adjustradii*) [inline]

Set the adjust radii coefficient of Gaussian kernel.

Parameters:
adjustradii The adjust radii coefficient of Gaussian kernel.

8.100.3.8 void mitkFootprint2DGaussian::SetCoefficient (float *coeff*) [inline]

Set the weight coefficient of Gaussian kernel.

Parameters:
coeff The weight coefficient of Gaussian kernel.

8.100.3.9 void mitkFootprint2DGaussian::SetVarianceMatrix (float *vmatrix*[4])

Set the variance matrix of Gaussian kernel.

Parameters:

vmatrix[0] Element of variance matrix at (row, column) = (0, 0).

vmatrix[1] Element of variance matrix at (row, column) = (1, 0).

vmatrix[2] Element of variance matrix at (row, column) = (0, 1).

vmatrix[3] Element of variance matrix at (row, column) = (1, 1).

The documentation for this class was generated from the following file:

- mitkFootprint2DGaussian.h

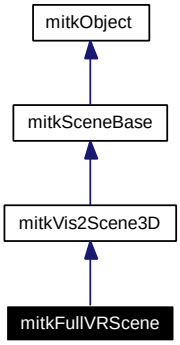
8.101 mitkFullVRScene Class Reference

mitkFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Optimized for multicore CPU.

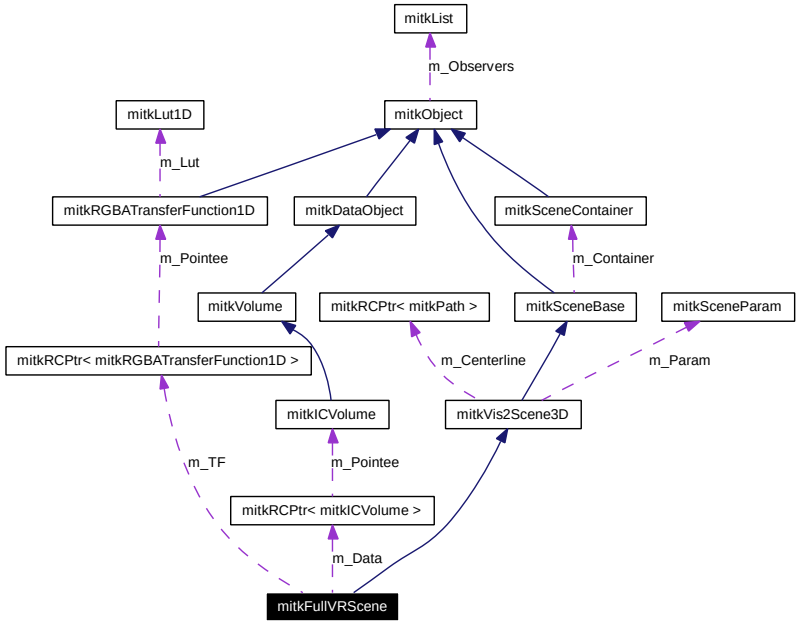
`#include <mitkFullVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkFullVRScene:



Collaboration diagram for mitkFullVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetCoherentData (mitkCoherentVolume *CVolume)

- void SetCoherentDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)
- mitkRGBATransferFunction1D * GetTransferFunction ()
- void SetTransferFunction (mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetShade (bool shade)
- bool GetShade ()
- void ShadeOn ()
- void ShadeOff ()
- void SetLightDirection (float x, float y, float z)
- void GetLightDirection (float &x, float &y, float &z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.101.1 Detailed Description

mitkFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Optimized for multicore CPU.

mitkFullVRScene is a 3D scene for shaded semi-transparent rendering from volume datasets. Optimized for multicore CPU. mitkFullVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container. Two kinds of data source are supported, ICVolume and CohrerentVolume.

8.101.2 Member Function Documentation

8.101.2.1 float mitkFullVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.101.2.2 void mitkFullVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.101.2.3 void mitkFullVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.101.2.4 float mitkFullVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.101.2.5 bool mitkFullVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:

: Return the illustrative shading status.

8.101.2.6 void mitkFullVRScene::GetLightDirection (float & *x*, float & *y*, float & *z*)

Return the light direction

Parameters:

x Return the x component of the light direction.

y Return the y component of the light direction.

z Return the z component of the light direction.

8.101.2.7 bool mitkFullVRScene::GetShade ()

Get the shading status

Returns:

Return the shading status.

8.101.2.8 float mitkFullVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.101.2.9 float mitkFullVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.101.2.10 float mitkFullVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.101.2.11 mitkRGBATransferFunction1D* mitkFullVRScene::GetTransferFunction ()

Get the transfer function

Returns:
Return the transfer function.

8.101.2.12 virtual void mitkFullVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.101.2.13 virtual void mitkFullVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.101.2.14 virtual void mitkFullVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.101.2.15 virtual void mitkFullVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.101.2.16 void mitkFullVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:

Whether turn the light rotation mode on.

8.101.2.17 void mitkFullVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:

value The ambient factor of the material of the isosurface

8.101.2.18 void mitkFullVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:

r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.101.2.19 void mitkFullVRScene::SetCoherentData (mitkCoherentVolume * *CVolume*)

Set the data object(CoherentVolume) to be rendered.

Parameters:

data The data object to be rendered.

8.101.2.20 void mitkFullVRScene::SetCoherentDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.101.2.21 void mitkFullVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.101.2.22 void mitkFullVRScene::SetData (mitkICVolume * *data*)

Set the data object(ICVolume) to be rendered.

Parameters:
data The data object to be rendered.

8.101.2.23 void mitkFullVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.101.2.24 void mitkFullVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.101.2.25 void mitkFullVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:
shade,: The illustrative shading status.

8.101.2.26 void mitkFullVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.

g Specify the green component of the light source color.

b Specify the blue component of the light source color.

8.101.2.27 void mitkFullVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.101.2.28 void mitkFullVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.101.2.29 void mitkFullVRScene::SetShade (bool *shade*)

Turn on or turn off shading.

Parameters:
shade The shading status.

8.101.2.30 void mitkFullVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.101.2.31 void mitkFullVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.101.2.32 void mitkFullVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:
stdSampleDistance The standard sample distance

8.101.2.33 void mitkFullVRScene::SetTransferFunction (mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

8.101.2.34 void mitkFullVRScene::ShadeOff ()

Turn off shading.

8.101.2.35 void mitkFullVRScene::ShadeOn ()

Turn on shading.

The documentation for this class was generated from the following file:

- mitkFullVRScene.h

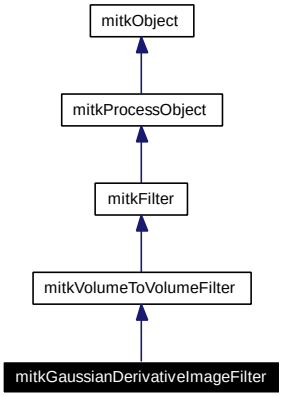
8.102 mitkGaussianDerivativeImageFilter Class Reference

mitkGaussianDerivativeImageFilter - a concrete class for implementation of Gaussian Derivative Filter

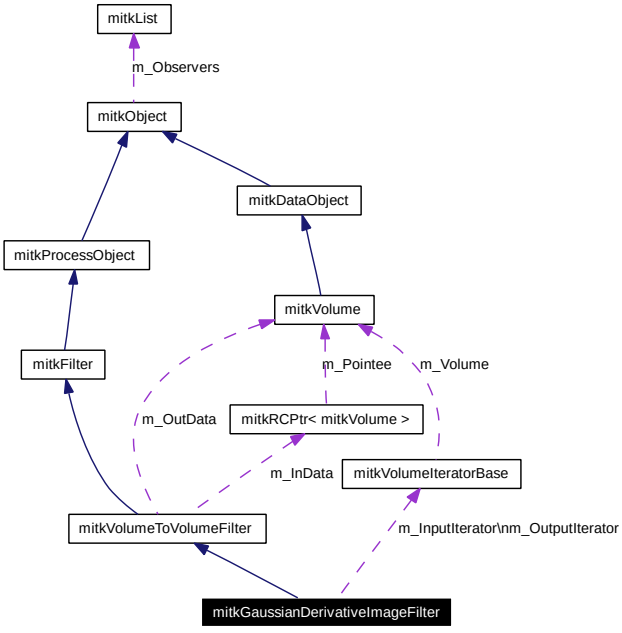
`#include <mitkGaussianDerivativeImageFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkGaussianDerivativeImageFilter:



Collaboration diagram for mitkGaussianDerivativeImageFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkGaussianDerivativeImageFilter ()
- virtual ~mitkGaussianDerivativeImageFilter ()

- void SetOutputDatatype (int dataType)

8.102.1 Detailed Description

mitkGaussianDerivativeImageFilter - a concrete class for implementation of Gaussian Derivative Filter

mitkGaussianDerivativeImageFilter - a concrete class for implementation of Gaussian Derivative Recursive Filter. It is used for calculating the gradient intensity of a image. The derivative based optimize method in registration framework will use this gradient intensities to compute the gradient descent direction.

In registration framework, this filter inputs the moving volume and outputs the gradient volume (2-channels in 2D and 3-channels in 3D).

8.102.2 Constructor & Destructor Documentation

8.102.2.1 mitkGaussianDerivativeImageFilter::mitkGaussianDerivativeImageFilter ()

Constructor.

8.102.2.2 virtual mitkGaussianDerivativeImageFilter::~mitkGaussianDerivativeImageFilter () [virtual]

Deconstructor.

8.102.3 Member Function Documentation

8.102.3.1 virtual void mitkGaussianDerivativeImageFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.102.3.2 void mitkGaussianDerivativeImageFilter::SetOutputDatatype (int *dataType*) [inline]

Set the data type for output volume.Default is MITK_FLOAT.

Parameters:

dataType Specify the output volume's data type.

The documentation for this class was generated from the following file:

- mitkGaussianDerivativeImageFilter.h

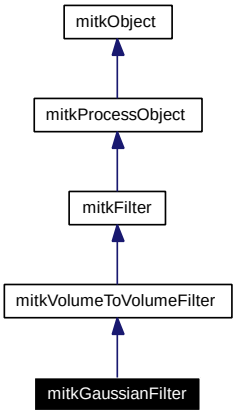
8.103 mitkGaussianFilter Class Reference

mitkGaussianFilter - a concrete class for implementation of Gaussian Smooth Filter

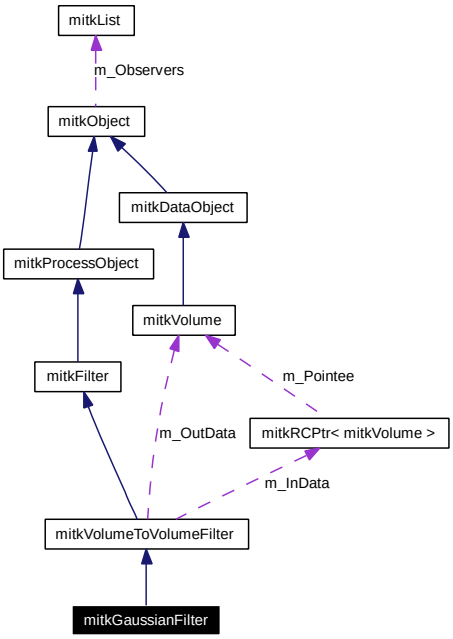
`#include <mitkGaussianFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkGaussianFilter:



Collaboration diagram for mitkGaussianFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetDimensionality (int Dimensionality)
- void SetStandardDeviation (float StandardDeviation)

8.103.1 Detailed Description

mitkGaussianFilter - a concrete class for implementation of Gaussian Smooth Filter

mitkGaussianFilter - a concrete class for implementation of Gaussian Smooth Filter.

8.103.2 Member Function Documentation

8.103.2.1 virtual void mitkGaussianFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.103.2.2 void mitkGaussianFilter::SetDimensionality (int *Dimensionality*)

Set the dimension of Gaussian Filter.

Parameters:
Dimensionality Specify the dimension. Default is 3.

8.103.2.3 void mitkGaussianFilter::SetStandardDeviation (float *StandardDeviation*)

Set the standard deviation of Gaussian Filter.

Parameters:
StandardDeviation Specify the standard deviation. Default is 1.

The documentation for this class was generated from the following file:

- mitkGaussianFilter.h

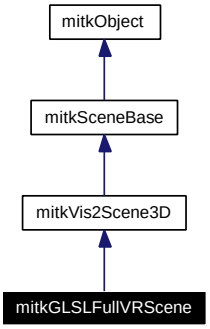
8.104 mitkGLSLFullVRScene Class Reference

mitkGLSLFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the GLSL interface.

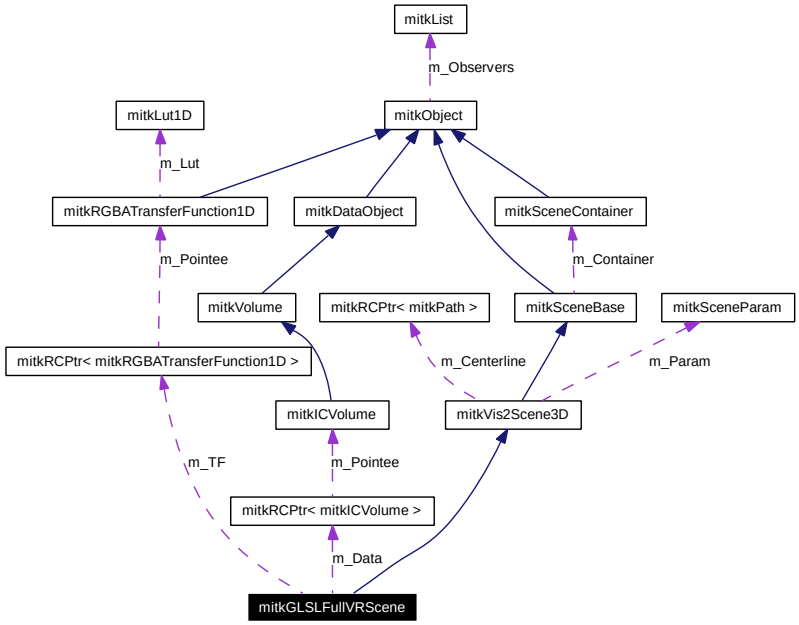
`#include <mitkGLSLFullVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkGLSLFullVRScene:



Collaboration diagram for mitkGLSLFullVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)

- void GetBackGroundColor (float &r, float &g, float &b)
- mitkRGBATransferFunction1D * GetTransferFunction ()
- void SetTransferFunction (mitkRGBATransferFunction1D *tf)
- void SetStdSampleDistance (float stdSampleDistance)
- float GetStdSampleDistance ()
- void SetShade (bool shade)
- bool GetShade ()
- void ShadeOn ()
- void ShadeOff ()
- void SetLightDirection (float x, float y, float z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void SetSampleDistance (float sd)
- float GetSampleDistance ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)

8.104.1 Detailed Description

mitkGLSLFullVRScene - a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the GLSL interface.

mitkGLSLFullVRScene is a 3D scene for shaded semi-transparent rendering from volume datasets. Accelerated with the GLSL interface. mitkGLSLFullVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.104.2 Member Function Documentation

8.104.2.1 float mitkGLSLFullVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.104.2.2 void mitkGLSLFullVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.104.2.3 float mitkGLSLFullVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.104.2.4 float mitkGLSLFullVRScene::GetSampleDistance ()

Get the sampling distance of ray traversal.

Returns:

Return the sampling distance of ray traversal.

8.104.2.5 bool mitkGLSLFullVRScene::GetShade ()

Get the shading status

Returns:

Return the shading status.

8.104.2.6 float mitkGLSLFullVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:

Return The specular factor of the material of the isosurface

8.104.2.7 float mitkGLSLFullVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:

Return The specular power of the material of the isosurface

8.104.2.8 float mitkGLSLFullVRScene::GetStdSampleDistance ()

Get the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Returns:
Return the standard sample distance

8.104.2.9 mitkRGBATransferFunction1D* mitkGLSLFullVRScene::GetTransferFunction ()

Get the transfer function

Returns:
Return the transfer function.

8.104.2.10 virtual void mitkGLSLFullVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.104.2.11 virtual void mitkGLSLFullVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.104.2.12 virtual void mitkGLSLFullVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.104.2.13 virtual void mitkGLSLFullVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.104.2.14 void mitkGLSLFullVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:
Whether turn the light rotation mode on.

8.104.2.15 void mitkGLSLFullVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:
value The ambient factor of the material of the isosurface

8.104.2.16 void mitkGLSLFullVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:
r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.104.2.17 void mitkGLSLFullVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.104.2.18 void mitkGLSLFullVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.104.2.19 void mitkGLSLFullVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.104.2.20 void mitkGLSLFullVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.

g Specify the green component of the light source color.

b Specify the blue component of the light source color.

8.104.2.21 void mitkGLSLFullVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
- ***x*** Specify the x component of the light direction.
- ***y*** Specify the y component of the light direction.
- ***z*** Specify the z component of the light direction.

8.104.2.22 void mitkGLSLFullVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
- ***intensity*** Light's intensity

8.104.2.23 void mitkGLSLFullVRScene::SetSampleDistance (float *sd*)

Set the sampling distance of ray traversal.

Parameters:
- ***sd*** The sampling distance of ray traversal.

8.104.2.24 void mitkGLSLFullVRScene::SetShade (bool *shade*)

Turn on or turn off shading.

Parameters:
- ***shade*** The shading status.

8.104.2.25 void mitkGLSLFullVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
- ***value*** The specular factor of the material of the isosurface

8.104.2.26 void mitkGLSLFullVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
- ***value*** The specular power of the material of the isosurface

8.104.2.27 void mitkGLSLFullVRScene::SetStdSampleDistance (float *stdSampleDistance*)

Set the standard sample distance where the opacity of each step equals to the alpha channel of the transfered value.

Parameters:
stdSampleDistance The standard sample distance

8.104.2.28 void mitkGLSLFullVRScene::SetTransferFunction (mitkRGBATransferFunction1D * *tf*)

Set the transfer function

Parameters:
tf The transfer function.

8.104.2.29 void mitkGLSLFullVRScene::ShadeOff ()

Turn off shading.

8.104.2.30 void mitkGLSLFullVRScene::ShadeOn ()

Turn on shading.

The documentation for this class was generated from the following file:

- mitkGLSLFullVRScene.h

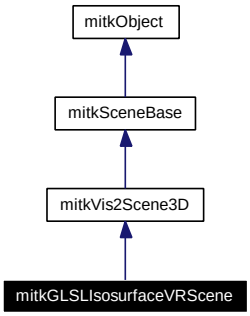
8.105 mitkGLSLIsosurfaceVRScene Class Reference

mitkGLSLIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the GLSL interface.

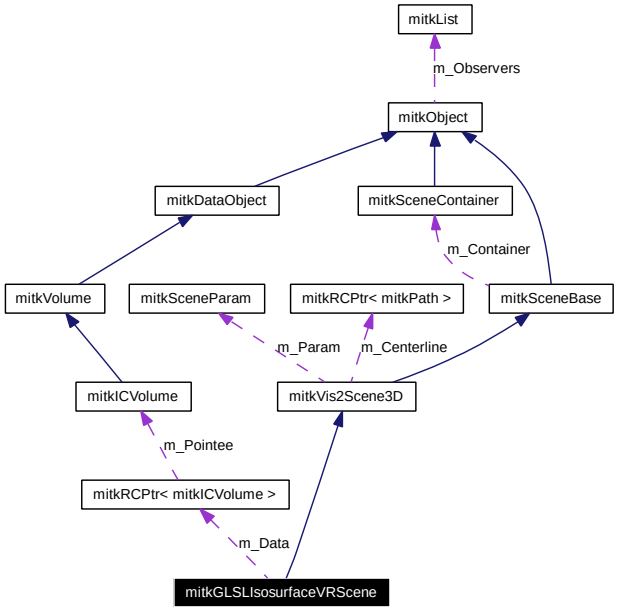
`#include <mitkGLSLIsosurfaceVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkGLSLIsosurfaceVRScene:



Collaboration diagram for mitkGLSLIsosurfaceVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- void SetIsovalue (float isovalue)
- float GetIsovalue ()
- void SetSurfaceColor (float r, float g, float b)
- void GetSurfaceColor (float &r, float &g, float &b)
- void SetLightDirection (float x, float y, float z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void SetSampleDistance (float sd)
- float GetSampleDistance ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)

8.105.1 Detailed Description

mitkGLSLIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Accelerated with the GLSL interface.

mitkGLSLIsosurfaceVRScene is a 3D scene to display isosurfaces from volume datasets. Accelerated with the GLSL interface. mitkGLSLIsosurfaceVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.105.2 Member Function Documentation

8.105.2.1 float mitkGLSLIsosurfaceVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.105.2.2 void mitkGLSLIsosurfaceVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:
- ***r*** Return the red component of the background color.
- ***g*** Return the green component of the background color.
- ***b*** Return the blue component of the background color.

8.105.2.3 float mitkGLSLIsosurfaceVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:
Return The diffuse factor of the material of the isosurface

8.105.2.4 float mitkGLSLIsosurfaceVRScene::GetIsovalue ()

Get the isovalue of the isosurface to be displayed.

Returns:
Return the isovalue.

8.105.2.5 float mitkGLSLIsosurfaceVRScene::GetSampleDistance ()

Get the sampling distance of ray traversal.

Returns:
Return the sampling distance of ray traversal.

8.105.2.6 float mitkGLSLIsosurfaceVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.105.2.7 float mitkGLSLIsosurfaceVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.105.2.8 void mitkGLSLIsosurfaceVRScene::GetSurfaceColor (float & *r*, float & *g*, float & *b*)

Get the color of the isosurface to be displayed

Parameters:
r Return the red component of the surface color.

g Return the green component of the surface color.

b Return the blue component of the surface color.

8.105.2.9 virtual void mitkGLSLIsosurfaceVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.105.2.10 virtual void mitkGLSLIsosurfaceVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.105.2.11 virtual void mitkGLSLIsosurfaceVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.105.2.12 virtual void mitkGLSLIsosurfaceVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.105.2.13 void mitkGLSLIsosurfaceVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:

Whether turn the light rotation mode on.

8.105.2.14 void mitkGLSLIsosurfaceVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:

value The ambient factor of the material of the isosurface

8.105.2.15 void mitkGLSLIsosurfaceVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:

r Specify the red component of the background color.

g Specify the green component of the background color.

b Specify the blue component of the background color.

8.105.2.16 void mitkGLSLIsosurfaceVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.105.2.17 void mitkGLSLIsosurfaceVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.105.2.18 void mitkGLSLIsosurfaceVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.105.2.19 void mitkGLSLIsosurfaceVRScene::SetIsovalue (float *isovalue*)

Set the isovalue of the isosurface to be displayed.

Parameters:
isovalue The isovalue.

8.105.2.20 void mitkGLSLIsosurfaceVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.

g Specify the green component of the light source color.

b Specify the blue component of the light source color.

8.105.2.21 void mitkGLSLIsosurfaceVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
x Specify the x component of the light direction.

y Specify the y component of the light direction.

z Specify the z component of the light direction.

8.105.2.22 void mitkGLSLIsosurfaceVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.105.2.23 void mitkGLSLIsosurfaceVRScene::SetSampleDistance (float *sd*)

Set the sampling distance of ray traversal.

Parameters:
sd The sampling distance of ray traversal.

8.105.2.24 void mitkGLSLIsosurfaceVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.105.2.25 void mitkGLSLIsosurfaceVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.105.2.26 void mitkGLSLIsosurfaceVRScene::SetSurfaceColor (float *r*, float *g*, float *b*)

Set the color of the isosurface to be displayed

Parameters:
r Specify the red component of the surface color.
g Specify the green component of the surface color.
b Specify the blue component of the surface color.

The documentation for this class was generated from the following file:

- mitkGLSLIsosurfaceVRScene.h

8.106 mitkGLSLLut1D Class Reference

mitkGLSLLut1D - A data class for 1D-LookUPTable,

`#include <mitkGLSLLut1D.h>`

8.106.1 Detailed Description

mitkGLSLLut1D - A data class for 1D-LookUPTable,

mitkGLSLLut1D is a data class for 1D-LookUPTable

The documentation for this class was generated from the following file:

- mitkGLSLLut1D.h

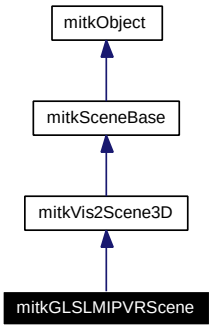
8.107 mitkGLSLMIPVRScene Class Reference

mitkGLSLMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Accelerated with the GLSL interface.

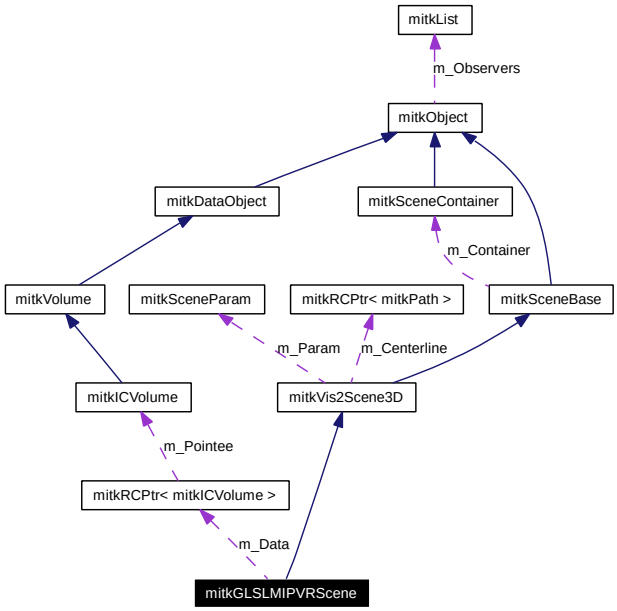
`#include <mitkGLSLMIPVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkGLSLMIPVRScene:



Collaboration diagram for mitkGLSLMIPVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetSampleDistance (float sd)
- float GetSampleDistance ()

- void SetWidthCenter (float windowWidth, float windowCenter)
- void SetWindowWidth (float windowWidth)
- void SetWindowCenter (float windowCenter)
- float GetWindowWidth ()
- float GetWindowCenter ()

8.107.1 Detailed Description

mitkGLSLMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Accelerated with the GLSL interface.

mitkGLSLMIPVRScene is a 3D scene for MIP Rendering from volume datasets. Accelerated with the GLSL interface. mitkGLSLMIPVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container.

8.107.2 Member Function Documentation

8.107.2.1 float mitkGLSLMIPVRScene::GetSampleDistance ()

Get the sampling distance of ray traversal.

Returns:
Return the sampling distance of ray traversal.

8.107.2.2 float mitkGLSLMIPVRScene::GetWindowCenter ()

Get the window center for the MIP

Returns:
Return the window center of the MIP

8.107.2.3 float mitkGLSLMIPVRScene::GetWindowWidth ()

Get the window width for the MIP

Returns:
Return the window width of the MIP

8.107.2.4 virtual void mitkGLSLMIPVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.107.2.5 void mitkGLSLMIPVRScene::SetData (mitkICVolume * *data*)

Set the data object to be rendered

Parameters:
data The data object to be rendered.

8.107.2.6 void mitkGLSLMIPVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.107.2.7 void mitkGLSLMIPVRScene::SetSampleDistance (float *sd*)

Set the sampling distance of ray traversal.

Parameters:
sd The sampling distance of ray traversal.

8.107.2.8 void mitkGLSLMIPVRScene::SetWidthCenter (float *windowWidth*, float *windowCenter*)

Set the window width and windows center for the MIP

Parameters:
windowWidth The window width of the MIP
windowCenter The window center of the MIP

8.107.2.9 void mitkGLSLMIPVRScene::SetWindowCenter (float *windowCenter*)

Set the window center for the MIP

Parameters:
windowCenter The window center of the MIP

8.107.2.10 void mitkGLSLMIPVRScene::SetWindowWidth (float *windowWidth*)

Set the window width for the MIP

Parameters:
windowWidth The window width of the MIP

The documentation for this class was generated from the following file:

- mitkGLSLMIPVRScene.h

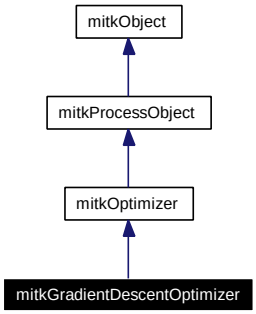
8.108 mitkGradientDescentOptimizer Class Reference

mitkGradientDescentOptimizer - a concrete class for implementation of the Gradient Descent algorithm.

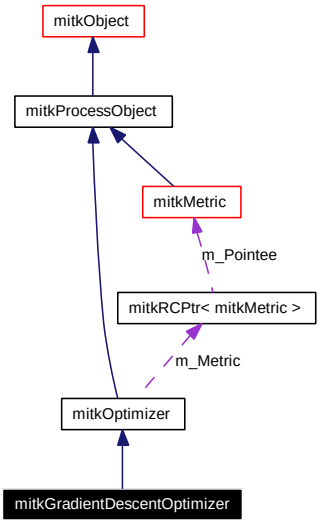
`#include <mitkGradientDescentOptimizer.h>`

Inherits mitkOptimizer.

Inheritance diagram for mitkGradientDescentOptimizer:



Collaboration diagram for mitkGradientDescentOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkGradientDescentOptimizer ()
- double GetCurrentStepLength ()
- double GetMaximumStepLength ()
- double GetMinimumStepLength ()
- void SetCurrentStepLength (double length)
- void SetMaximumStepLength (double length)
- void SetMinimumStepLength (double length)
- void SetGradientMagnitudeTolerance (double tolerance)

- void SetRelaxationFactor (double factor)

8.108.1 Detailed Description

mitkGradientDescentOptimizer - a concrete class for implementation of the Gradient Descent algorithm.

mitkGradientDescentOptimizer - a concrete class for implementation of the Gradient Descent algorithm. Gradient Descent algorithm is a derivative based optimize method which needs image's gradient intensities to determine the optimization directions. In each recursive step, the algorithm will get current position in parameter space by evaluating the cost function and compute the next position following the equation:

nextPosition = currentPostion + direction $*$ step length

The Gradient Descent algorithm is a simple, robust method and the convergence speed is fast.

8.108.2 Constructor & Destructor Documentation

8.108.2.1 mitkGradientDescentOptimizer::mitkGradientDescentOptimizer ()

Constructor.

8.108.3 Member Function Documentation

8.108.3.1 double mitkGradientDescentOptimizer::GetCurrentStepLength () [inline]

Get the current step length of optimization.

Returns:
Return the current step length.

8.108.3.2 double mitkGradientDescentOptimizer::GetMaximumStepLength () [inline]

Get the maximum step length of optimization.

Returns:
Return the maximum step length.

8.108.3.3 double mitkGradientDescentOptimizer::GetMinimumStepLength () [inline]

Get the minimum step length of optimization, implement in child class.

Returns:
Return the minimum step length.

8.108.3.4 virtual void mitkGradientDescentOptimizer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkOptimizer.

8.108.3.5 void mitkGradientDescentOptimizer::SetCurrentStepLength (double *length*) [inline]

Set the current step length of optimization.

Parameters:
length The current step length.

8.108.3.6 void mitkGradientDescentOptimizer::SetGradientMagnitudeTolerance (double *tolerance*) [inline]

Set the gradient magnitude tolerance value.

Parameters:
tolerance The tolerance value.

8.108.3.7 void mitkGradientDescentOptimizer::SetMaximumStepLength (double *length*) [inline]

Set the maximum step length of optimization.

Parameters:
length The maximum step length.

8.108.3.8 void mitkGradientDescentOptimizer::SetMinimumStepLength (double *length*) [inline]

Set the minimum step length of optimization.

Parameters:
length The minimum step length.

8.108.3.9 void mitkGradientDescentOptimizer::SetRelaxationFactor (double *factor*) [inline]

Set the relaxation factor for step updating.

Parameters:
factor The factor value.

The documentation for this class was generated from the following file:

- mitkGradientDescentOptimizer.h

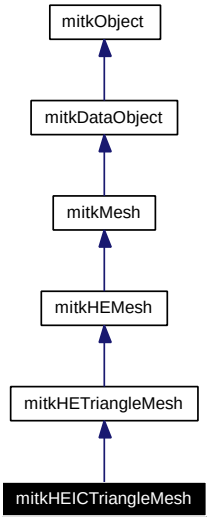
8.109 mitkHEICTriangleMesh Class Reference

mitkHEICTriangleMesh - a concrete class for in-core triangle meshes represented by Half Edges

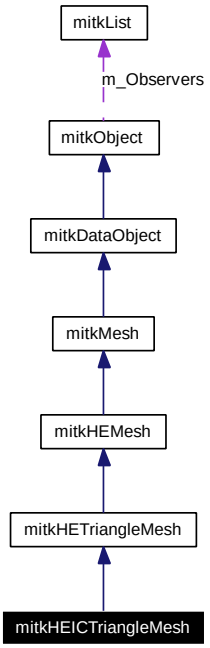
`#include <mitkHEICTriangleMesh.h>`

Inherits mitkHETriangleMesh.

Inheritance diagram for mitkHEICTriangleMesh:



Collaboration diagram for mitkHEICTriangleMesh:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkHEICTriangleMesh ()
- virtual int GetDataObjectType () const
- virtual void Initialize ()
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void SetVertexNumber (size_type number)
- virtual void SetFaceNumber (size_type number)
- virtual Vertex & GetVertex (VertexHandle v) const
- virtual Edge & GetEdge (EdgeHandle e) const
- virtual Edge & GetEdge (HalfEdgeHandle he) const
- virtual HalfEdge & GetHalfEdge (HalfEdgeHandle he) const
- virtual Face & GetFace (FaceHandle f) const
- virtual VertexHandle GetHandle (Vertex const &vert) const
- virtual EdgeHandle GetHandle (Edge const &edge) const
- virtual HalfEdgeHandle GetHandle (HalfEdge const &he) const
- virtual FaceHandle GetHandle (Face const &face) const
- virtual float * GetVertexData ()
- virtual index_type * GetFaceData ()
- void ClearTempVertArray ()
- void ClearTempFaceArray ()

8.109.1 Detailed Description

mitkHEICTriangleMesh - a concrete class for in-core triangle meshes represented by Half Edges

mitkHEICTriangleMesh is a concrete class for in-core triangle meshes represented by Half Edges. It supports in-core data sets which can be loaded into the main memory entirely.

See also:
mitkHEMesh and mitkHETriangleMesh for access interfaces
mitkHEOoCTriangleMesh for out-of-core data set

8.109.2 Constructor & Destructor Documentation

8.109.2.1 mitkHEICTriangleMesh::mitkHEICTriangleMesh ()

Default constructor.

8.109.3 Member Function Documentation

8.109.3.1 void mitkHEICTriangleMesh::ClearTempFaceArray ()

Clear the temporary memory allocated in the function GetFaceData().

Warning:
Do not call this function until you no longer use the face data gotten from GetFaceData();

8.109.3.2 void mitkHEICTriangleMesh::ClearTempVertArray ()

Clear the temporary memory allocated in the function GetVertexData().

Warning:
Do not call this function until you no longer use the vertex data gotten from GetVertexData();

8.109.3.3 virtual void mitkHEICTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkHEMesh.

8.109.3.4 virtual unsigned long long mitkHEICTriangleMesh::GetActualMemorySize () const [virtual]

Get the actual size of the data in bytes.

Returns:
Return the actual size of the data in bytes.

Implements mitkDataObject.

8.109.3.5 virtual int mitkHEICTriangleMesh::GetDataObjectType () const [inline, virtual]

Return what type of data object this is.

Returns:
Return the type of this data object.

Reimplemented from mitkHEMesh.

8.109.3.6 virtual Edge& mitkHEICTriangleMesh::GetEdge (HalfEdgeHandle *he*) const [inline, virtual]

Get edge by half edge handle.

Parameters:
he one half edge handle of the required edge

Returns:
Return the reference to the required edge.

Note:
An edge is composed by two opposite half edges.

8.109.3.7 virtual Edge& mitkHEICTriangleMesh::GetEdge (EdgeHandle *e*) const [inline, virtual]

Get edge by handle.

Parameters:
e the handle of the required edge

Returns:
Return the reference to the required edge.

8.109.3.8 virtual Face& mitkHEICTriangleMesh::GetFace (FaceHandle *f*) const [inline, virtual]

Get face by handle.

Parameters:
f the handle of the required face

Returns:
Return the reference to the required face.

8.109.3.9 virtual index_type* mitkHEICTriangleMesh::GetFaceData () [virtual]

Get data pointer of this face data. This is for compatibility with mitkTriangleMesh which is used more frequently.

Returns:
Return a unsigned int pointer to the face data (indices to vertices).

Warning:
- This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes;
- DO NOT delete the pointer you got from this function. If you must release the memory, call ClearTempVertArray() to do this.

Implements mitkMesh.

8.109.3.10 virtual HalfEdge& mitkHEICTriangleMesh::GetHalfEdge (HalfEdgeHandle *he*) const [inline, virtual]

Get half edge by handle.

Parameters:
he the handle of the required half edge

Returns:
Return the reference to the required half edge.

8.109.3.11 virtual FaceHandle mitkHEICTriangleMesh::GetHandle (Face const & *face*) const [inline, virtual]

Get the handle of a face.

Parameters:
face the reference to the face

Returns:
Return the handle of the face.

8.109.3.12 virtual HalfEdgeHandle mitkHEICTriangleMesh::GetHandle (HalfEdge const & *he*) const [inline, virtual]

Get the handle of a half edge.

Parameters:
he the reference to the half edge

Returns:
Return the handle of the half edge.

8.109.3.13 virtual EdgeHandle mitkHEICTriangleMesh::GetHandle (Edge const & *edge*) const [inline, virtual]

Get the handle of an edge.

Parameters:
edge the reference to the edge

Returns:
Return the handle of the edge.

8.109.3.14 virtual VertexHandle mitkHEICTriangleMesh::GetHandle (Vertex const & *vert*) const [inline, virtual]

Get the handle of a vertex.

Parameters:
vert the reference to the vertex

Returns:
Return the handle of the vertex.

8.109.3.15 virtual Vertex& mitkHEICTriangleMesh::GetVertex (VertexHandle *v*) const [inline, virtual]

Get vertex by handle.

Parameters:
v the handle of the required vertex

Returns:
Return the reference to the required vertex.

8.109.3.16 virtual float* mitkHEICTriangleMesh::GetVertexData () [virtual]

Get data pointer of this vertex data. This is for compatibility with mitkTriangleMesh which is used more frequently.

Returns:
Return a float pointer to the vertex data.

Warning:
This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes.

Implements mitkMesh.

8.109.3.17 virtual void mitkHEICTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkHETriangleMesh.

8.109.3.18 virtual void mitkHEICTriangleMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkHETriangleMesh.

8.109.3.19 virtual void mitkHEICTriangleMesh::SetFaceNumber (size_type *number*) [virtual]

Set the faces' number and allocate memory.

Parameters:
number the number of faces

Warning:
This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes.

Implements mitkMesh.

8.109.3.20 virtual void mitkHEICTriangleMesh::SetVertexNumber (size_type *number*) [virtual]

Set the vertices' number and allocate memory.

Parameters:
number the number of vertices

Note:
This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes.

Implements mitkMesh.

8.109.3.21 virtual void mitkHEICTriangleMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkHEMesh.

The documentation for this class was generated from the following file:

- mitkHEICTriangleMesh.h

8.110 mitkHEMesh Class Reference

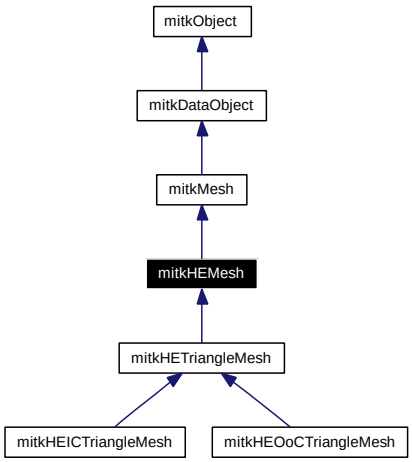
mitkHEMesh - a concrete class for polygon meshes represented by Half Edges

`#include <mitkHEMesh.h>`

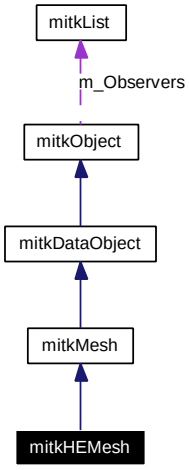
Inherits mitkMesh.

Inherited by mitkHETriangleMesh.

Inheritance diagram for mitkHEMesh:



Collaboration diagram for mitkHEMesh:



Public Types

- typedef VertexVertexIterT< mitkHEMesh > VertexVertexIterator
- typedef VertexOHalfedgeIterT< mitkHEMesh > VertexOHalfedgeIterator
- typedef VertexIHalfedgeIterT< mitkHEMesh > VertexIHalfedgeIterator
- typedef VertexEdgeIterT< mitkHEMesh > VertexEdgeIterator
- typedef VertexFaceIterT< mitkHEMesh > VertexFaceIterator

- typedef FaceVertexIterT< mitkHEMesh > FaceVertexIterator
- typedef FaceHalfedgeIterT< mitkHEMesh > FaceHalfedgeIterator
- typedef FaceEdgeIterT< mitkHEMesh > FaceEdgeIterator
- typedef FaceFaceIterT< mitkHEMesh > FaceFaceIterator
- typedef VertexIterT< mitkHEMesh > VertexIterator
- typedef EdgeIterT< mitkHEMesh > EdgeIterator
- typedef FaceIterT< mitkHEMesh > FaceIterator

Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkHEMesh ()
- virtual int GetDataObjectType () const
- virtual void Initialize ()
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual size_type GetVertexNumber () const
- virtual size_type GetFaceNumber () const
- size_type GetEdgeNumber () const
- void ClearGarbage ()
- VertexHandle AddVertex (Vertex const &vert)
- VertexHandle AddVertex (Point3f const &point, Point3f const &normal)
- VertexHandle AddVertex (float x, float y, float z, float nx=0.0f, float ny=0.0f, float nz=0.0f)
- VertexHandle AddVertex (float v[6])
- FaceHandle AddFace (unsigned int n, VertexHandle *vertices)
- void DeleteVertex (VertexHandle vert)
- void DeleteEdge (EdgeHandle edge, bool deleteIsolatedVertices)
- void DeleteFace (FaceHandle face, bool deleteIsolatedVertices)
- HalfEdgeHandle FindHalfEdge (VertexHandle vert0, VertexHandle vert1)
- bool IsBoundary (HalfEdgeHandle halfedge) const
- bool IsBoundary (VertexHandle vert) const
- bool IsBoundary (EdgeHandle edge) const
- bool IsManifold (VertexHandle vert)
- virtual Vertex & GetVertex (VertexHandle v) const =0
- virtual Edge & GetEdge (EdgeHandle e) const =0
- virtual Edge & GetEdge (HalfEdgeHandle he) const =0
- virtual HalfEdge & GetHalfEdge (HalfEdgeHandle he) const =0
- virtual Face & GetFace (FaceHandle f) const =0
- virtual VertexHandle GetHandle (Vertex const &vert) const =0
- virtual EdgeHandle GetHandle (Edge const &edge) const =0
- virtual HalfEdgeHandle GetHandle (HalfEdge const &he) const =0
- virtual FaceHandle GetHandle (Face const &face) const =0
- EdgeHandle GetEdgeHandle (HalfEdgeHandle he) const
- HalfEdgeHandle PairHalfEdge (HalfEdgeHandle he) const
- HalfEdgeHandle NextHalfEdge (HalfEdgeHandle he) const
- HalfEdgeHandle PrevHalfEdge (HalfEdgeHandle he) const
- VertexHandle EndVertex (HalfEdgeHandle he) const
- FaceHandle AdjFace (HalfEdgeHandle he) const
- HalfEdgeHandle OutHalfEdge (VertexHandle v) const
- HalfEdgeHandle OneHalfEdge (FaceHandle f) const

- bool IsValid (VertexHandle v) const
- bool IsValid (EdgeHandle e) const
- bool IsValid (HalfEdgeHandle he) const
- bool IsValid (FaceHandle f) const
- size_type GetValidVertexNumber () const
- size_type GetValidEdgeNumber () const
- size_type GetValidFaceNumber () const
- virtual bool TestClockwise ()
- void ClearFlags ()

8.110.1 Detailed Description

mitkHEMesh - a concrete class for polygon meshes represented by Half Edges

mitkHEMesh is a concrete class for polygon meshes which are represented by Half Edge Type. This implementation of half edge references to OpenMesh written by Computer Graphics Group, RWTH Aachen.

Some Structs Used in this Class

HEVertex

Besides the point coordinates and normal of the vertex, Vertex struct also contains the handle of outgoing half edge. It is defined as follows:

```
struct HEVertex : public _flag_base
{
    union
    {
        Vertex3f vert;
        struct
        {
            Point3f point;
            Point3f normal;
        };
    };
    HEHalfEdgeHandle outHalfEdge; // handle of outgoing half edge
};
```

HEHalfEdge

It contains the handles of its end vertex, opposite half edge, next half edge, previous half edge and adjacent face. It is defined as follows:

```
struct HEHalfEdge
{
    HEVertexHandle endVertex;      // handle of end vertex
    HEHalfEdgeHandle pairHalfEdge; // handle of opposite half edge
    HEHalfEdgeHandle nextHalfEdge; // handle of next half edge
    HEHalfEdgeHandle prevHalfEdge; // handle of previous half edge
    HEFaceHandle face;
};
```

HEEdge

It contains two opposite half edges, defined as follows:

```
struct HEEdge : public _flag_base
{
    HEHalfEdge halfEdge[2];
};
```

HEFace

It contains the handle of one of its half edges, defined as follows:

```
struct _face : public _flag_base
{
    HEHalfEdgeHandle oneHalfEdge; // handle of one half edge
};
```

Furthermore, HEVertex, HEEdge and HEFace are all derived from _flag_base, which provides a set of methods to make some special marks on these items when processing a mitkHEMesh in your algorithm. You can make and unmake 31 different marks from "flag0" to "flag30" through the methods _flag_base::SetFlag(FLAGS f) and _flag_base::UnSetFlag(FLAGS f), and test these flags through the method _flag_base::GetFlag(FLAGS f). _flag_base::SetDeleted(bool del=true) is reserved by MITK, which means some procedures inside MITK will make "delete" marks after deleting some items. You can test if an item is deleted by MITK through the method _flag_base::IsDeleted().

Handles

Handles including HEVertexHandle, HEHalfEdgeHandle, HEEdgeHandle and HEFaceHandle are encapsulations of index. Methods in mitkHEMesh use handles to deal with items instead of using indices directly. You can get index from a handle through the method *HandleIdx(), and test a handle's validation through the method *HandleIsValid().

The detailed definition can be found in mitkHalfEdgeStructures.h and mitkGeometryTypes.h.

8.110.2 Member Typedef Documentation

8.110.2.1 typedef EdgeIterT<mitkHEMesh> mitkHEMesh::EdgeIterator

EdgeIterator is an iterator class for iterating over all edges of a mitkHEMesh object.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
mitkHEMesh::EdgeIterator iter(mesh);
for ( ; iter; ++iter)
{
    // iter is the pointer to current edge
    // *iter is the reference to current edge
    // iter.Handle() returns the handle of current edge
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.2 typedef FaceEdgeIterT<mitkHEMesh> mitkHEMesh::FaceEdgeIterator

FaceEdgeIterator is an iterator class for iterating over a face's edges.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the face
mitkHEMesh::FaceEdgeIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current edge
    // *iter is the reference to current edge
    // iter.Handle() returns the handle of current edge
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.3 typedef FaceFaceIterT<mitkHEMesh> mitkHEMesh::FaceFaceIterator

FaceFaceIterator is an iterator class for iterating over a face's all edge-neighboring faces.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the face
mitkHEMesh::FaceFaceIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current edge-neighboring face
    // *iter is the reference to current edge-neighboring face
    // iter.Handle() returns the handle of current edge-neighboring face
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.4 typedef FaceHalfedgeIterT<mitkHEMesh> mitkHEMesh::FaceHalfedgeIterator

FaceHalfedgeIterator is an iterator class for iterating over a face's half edges.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the face
mitkHEMesh::FaceHalfedgeIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current half edge
    // *iter is the reference to current half edge
    // iter.Handle() returns the handle of current half edge
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.5 typedef FaceIterT<mitkHEMesh> mitkHEMesh::FaceIterator

FaceIterator is an iterator class for iterating over all faces of a mitkHEMesh object.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
mitkHEMesh::FaceIterator iter(mesh);
for ( ; iter; ++iter)
{
    // iter is the pointer to current face
    // *iter is the reference to current face
    // iter.Handle() returns the handle of current face
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.6 typedef FaceVertexIterT<mitkHEMesh> mitkHEMesh::FaceVertexIterator

FaceVertexIterator is an iterator class for iterating over a face's vertices.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the face
mitkHEMesh::FaceVertexIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current vertex
    // *iter is the reference to current vertex
    // iter.Handle() returns the handle of current vertex
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.7 typedef VertexEdgeIterT<mitkHEMesh> mitkHEMesh::VertexEdgeIterator

VertexEdgeIterator is an iterator class for iterating over all incident edges around a vertex.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the center vertex
mitkHEMesh::VertexEdgeIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current incident edge
    // *iter is the reference to current incident edge
    // iter.Handle() returns the handle of current incident edge
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.8 typedef VertexFaceIterT<mitkHEMesh> mitkHEMesh::VertexFaceIterator

VertexFaceIterator is an iterator class for iterating over all adjacent faces around a vertex.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the center vertex
mitkHEMesh::VertexFaceIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current adjacent face
    // *iter is the reference to current adjacent face
    // iter.Handle() returns the handle of current adjacent face
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.9 typedef VertexIHalfedgeIterT<mitkHEMesh> mitkHEMesh::VertexIHalfedgeIterator

VertexIHalfedgeIterator is an iterator class for iterating over all incoming half edges around a vertex.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the center vertex
mitkHEMesh::VertexIHalfedgeIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current incoming half edge
```

```
    // *iter is the reference to current incoming half edge
    // iter.Handle() returns the handle of current incoming half edge
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.10 typedef VertexIterT<mitkHEMesh> mitkHEMesh::VertexIterator

VertexIterator is an iterator class for iterating over all vertices of a mitkHEMesh object.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
mitkHEMesh::VertexIterator iter(mesh);
for ( ; iter; ++iter)
{
    // iter is the pointer to current vertex
    // *iter is the reference to current vertex
    // iter.Handle() returns the handle of current vertex
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.11 typedef VertexOHalfedgeIterT<mitkHEMesh> mitkHEMesh::VertexOHalfedgeIterator

VertexOHalfedgeIterator is an iterator class for iterating over all outgoing half edges around a vertex.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the center vertex
mitkHEMesh::VertexOHalfedgeIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current outgoing half edge
    // *iter is the reference to current outgoing half edge
    // iter.Handle() returns the handle of current outgoing half edge
    do something with iter, *iter or iter.Handle();
}
```

8.110.2.12 typedef VertexVertexIterT<mitkHEMesh> mitkHEMesh::VertexVertexIterator

VertexVertexIterator is an iterator class for iterating over all neighboring vertices around a vertex.

The code snippet for using this class:

```
// mesh is a pointer to an object of mitkHEMesh
// center is the handle of the center vertex
mitkHEMesh::VertexVertexIterator iter(mesh, center);
for ( ; iter; ++iter)
{
    // iter is the pointer to current neighboring vertex
    // *iter is the reference to current neighboring vertex
    // iter.Handle() returns the handle of current neighboring vertex
    do something with iter, *iter or \e r.Handle();
}
```

8.110.3 Constructor & Destructor Documentation

8.110.3.1 mitkHEMesh::mitkHEMesh ()

Default constructor.

8.110.4 Member Function Documentation

8.110.4.1 FaceHandle mitkHEMesh::AddFace (unsigned int *n*, VertexHandle * *vertices*)

Add Face.

Parameters:

n the number of vertices of the face to add

vertices array of the VertexHandle which compose the face

Returns:

Return the handle of the face added.

Warning:

Each VertexHandle in the array must be valid, i.e. represents a existent vertex in the mesh (has been "Add*"ed before).

8.110.4.2 VertexHandle mitkHEMesh::AddVertex (float *v*[6]) [inline]

Add vertex.

Parameters:

v[0] the x-coordinate of the vertex to add

v[1] the y-coordinate of the vertex to add

v[2] the z-coordinate of the vertex to add

v[3] the x-coordinate of the vertex's normal

v[4] the y-coordinate of the vertex's normal

v[5] the z-coordinate of the vertex's normal

Returns:

Return the handle of the vertex added.

8.110.4.3 VertexHandle mitkHEMesh::AddVertex (float *x*, float *y*, float *z*, float *nx* = 0.0f, float *ny* = 0.0f, float *nz* = 0.0f) [inline]

Add vertex.

Parameters:

x the x-coordinate of the vertex to add

y the y-coordinate of the vertex to add

z the z-coordinate of the vertex to add

nx the x-coordinate of the vertex's normal, the default value is 0.0f

ny the y-coordinate of the vertex's normal, the default value is 0.0f

nz the z-coordinate of the vertex's normal, the default value is 0.0f

Returns:
Return the handle of the vertex added.

8.110.4.4 VertexHandle mitkHEMesh::AddVertex (Point3f const & *point*, Point3f const & *normal*) [inline]

Add vertex.

Parameters:
point the coordinates of the vertex to add

normal the normal of the vertex to add

Returns:
Return the handle of the vertex added.

8.110.4.5 VertexHandle mitkHEMesh::AddVertex (Vertex const & *vert*) [inline]

Add vertex.

Parameters:
vert the vertex to add

Returns:
Return the handle of the vertex added.

8.110.4.6 FaceHandle mitkHEMesh::AdjFace (HalfEdgeHandle *he*) const [inline]

Get the handle of one half edge's adjacent face.

Parameters:
he the handle of the half edge

Returns:
Return the handle of the face.

Note:
A face is made up of an inner loop of half edges.

8.110.4.7 void mitkHEMesh::ClearFlags ()

Clear all the user-set flags except MITK_FLAG_DELETED.

8.110.4.8 void mitkHEMesh::ClearGarbage ()

Clear the garbage generated by deletion.

Note:
Call this function before any kinds of iteration.

8.110.4.9 virtual void mitkHEMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

8.110.4.10 void mitkHEMesh::DeleteEdge (EdgeHandle *edge*, bool *deleteIsolatedVertices*)

Delete edge.

Parameters:
edge the handle of the edge to be deleted
deleteIsolatedVertices whether to delete the isolated vertices after deleting the edge

8.110.4.11 void mitkHEMesh::DeleteFace (FaceHandle *face*, bool *deleteIsolatedVertices*)

Delete face.

Parameters:
face the handle of the face to be deleted
deleteIsolatedVertices whether to delete the isolated vertices after deleting the face

8.110.4.12 void mitkHEMesh::DeleteVertex (VertexHandle *vert*)

Delete vertex.

Parameters:
vert the handle of the vertex to be deleted

8.110.4.13 VertexHandle mitkHEMesh::EndVertex (HalfEdgeHandle *he*) const [inline]

Get the handle of one half edge's end vertex.

Parameters:
he the handle of the half edge

Returns:
Return the handle of the end vertex.

8.110.4.14 HalfEdgeHandle mitkHEMesh::FindHalfEdge (VertexHandle *vert0*, VertexHandle *vert1*)

Find halfedge from vert0 to vert1.

Parameters:

vert0 the handle of the start vertex

vert1 the handle of the end vertex

Returns:

Return the handle of the half edge between vert0 and vert1. If no half edge is found, the returned handle will be invalid. Use HalfEdgeHandle::IsValid() to test it.

8.110.4.15 virtual int mitkHEMesh::GetDataObjectType () const [inline, virtual]

Return what type of data object this is.

Returns:

Return the type of this data object.

Reimplemented from mitkMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

8.110.4.16 virtual Edge& mitkHEMesh::GetEdge (HalfEdgeHandle *he*) const [pure virtual]

Get edge by half edge handle.

Parameters:

he one half edge handle of the required edge

Returns:

Return the reference to the required edge.

Note:

An edge is composed by two opposite half edges.

8.110.4.17 virtual Edge& mitkHEMesh::GetEdge (EdgeHandle *e*) const [pure virtual]

Get edge by handle.

Parameters:

e the handle of the required edge

Returns:

Return the reference to the required edge.

8.110.4.18 EdgeHandle mitkHEMesh::GetEdgeHandle (HalfEdgeHandle *he*) const [inline]

Get the handle of an edge by one of its half edge's handle.

Parameters:
he one half edge's handle of the edge

Returns:
Return the handle of the edge.

Note:
An edge is composed by two opposite half edges.

8.110.4.19 size_type mitkHEMesh::GetEdgeNumber () const [inline]

Get edge number.

Returns:
Return the number of edges.

8.110.4.20 virtual Face& mitkHEMesh::GetFace (FaceHandle *f*) const [pure virtual]

Get face by handle.

Parameters:
f the handle of the required face

Returns:
Return the reference to the required face.

8.110.4.21 virtual size_type mitkHEMesh::GetFaceNumber () const [inline, virtual]

Get face number.

Returns:
Return the number of faces.

Implements mitkMesh.

8.110.4.22 virtual HalfEdge& mitkHEMesh::GetHalfEdge (HalfEdgeHandle *he*) const [pure virtual]

Get half edge by handle.

Parameters:
he the handle of the required half edge

Returns:
Return the reference to the required half edge.

8.110.4.23 virtual FaceHandle mitkHEMesh::GetHandle (Face const & *face*) const [pure virtual]

Get the handle of a face.

Parameters:
face the reference to the face

Returns:
Return the handle of the face.

8.110.4.24 virtual HalfEdgeHandle mitkHEMesh::GetHandle (HalfEdge const & *he*) const [pure virtual]

Get the handle of a half edge.

Parameters:
he the reference to the half edge

Returns:
Return the handle of the half edge.

8.110.4.25 virtual EdgeHandle mitkHEMesh::GetHandle (Edge const & *edge*) const [pure virtual]

Get the handle of an edge.

Parameters:
edge the reference to the edge

Returns:
Return the handle of the edge.

8.110.4.26 virtual VertexHandle mitkHEMesh::GetHandle (Vertex const & *vert*) const [pure virtual]

Get the handle of a vertex.

Parameters:
vert the reference to the vertex

Returns:
Return the handle of the vertex.

8.110.4.27 size_type mitkHEMesh::GetValidEdgeNumber () const [inline]

Get the number of current valid edges in this mesh.

Returns:
Return the number of the valid edges.

Note:
For some reasons of efficiency, deleting some components from the mesh does not really remove the components from the memory, but just make marks on them. Therefore, after the delete operations, the total number of the components in the memory does not equal to the number of the valid ones. So, use this function to get the right number. Furthermore, ClearGarbage() will remove the deleted components from the memory at last.

8.110.4.28 size_type mitkHEMesh::GetValidFaceNumber () const [inline]

Get the number of current valid faces in this mesh.

Returns:
Return the number of the valid faces.

Note:
For some reasons of efficiency, deleting some components from the mesh does not really remove the components from the memory, but just make marks on them. Therefore, after the delete operations, the total number of the components in the memory does not equal to the number of the valid ones. So, use this function to get the right number. Furthermore, ClearGarbage() will remove the deleted components from the memory at last.

8.110.4.29 size_type mitkHEMesh::GetValidVertexNumber () const [inline]

Get the number of current valid vertices in this mesh.

Returns:
Return the number of the valid vertices.

Note:
For some reasons of efficiency, deleting some components from the mesh does not really remove the components from the memory but just make marks on them. Therefore, after the delete operations, the total number of the components in the memory does not equal to the number of the valid ones. So, use this function to get the right number. Furthermore, ClearGarbage() will remove the deleted components from the memory at last.

8.110.4.30 virtual Vertex& mitkHEMesh::GetVertex (VertexHandle *v*) const [pure virtual]

Get vertex by handle.

Parameters:
v the handle of the required vertex

Returns:
Return the reference to the required vertex.

8.110.4.31 virtual size_type mitkHEMesh::GetVertexNumber () const [inline, virtual]

Get vertex number.

Returns:
Return the number of vertices.

Implements mitkMesh.

8.110.4.32 virtual void mitkHEMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkMesh.

Reimplemented in mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, and mitkHETriangleMesh.

8.110.4.33 bool mitkHEMesh::IsBoundary (EdgeHandle *edge*) const [inline]

Boundary test for edge.

Parameters:
edge the handle of the edge to be tested

Returns:
Return true if edge is a boundary edge, otherwise return false.

8.110.4.34 bool mitkHEMesh::IsBoundary (VertexHandle *vert*) const [inline]

Boundary test for vertex.

Parameters:
vert the handle of the vertex to be tested

Returns:
Return true if vert is a boundary vertex, otherwise return false.

8.110.4.35 bool mitkHEMesh::IsBoundary (HalfEdgeHandle *halfedge*) const [inline]

Boundary test for half edge.

Parameters:
halfedge the handle of the half edge to be tested

Returns:
Return true if halfedge is a boundary half edge, otherwise return false.

8.110.4.36 bool mitkHEMesh::IsManifold (VertexHandle *vert*) [inline]

Manifold test for vertex. The vertex is non-manifold if more than one gap exists, i.e. more than one outgoing boundary halfedge. If one boundary halfedge exists, the vertex' halfedge must be a boundary halfedge. If iterating around the vertex finds another boundary halfedge, the vertex is non-manifold.

Parameters:
vert the handle of the vertex to be tested

Returns:
Return true if the vertex is manifold, otherwise return false.

8.110.4.37 bool mitkHEMesh::IsValid (FaceHandle *f*) const [inline]

Test the validity of one face handle.

Parameters:
f the handle of the face to be tested

Returns:
Return true if this handle is valid, otherwise return false.

8.110.4.38 bool mitkHEMesh::IsValid (HalfEdgeHandle *he*) const [inline]

Test the validity of one half edge handle.

Parameters:
he the handle of the half edge to be tested

Returns:
Return true if this handle is valid, otherwise return false.

8.110.4.39 bool mitkHEMesh::IsValid (EdgeHandle *e*) const [inline]

Test the validity of one edge handle.

Parameters:
e the handle of the edge to be tested

Returns:
Return true if this handle is valid, otherwise return false.

8.110.4.40 bool mitkHEMesh::IsValid (VertexHandle *v*) const [inline]

Test the validity of one vertex handle.

Parameters:
v the handle of the vertex to be tested

Returns:
Return true if this handle is valid, otherwise return false.

8.110.4.41 HalfEdgeHandle mitkHEMesh::NextHalfEdge (HalfEdgeHandle *he*) const [inline]

Get the handle of one half edge's next half edge.

Parameters:
he the handle of the half edge

Returns:
Return the handle of the next half edge.

8.110.4.42 HalfEdgeHandle mitkHEMesh::OneHalfEdge (FaceHandle *f*) const [inline]

Get the handle of one half edge in one face.

Parameters:
f the handle of the face

Returns:
Return the handle of one of the face's half edges.

Note:
A face is made up of an inner loop of half edges.

8.110.4.43 HalfEdgeHandle mitkHEMesh::OutHalfEdge (VertexHandle *v*) const [inline]

Get the handle of one vertex's out half edge.

Parameters:
v the handle of the vertex

Returns:
Return the handle of the out half edge.

8.110.4.44 HalfEdgeHandle mitkHEMesh::PairHalfEdge (HalfEdgeHandle *he*) const [inline]

Get the handle of one half edge's opposite.

Parameters:
he the handle of the half edge

Returns:
Return the handle of the opposite half edge.

8.110.4.45 HalfEdgeHandle mitkHEMesh::PrevHalfEdge (HalfEdgeHandle *he*) const [inline]

Get the handle of one half edge's previous half edge.

Parameters:
he the handle of the half edge

Returns:
Return the handle of the previous half edge.

8.110.4.46 virtual void mitkHEMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkMesh.

Reimplemented in mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, and mitkHETriangleMesh.

8.110.4.47 virtual void mitkHEMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

8.110.4.48 virtual bool mitkHEMesh::TestClockwise () [virtual]

Test the orientation of front-facing polygons.

Returns:
Return true if the orientation of front-facing polygons is clockwise, otherwise return false.

Implements mitkMesh.

The documentation for this class was generated from the following file:

- mitkHEMesh.h

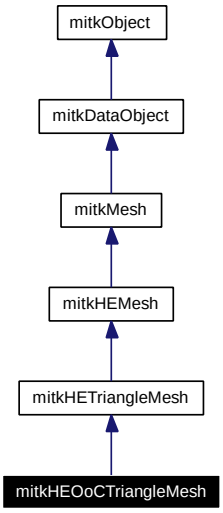
8.111 mitkHEOoCTriangleMesh Class Reference

mitkHEOoCTriangleMesh - a concrete class for out-of-core triangle meshes represented by Half Edges

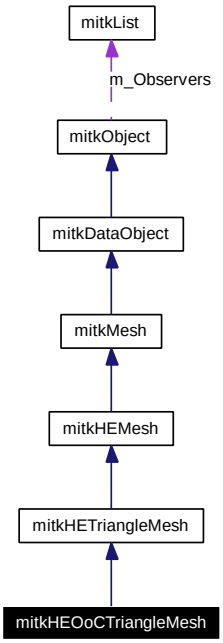
`#include <mitkHEOoCTriangleMesh.h>`

Inherits mitkHETriangleMesh.

Inheritance diagram for mitkHEOoCTriangleMesh:



Collaboration diagram for mitkHEOoCTriangleMesh:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkHEOoCTriangleMesh ()
- void SetVertexBlockSize (unsigned int vn)
- void SetEdgeBlockSize (unsigned int en)
- void SetFaceBlockSize (unsigned int fn)
- void SetVertexBufferSize (size_type s)
- void SetEdgeBufferSize (size_type s)
- void SetFaceBufferSize (size_type s)
- void SetBufferedVertexBlockNum (unsigned int n)
- unsigned int GetBufferedVertexBlockNum ()
- void SetBufferedEdgeBlockNum (unsigned int n)
- unsigned int GetBufferedEdgeBlockNum ()
- void SetBufferedFaceBlockNum (unsigned int n)
- unsigned int GetBufferedFaceBlockNum ()
- void SetPathOfDiskBuffer (char const *path)
- char const * GetPathOfDiskBuffer ()
- virtual int GetDataObjectType () const
- virtual void Initialize ()
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void SetVertexNumber (size_type number)
- virtual void SetFaceNumber (size_type number)
- virtual Vertex & GetVertex (VertexHandle v) const
- virtual Edge & GetEdge (EdgeHandle e) const
- virtual Edge & GetEdge (HalfEdgeHandle he) const
- virtual HalfEdge & GetHalfEdge (HalfEdgeHandle he) const
- virtual Face & GetFace (FaceHandle f) const
- virtual Vertex const & GetVertexForRead (VertexHandle v) const
- virtual Edge const & GetEdgeForRead (EdgeHandle e) const
- virtual Edge const & GetEdgeForRead (HalfEdgeHandle he) const
- virtual HalfEdge const & GetHalfEdgeForRead (HalfEdgeHandle he) const
- virtual Face const & GetFaceForRead (FaceHandle f) const
- virtual VertexHandle GetHandle (Vertex const &vert) const
- virtual EdgeHandle GetHandle (Edge const &edge) const
- virtual HalfEdgeHandle GetHandle (HalfEdge const &he) const
- virtual FaceHandle GetHandle (Face const &face) const
- virtual float * GetVertexData ()
- virtual index_type * GetFaceData ()

8.111.1 Detailed Description

mitkHEOoCTriangleMesh - a concrete class for out-of-core triangle meshes represented by Half Edges

mitkHEOoCTriangleMesh is a concrete class for out-of-core triangle meshes represented by Half Edges. It can hold an out-of-core data set and provide the same access interfaces as mitkHETriangleMesh.

See also:

mitkHEMesh and mitkHETriangleMesh for access interfaces
mitkHEICTriangleMesh for in-core data set

8.111.2 Constructor & Destructor Documentation

8.111.2.1 mitkHEOoCTriangleMesh::mitkHEOoCTriangleMesh ()

Default constructor.

8.111.3 Member Function Documentation

8.111.3.1 virtual void mitkHEOoCTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkHEMesh.

8.111.3.2 virtual unsigned long long mitkHEOoCTriangleMesh::GetActualMemorySize () const [virtual]

Get the actual size of the data in bytes.

Returns:
Return the actual size of the data in bytes.

Implements mitkDataObject.

8.111.3.3 unsigned int mitkHEOoCTriangleMesh::GetBufferedEdgeBlockNum ()

Get the number of buffered edge blocks in memory.

Returns:
Return the number of buffered vertex blocks.

8.111.3.4 unsigned int mitkHEOoCTriangleMesh::GetBufferedFaceBlockNum ()

Get the number of buffered face blocks in memory.

Returns:
Return the number of buffered face blocks.

8.111.3.5 unsigned int mitkHEOoCTriangleMesh::GetBufferedVertexBlockNum ()

Get the number of buffered vertex blocks in memory.

Returns:
Return the number of buffered vertex blocks.

8.111.3.6 virtual int mitkHEOoCTriangleMesh::GetDataObjectType () const [inline, virtual]

Return what type of data object this is.

Returns:
Return the type of this data object.

Reimplemented from mitkHEMesh.

8.111.3.7 virtual Edge& mitkHEOoCTriangleMesh::GetEdge (HalfEdgeHandle *he*) const [virtual]

Get edge by half edge handle.

Parameters:
he one half edge handle of the required edge

Returns:
Return the reference to the required edge.

Note:
An edge is composed by two opposite half edges.

8.111.3.8 virtual Edge& mitkHEOoCTriangleMesh::GetEdge (EdgeHandle *e*) const [virtual]

Get edge by handle.

Parameters:
e the handle of the required edge

Returns:
Return the reference to the required edge.

8.111.3.9 virtual Edge const& mitkHEOoCTriangleMesh::GetEdgeForRead (HalfEdgeHandle *he*) const [virtual]

Get edge by half edge handle for read only.

Parameters:
he one half edge handle of the required edge

Returns:
Return the reference to the required edge.

Note:
An edge is composed by two opposite half edges.

8.111.3.10 virtual Edge const& mitkHEOoCTriangleMesh::GetEdgeForRead (EdgeHandle *e*) const [virtual]

Get edge by handle for read only.

Parameters:
e the handle of the required edge

Returns:
Return the reference to the required edge.

8.111.3.11 virtual Face& mitkHEOoCTriangleMesh::GetFace (FaceHandle *f*) const [virtual]

Get face by handle.

Parameters:
f the handle of the required face

Returns:
Return the reference to the required face.

8.111.3.12 virtual index_type* mitkHEOoCTriangleMesh::GetFaceData () [inline, virtual]

Get data pointer of this face data. This is for compatibility with mitkTriangleMesh which is used more frequently. For out-of-core data sets, it will always return NULL, because the impossibility of fitting an out-of-core data set into the main memory.

Returns:
Always return NULL.

Warning:
This function is obsolete and does nothing in this class. We strongly advise against using it in new codes;

Implements mitkMesh.

8.111.3.13 virtual Face const& mitkHEOoCTriangleMesh::GetFaceForRead (FaceHandle *f*) const [virtual]

Get face by handle for read only.

Parameters:
f the handle of the required face

Returns:
Return the reference to the required face.

8.111.3.14 virtual HalfEdge& mitkHEOoCTriangleMesh::GetHalfEdge (HalfEdgeHandle *he*) const [virtual]

Get half edge by handle.

Parameters:
he the handle of the required half edge

Returns:
Return the reference to the required half edge.

8.111.3.15 virtual HalfEdge const& mitkHEOoCTriangleMesh::GetHalfEdgeForRead (HalfEdgeHandle *he*) const [virtual]

Get half edge by handle for read only.

Parameters:
he the handle of the required half edge

Returns:
Return the reference to the required half edge.

8.111.3.16 virtual FaceHandle mitkHEOoCTriangleMesh::GetHandle (Face const & *face*) const [virtual]

Get the handle of a face.

Parameters:
face the reference to the face

Returns:
Return the handle of the face.

8.111.3.17 virtual HalfEdgeHandle mitkHEOoCTriangleMesh::GetHandle (HalfEdge const & *he*) const [inline, virtual]

Get the handle of a half edge.

Parameters:
he the reference to the half edge

Returns:
Return the handle of the half edge.

8.111.3.18 virtual EdgeHandle mitkHEOoCTriangleMesh::GetHandle (Edge const & *edge*) const [virtual]

Get the handle of an edge.

Parameters:
edge the reference to the edge

Returns:
Return the handle of the edge.

8.111.3.19 virtual VertexHandle mitkHEOoCTriangleMesh::GetHandle (Vertex const & *vert*) const [virtual]

Get the handle of a vertex.

Parameters:
vert the reference to the vertex

Returns:
Return the handle of the vertex.

8.111.3.20 char const* mitkHEOoCTriangleMesh::GetPathOfDiskBuffer ()

Get the full path of the disk buffer which contains all the data of the mesh.

Returns:
Return the full path of the disk buffer.

8.111.3.21 virtual Vertex& mitkHEOoCTriangleMesh::GetVertex (VertexHandle *v*) const [virtual]

Get vertex by handle.

Parameters:
v the handle of the required vertex

Returns:
Return the reference to the required vertex.

8.111.3.22 virtual float* mitkHEOoCTriangleMesh::GetVertexData () [inline, virtual]

Get data pointer of this vertex data. This is for compatibility with mitkTriangleMesh which is used more frequently. For out-of-core data sets, it will always return NULL, because the impossibility of fitting an out-of-core data set into the main memory.

Returns:
Always return NULL.

Warning:
This function is obsolete and does nothing in this class. We strongly advise against using it in new codes.

Implements mitkMesh.

8.111.3.23 virtual Vertex const& mitkHEOoCTriangleMesh::GetVertexForRead (VertexHandle *v*) const [virtual]

Get vertex by handle for read only.

Parameters:
v the handle of the required vertex

Returns:
Return the reference to the required vertex.

8.111.3.24 virtual void mitkHEOoCTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkHETriangleMesh.

8.111.3.25 virtual void mitkHEOoCTriangleMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkHETriangleMesh.

8.111.3.26 void mitkHEOoCTriangleMesh::SetBufferedEdgeBlockNum (unsigned int *n*)

Set the number of buffered edge blocks in memory.

Parameters:
n the number of buffered edge blocks

Note:
When calling this function, the buffer size is also calculated by (edge block size) $*$ (vertex block number), so it is unnecessary to call SetEdgeBufferSize() again.

8.111.3.27 void mitkHEOoCTriangleMesh::SetBufferedFaceBlockNum (unsigned int *n*)

Set the number of buffered face blocks in memory.

Parameters:
n the number of buffered face blocks

Note:
When calling this function, the buffer size is also calculated by (face block size) * (face block number), so it is unnecessary to call SetFaceBufferSize() again.

8.111.3.28 void mitkHEOoCTriangleMesh::SetBufferedVertexBlockNum (unsigned int *n*)

Set the number of buffered vertex blocks in memory.

Parameters:
n the number of buffered vertex blocks

Note:
When calling this function, the buffer size is also calculated by (vertex block size) * (vertex block number), so it is unnecessary to call SetVertexBufferSize() again.

8.111.3.29 void mitkHEOoCTriangleMesh::SetEdgeBlockSize (unsigned int *en*)

Set the size of the edge block (number of edges of one block).

Parameters:
en the number of edges of one block

8.111.3.30 void mitkHEOoCTriangleMesh::SetEdgeBufferSize (size_type *s*)

Set the size of the memory buffer for containing cached edge blocks.

Parameters:
s the size of the buffer in bytes

Note:
When calling this function, the number of buffered edge blocks is also calculated by (buffer size) / (edge block size), so it is unnecessary to call SetBufferedEdgeBlockNum() again.

8.111.3.31 void mitkHEOoCTriangleMesh::SetFaceBlockSize (unsigned int *fn*)

Set the size of the face block (number of triangles of one block).

Parameters:
fn the number of triangles of one block

8.111.3.32 void mitkHEOoCTriangleMesh::SetFaceBufferSize (size_type *s*)

Set the size of the memory buffer for containing cached face blocks.

Parameters:
s the size of the buffer in bytes

Note:
When calling this function, the number of buffered slice is also calculated by (buffer size) / (face block size), so it is unnecessary to call SetBufferedFaceBlockNum() again.

8.111.3.33 virtual void mitkHEOoCTriangleMesh::SetFaceNumber (size_type *number*) [virtual]

Set the faces' number and allocate memory.

Parameters:
number the number of faces

Warning:
This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes.

Implements mitkMesh.

8.111.3.34 void mitkHEOoCTriangleMesh::SetPathOfDiskBuffer (char const * *path*)

Set the full path of the disk buffer to contain all the data of the mesh.

Parameters:
path the full path of the disk buffer

8.111.3.35 void mitkHEOoCTriangleMesh::SetVertexBlockSize (unsigned int *vn*)

Set the size of the vertex block (number of vertices of one block).

Parameters:
vn the number of vertices of one block

8.111.3.36 void mitkHEOoCTriangleMesh::SetVertexBufferSize (size_type *s*)

Set the size of the memory buffer for containing cached vertex blocks.

Parameters:
s the size of the buffer in bytes

Note:
When calling this function, the number of buffered vertex blocks is also calculated by (buffer size) / (vertex block size), so it is unnecessary to call SetBufferedVertexBlockNum() again.

8.111.3.37 virtual void mitkHEOoCTriangleMesh::SetVertexNumber (size_type *number*) [virtual]

Set the vertices' number and allocate memory.

Parameters:
number the number of vertices

Note:
This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes.

Implements mitkMesh.

8.111.3.38 virtual void mitkHEOoCTriangleMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:

src pointer to the source mitkDataObject

Reimplemented from mitkHEMesh.

The documentation for this class was generated from the following file:

- mitkHEOoCTriangleMesh.h

8.112 mitkHETriangleMesh Class Reference

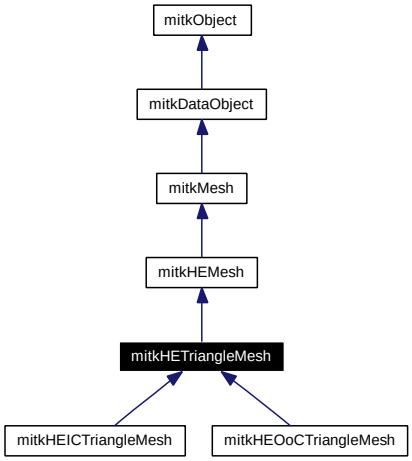
mitkHETriangleMesh - an abstract class for triangle meshes represented by Half Edges

`#include <mitkHETriangleMesh.h>`

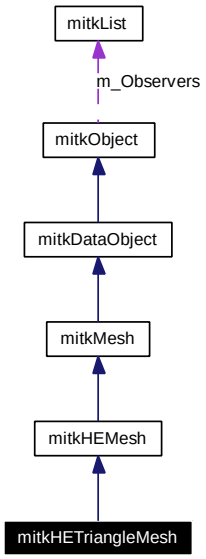
Inherits mitkHEMesh.

Inherited by mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

Inheritance diagram for mitkHETriangleMesh:



Collaboration diagram for mitkHETriangleMesh:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkHETriangleMesh ()

- bool CreateFrom (mitkTriangleMesh ∗mesh)
- bool CreateFrom (size_type vertNum, Vertex3f const ∗verts, size_type faceNum, TriangleFace const ∗faces, float const ∗boundingBox=NULL)
- virtual int GetDataObjectType ()
- virtual void Initialize ()
- FaceHandle AddFace (VertexHandle v0, VertexHandle v1, VertexHandle v2)
- FaceHandle AddFace (VertexHandle vertices[3])
- bool IsCollapseOk (HalfEdgeHandle he)
- VertexHandle Collapse (HalfEdgeHandle he)
- HalfEdgeHandle VertexSplit (VertexHandle v0, VertexHandle v1, VertexHandle vL, VertexHandle vR)
- HalfEdgeHandle VertexSplit (Vertex &vert, VertexHandle v1, VertexHandle vL, VertexHandle vR)
- bool IsFlipOk (EdgeHandle edge) const
- void Flip (EdgeHandle edge)

8.112.1 Detailed Description

mitkHETriangleMesh - an abstract class for triangle meshes represented by Half Edges

mitkHETriangleMesh is an abstract class for triangle meshes represented by Half Edges. This implementation of half edge references to OpenMesh written by Computer Graphics Group, RWTH Aachen.

See also:
mitkHEMesh, mitkHEICTriangleMesh (for in-core storage) and mitkOoCTriangleMesh (for out-of-core storage)

8.112.2 Constructor & Destructor Documentation

8.112.2.1 mitkHETriangleMesh::mitkHETriangleMesh ()

Default constructor.

8.112.3 Member Function Documentation

8.112.3.1 FaceHandle mitkHETriangleMesh::AddFace (VertexHandle *vertices*[3]) [inline]

Add face.

Parameters:
vertices[0] the handle of the first vertex
vertices[1] the handle of the second vertex
vertices[2] the handle of the third vertex

Returns:
Return the handle of the face added.

Warning:
Each VertexHandle must be valid, i.e. represents a existent vertex in the mesh (has been "Add∗"ed before).

8.112.3.2 FaceHandle mitkHETriangleMesh::AddFace (VertexHandle *v0*, VertexHandle *v1*, VertexHandle *v2*) [inline]

Add face.

Parameters:

v0 the handle of the first vertex

v1 the handle of the second vertex

v2 the handle of the third vertex

Returns:

Return the handle of the face added.

Warning:

Each VertexHandle must be valid, i.e. represents a existent vertex in the mesh (has been "Add∗"ed before).

8.112.3.3 VertexHandle mitkHETriangleMesh::Collapse (HalfEdgeHandle *he*)

Collapse the from-vertex of a half edge into its to-vertex.

Parameters:

he the handle of the half edge to be collapsed

Returns:

Return the handle of the vertex the half edge collapsed into.

8.112.3.4 bool mitkHETriangleMesh::CreateFrom (size_type *vertNum*, Vertex3f const ∗ *verts*, size_type *faceNum*, TriangleFace const ∗ *faces*, float const ∗ *boundingBox* = NULL)

Create mitkHETriangleMesh object from the arrays of vertices and faces.

Parameters:

vertNum number of vertices in the vertex array

verts the vertex array

faceNum number of faces in the face array

faces the face array

boundingBox the 6-float array for bounding box

Returns:

Return true if this operation succeed, otherwise return false.

Note:

Each Vertex3f struct contains 3 float values for coordinates and 3 float values for normal. Each TriangleFace struct contains 3 indices to vertex array. If boundingBox is NULL, this function will calculate the actual bounding box itself.

8.112.3.5 bool mitkHETriangleMesh::CreateFrom (mitkTriangleMesh * *mesh*)

Create mitkHETriangleMesh object from a mitkTriangleMesh object which is not based on half edge structure.

Parameters:
mesh the pointer to a mitkTriangleMesh object this mitkHETriangleMesh object is created from

Returns:
Return true if this operation succeed, otherwise return false.

8.112.3.6 void mitkHETriangleMesh::Flip (EdgeHandle *edge*)

Flip edge.

Parameters:
edge the handle of the edge to be flipped

8.112.3.7 virtual int mitkHETriangleMesh::GetDataObjectType () `[inline, virtual]`

Return what type of data object this is.

Returns:
Return the type of this data object.

8.112.3.8 virtual void mitkHETriangleMesh::Initialize () `[virtual]`

Make the output data ready for new data to be inserted.

Reimplemented from mitkHEMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

8.112.3.9 bool mitkHETriangleMesh::IsCollapseOk (HalfEdgeHandle *he*)

Check whether collapsing half edge he is ok or would lead to topological inconsistencies.

Parameters:
he the handle of the half edge to be tested

Returns:
Return true if collapsing is ok, otherwise return false.

8.112.3.10 bool mitkHETriangleMesh::IsFlipOk (EdgeHandle *edge*) const

Check whether flipping edge is topologically correct.

Parameters:
edge the handle of the edge to be flipped

Returns:
Return true if flipping is ok, otherwise return false.

8.112.3.11 virtual void mitkHETriangleMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkHEMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

8.112.3.12 HalfEdgeHandle mitkHETriangleMesh::VertexSplit (Vertex & *vert*, VertexHandle *v1*, VertexHandle *vL*, VertexHandle *vR*) [inline]

Split vertex v1 into edge v0v1. It is the inverse operation to Collapse().

Parameters:
v0 a new vertex
v1 the handle of the vertex to be splited
vL the handle of the vertex to the left hand of the splited vertex
vL the handle of the vertex to the right hand of the splited vertex

Returns:
Return the handle of the half edge from v0 to v1.

8.112.3.13 HalfEdgeHandle mitkHETriangleMesh::VertexSplit (VertexHandle *v0*, VertexHandle *v1*, VertexHandle *vL*, VertexHandle *vR*)

Split vertex v1 into edge v0v1. It is the inverse operation to Collapse().

Parameters:
v0 the handle of a new vertex
v1 the handle of the vertex to be splited
vL the handle of the vertex to the left hand of the splited vertex
vL the handle of the vertex to the right hand of the splited vertex

Returns:
Return the handle of the half edge from v0 to v1.

Warning:
v0 must be valid, i.e. represents a existent vertex in the mesh (has been "Add∗"ed before).

The documentation for this class was generated from the following file:

- mitkHETriangleMesh.h

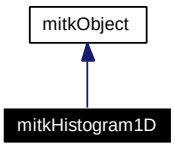
8.113 mitkHistogram1D Class Reference

mitkHistogram1D - a tool to generate a 1D histogram from a volume

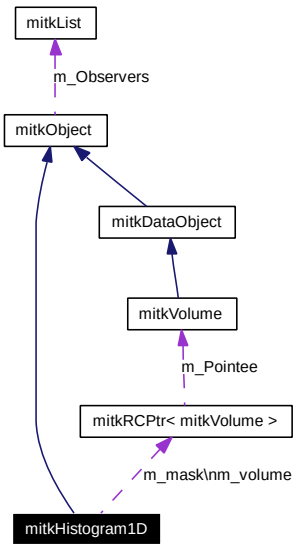
`#include <mitkHistogram1D.h>`

Inherits mitkObject.

Inheritance diagram for mitkHistogram1D:



Collaboration diagram for mitkHistogram1D:



Public Member Functions

- void SetSourceVolume (mitkVolume *vol, int chIdx=-1, unsigned int *extent=NULL, mitkVolume *mask=NULL)
- void SetRange (double min, double max, unsigned int intervalNum)
- void GetRange (double &min, double &max, unsigned int &intervalNum) const
- double GetMaxPixVal ()
- double GetMinPixVal ()
- double GetMeanPixVal ()
- double GetStandardDeviation ()
- unsigned long GetTotalPixels ()
- unsigned long * GetStatistics ()
- bool IsModified ()
- void SetUnmodified ()
- mitkVolume * GetSourceVolume ()

8.113.1 Detailed Description

mitkHistogram1D - a tool to generate a 1D histogram from a volume

mitkHistogram1D is a tool to generate a 1D histogram from a volume

8.113.2 Member Function Documentation

8.113.2.1 double mitkHistogram1D::GetMaxPixVal ()

Get the maximum pixel value within the specified extent

Returns:
Return the maximum pixel value within the specified extent

8.113.2.2 double mitkHistogram1D::GetMeanPixVal ()

Get the average pixel value within the specified extent, this will trigger the calculation

Returns:
Return the average pixel value within the specified extent

8.113.2.3 double mitkHistogram1D::GetMinPixVal ()

Get the minimum pixel value within the specified extent this will trigger the calculation

Returns:
Return the minimum pixel value within the specified extent

8.113.2.4 void mitkHistogram1D::GetRange (double & *min*, double & *max*, unsigned int & *intervalNum*) const [inline]

////////////////// Get the value range and interval where statistics are calculated,

Parameters:
min Recieve the min value of histogram

max Recieve the max value of histogram

intervalNum Recieve number of intervals between min and max

8.113.2.5 mitkVolume* mitkHistogram1D::GetSourceVolume () [inline]

Get the volume whose histogram is generated

Returns:
Return the volume whose histogram is generated

8.113.2.6 double mitkHistogram1D::GetStandardDeviation ()

Get the standard deviation of pixel value within the specified extent, this will trigger the calculation

Returns:
Return the standard deviation of pixel value within the specified extent

8.113.2.7 unsigned long* mitkHistogram1D::GetStatistics ()

Get histogram statistics, this will trigger the calculation

Returns:
Return the histogram statistics, the array is as large as intervalNum

8.113.2.8 unsigned long mitkHistogram1D::GetTotalPixels ()

Get the total number of pixels within the specified extent

Returns:
Return the total number of pixels within the specified extent

8.113.2.9 bool mitkHistogram1D::IsModified ()

Get the modified status of the histogram

Returns:
whether the histogram is modified after last call to SetUnmodified()

8.113.2.10 void mitkHistogram1D::SetRange (double *min*, double *max*, unsigned int *intervalNum*)

////////////////// Set the value range and interval where to calculate statistics, this will allocate memory for statistics

Parameters:
min Specify the min value of histogram
max Specify the max value of histogram
intervalNum Specify number of intervals between min and max

8.113.2.11 void mitkHistogram1D::SetSourceVolume (mitkVolume * *vol*, int *chIdx* = -1, unsigned int * *extent* = NULL, mitkVolume * *mask* = NULL)

Set the volume whose histogram is to be generated

Parameters:
vol Specify pointer to the volume
chIdx Specify the channel of the the histogram, if the value is greater or equal than the number of channel present or smaller than zero, an average of all channels will be genrated.
extent Specify (optionally) the region of the volume to generate the histogram. NULL means to take the whole volume into consideration.

8.113.2.12 void mitkHistogram1D::SetUnmodified ()

Set the modified status the false

The documentation for this class was generated from the following file:

- mitkHistogram1D.h

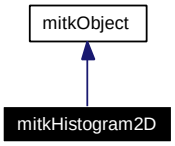
8.114 mitkHistogram2D Class Reference

mitkHistogram2D - a tool to generate a 2D histogram from two volumes

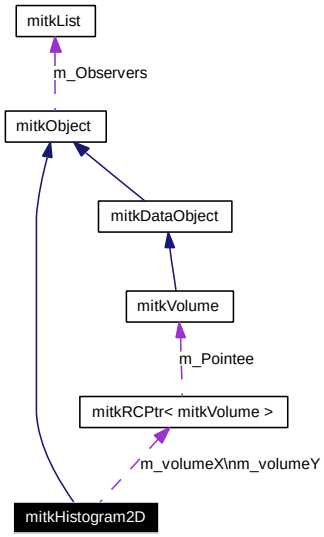
`#include <mitkHistogram2D.h>`

Inherits mitkObject.

Inheritance diagram for mitkHistogram2D:



Collaboration diagram for mitkHistogram2D:



Public Member Functions

- void SetSourceVolumeX (mitkVolume *vol, int chIdx=1, unsigned int *extent=NULL)
- void SetSourceVolumeY (mitkVolume *vol, int chIdx=1, unsigned int *extent=NULL)
- void SetRange (double minX, double maxX, unsigned int intervalNumX, double minY, double maxY, unsigned int intervalNumY)
- void GetRange (double &minX, double &maxX, unsigned int &intervalNumX, double &minY, double &maxY, unsigned int &intervalNumY)
- double GetMaxPixValX ()
- double GetMinPixValX ()
- double GetMaxPixValY ()
- double GetMinPixValY ()
- double GetMeanPixValX ()
- double GetMeanPixValY ()
- double GetStandardDeviationX ()

- double GetStandardDeviationY ()
- unsigned long GetTotalPixels ()
- unsigned long * GetStatistics ()
- bool IsModified ()
- void SetUnmodified ()
- mitkVolume * GetSourceVolumeX ()
- mitkVolume * GetSourceVolumeY ()

8.114.1 Detailed Description

mitkHistogram2D - a tool to generate a 2D histogram from two volumes

mitkHistogram2D is a tool to generate a 2D histogram from two volumes

8.114.2 Member Function Documentation

8.114.2.1 double mitkHistogram2D::GetMaxPixValX ()

Get the maximum pixel value within the specified extent of VolumeX

Returns:
Return the maximum pixel value within the specified extent

8.114.2.2 double mitkHistogram2D::GetMaxPixValY ()

Get the maximum pixel value within the specified extent of VolumeY

Returns:
Return the maximum pixel value within the specified extent

8.114.2.3 double mitkHistogram2D::GetMeanPixValX ()

Get the average pixel value within the specified extent of VolumeX, this will trigger the calculation

Returns:
Return the average pixel value within the specified extent

8.114.2.4 double mitkHistogram2D::GetMeanPixValY ()

Get the average pixel value within the specified extent of VolumeY, this will trigger the calculation

Returns:
Return the average pixel value within the specified extent

8.114.2.5 double mitkHistogram2D::GetMinPixValX ()

Get the minimum pixel value within the specified extent of VolumeX this will trigger the calculation

Returns:
Return the minimum pixel value within the specified extent

8.114.2.6 double mitkHistogram2D::GetMinPixValY ()

Get the minimum pixel value within the specified extent of VolumeY this will trigger the calculation

Returns:
Return the minimum pixel value within the specified extent

8.114.2.7 void mitkHistogram2D::GetRange (double & *minX*, double & *maxX*, unsigned int & *intervalNumX*, double & *minY*, double & *maxY*, unsigned int & *intervalNumY*) [inline]

////////////////// Get the value range and interval where statistics are calculated,

Parameters:
minX Receive the min value of the histogram's X-Axial
maxX Receive the max value of the histogram's X-Axial
intervalNum Receive number of intervals between minX and maxX
minY Receive the min value of the histogram's Y-Axial
maxY Receive the max value of the histogram's Y-Axial
intervalNum Receive number of intervals between minY and maxY

8.114.2.8 mitkVolume* mitkHistogram2D::GetSourceVolumeX () [inline]

Get the first volume (the X-Axial volume)

Returns:
Return the first volume

8.114.2.9 mitkVolume* mitkHistogram2D::GetSourceVolumeY () [inline]

Get the second volume (the Y-Axial volume)

Returns:
Return the second volume

8.114.2.10 double mitkHistogram2D::GetStandardDeviationX ()

Get the standard deviation of pixel value within the specified extent, of VolumeX, this will trigger the calculation

Returns:
Return the standard deviation of pixel value within the specified extent

8.114.2.11 double mitkHistogram2D::GetStandardDeviationY ()

Get the standard deviation of pixel value within the specified extent, of VolumeY, this will trigger the calculation

Returns:
Return the standard deviation of pixel value within the specified extent

8.114.2.12 unsigned long* mitkHistogram2D::GetStatistics ()

Get histogram statistics, this will trigger the calculation

Returns:
Return the histogram statistics, the array is as large as intervalNumX*intervalNumY

8.114.2.13 unsigned long mitkHistogram2D::GetTotalPixels ()

Get the total number of pixels within the specified extent

Returns:
Return the total number of pixels within the specified extent

8.114.2.14 bool mitkHistogram2D::IsModified ()

Get the modified status of the histogram

Returns:
whether the histogram is modified after last call to SetUnmodified()

8.114.2.15 void mitkHistogram2D::SetRange (double *minX*, double *maxX*, unsigned int *intervalNumX*, double *minY*, double *maxY*, unsigned int *intervalNumY*)

////////////////// Set the value range and interval where to calculate statistics, this will allocate memory for statistics

Parameters:

minX Specify the min value of the histogram's X-Axial

maxX Specify the max value of the histogram's X-Axial

intervalNum Specify number of intervals between minX and maxX

minY Specify the min value of the histogram's Y-Axial

maxY Specify the max value of the histogram's Y-Axial

intervalNum Specify number of intervals between minY and maxY

8.114.2.16 void mitkHistogram2D::SetSourceVolumeX (mitkVolume * *vol*, int *chIdx* = 1, unsigned int * *extent* = NULL)

Set the first volume (the X-Axial volume)

Parameters:

vol Specify pointer to the volume

chIdx Specify the channel of the the histogram, if the value is greater or equal than the number of channel present, an average of all channels will be genrated.

extent Specify (optionally) the region of the volume to generate the histogram. NULL means to take the whole volume into consideration.

8.114.2.17 void mitkHistogram2D::SetSourceVolumeY (mitkVolume * *vol*, int *chIdx* = 1, unsigned int * *extent* = NULL)

Set the second volume (the Y-Axial volume)

Parameters:

vol Specify pointer to the volume

chIdx Specify the channel of the the histogram, if the value is greater or equal than the number of channel present, an average of all channels will be genrated.

extent Specify (optionally) the region of the volume to generate the histogram. NULL means to take the whole volume into consideration.

8.114.2.18 void mitkHistogram2D::SetUnmodified ()

Set the modified status the false

The documentation for this class was generated from the following file:

- mitkHistogram2D.h

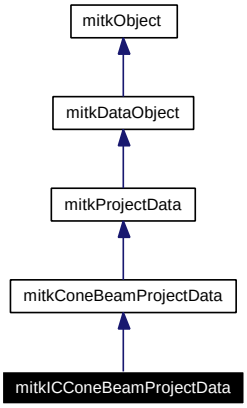
8.115 mitkICConeBeamProjectData Class Reference

mitkICConeBeamProjectData - concret class to represent cone-beam project data.

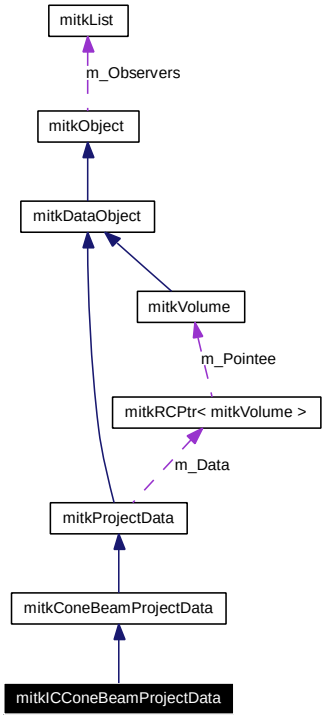
`#include <mitkICConeBeamProjectData.h>`

Inherits mitkConeBeamProjectData.

Inheritance diagram for mitkICConeBeamProjectData:



Collaboration diagram for mitkICConeBeamProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void Initialize ()
- virtual void const * GetAngleSliceForRead (int sliceIdx)

8.115.1 Detailed Description

mitkICConeBeamProjectData - concret class to represent cone-beam project data.

mitkICConeBeamProjectData - concret class to represent cone-beam project data.

8.115.2 Member Function Documentation

8.115.2.1 virtual void mitkICConeBeamProjectData::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamProjectData.

8.115.2.2 virtual unsigned long long mitkICConeBeamProjectData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkConeBeamProjectData.

8.115.2.3 virtual void const* mitkICConeBeamProjectData::GetAngleSliceForRead (int *sliceIdx*) [virtual]

GetAngleSliceForRead GetAngleSliceForWrite and GetAngleSliceForReadWrite

Implements mitkConeBeamProjectData.

8.115.2.4 virtual void mitkICConeBeamProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkConeBeamProjectData.

8.115.2.5 virtual void mitkICConeBeamProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkConeBeamProjectData.

8.115.2.6 virtual void mitkICConeBeamProjectData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamProjectData.

The documentation for this class was generated from the following file:

- mitkICConeBeamProjectData.h

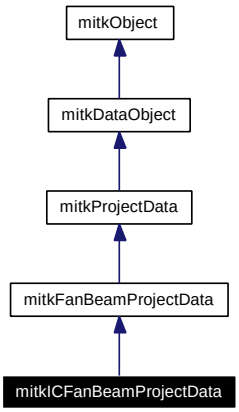
8.116 mitkICFanBeamProjectData Class Reference

mitkICFanBeamProjectData - concret class to represent fan-beam project data.

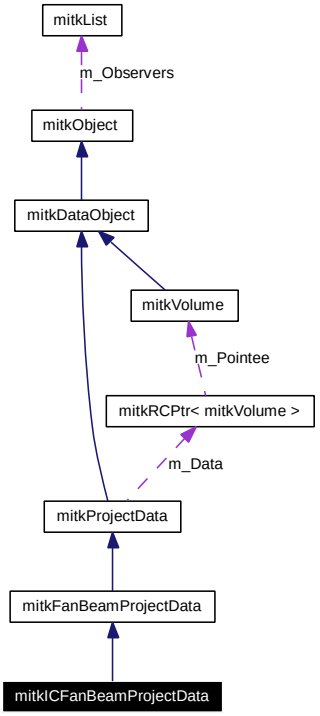
`#include <mitkICFanBeamProjectData.h>`

Inherits mitkFanBeamProjectData.

Inheritance diagram for mitkICFanBeamProjectData:



Collaboration diagram for mitkICFanBeamProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject ∗src)
- virtual void DeepCopy (mitkDataObject ∗src)
- virtual void Initialize ()
- virtual void const ∗ GetSliceForRead (int sliceIdx)

8.116.1 Detailed Description

mitkICFanBeamProjectData - concret class to represent fan-beam project data.

mitkICFanBeamProjectData - concret class to represent fan-beam project data.

8.116.2 Member Function Documentation

8.116.2.1 virtual void mitkICFanBeamProjectData::DeepCopy (mitkDataObject ∗ *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkFanBeamProjectData.

8.116.2.2 virtual unsigned long long mitkICFanBeamProjectData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkFanBeamProjectData.

8.116.2.3 virtual void const∗ mitkICFanBeamProjectData::GetSliceForRead (int *sliceIdx*) [virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implements mitkFanBeamProjectData.

8.116.2.4 virtual void mitkICFanBeamProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkFanBeamProjectData.

8.116.2.5 virtual void mitkICFanBeamProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFanBeamProjectData.

8.116.2.6 virtual void mitkICFanBeamProjectData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkFanBeamProjectData.

The documentation for this class was generated from the following file:

- mitkICFanBeamProjectData.h

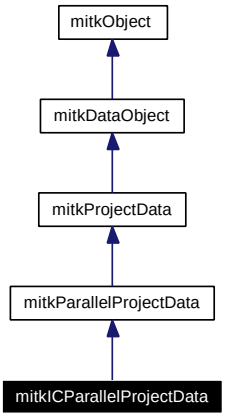
8.117 mitkICParallelProjectData Class Reference

mitkICParallelProjectData - concret class to represent Parallel project data.

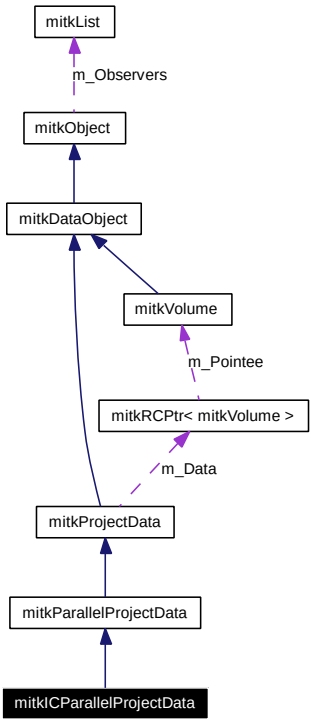
`#include <mitkICParallelProjectData.h>`

Inherits mitkParallelProjectData.

Inheritance diagram for mitkICParallelProjectData:



Collaboration diagram for mitkICParallelProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void Initialize ()
- virtual void const * GetSliceForRead (int sliceIdx)

8.117.1 Detailed Description

mitkICParallelProjectData - concret class to represent Parallel project data.

mitkICParallelProjectData - concret class to represent Parallel project data.

8.117.2 Member Function Documentation

8.117.2.1 virtual void mitkICParallelProjectData::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkParallelProjectData.

8.117.2.2 virtual unsigned long long mitkICParallelProjectData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkDataObject.

8.117.2.3 virtual void const* mitkICParallelProjectData::GetSliceForRead (int *sliceIdx*) [virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implements mitkParallelProjectData.

8.117.2.4 virtual void mitkICParallelProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkParallelProjectData.

8.117.2.5 virtual void mitkICParallelProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkParallelProjectData.

8.117.2.6 virtual void mitkICParallelProjectData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkParallelProjectData.

The documentation for this class was generated from the following file:

- mitkICParallelProjectData.h

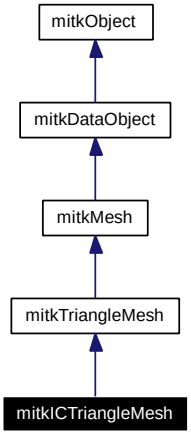
8.118 mitkICTriangleMesh Class Reference

mitkICTriangleMesh - a concrete class for in-core triangle meshes

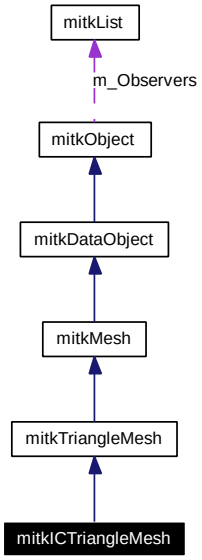
`#include <mitkICTriangleMesh.h>`

Inherits mitkTriangleMesh.

Inheritance diagram for mitkICTriangleMesh:



Collaboration diagram for mitkICTriangleMesh:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkICTriangleMesh ()
- virtual int GetDataObjectType () const
- virtual void Initialize ()

- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void SetVertexNumber (size_type number)
- virtual void SetFaceNumber (size_type number)
- virtual float * GetVertexData ()
- virtual index_type * GetFaceData ()
- virtual void ReverseNormals ()
- virtual bool TestClockwise ()

8.118.1 Detailed Description

mitkICTriangleMesh - a concrete class for in-core triangle meshes

mitkICTriangleMesh is a concrete implementation of triangle mesh, for representation of a in-core 3D object (can be loaded to the main memory entirely). It is made up of triangle faces.

See also:
mitkOoCTriangleMesh for out-of-core implementation of triangle mesh

8.118.2 Constructor & Destructor Documentation

8.118.2.1 mitkICTriangleMesh::mitkICTriangleMesh ()

Default constructor of this class.

8.118.3 Member Function Documentation

8.118.3.1 virtual void mitkICTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

8.118.3.2 virtual unsigned long long mitkICTriangleMesh::GetActualMemorySize () const [virtual]

Get the actual size of the data in bytes.

Returns:
Return the actual size of the data in bytes.

Implements mitkDataObject.

8.118.3.3 virtual int mitkICTriangleMesh::GetDataObjectType () const [inline, virtual]

Return what type of data object this is.

Returns:
Return the type of this data object.

Reimplemented from mitkTriangleMesh.

8.118.3.4 virtual index_type* mitkICTriangleMesh::GetFaceData () [inline, virtual]

Get data pointer of this face data.

Returns:
Return a unsigned int pointer to the face data (indices to vertices).

Implements mitkMesh.

8.118.3.5 virtual float* mitkICTriangleMesh::GetVertexData () [inline, virtual]

Get data pointer of this vertex data.

Returns:
Return a float pointer to the vertex data.

Implements mitkMesh.

8.118.3.6 virtual void mitkICTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkTriangleMesh.

8.118.3.7 virtual void mitkICTriangleMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTriangleMesh.

8.118.3.8 virtual void mitkICTriangleMesh::ReverseNormals () [virtual]

Reverse normals.

Reimplemented from mitkMesh.

8.118.3.9 virtual void mitkICTriangleMesh::SetFaceNumber (size_type *number*) [virtual]

Set the mesh's faces' number and allocate memory.

Parameters:
number the number of faces

Implements mitkMesh.

8.118.3.10 virtual void mitkICTriangleMesh::SetVertexNumber (size_type *number*) [virtual]

Set the mesh's vertices' number and allocate memory.

Parameters:
number the number of vertices

Implements mitkMesh.

8.118.3.11 virtual void mitkICTriangleMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

8.118.3.12 virtual bool mitkICTriangleMesh::TestClockwise () [virtual]

Test the orientation of front-facing triangles.

Returns:
Return true if the orientation of front-facing triangles is clockwise, otherwise return false.

Implements mitkMesh.

The documentation for this class was generated from the following file:

- mitkICTriangleMesh.h

8.119 mitkICVolume Class Reference

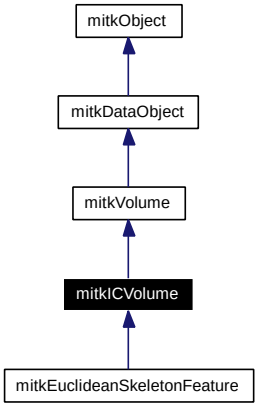
mitkICVolume - a concrete data object to represent an in-core multi-dimensional medical image dataset

`#include <mitkICVolume.h>`

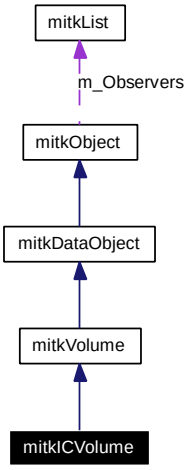
Inherits mitkVolume.

Inherited by mitkEuclideanSkeletonFeature.

Inheritance diagram for mitkICVolume:



Collaboration diagram for mitkICVolume:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkICVolume ()
- virtual int GetDataObjectType () const
- virtual void const * GetData () const
- virtual void * GetData ()
- virtual void const * GetSliceForRead (int sliceIdx)

- virtual void * GetSliceForWrite (int sliceIdx)
- virtual void * GetSliceForReadWrite (int sliceIdx)
- virtual bool ReadSliceData (int sliceIdx, void *dst)
- virtual bool ReadYZSliceData (int xIdx, void *dst)
- virtual bool ReadXZSliceData (int yIdx, void *dst)
- virtual bool GetArbitrarySlice (int w, int h, double o[3], double ux[3], double uy[3], void *dst)
- virtual bool WriteSliceData (int sliceIdx, void const *src)
- virtual bool ReadSubVolume (int x, int y, int z, int w, int h, int d, int &tw, int &th, int &td, void *dst)
- virtual bool WriteSubVolume (int x, int y, int z, int w, int h, int d, int &tw, int &th, int &td, void const *src)
- virtual bool Allocate ()
- virtual unsigned long long GetActualMemorySize () const
- virtual void Initialize ()
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)

8.119.1 Detailed Description

mitkICVolume - a concrete data object to represent an in-core multi-dimensional medical image dataset

mitkICVolume is a concrete data object to represent an in-core multi-dimensional medical image dataset.

8.119.2 Constructor & Destructor Documentation

8.119.2.1 mitkICVolume::mitkICVolume ()

Default constructor.

8.119.3 Member Function Documentation

8.119.3.1 virtual bool mitkICVolume::Allocate () [virtual]

Allocate necessary space for holding the image data. It calculate the memory size using current Dimensions, DataType and NumberOfChannel settings. The equation is shown as follow:

Width * Height * SliceNum * sizeof(DataType) * NumberOfChannel

The previous allocated data will be deleted if exists.

Returns:
Return true if successful, otherwise return false.

Implements mitkVolume.

8.119.3.2 virtual void mitkICVolume::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Reimplemented from mitkVolume.

8.119.3.3 virtual unsigned long long mitkICVolume::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by the volume data. The unit is BYTE.

Returns:
Return the actual memory size occupied by this volume. The unit is BYTE.

Implements mitkDataObject.

8.119.3.4 virtual bool mitkICVolume::GetArbitrarySlice (int *w*, int *h*, double *o*[3], double *ux*[3], double *uy*[3], void * *dst*) [virtual]

Get arbitrary directional slice from the volume to a specified memory buffer (i.e. re-slicing).

Parameters:
w width of the new slice (in pixels)

h height of the new slice (in pixels)

o start position (left-bottom) of the new slice in the volume space

ux the step (in pixels) vector of moving to the next pixel along the x-direction of the new slice in the volume space

uy the step (in pixels) vector of moving to the next pixel along the y-direction of the new slice in the volume space

dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be w * h * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.119.3.5 virtual void* mitkICVolume::GetData () [inline, virtual]

Get data pointer of the volume data (changeable).

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkVolume.

8.119.3.6 virtual void const* mitkICVolume::GetData () const [inline, virtual]

Get data pointer of the volume data (unchangeable).

Returns:
Return a void pointer to const data.

Note:
The returned type is void const *, it must be converted to some useful data type according to the return value of GetDataType().

Implements mitkVolume.

8.119.3.7 virtual int mitkICVolume::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_IN_CORE_VOLUME

Reimplemented from mitkVolume.

8.119.3.8 virtual void const* mitkICVolume::GetSliceForRead (int *sliceIdx*) [virtual]

Get data pointer of a specified slice in the volume (for read only).

Parameters:
sliceIdx the index of the slice to get (in [0, GetImageNum()-1])

Returns:
Return a const void pointer to data.

Note:
The returned type is void const *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkVolume.

8.119.3.9 virtual void* mitkICVolume::GetSliceForReadWrite (int *sliceIdx*) [virtual]

Get data pointer of a specified slice in the volume (changeable).

Parameters:
sliceIdx the index of the slice to get (in [0, GetImageNum()-1])

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkVolume.

8.119.3.10 virtual void* mitkICVolume::GetSliceForWrite (int *sliceIdx*) [virtual]

Get data pointer of a specified slice in the volume (for write only).

Parameters:
sliceIdx the index of the slice to get (in [0, GetImageNum()-1])

Returns:
Return a const void pointer to data.

Note:
The returned type is void *, it should be converted to some useful data type according to the return value of GetDataType(). This function returns an empty buffer for writing the specified slice the first time.

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkVolume.

8.119.3.11 virtual void mitkICVolume::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Reimplemented from mitkVolume.

8.119.3.12 virtual void mitkICVolume::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolume.

8.119.3.13 virtual bool mitkICVolume::ReadSliceData (int *sliceIdx*, void * *dst*) [virtual]

Copy slice data from the volume to a specified memory buffer.

Parameters:
sliceIdx the index of the slice to read (in [0, GetImageNum()-1])

dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.119.3.14 virtual bool mitkICVolume::ReadSubVolume (int *x*, int *y*, int *z*, int *w*, int *h*, int *d*, int & *tw*, int & *th*, int & *td*, void * *dst*) [virtual]

Copy a sub-volume (start position = (x, y, z), size = w * h * d) from the volume to a specified memory buffer.

Parameters:
x the x-coordinate of the start point
y the y-coordinate of the start point
z the z-coordinate of the start point
w the width of the sub-volume
h the height of the sub-volume
d the depth of the sub-volume
tw return the true width of the sub-volume copied (in case that x+w > GetWidth())
th return the true height of the sub-volume copied (in case that y+h > GetHeight())
td return the true depth of the sub-volume copied (in case that z+d > GetSliceNum())
dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be w * h * d * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.119.3.15 virtual bool mitkICVolume::ReadXZSliceData (int *yIdx*, void * *dst*) [virtual]

Copy y-directional slice from the volume to a specified memory buffer.

Parameters:
yIdx the position of the y-directional slice to read (in [0, GetHeight()-1])
dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be GetWidth() * GetImageNum() * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.119.3.16 virtual bool mitkICVolume::ReadYZSliceData (int *xIdx*, void * *dst*) [virtual]

Copy x-directional slice from the volume to a specified memory buffer.

Parameters:

xIdx the position of the x-directional slice to read (in [0, GetWidth()-1])

dst the pointer to the destination memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of destination memory buffer should be GetHeight() * GetImageNum() * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.119.3.17 virtual void mitkICVolume::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:

Don't call this function directly.

Reimplemented from mitkVolume.

8.119.3.18 virtual bool mitkICVolume::WriteSliceData (int *sliceIdx*, void const * *src*) [virtual]

Copy slice data from a specified memory buffer to the volume.

Parameters:

sliceIdx the index of the slice to write (in [0, GetImageNum()-1])

src the pointer to the source memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of source memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.119.3.19 virtual bool mitkICVolume::WriteSubVolume (int *x*, int *y*, int *z*, int *w*, int *h*, int *d*, int & *tw*, int & *th*, int & *td*, void const * *src*) [virtual]

Copy a sub-volume (start position = (x, y, z), size = w * h * d) from a specified memory buffer to the volume.

Parameters:

x the x-coordinate of the start point

y the y-coordinate of the start point

z the z-coordinate of the start point

w the width of the sub-volume

h the height of the sub-volume

d the depth of the sub-volume

tw return the true width of the sub-volume copied (in case that x+w > GetWidth())

th return the true height of the sub-volume copied (in case that y+h > GetHeight())

td return the true depth of the sub-volume copied (in case that z+d > GetSliceNum())

src the pointer to the source memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of source memory buffer should be w ∗ h ∗ d ∗ GetChannelNum() ∗ GetDataTypeSize().

Implements mitkVolume.

The documentation for this class was generated from the following file:

- mitkICVolume.h

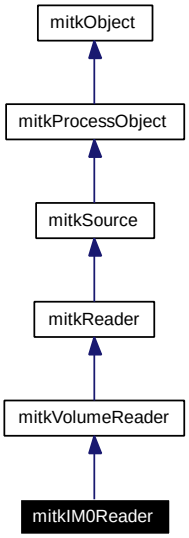
8.120 mitkIM0Reader Class Reference

mitkIM0Reader - a concrete reader for reading IM0 volume file

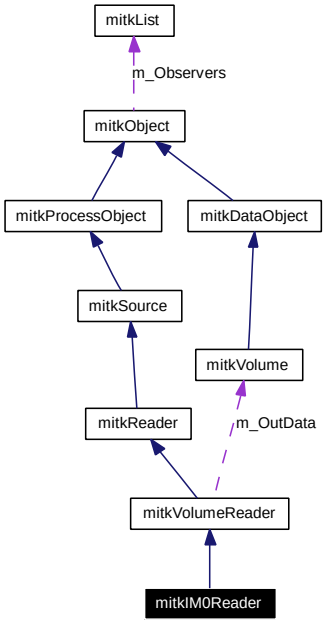
`#include <mitkIM0Reader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkIM0Reader:



Collaboration diagram for mitkIM0Reader:



8.120.1 Detailed Description

mitkIM0Reader - a concrete reader for reading IM0 volume file

mitkIM0Reader reads a IM0 volume file to a volume. IM0 file is a 3D data file, and the spacing information can be obtained from the file header. To use this reader, the code snippet is:

```
mitkIM0Reader *aReader = new mitkIM0Reader;
aReader->AddFileName(filename); //Only require one file name
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

Warning:
The IM0 file format is originally used in 3DVIEWNIX software developed by MIPG group, University of Pennsylvania. Maybe the file format is copyrighted. Please use it in your own risk.

The documentation for this class was generated from the following file:

- mitkIM0Reader.h

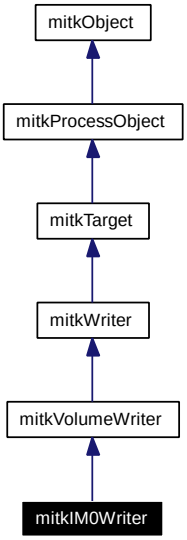
8.121 mitkIM0Writer Class Reference

mitkIM0Writer - a concrete writer for writing a volume to IM0 volume file

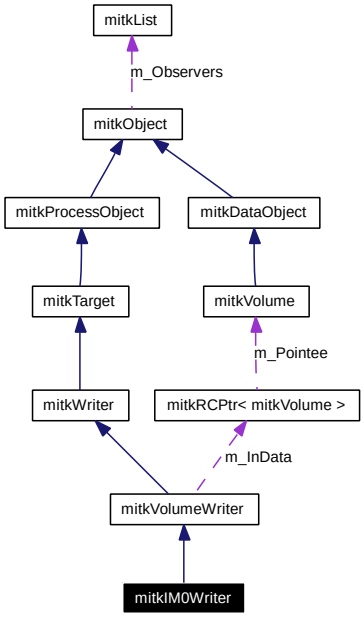
`#include <mitkIM0Writer.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkIM0Writer:



Collaboration diagram for mitkIM0Writer:



8.121.1 Detailed Description

mitkIM0Writer - a concrete writer for writing a volume to IM0 volume file

mitkIM0Writer writes a volume to a single IM0 file. Because the IM0 file format can contain 3D dataset, a single file name is sufficient for this writer to work. To use this writer, the code snippet is:

```
mitkIM0Writer *aWriter = new mitkIM0Writer;
aWriter->SetInput(aVolume);
aWriter->AddFileName(filename);
aWriter->Run();
```

Warning:
The IM0 file format is originally used in 3DVIEWNIX software developed by MIPG group, University of Pennsylvania. Maybe the file format is copyrighted. Please use it in your own risk.

The documentation for this class was generated from the following file:

- mitkIM0Writer.h

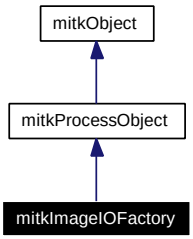
8.122 mitkImageIOFactory Class Reference

mitkImageIOFactory -

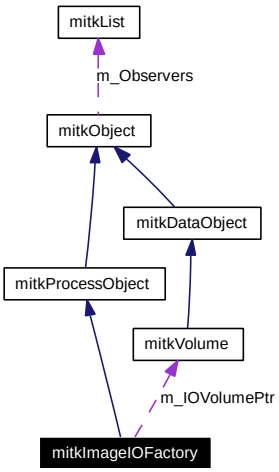
`#include <mitkImageIOFactory.h>`

Inherits mitkProcessObject.

Inheritance diagram for mitkImageIOFactory:



Collaboration diagram for mitkImageIOFactory:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.122.1 Detailed Description

mitkImageIOFactory -

mitkImageIOFactory

8.122.2 Member Function Documentation

8.122.2.1 virtual void mitkImageIOFactory::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProcessObject.

The documentation for this class was generated from the following file:

- mitkImageIOFactory.h

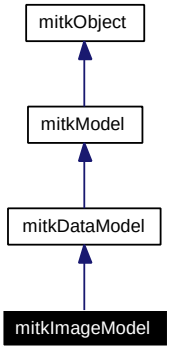
8.123 mitkImageModel Class Reference

mitkImageModel - a concrete model class used to display a 2D image in mitkImageScene

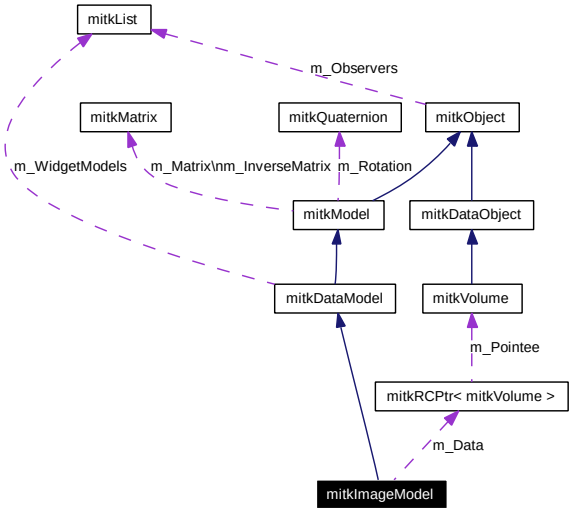
`#include <mitkImageModel.h>`

Inherits mitkDataModel.

Inheritance diagram for mitkImageModel:



Collaboration diagram for mitkImageModel:



Public Types

- enum ViewMode { VIEW_XY, VIEW_YZ, VIEW_XZ }

Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetData (mitkVolume *data)
- mitkVolume * GetData ()

- virtual int Render (mitkScene ∗scene)
- void SetViewMode (ViewMode viewMode)
- ViewMode GetViewMode () const
- void SetCurrentSliceNumber (int sliceNo)
- int GetCurrentSliceNumber () const
- void NextSlice ()
- void PrevSlice ()
- int GetTotalSliceNumber () const
- int GetWidth () const
- int GetHeight () const
- float GetSpacingX () const
- float GetSpacingY ()
- float GetSpacingZ () const
- void SetOpacity (float opacity)
- float GetOpacity () const
- virtual bool IsOpaque ()
- void CleanUp ()
- void AdjustWidthCenter (int viewWidth, int viewHeight, float deltX, float deltY)
- float GetWindowWidth () const
- float GetWindowCenter () const
- void SetWindowWidth (float winWidth)
- void SetWindowCenter (float winCenter)
- void ResetWindowWidthCenter ()
- void GetDataValueAndCoordinate (float objectX, float objectY, int &vx, int &vy, int &vz, float &rValue, float &gValue, float &bValue) const
- void GetCoordinate (float objectX, float objectY, int &vx, int &vy, int &vz) const
- void GetXYCoordinate (float objectX, float objectY, int &vx, int &vy) const
- void GetXYCoordinate (float objectX, float objectY, float &vx, float &vy) const
- void GetXYCoordinateDelta (float odx, float ody, float &vdx, float &vdy) const
- void GetObjectCoordinate (int vx, int vy, float &objX, float &objY)
- void GetObjectCoordinate (int vx, int vy, int vz, float &objX, float &objY)
- float GetActualLength (float opt0[2], float opt1[2]) const
- float GetActualLength (float x0, float y0, float x1, float y1) const
- float GetActualXLength (float objXLen) const
- float GetActualYLength (float objYLen) const
- unsigned int GetTextureID () const
- void GetIncrements (int incs[3]) const
- mitkVolume ∗ GetCurrentSlice ()
- void EnablePseudocolor (bool enable=true)
- void UpdatePseudocolor (bool rectChanged)

8.123.1 Detailed Description

mitkImageModel - a concrete model class used to display a 2D image in mitkImageScene

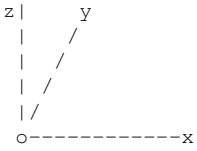
mitkImageModel is a concrete model class used to display a 2D image in mitkImageScene. It can access a volume as input and display it in image scene. To use it, the code snippet is:

```
mitkImageModel *aModel = new mitkImageModel;
aModel->SetData(aVolume); // Set the volume to be displayed
aModel->SetViewMode(mitkImageModel::VIEW_XY);
aModel->SetCurrentSliceNumber(0); //Display the first slice
aImageScene->AddModel(aModel);
aImageScene->Update();
aModel->NextSlice();  //Display the next slice
aImageScene->Update();
```

8.123.2 Member Enumeration Documentation

8.123.2.1 enum mitkImageModel::ViewMode

Set the view mode of a volume.



Enumerator:

VIEW_XY default mode, display the volume in the direction of parallel to XoY plane

VIEW_YZ display the volume in the direction of parallel to YoZ plane

VIEW_XZ display the volume in the direction of parallel to XoZ plane

8.123.3 Member Function Documentation

8.123.3.1 void mitkImageModel::AdjustWidthCenter (int *viewWidth*, int *viewHeight*, float *deltX*, float *deltY*)

Adjust the window width and window center of this image model

Parameters:

viewWidth the width of the view

viewHeight the height of the view

deltX Specify the change of the window width

deltY Specify the change of the window center

8.123.3.2 void mitkImageModel::CleanUp ()

Internal function, Don't call it directly.

8.123.3.3 void mitkImageModel::EnablePseudocolor (bool *enable* = `true`) `[inline]`

Enable pseudo-color function.

Parameters:

enable if enable is true, the pseudo-color function is enabled, otherwise it is disabled.

8.123.3.4 float mitkImageModel::GetActualLength (float *x0*, float *y0*, float *x1*, float *y1*) const [inline]

Get the actual length of the line in volume coordinate system at used specified object coordinate.

Parameters:

x0 specify the x coordinate of one point in object coordinate

y0 specify the y coordinate of one point in object coordinate

x1 specify the x coordinate of the other point in object coordinate

y1 specify the y coordinate of the other point in object coordinate

8.123.3.5 float mitkImageModel::GetActualLength (float *opt0*[2], float *opt1*[2]) const [inline]

Get the actual length of the line in volume coordinate system at used specified object coordinate.

Parameters:

opt0[0] specify the x coordinate of one point in object coordinate

opt0[1] specify the y coordinate of one point in object coordinate

opt1[0] specify the x coordinate of the other point in object coordinate

opt1[1] specify the y coordinate of the other point in object coordinate

8.123.3.6 float mitkImageModel::GetActualXLength (float *objXLen*) const [inline]

Get the actual length of the line in volume coordinate system at used specified object coordinate along x-axis.

Parameters:

objXLen specify the length along x-axis in object coordinate

8.123.3.7 float mitkImageModel::GetActualYLength (float *objYLen*) const [inline]

Get the actual length of the line in volume coordinate system at used specified object coordinate.

Parameters:

objYLen specify the length along y-axis in object coordinate

8.123.3.8 void mitkImageModel::GetCoordinate (float *objectX*, float *objectY*, int & *vx*, int & *vy*, int & *vz*) const [inline]

Get the volume coordinate at a specified object coordinate.

Parameters:

objectX Specify the x coordinate in object coordinate

objectY Specify the y coordinate in object coordinate

vx Return the x coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

vy Return the y coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

vz Return the z coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

8.123.3.9 mitkVolume* mitkImageModel::GetCurrentSlice ()

Get data of the slice currently displayed.

Returns:
Return a one-slice volume which contains the data of the slice currently displayed.

Note:
The returned object pointer should be deleted properly by yourself.

8.123.3.10 int mitkImageModel::GetCurrentSliceNumber () const [inline]

Get the current slice number displayed in ImageView

Returns:
Return the current slice number

8.123.3.11 mitkVolume* mitkImageModel::GetData ()

Get the volume data.

Returns:
Return pointer to the volume data.

8.123.3.12 void mitkImageModel::GetDataValueAndCoordinate (float *objectX*, float *objectY*, int & *vx*, int & *vy*, int & *vz*, float & *rValue*, float & *gValue*, float & *bValue*) const

Get the data value and volume coordinate at a specified object coordinate.

Parameters:
objectX Specify the x coordinate in object coordinate

objectY Specify the y coordinate in object coordinate

vx Return the x coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

vy Return the y coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

vz Return the z coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

rValue Return the red component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue

gValue Return the green component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue

bValue Return the blue component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue

8.123.3.13 int mitkImageModel::GetHeight () const [inline]

Internal function, Don't call it directly.

8.123.3.14 void mitkImageModel::GetIncrements (int *incs*[3]) const [inline]

Get the increments in x, y and z directions according to current view mode.

Parameters:

incs[0] the increment in x directions

incs[1] the increment in y directions

incs[2] the increment in z directions

8.123.3.15 void mitkImageModel::GetObjectCoordinate (int *vx*, int *vy*, int *vz*, float & *objX*, float & *objY*)

Get the coordinates in the object space according to the given volume coordinates.

Parameters:

vx specify the x-coordinate in the volume space

vy specify the y-coordinate in the volume space

vz specify the z-coordinate in the volume space

objX return the x-coordinate in the object space

objY return the y-coordinate in the object space

8.123.3.16 void mitkImageModel::GetObjectCoordinate (int *vx*, int *vy*, float & *objX*, float & *objY*)

Get the coordinates in the object space according to the given volume coordinates.

Parameters:

vx specify the x-coordinate in the volume space (according to view-mode)

vy specify the y-coordinate in the volume space (according to view-mode)

objX return the x-coordinate in the object space

objY return the y-coordinate in the object space

8.123.3.17 float mitkImageModel::GetOpacity () const [inline]

Get the opacity of this model.

Returns:

Return the opacity of this model.

8.123.3.18 float mitkImageModel::GetSpacingX () const [inline]

Internal function, Don't call it directly.

8.123.3.19 float mitkImageModel::GetSpacingY () [inline]

Internal function, Don't call it directly.

8.123.3.20 float mitkImageModel::GetSpacingZ () const [inline]

Internal function, Don't call it directly.

8.123.3.21 unsigned int mitkImageModel::GetTextureID () const [inline]

Get the texture ID.

Returns:
Return the texture ID.

8.123.3.22 int mitkImageModel::GetTotalSliceNumber () const [inline]

Display previous slice. If current slice is the first slice in the volume, then display the last slice.

8.123.3.23 ViewMode mitkImageModel::GetViewMode () const [inline]

Get the view mode of the volume.

See also:
ViewMode

Returns:
Return the view mode used by this ImageModel

8.123.3.24 int mitkImageModel::GetWidth () const [inline]

Internal function, Don't call it directly.

8.123.3.25 float mitkImageModel::GetWindowCenter () const [inline]

Get the window center of this image model

Returns:
Return the window center of this image model

8.123.3.26 float mitkImageModel::GetWindowWidth () const [inline]

Get the window width of this image model

Returns:
Return the window width of this image model

8.123.3.27 void mitkImageModel::GetXYCoordinate (float *objectX*, float *objectY*, float & *vx*, float & *vy*) const [inline]

Get the X-Y volume coordinate at a specified object coordinate according to the view mode.

Parameters:

objectX Specify the x coordinate in object coordinate

objectY Specify the y coordinate in object coordinate

vx Return the x coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

vy Return the y coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

8.123.3.28 void mitkImageModel::GetXYCoordinate (float *objectX*, float *objectY*, int & *vx*, int & *vy*) const [inline]

Get the X-Y volume coordinate at a specified object coordinate according to the view mode.

Parameters:

objectX Specify the x coordinate in object coordinate

objectY Specify the y coordinate in object coordinate

vx Return the x coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

vy Return the y coordinate in the volume. It is calculate by transforming the point(objectX,objectY) from object coordinate to volume coordinate.

8.123.3.29 void mitkImageModel::GetXYCoordinateDelta (float *odx*, float *ody*, float & *vdx*, float & *vdy*) const [inline]

Get the movement in volume coordinate at a specified object coordinate according to the view mode.

Parameters:

odx Specify the movement along x-axis in object coordinate

ody Specify the movement along y-axis in object coordinate

vdx Return the movement along x-axis in the volume coordinate

vdy Return the movement along y-axis in the volume coordinate

8.123.3.30 virtual bool mitkImageModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:

Return true if this model is opaque.

Implements mitkModel.

8.123.3.31 void mitkImageModel::NextSlice ()

Display next slice. If current slice is the last slice in the volume, then display the first slice.

8.123.3.32 void mitkImageModel::PrevSlice ()

Display previous slice. If current slice is the first slice in the volume, then display the last slice.

8.123.3.33 virtual void mitkImageModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkDataModel.

8.123.3.34 virtual int mitkImageModel::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Warning:
Don't call this function directly.

Reimplemented from mitkModel.

8.123.3.35 void mitkImageModel::ResetWindowWidthCenter ()

Reset the window width and center to the default value.

8.123.3.36 void mitkImageModel::SetCurrentSliceNumber (int *sliceNo*)

Set the current slice number for displaying.

Parameters:
sliceNo Specify the current slice number

8.123.3.37 void mitkImageModel::SetData (mitkVolume * *data*)

Set the volume data.

Parameters:
data pointer to an mitkVolume

8.123.3.38 void mitkImageModel::SetOpacity (float *opacity*) [inline]

Set the opacity of this model.

Parameters:
opacity Specify the opacity of this model.

8.123.3.39 void mitkImageModel::SetViewMode (ViewMode *viewMode*)

Set the view mode of the volume.

See also:
ViewMode

Parameters:
viewMode Specify the view mode

8.123.3.40 void mitkImageModel::SetWindowCenter (float *winCenter*)

Set the window center of this image model

Parameters:
winCenter Specify the new window center of this image model

8.123.3.41 void mitkImageModel::SetWindowWidth (float *winWidth*)

Set the window width of this image model

Parameters:
winWidth Specify the new window width of this image model

8.123.3.42 void mitkImageModel::UpdatePseudocolor (bool *rectChanged*)

Update pseudo-color region if the region is changed.

Parameters:
rectChanged if the size or position of the pseudo-color region is changed

The documentation for this class was generated from the following file:

- mitkImageModel.h

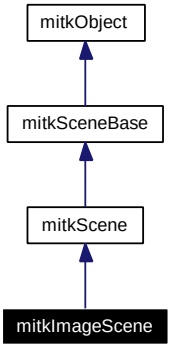
8.124 mitkImageScene Class Reference

mitkImageScene - a Scene to display 2D images

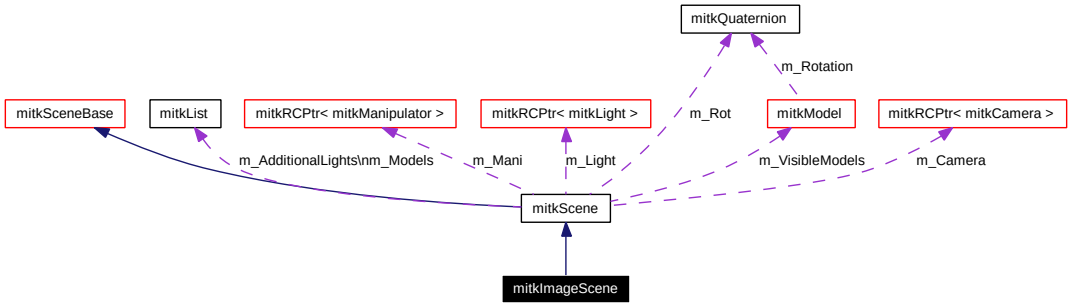
`#include <mitkImageScene.h>`

Inherits mitkScene.

Inheritance diagram for mitkImageScene:



Collaboration diagram for mitkImageScene:



Public Member Functions

- virtual void InitGL ()
- virtual void ResizeGL (int newX, int newY)
- virtual void RenderGL ()
- virtual mitkManipulator * CreateManipulator ()
- void GetDataValueAndCoordinate (int mIdx, int x, int y, int &vx, int &vy, int &vz, float &rValue, float &gValue, float &bValue)
- void GetVolumeCoordinate (int mIdx, int x, int y, int &vx, int &vy, int &vz)
- void GetViewCoordinate (int mIdx, int vx, int vy, int vz, int &sx, int &sy)
- void AdjustWidthCenter (float deltX, float deltY)
- float GetWindowWidth ()
- float GetWindowCenter ()
- void SetWindowWidth (float winWidth)
- void SetWindowCenter (float winCenter)
- void ResetWindowWidthCenter ()

- void EnableCrossArrow ()
- void DisableCrossArrow ()
- void SetCrossArrow (bool isEnabled)
- bool GetCrossArrow ()
- void SetCrossArrowWidth (int width)
- void SetCrossArrowHeight (int height)
- int GetCrossArrowWidth ()
- int GetCrossArrowHeight ()
- void SetCrossArrowPositionX (float xPos)
- void SetCrossArrowPositionY (float yPos)
- void SetCrossArrowPosition (int mIdx, int vx, int vy)
- float GetCrossArrowPositionX ()
- float GetCrossArrowPositionY ()
- void ResetCrossArrowPosition ()
- virtual void ResetScene ()
- virtual void Translate (float deltX, float deltY, float deltZ=0)
- virtual void Rotate (float deltX, float deltY, float deltZ)
- virtual void Rotate (float ax, float ay, float az, float angle)
- virtual void Rotate (const mitkQuaternion &q)
- virtual void SetRotation (float x, float y, float z)
- virtual void SetRotation (float ax, float ay, float az, float angle)
- virtual void SetRotation (const mitkQuaternion &q)
- virtual void Scale (float delt)

8.124.1 Detailed Description

mitkImageScene - a Scene to display 2D images

mitkImageScene is a 2d Scene to display 2-dimensional images. The purpose of providing it is for the convenience. To display 2d images, you must create one or several image models(mitkImageModel) firstly, then add them using the function

```
AddModel()
```

. Same as mitkScene, mitkImageScene can be easily integrated into any user interface toolkit, such as MFC in Microsoft Visual C++, VCL in Borland C++ Builder, Trolltech Qt, etc. The following code snippet demonstrate this point:

```
mitkImageScene *aScene = new mitkImageScene;
//Set the parent window, and fill out the whole parent window
aScene->SetParent(parentWindowId);
aScene->SetLeft(0);
aScene->SetTop(0);
aScene->SetWidth(parentWindowWidth);
aScene->SetHeight(parentWindowHeight);
aScene->Show();
```

So mitkImageScene only requires a parent window id and then can integrate into that window seamless.

8.124.2 Member Function Documentation

8.124.2.1 void mitkImageScene::AdjustWidthCenter (float *deltX*, float *deltY*)

Adjust the window width and window center of this scene

Parameters:

deltX Specify the change of the window width

deltY Specify the change of the window center

8.124.2.2 virtual mitkManipulator* mitkImageScene::CreateManipulator () [virtual]

Create a default manipulator for the mouse events processing. Subclass can override this function to create a proper manipulator.

Reimplemented from mitkScene.

8.124.2.3 void mitkImageScene::DisableCrossArrow () [inline]

Set the cross arrow to disable. If the cross arrow is enabled, the scene will draw a cross arrow and a rectangle centered in the cross.

8.124.2.4 void mitkImageScene::EnableCrossArrow () [inline]

Set the cross arrow to enable. If the cross arrow is enabled, the scene will draw a cross arrow and a rectangle centered in the cross.

8.124.2.5 bool mitkImageScene::GetCrossArrow () [inline]

Get the status of cross arrow. If the cross arrow is enabled, the scene will draw a cross arrow and a rectangle centered in the cross.

Returns:

Return true, the cross arrow is enabled. Return false, the cross arrow is disabled.

8.124.2.6 int mitkImageScene::GetCrossArrowHeight () [inline]

Get the height of the rectangle associated with the cross arrow.

Returns:

Return the height of the rectangle.

8.124.2.7 float mitkImageScene::GetCrossArrowPositionX () [inline]

Get the x coordinate of the cross arrow center.

Returns:

Return the x coordinate of the cross arrow center.

8.124.2.8 float mitkImageScene::GetCrossArrowPositionY () [inline]

Get the y coordinate of the cross arrow center.

Returns:
Return the y coordinate of the cross arrow center.

8.124.2.9 int mitkImageScene::GetCrossArrowWidth () [inline]

Get the width of the rectangle associated with the cross arrow.

Returns:
Return the width of the rectangle.

8.124.2.10 void mitkImageScene::GetDataValueAndCoordinate (int *mIdx*, int *x*, int *y*, int & *vx*, int & *vy*, int & *vz*, float & *rValue*, float & *gValue*, float & *bValue*)

Get the data value and volume coordinate at a specified view coordinate.

Parameters:
mIdx Specify the ith model. The data value and volume coordinate are got from this model.
x Specify the x coordinate in this scene
y Specify the y coordinate in this scene
vx Return the x coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.
vy Return the y coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.
vz Return the z coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.
rValue Return the red component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue
gValue Return the green component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue
bValue Return the blue component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue

8.124.2.11 void mitkImageScene::GetViewCoordinate (int *mIdx*, int *vx*, int *vy*, int *vz*, int & *sx*, int & *sy*)

Get the view (screen) coordinate at a specified volume coordinate.

Parameters:
mIdx Specify the ith model. The data value and volume coordinate are got from this model.
vx the x coordinate in the volume.
vy the y coordinate in the volume.
vz the z coordinate in the volume.
x return the x coordinate in this scene
y return the y coordinate in this scene

8.124.2.12 void mitkImageScene::GetVolumeCoordinate (int *mIdx*, int *x*, int *y*, int & *vx*, int & *vy*, int & *vz*)

Get the volume coordinate at a specified view coordinate.

Parameters:

mIdx Specify the ith model. The data value and volume coordinate are got from this model.

x Specify the x coordinate in this scene

y Specify the y coordinate in this scene

vx Return the x coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

vy Return the y coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

vz Return the z coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

8.124.2.13 float mitkImageScene::GetWindowCenter ()

Get the window center of this scene

Returns:

Return the window center of this scene

8.124.2.14 float mitkImageScene::GetWindowWidth ()

Get the window width of this scene

Returns:

Return the window width of this scene

8.124.2.15 virtual void mitkImageScene::InitGL () [virtual]

Initialize OpenGL variables and resources.

Reimplemented from mitkScene.

8.124.2.16 virtual void mitkImageScene::RenderGL () [virtual]

Handle the render event pass by the container

Reimplemented from mitkScene.

8.124.2.17 void mitkImageScene::ResetCrossArrowPosition ()

Reset the center position and the rectangle size to the default value.

8.124.2.18 virtual void mitkImageScene::ResetScene () [virtual]

Reset the geometric position of all of the models in this scene.

Reimplemented from mitkScene.

8.124.2.19 void mitkImageScene::ResetWindowWidthCenter ()

Reset the window width and center to the default value.

8.124.2.20 virtual void mitkImageScene::ResizeGL (int *newX*, int *newY*) [virtual]

Handle the resize event pass by the container

Reimplemented from mitkScene.

8.124.2.21 virtual void mitkImageScene::Rotate (const mitkQuaternion & *q*) [inline, virtual]

Rotate all of the models in this scene.

Parameters:

q the quaternion of the rotation

Note:

The rotation is incremental.

Reimplemented from mitkScene.

8.124.2.22 virtual void mitkImageScene::Rotate (float *ax*, float *ay*, float *az*, float *angle*) [inline, virtual]

Rotate all of the models in this scene around axis (ax,ay,az) for angle degrees.

Parameters:

ax the x-coordinate of rotation axis

ay the y-coordinate of rotation axis

az the z-coordinate of rotation axis

angle the angle of rotation in degrees

Note:

The rotation is incremental.

Reimplemented from mitkScene.

8.124.2.23 virtual void mitkImageScene::Rotate (float *deltX*, float *deltY*, float *deltZ*) [virtual]

Rotate all of the models in this scene around x, y, z axis.

Parameters:

deltX Specify the rotation around x-axis in degrees.

deltY Specify the rotation around y-axis in degrees.

deltZ Specify the rotation around z-axis in degrees.

Note:
The rotation is incremental.

Reimplemented from mitkScene.

8.124.2.24 virtual void mitkImageScene::Scale (float *delt*) [virtual]

Scale all of the models in this scene in x, y, z directions.

Parameters:
delt Specify the scale in x, y, z directions. This function scales the models equally in each direction.

Reimplemented from mitkScene.

8.124.2.25 void mitkImageScene::SetCrossArrow (bool *isEnabled*) [inline]

Set the status of cross arrow. If the cross arrow is enabled, the scene will draw a cross arrow and a rectangle centered in the cross.

Parameters:
isEnabled isEnabled=true, Set the cross arrow to enable. isEnabled=false, Set the cross arrow to disable.

8.124.2.26 void mitkImageScene::SetCrossArrowHeight (int *height*) [inline]

Set the height of the rectangle associated with the cross arrow.

Parameters:
height Specify the height of the rectangle.

8.124.2.27 void mitkImageScene::SetCrossArrowPosition (int *mIdx*, int *vx*, int *vy*)

Set the cross arrow position according the volume coordinates of the mIdx'th model.

Parameters:
mIdx the index of the model (should be a mitkImageModel)

vx the x-coordinate in the volume space

vy the y-coordinate in the volume space

8.124.2.28 void mitkImageScene::SetCrossArrowPositionX (float *xPos*) [inline]

Set the x coordinate of the cross arrow center.

Parameters:
xPos Specify the x coordinate of the cross arrow center.

8.124.2.29 void mitkImageScene::SetCrossArrowPositionY (float *yPos*) [inline]

Set the y coordinate of the cross arrow center.

Parameters:
yPos Specify the y coordinate of the cross arrow center.

8.124.2.30 void mitkImageScene::SetCrossArrowWidth (int *width*) [inline]

Set the width of the rectangle associated with the cross arrow.

Parameters:
width Specify the width of the rectangle.

8.124.2.31 virtual void mitkImageScene::SetRotation (const mitkQuaternion & *q*) [inline, virtual]

Set rotations of all models in this scene to the quaternion "q".

Parameters:
q the quaternion of the rotation

Reimplemented from mitkScene.

8.124.2.32 virtual void mitkImageScene::SetRotation (float *ax*, float *ay*, float *az*, float *angle*) [inline, virtual]

Set rotations of all models in this scene.

Parameters:
ax the x-coordinate of rotation axis
ay the y-coordinate of rotation axis
az the z-coordinate of rotation axis
angle the angle of rotation in degrees

Reimplemented from mitkScene.

8.124.2.33 virtual void mitkImageScene::SetRotation (float *x*, float *y*, float *z*) [virtual]

Set rotations of all models in this scene.

Parameters:
x Specify the rotation around x-axis in degrees.
y Specify the rotation around y-axis in degrees.
z Specify the rotation around z-axis in degrees.

Reimplemented from mitkScene.

8.124.2.34 void mitkImageScene::SetWindowCenter (float *winCenter*)

Set the window center of this scene

Parameters:

winCenter Specify the new window center of this scene

8.124.2.35 void mitkImageScene::SetWindowWidth (float *winWidth*)

Set the window width of this scene

Parameters:

winWidth Specify the new window width of this scene

8.124.2.36 virtual void mitkImageScene::Translate (float *deltX*, float *deltY*, float *deltZ* = 0) [virtual]

Translate all of the models in this scene in x, y, z directions.

Parameters:

deltX Specify the translation in x direction

deltY Specify the translation in y direction

deltZ Specify the translation in z direction

Reimplemented from mitkScene.

The documentation for this class was generated from the following file:

- mitkImageScene.h

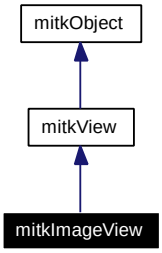
8.125 mitkImageView Class Reference

mitkImageView - a view to display 2D images

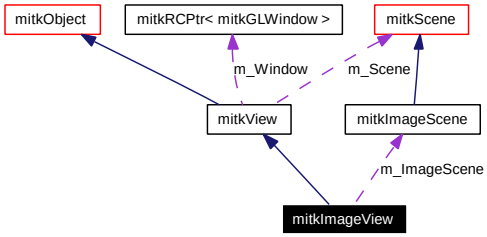
`#include <mitkImageView.h>`

Inherits mitkView.

Inheritance diagram for mitkImageView:



Collaboration diagram for mitkImageView:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void GetDataValueAndCoordinate (int mIdx, int x, int y, int &vx, int &vy, int &vz, float &rValue, float &gValue, float &bValue)
- void GetVolumeCoordinate (int mIdx, int x, int y, int &vx, int &vy, int &vz)
- void GetViewCoordinate (int mIdx, int vx, int vy, int vz, int &sx, int &sy)
- void AdjustWidthCenter (float deltX, float deltY)
- float GetWindowWidth ()
- float GetWindowCenter ()
- void SetWindowWidth (float winWidth)
- void SetWindowCenter (float winCenter)
- void ResetWindowWidthCenter ()
- void EnableCrossArrow ()
- void DisableCrossArrow ()
- void SetCrossArrow (bool isEnabled)
- bool GetCrossArrow ()
- void SetCrossArrowWidth (int width)
- void SetCrossArrowHeight (int height)
- int GetCrossArrowWidth ()
- int GetCrossArrowHeight ()

- void SetCrossArrowPositionX (float xPos)
- void SetCrossArrowPositionY (float yPos)
- void SetCrossArrowPosition (int mIdx, int vx, int vy)
- float GetCrossArrowPositionX ()
- float GetCrossArrowPositionY ()
- void ResetCrossArrowPosition ()

8.125.1 Detailed Description

mitkImageView - a view to display 2D images

mitkImageView is a 2d view to display 2-dimensional images. The purpose of providing it is for the convenience. To display 2d images, you must create one or several image models(mitkImageModel) firstly, then add them using the function

```
AddModel()
```

. Same as mitkView, mitkImageView can be easily integrated into any user interface toolkit, such as MFC in Microsoft Visual C++, VCL in Borland C++ Builder, Trolltech Qt, etc. The following code snippet demonstrate this point:

```
mitkImageView *aView = new mitkImageView;
//Set the parent window, and fill out the whole parent window
aView->SetParent(parentWindowId);
aView->SetLeft(0);
aView->SetTop(0);
aView->SetWidth(parentWindowWidth);
aView->SetHeight(parentWindowHeight);
aView->Show();
```

So mitkImageView only requires a parent window id and then can integrate into that window seamless. From Mitk 2.3, mitkImageView is decomposed into mitkGLWindow and mitkImageScene, it is recommanded to use those two classes instead of this one. This class serves as a wrapper that provides the compatibility to the tranditional routines.

8.125.2 Member Function Documentation

8.125.2.1 void mitkImageView::AdjustWidthCenter (float *deltX*, float *deltY*) [inline]

Adjust the window width and window center of this view

Parameters:

deltX Specify the change of the window width

deltY Specify the change of the window center

8.125.2.2 void mitkImageView::DisableCrossArrow () [inline]

Set the cross arrow to disable. If the cross arrow is enabled, the view will draw a cross arrow and a rectangle centered in the cross.

8.125.2.3 void mitkImageView::EnableCrossArrow () [inline]

Set the cross arrow to enable. If the cross arrow is enabled, the view will draw a cross arrow and a rectangle centered in the cross.

8.125.2.4 bool mitkImageView::GetCrossArrow () [inline]

Get the status of cross arrow. If the cross arrow is enabled, the view will draw a cross arrow and a rectangle centered in the cross.

Returns:
Return true, the cross arrow is enabled. Return false, the cross arrow is disabled.

8.125.2.5 int mitkImageView::GetCrossArrowHeight () [inline]

Get the height of the rectangle associated with the cross arrow.

Returns:
Return the height of the rectangle.

8.125.2.6 float mitkImageView::GetCrossArrowPositionX () [inline]

Get the x coordinate of the cross arrow center.

Returns:
Return the x coordinate of the cross arrow center.

8.125.2.7 float mitkImageView::GetCrossArrowPositionY () [inline]

Get the y coordinate of the cross arrow center.

Returns:
Return the y coordinate of the cross arrow center.

8.125.2.8 int mitkImageView::GetCrossArrowWidth () [inline]

Get the width of the rectangle associated with the cross arrow.

Returns:
Return the width of the rectangle.

8.125.2.9 void mitkImageView::GetDataValueAndCoordinate (int *mIdx*, int *x*, int *y*, int & *vx*, int & *vy*, int & *vz*, float & *rValue*, float & *gValue*, float & *bValue*) [inline]

Get the data value and volume coordinate at a specified view coordinate.

Parameters:
mIdx Specify the ith model. The data value and volume coordinate are got from this model.

x Specify the x coordinate in this view

y Specify the y coordinate in this view

vx Return the x coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

vy Return the y coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

vz Return the z coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

rValue Return the red component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue

gValue Return the green component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue

bValue Return the blue component of the data value at point(vx, vy, vz) in the volume. If the volume is gray image(single channel), then rValue = gValue = bValue

8.125.2.10 void mitkImageView::GetViewCoordinate (int *mIdx*, int *vx*, int *vy*, int *vz*, int & *sx*, int & *sy*) [inline]

Get the view (screen) coordinate at a specified volume coordinate.

Parameters:

mIdx Specify the ith model. The data value and volume coordinate are got from this model.

vx the x coordinate in the volume.

vy the y coordinate in the volume.

vz the z coordinate in the volume.

x return the x coordinate in this view

y return the y coordinate in this view

8.125.2.11 void mitkImageView::GetVolumeCoordinate (int *mIdx*, int *x*, int *y*, int & *vx*, int & *vy*, int & *vz*) [inline]

Get the volume coordinate at a specified view coordinate.

Parameters:

mIdx Specify the ith model. The data value and volume coordinate are got from this model.

x Specify the x coordinate in this view

y Specify the y coordinate in this view

vx Return the x coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

vy Return the y coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

vz Return the z coordinate in the volume. It is calculate by transforming the point(x,y) from view coordinate to volume coordinate.

8.125.2.12 float mitkImageView::GetWindowCenter () [inline]

Get the window center of this view

Returns:
Return the window center of this view

8.125.2.13 float mitkImageView::GetWindowWidth () [inline]

Get the window width of this view

Returns:
Return the window width of this view

8.125.2.14 virtual void mitkImageView::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkView.

8.125.2.15 void mitkImageView::ResetCrossArrowPosition () [inline]

Reset the center position and the rectangle size to the default value.

8.125.2.16 void mitkImageView::ResetWindowWidthCenter () [inline]

Reset the window width and center to the default value.

8.125.2.17 void mitkImageView::SetCrossArrow (bool *isEnabled*) [inline]

Set the status of cross arrow. If the cross arrow is enabled, the view will draw a cross arrow and a rectangle centered in the cross.

Parameters:
isEnabled isEnabled=true, Set the cross arrow to enable. isEnabled=false, Set the cross arrow to disable.

8.125.2.18 void mitkImageView::SetCrossArrowHeight (int *height*) [inline]

Set the height of the rectangle associated with the cross arrow.

Parameters:
height Specify the height of the rectangle.

8.125.2.19 void mitkImageView::SetCrossArrowPosition (int *mIdx*, int *vx*, int *vy*) [inline]

Set the cross arrow position according the volume coordinates of the mIdx'th model.

Parameters:

mIdx the index of the model (should be a mitkImageModel)

vx the x-coordinate in the volume space

vy the y-coordinate in the volume space

8.125.2.20 void mitkImageView::SetCrossArrowPositionX (float *xPos*) [inline]

Set the x coordinate of the cross arrow center.

Parameters:

xPos Specify the x coordinate of the cross arrow center.

8.125.2.21 void mitkImageView::SetCrossArrowPositionY (float *yPos*) [inline]

Set the y coordinate of the cross arrow center.

Parameters:

yPos Specify the y coordinate of the cross arrow center.

8.125.2.22 void mitkImageView::SetCrossArrowWidth (int *width*) [inline]

Set the width of the rectangle associated with the cross arrow.

Parameters:

width Specify the width of the rectangle.

8.125.2.23 void mitkImageView::SetWindowCenter (float *winCenter*) [inline]

Set the window center of this view

Parameters:

winCenter Specify the new window center of this view

8.125.2.24 void mitkImageView::SetWindowWidth (float *winWidth*) [inline]

Set the window width of this view

Parameters:

winWidth Specify the new window width of this view

The documentation for this class was generated from the following file:

- mitkImageView.h

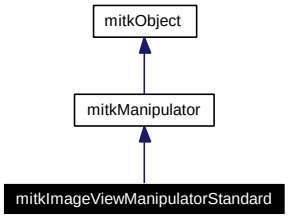
8.126 mitkImageViewManipulatorStandard Class Reference

mitkImageViewManipulatorStandard - a concrete manipulator to process mouse events in an image scene

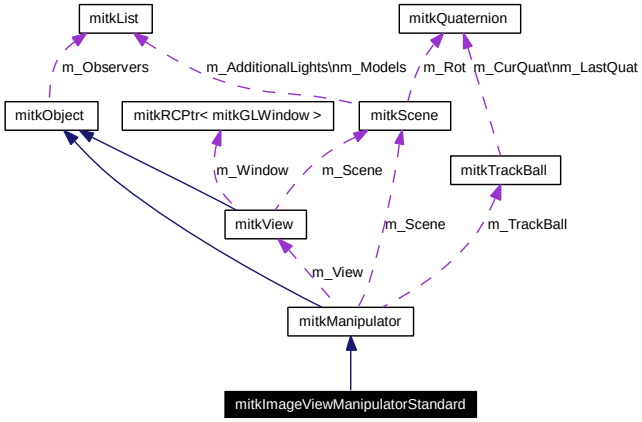
`#include <mitkImageViewManipulatorStandard.h>`

Inherits mitkManipulator.

Inheritance diagram for mitkImageViewManipulatorStandard:



Collaboration diagram for mitkImageViewManipulatorStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)

8.126.1 Detailed Description

mitkImageViewManipulatorStandard - a concrete manipulator to process mouse events in an image scene

mitkImageViewManipulatorStandard is a concrete manipulator to process mouse events in an image scene. It is the default manipulator of a image scene. The behavior of this manipulator is shown as follows:

Mouse Left Key — Translation

Mouse Middle Key — Adjusting window width / center

Mouse Right Key — Zoom in / out

8.126.2 Member Function Documentation

8.126.2.1 virtual void mitkImageViewManipulatorStandard::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkManipulator.

8.126.2.2 virtual void mitkImageViewManipulatorStandard::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

Implements mitkManipulator.

8.126.2.3 virtual void mitkImageViewManipulatorStandard::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkManipulator.

8.126.2.4 virtual void mitkImageViewManipulatorStandard::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkManipulator.

The documentation for this class was generated from the following file:

- mitkImageViewManipulatorStandard.h

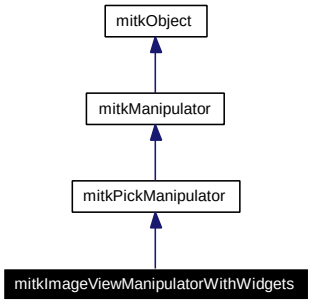
8.127 mitkImageViewManipulatorWithWidgets Class Reference

mitkImageViewManipulatorWithWidgets - manipulator of an image view contains widgets

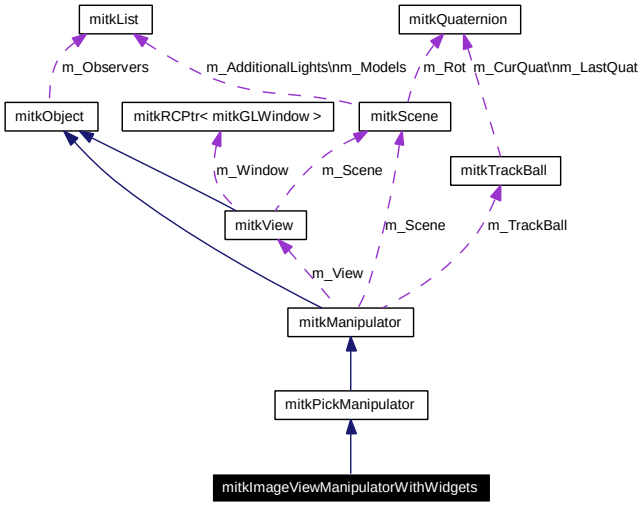
`#include <mitkImageViewManipulatorWithWidgets.h>`

Inherits mitkPickManipulator.

Inheritance diagram for mitkImageViewManipulatorWithWidgets:



Collaboration diagram for mitkImageViewManipulatorWithWidgets:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)

8.127.1 Detailed Description

mitkImageViewManipulatorWithWidgets - manipulator of an image view contains widgets

mitkImageViewManipulatorWithWidgets is a manipulator of an image view contains widgets. It can select a widget in the scene and drive the widget to manipulate the object in the scene.

8.127.2 Member Function Documentation

8.127.2.1 virtual void mitkImageViewManipulatorWithWidgets::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkManipulator.

8.127.2.2 virtual void mitkImageViewManipulatorWithWidgets::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

Implements mitkManipulator.

8.127.2.3 virtual void mitkImageViewManipulatorWithWidgets::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkManipulator.

8.127.2.4 virtual void mitkImageViewManipulatorWithWidgets::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkPickManipulator.

The documentation for this class was generated from the following file:

- mitkImageViewManipulatorWithWidgets.h

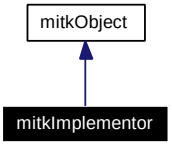
8.128 mitkImplementor Class Reference

mitkImplementor - an abstract class to define an interface to implement some OS dependent operations

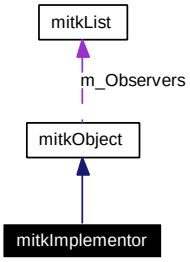
`#include <mitkImplementor.h>`

Inherits mitkObject.

Inheritance diagram for mitkImplementor:



Collaboration diagram for mitkImplementor:



8.128.1 Detailed Description

mitkImplementor - an abstract class to define an interface to implement some OS dependent operations

mitkImplementor is an abstract class to define an interface to implement some OS dependent operations. The purpose of it is to provide an elegant solution to encapsulate the OS dependent codes. Its subclass is created automatically according to the current running OS. User needn't take care for its implementation details.

The documentation for this class was generated from the following file:

- mitkImplementor.h

8.129 mitkInfoReader Class Reference

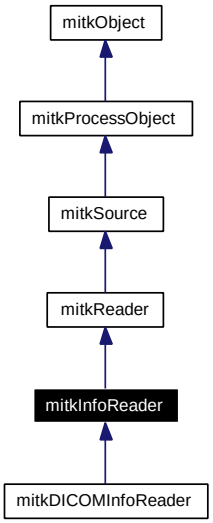
mitkInfoReader - an abstract class represents an information reader

`#include <mitkInfoReader.h>`

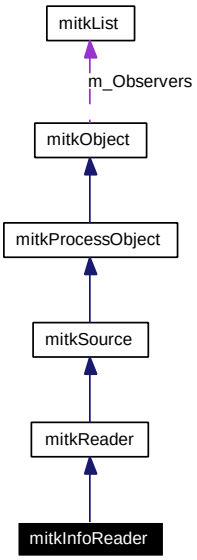
Inherits mitkReader.

Inherited by mitkDICOMInfoReader.

Inheritance diagram for mitkInfoReader:



Collaboration diagram for mitkInfoReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.129.1 Detailed Description

mitkInfoReader - an abstract class represents an information reader

mitkInfoReader is an abstract class represents an information reader. Generally, it reads the header information of some files (e.g. DICOM files).

8.129.2 Member Function Documentation

8.129.2.1 virtual void mitkInfoReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkReader.

Reimplemented in mitkDICOMInfoReader.

The documentation for this class was generated from the following file:

- mitkInfoReader.h

8.130 mitkInterpolateFilter Class Reference

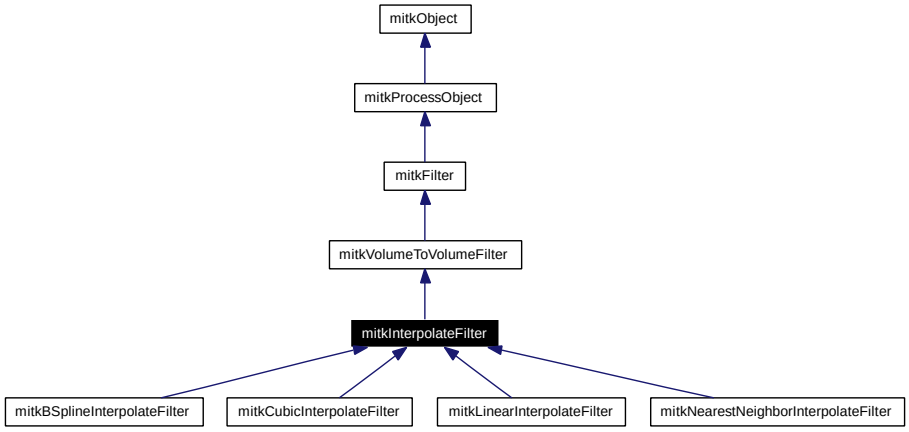
mitkInterpolateFilter - an abstract class specifies interface for interpolating values when objects are resampled through the Transform

`#include <mitkInterpolateFilter.h>`

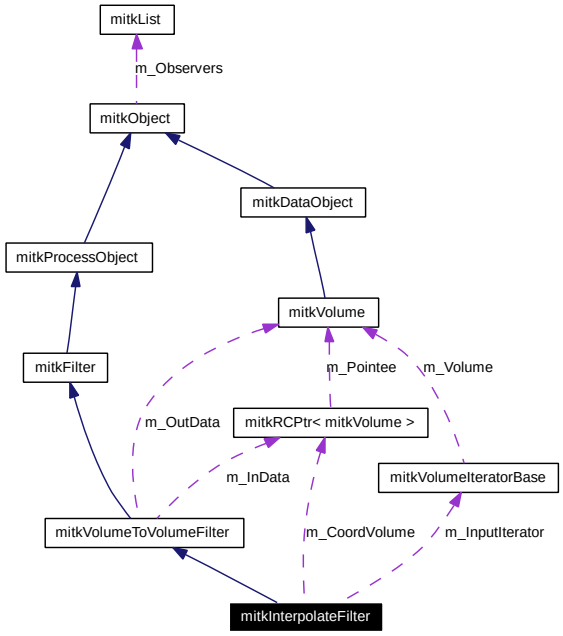
Inherits mitkVolumeToVolumeFilter.

Inherited by mitkBSplineInterpolateFilter, mitkCubicInterpolateFilter, mitkLinearInterpolateFilter, and mitkNearestNeighborInterpolateFilter.

Inheritance diagram for mitkInterpolateFilter:



Collaboration diagram for mitkInterpolateFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetCorrdinates (mitkVolume *coordinates)
- mitkVolume * GetCoordinates ()
- void SetRegion (const VectorIndexType &r)
- void GetRegion (int r[6])
- virtual void Update ()
- virtual bool InterpolatePoint (float x, float y, float z, float &value)
- virtual bool InterpolatePoint (float x, float y, float z, float *value)
- virtual bool InterpolatePoint (const VectorParameterType &index, ScalarPixelType &value)
- virtual bool InterpolatePoint (const VectorParameterType &index, VectorPixelType &value)
- void SetOutputDatatype (int dataType)
- void SetDefaultPixelValue (ScalarPixelType val)

8.130.1 Detailed Description

mitkInterpolateFilter - an abstract class specifies interface for interpolating values when objects are resampled through the Transform

mitkInterpolateFilter is an abstract class specifies interface for interpolating values when objects are resampled through the Transform.

mitkInterpolateFilter and its child class will only works inside the registration framework. If you want to perform geometric transform and interpolation outside the framework, please use mitkGeometricTransformFilter.

Note:
data type of the output is float.

8.130.2 Member Function Documentation

8.130.2.1 mitkVolume* mitkInterpolateFilter::GetCoordinates ()

Get coordinates for interpolating pixels.

Returns:
Return the pointer to the coordinates.

8.130.2.2 void mitkInterpolateFilter::GetRegion (int *r*[6]) [inline]

Get valid region for interpolation.

Parameters:
r[0] Return the first (smaller) index in x axis, the unit is pixel.
r[1] Return the second (bigger) index in x axis, the unit is pixel.
r[2] Return the first (smaller) index in y axis, the unit is pixel.
r[3] Return the second (bigger) index in y axis, the unit is pixel.
r[4] Return the first (smaller) index in z axis, the unit is pixel.
r[5] Return the second (bigger) index in z axis, the unit is pixel.

8.130.2.3 virtual bool mitkInterpolateFilter::InterpolatePoint (const VectorParameterType & *index*, VectorPixelType & *value*) [virtual]

Perform a point interpolation for multi-channel(color) images.

Parameters:

index The index of the point in image.

value The point's intensity value after interpolation.

Returns:

Return true if the interpolation was performed without error.

8.130.2.4 virtual bool mitkInterpolateFilter::InterpolatePoint (const VectorParameterType & *index*, ScalarPixelType & *value*) [virtual]

Perform a point interpolation for single channel images.

Parameters:

index The index of the point in image.

value The point's intensity value after interpolation.

Returns:

Return true if the interpolation was performed without error.

8.130.2.5 virtual bool mitkInterpolateFilter::InterpolatePoint (float *x*, float *y*, float *z*, float * *value*) [virtual]

Perform a point interpolation for multi-channel(color) images.

Parameters:

x The x index of the point in image.

y The y index of the point in image.

z The z index of the point in image.

value The point's intensity value after interpolation.

Returns:

Return true if the interpolation was performed without error.

8.130.2.6 virtual bool mitkInterpolateFilter::InterpolatePoint (float *x*, float *y*, float *z*, float & *value*) [virtual]

Perform a point interpolation for single channel images.

Parameters:

x The x index of the point in image.

y The y index of the point in image.

z The z index of the point in image.

value The point's intensity value after interpolation.

Returns:

Return true if the interpolation was performed without error.

8.130.2.7 virtual void mitkInterpolateFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

Reimplemented in mitkBSplineInterpolateFilter, mitkCubicInterpolateFilter, mitkLinearInterpolateFilter, and mitkNearestNeighborInterpolateFilter.

8.130.2.8 void mitkInterpolateFilter::SetCorrdinates (mitkVolume * *coordinates*) [inline]

Set coordinates for interpolating pixels.

Parameters:
coordinates The pointer to the coordinates for interpolating pixels.

8.130.2.9 void mitkInterpolateFilter::SetDefaultPixelValue (ScalarPixelType *val*) [inline]

Set the default pixel value for background.

Parameters:
val The pixel value.

8.130.2.10 void mitkInterpolateFilter::SetOutputDatatype (int *dataType*) [inline]

Set the data type for output volume.

Parameters:
dataType Specify the output volume's data type.

8.130.2.11 void mitkInterpolateFilter::SetRegion (const VectorIndexType & *r*) [inline]

Set valid region for interpolation.

Parameters:
r[0] The first (smaller) index in x axis, the unit is pixel.
r[1] The second (bigger) index in x axis, the unit is pixel.
r[2] The first (smaller) index in y axis, the unit is pixel.
r[3] The second (bigger) index in y axis, the unit is pixel.
r[4] The first (smaller) index in z axis, the unit is pixel.
r[5] The second (bigger) index in z axis, the unit is pixel.

8.130.2.12 virtual void mitkInterpolateFilter::Update () [virtual]

Initialize the Interpolator . Perform before Run() function.

The documentation for this class was generated from the following file:

- mitkInterpolateFilter.h

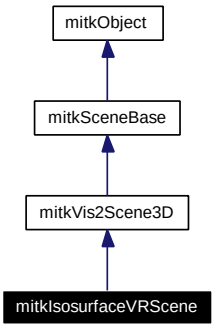
8.131 mitkIsosurfaceVRScene Class Reference

mitkIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Optimized for multicore CPU.

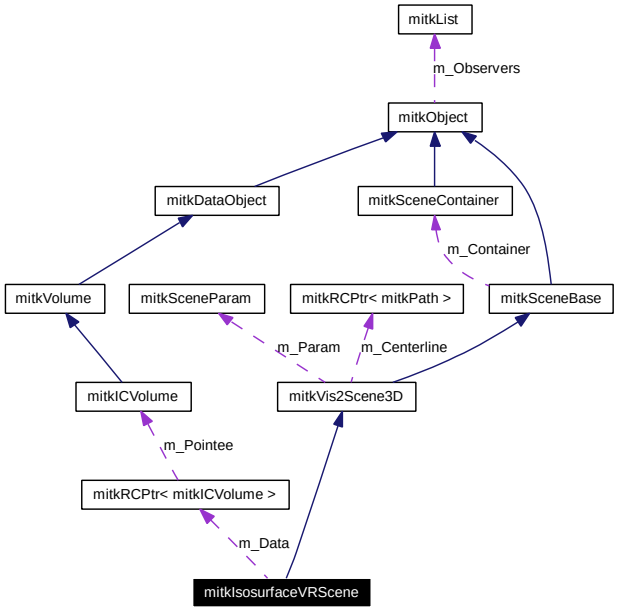
`#include <mitkIsosurfaceVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkIsosurfaceVRScene:



Collaboration diagram for mitkIsosurfaceVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetCoherentData (mitkCoherentVolume *CVolume)
- void SetCoherentDataModified ()

- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)
- void SetIsovalue (float isovalue)
- float GetIsovalue ()
- void SetSurfaceColor (float r, float g, float b)
- void GetSurfaceColor (float &r, float &g, float &b)
- void SetLightDirection (float x, float y, float z)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- void RotateLightDirection (bool turnOn)
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void SetIllustrativeShading (int illustrative)
- bool GetIllustrativeShading ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.131.1 Detailed Description

mitkIsosurfaceVRScene - a 3D scene to display isosurfaces from volume datasets. Optimized for multicore CPU.

mitkIsosurfaceVRScene is a 3D scene to display isosurfaces from volume datasets. Optimized for multicore CPU. mitkIsosurfaceVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container. Two kinds of data source are supported, ICVolume and CohrerentVolume.

8.131.2 Member Function Documentation

8.131.2.1 float mitkIsosurfaceVRScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.131.2.2 void mitkIsosurfaceVRScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:

r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.131.2.3 void mitkIsosurfaceVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:

CroppingBounds,: the cropping boundary

8.131.2.4 float mitkIsosurfaceVRScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:

Return The diffuse factor of the material of the isosurface

8.131.2.5 bool mitkIsosurfaceVRScene::GetIllustrativeShading ()

Get the illustrative shading status

Returns:

: Return the illustrative shading status.

8.131.2.6 float mitkIsosurfaceVRScene::GetIsovalue ()

Get the isovalue of the isosurface to be displayed.

Returns:

Return the isovalue.

8.131.2.7 float mitkIsosurfaceVRScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:

Return The specular factor of the material of the isosurface

8.131.2.8 float mitkIsosurfaceVRScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.131.2.9 void mitkIsosurfaceVRScene::GetSurfaceColor (float & *r*, float & *g*, float & *b*)

Get the color of the isosurface to be displayed

Parameters:
r Return the red component of the surface color.
g Return the green component of the surface color.
b Return the blue component of the surface color.

8.131.2.10 virtual void mitkIsosurfaceVRScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkVis2Scene3D.

8.131.2.11 virtual void mitkIsosurfaceVRScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkVis2Scene3D.

8.131.2.12 virtual void mitkIsosurfaceVRScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkVis2Scene3D.

8.131.2.13 virtual void mitkIsosurfaceVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.131.2.14 void mitkIsosurfaceVRScene::RotateLightDirection (bool *turnOn*)

Turn this on to enter the light rotation interaction mode.

Parameters:
Whether turn the light rotation mode on.

8.131.2.15 void mitkIsosurfaceVRScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:
value The ambient factor of the material of the isosurface

8.131.2.16 void mitkIsosurfaceVRScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:
r Specify the red component of the background color.
g Specify the green component of the background color.
b Specify the blue component of the background color.

8.131.2.17 void mitkIsosurfaceVRScene::SetCoherentData (mitkCoherentVolume * *CVolume*)

Set the data object(CoherentVolume) to be rendered.

Parameters:
data The data object to be rendered.

8.131.2.18 void mitkIsosurfaceVRScene::SetCoherentDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.131.2.19 void mitkIsosurfaceVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.131.2.20 void mitkIsosurfaceVRScene::SetData (mitkICVolume * *data*)

Set the data object(ICVolume) to be rendered.

Parameters:
data The data object to be rendered.

8.131.2.21 void mitkIsosurfaceVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.131.2.22 void mitkIsosurfaceVRScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.131.2.23 void mitkIsosurfaceVRScene::SetIllustrativeShading (int *illustrative*)

Turn on or turn off illustrative shading.

Parameters:
shade,: The illustrative shading status.

8.131.2.24 void mitkIsosurfaceVRScene::SetIsovalue (float *isovalue*)

Set the isovalue of the isosurface to be displayed.

Parameters:
isovalue The isovalue.

8.131.2.25 void mitkIsosurfaceVRScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.
g Specify the green component of the light source color.
b Specify the blue component of the light source color.

8.131.2.26 void mitkIsosurfaceVRScene::SetLightDirection (float *x*, float *y*, float *z*)

Set the light direction

Parameters:
x Specify the x component of the light direction.
y Specify the y component of the light direction.
z Specify the z component of the light direction.

8.131.2.27 void mitkIsosurfaceVRScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.131.2.28 void mitkIsosurfaceVRScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.131.2.29 void mitkIsosurfaceVRScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.131.2.30 void mitkIsosurfaceVRScene::SetSurfaceColor (float *r*, float *g*, float *b*)

Set the color of the isosurface to be displayed

Parameters:
r Specify the red component of the surface color.
g Specify the green component of the surface color.
b Specify the blue component of the surface color.

The documentation for this class was generated from the following file:

- mitkIsosurfaceVRScene.h

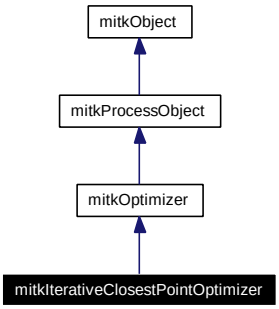
8.132 mitkIterativeClosestPointOptimizer Class Reference

mitkIterativeClosestPointOptimizer - a concrete class to iteratively align two point sets.

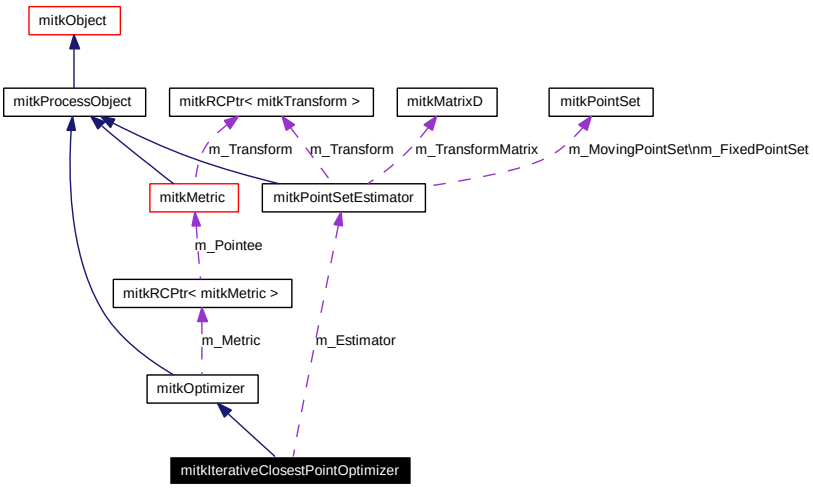
`#include <mitkIterativeClosestPointOptimizer.h>`

Inherits mitkOptimizer.

Inheritance diagram for mitkIterativeClosestPointOptimizer:



Collaboration diagram for mitkIterativeClosestPointOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkIterativeClosestPointOptimizer ()
- mitkPointSetEstimator * GetEstimator ()
- void SetEstimator (mitkPointSetEstimator *estimator)
- void SetInitializedByCentroidsFlag (bool flag)
- void SetMaxMeanDistance (double val)

8.132.1 Detailed Description

mitkIterativeClosestPointOptimizer - a concrete class to iteratively align two point sets.

mitkIterativeClosestPointOptimizer is a concrete class to iteratively align two point sets. The core of the algorithm is to match each point in moving point set with the closest point in fixed point set, then apply the transformation that modify the moving point set to best match the fixed point set (in a least square sense). This has to be iterated to get proper convergence of the two point sets.

8.132.2 Constructor & Destructor Documentation

8.132.2.1 mitkIterativeClosestPointOptimizer::mitkIterativeClosestPointOptimizer ()

Constructor.

8.132.3 Member Function Documentation

8.132.3.1 mitkPointSetEstimator* mitkIterativeClosestPointOptimizer::GetEstimator ()

Get the transform parameter estimator.

Returns:
Return the pointer of estimator.

8.132.3.2 virtual void mitkIterativeClosestPointOptimizer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkOptimizer.

8.132.3.3 void mitkIterativeClosestPointOptimizer::SetEstimator (mitkPointSetEstimator * *estimator*) [inline]

Set the transform parameter estimator.

Parameters:
estimator The pointer to the estimator.

8.132.3.4 void mitkIterativeClosestPointOptimizer::SetInitializedByCentroidsFlag (bool *flag*) [inline]

Starts the process by translating moving ps' centroid to fixed ps' centroid.

Parameters:
flag Initialize the transform by centroids if flag is true.

8.132.3.5 void mitkIterativeClosestPointOptimizer::SetMaxMeanDistance (double *val*) [inline]

Set the maximum mean distance between two iteration. If the mean distance is lower than this, the convergence stops.

Parameters:

val The mean distance value.

The documentation for this class was generated from the following file:

- mitkIterativeClosestPointOptimizer.h

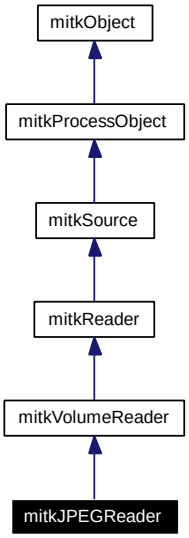
8.133 mitkJPEGReader Class Reference

mitkJPEGReader - a concrete reader for reading JPEG image files

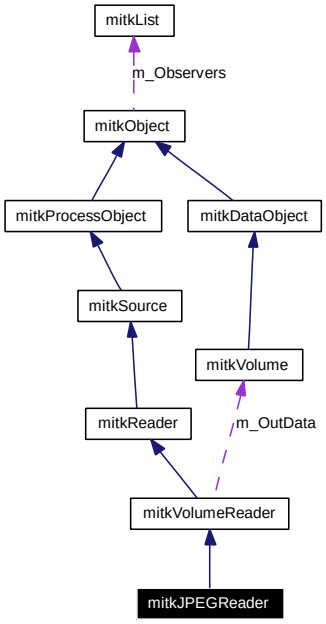
`#include <mitkJPEGReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkJPEGReader:



Collaboration diagram for mitkJPEGReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetSpacingX (float px)
- void SetSpacingY (float py)
- void SetSpacingZ (float pz)
- void SetSpacings (float s[3])

8.133.1 Detailed Description

mitkJPEGReader - a concrete reader for reading JPEG image files

mitkJPEGReader reads a set of JPEG image files to a volume. Because JPEG file doesn't have the spacing information, you must set them using the SetSpacingX, SetSpacingY, SetSpacingZ functions. To use this reader, the code snippet is:

```
mitkJPEGReader *aReader = new mitkJPEGReader;
aReader->SetSpacingX(sx);
aReader->SetSpacingY(sy);
aReader->SetSpacingZ(sz);
aReader->AddFileName(file1);
aReader->AddFileName(file2);
...  ...  ...
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

Warning:
All of the images must have equal width and height. Otherwise they can't form a volume. Now MITK doesn't support JPEG 2000 standard.

8.133.2 Member Function Documentation

8.133.2.1 virtual void mitkJPEGReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

8.133.2.2 void mitkJPEGReader::SetSpacings (float *s*[3]) [inline]

Set spacing information in x, y, z axis respectively, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.

py the spacing (mm) in two adjacent voxels in y axis.

pz the spacing (mm) in two adjacent voxels in z axis.

8.133.2.3 void mitkJPEGReader::SetSpacingX (float *px*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.

8.133.2.4 void mitkJPEGReader::SetSpacingY (float *py*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:
py the spacing (mm) in two adjacent voxels in y axis.

8.133.2.5 void mitkJPEGReader::SetSpacingZ (float *pz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:
pz the spacing (mm) in two adjacent voxels in z axis.

The documentation for this class was generated from the following file:

- mitkJPEGReader.h

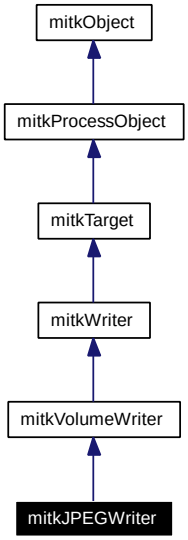
8.134 mitkJPEGWriter Class Reference

mitkJPEGWriter - a concrete writer for writing a volume to JPEG image files

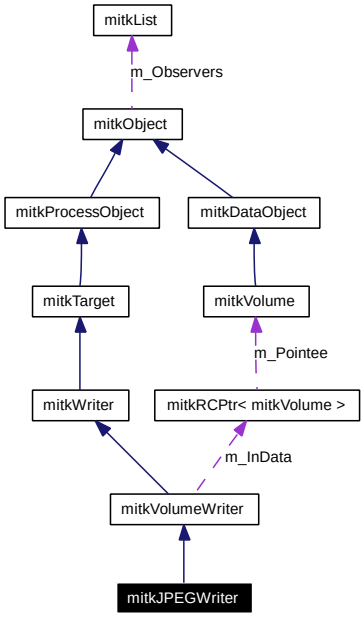
`#include <mitkJPEGWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkJPEGWriter:



Collaboration diagram for mitkJPEGWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetQuality (int quality)

8.134.1 Detailed Description

mitkJPEGWriter - a concrete writer for writing a volume to JPEG image files

mitkJPEGWriter writes a volume to a set of JPEG image files. Because the volume is a 3D dataset, it may contain many slices. So the file names must be generated and passed to writer properly. To use this writer, the code snippet is:

```
mitkJPEGWriter *aWriter = new mitkJPEGWriter;
aWriter->SetInput(aVolume);
int imageNum = aVolume->GetImageNum();
Gerenate file names into files[imageNum];
for(int i = 0; i < imageNum; i++)
    aWriter->AddFileName(files[i]);
aWriter->Run();
```

8.134.2 Member Function Documentation

8.134.2.1 virtual void mitkJPEGWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

8.134.2.2 void mitkJPEGWriter::SetQuality (int *quality*) [inline]

Set the quality of output JPEG images.

Parameters:
quality The quality of output JPEG images. The maximum value is 100, and the default value is 95.

The documentation for this class was generated from the following file:

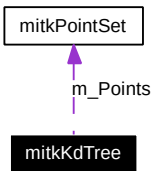
- mitkJPEGWriter.h

8.135 mitkKdTree Class Reference

mitkKdTree - a class for implementation of k-d tree data structure

`#include <mitkKdTree.h>`

Collaboration diagram for mitkKdTree:



Public Member Functions

- mitkKdTree (mitkPointSet *points, int points_per_leaf=4)
- ~mitkKdTree ()
- void GetPointsKNearest (double *queryPoint, int k, vector< int > &closestIndices)
- void GetPointsInBoundingBox (mitkBoundingBox< double > &box, vector< int > &indices_in_box)
- void GetPointsInRadius (double *query_point, double radius, vector< int > &indices_in_radius)
- void SetFlagUseHeap (bool flag)

8.135.1 Detailed Description

mitkKdTree - a class for implementation of k-d tree data structure

mitkKdTree is a class for the implementation of k-d tree data structure. It's designed to accelerate point set k nearest neighbor searching and points indexing. Some codes are borrowed from rsdl library.

8.135.2 Constructor & Destructor Documentation

8.135.2.1 mitkKdTree::mitkKdTree (mitkPointSet * *points*, int *points_per_leaf* = 4)

Constructor with tree building.

Parameters:

points The input point set.

points_per_leaf Max number of points can be stored in a leaf.

8.135.2.2 mitkKdTree::~mitkKdTree ()

Deconstructor.

8.135.3 Member Function Documentation

8.135.3.1 void mitkKdTree::GetPointsInBoundingBox (mitkBoundingBox< double > & *box*, vector< int > & *indices_in_box*)

Get points in a bounding box.

Parameters:

box The query bounding box

indices_in_box Return a vector contains result point's indices

8.135.3.2 void mitkKdTree::GetPointsInRadius (double * *query_point*, double *radius*, vector< int > & *indices_in_radius*)

Get points in a given circle/sphere.

Parameters:

query_point The query point

radius The radius of the given circle

indices_in_radius Return a vector contains result point's indices

8.135.3.3 void mitkKdTree::GetPointsKNearest (double * *queryPoint*, int *k*, vector< int > & *closestIndices*)

Get k nearest point to a query point.

Parameters:

queryPoint The query point stored in an array

k The number of points to search

closestIndices Return a vector contains result point's indices

8.135.3.4 void mitkKdTree::SetFlagUseHeap (bool *flag*) [inline]

Set knn search using heap flag.

Parameters:

flag Use heap in knn search if flag is true

The documentation for this class was generated from the following file:

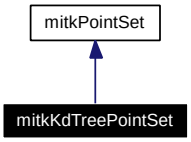
- mitkKdTree.h

8.136 mitkKdTreePointSet Class Reference

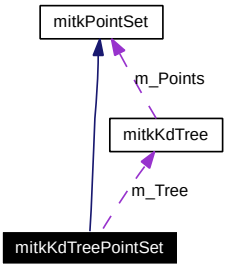
`#include <mitkKdTreePointSet.h>`

Inherits mitkPointSet.

Inheritance diagram for mitkKdTreePointSet:



Collaboration diagram for mitkKdTreePointSet:



Public Member Functions

- mitkKdTreePointSet ()
- virtual ~mitkKdTreePointSet ()
- mitkKdTreePointSet (mitkPointSet *pointSet)
- virtual void Initialize ()
- virtual void GetNearestPoint (double *query_point, double *result_point, int &i)
- virtual void GetKNearestPoints (double *query_point, unsigned int k, vector< int > &indices)
- virtual void GetKNearestPoints (vector< double > *query_point, unsigned int k, vector< int > &indices)
- virtual void GetNearestPoints (mitkPointSet *query_pointset, vector< int > &indices)
- virtual void GetPointsInBoundingBox (mitkBoundingBox< double > &box, vector< int > &indices)
- virtual void GetPointsInRadius (double *query_point, double radius, vector< int > &indices)

8.136.1 Detailed Description

mitkKdTreePointSet - a class to represent geometry in the form of a set of points in two or three dimensional space using k-d tree data structure.

mitkKdTreePointSet is a class to represent geometry in the form of a set of points in two or three dimensional space using k-d tree data structure. A fast k nearest neighbor point searching is supported in this class.

8.136.2 Constructor & Destructor Documentation

8.136.2.1 mitkKdTreePointSet::mitkKdTreePointSet ()

Constructor.

8.136.2.2 virtual mitkKdTreePointSet::~mitkKdTreePointSet () [virtual]

De-constructor.

8.136.2.3 mitkKdTreePointSet::mitkKdTreePointSet (mitkPointSet * *pointSet*)

Constructor initialized with a point set.

Parameters:

pointSet The input point set.

8.136.3 Member Function Documentation

8.136.3.1 virtual void mitkKdTreePointSet::GetKNearestPoints (vector< double > * *query_point*, unsigned int *k*, vector< int > & *indices*) [virtual]

Get k nearest points to the query point.

Parameters:

query_point The query point stored in a vector

k The number of points to search

indices Return a vector contains result point's indices

Reimplemented from mitkPointSet.

8.136.3.2 virtual void mitkKdTreePointSet::GetKNearestPoints (double * *query_point*, unsigned int *k*, vector< int > & *indices*) [virtual]

Get k nearest point to the query point.

Parameters:

query_point The query point stored in an array

k The number of points to search

indices Return a vector contains result point's indices

Reimplemented from mitkPointSet.

8.136.3.3 virtual void mitkKdTreePointSet::GetNearestPoint (double * *query_point*, double * *result_point*, int & *i*) [virtual]

Get a nearest point to the query point.

Parameters:

query_point The query point

result_point Return the pointer to result point

i Return the index to result point

Reimplemented from mitkPointSet.

8.136.3.4 virtual void mitkKdTreePointSet::GetNearestPoints (mitkPointSet * *query_pointset*, vector< int > & *indices*) [virtual]

Get nearest point to a set of query points.

Parameters:

query_pointset The query points stored in a mitkPointSet

indices Return a vector contains result point's indices

Reimplemented from mitkPointSet.

8.136.3.5 virtual void mitkKdTreePointSet::GetPointsInBoundingBox (mitkBoundingBox< double > & *box*, vector< int > & *indices*) [virtual]

Get points in a bounding box.

Parameters:

box The query bounding box

indices Return a vector contains result point's indices

Reimplemented from mitkPointSet.

8.136.3.6 virtual void mitkKdTreePointSet::GetPointsInRadius (double * *query_point*, double *radius*, vector< int > & *indices*) [virtual]

Get points in a given circle/sphere.

Parameters:

query_point The query point

radius The radius of the given circle

indices Return a vector contains result point's indices

Reimplemented from mitkPointSet.

8.136.3.7 virtual void mitkKdTreePointSet::Initialize () [virtual]

Initialize the point set. (build tree)

Reimplemented from mitkPointSet.

The documentation for this class was generated from the following file:

- mitkKdTreePointSet.h

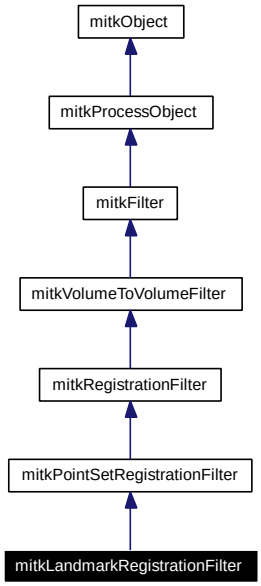
8.137 mitkLandmarkRegistrationFilter Class Reference

mitkLandmarkRegistrationFilter - implementation of landmark-based registration framework.

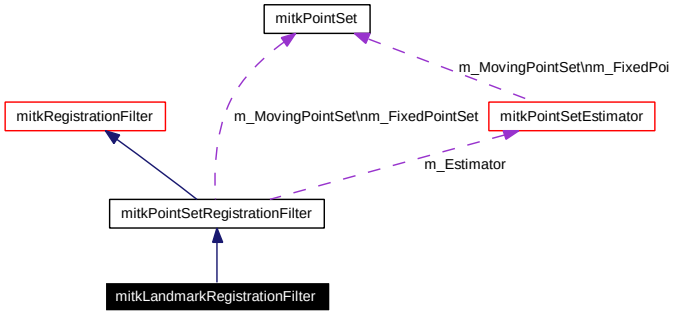
`#include <mitkLandmarkRegistrationFilter.h>`

Inherits mitkPointSetRegistrationFilter.

Inheritance diagram for mitkLandmarkRegistrationFilter:



Collaboration diagram for mitkLandmarkRegistrationFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkLandmarkRegistrationFilter ()

8.137.1 Detailed Description

mitkLandmarkRegistrationFilter - implementation of landmark-based registration framework.

mitkLandmarkRegistrationFilter is an implementation of landmark-based registration framework. The algorithm requires users to manually locate two landmark point sets from images, and it will compute the final transform parameter directly using a Unit Quaternion / SVD based closed-form solution.

8.137.2 Constructor & Destructor Documentation

8.137.2.1 mitkLandmarkRegistrationFilter::mitkLandmarkRegistrationFilter ()

Constructor.

8.137.3 Member Function Documentation

8.137.3.1 virtual void mitkLandmarkRegistrationFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkPointSetRegistrationFilter.

The documentation for this class was generated from the following file:

- mitkLandmarkRegistrationFilter.h

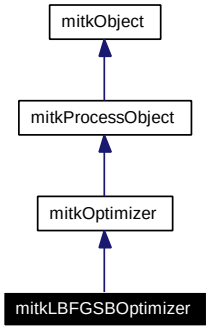
8.138 mitkLBFGSBOptimizer Class Reference

mitkLBFGSBOptimizer -

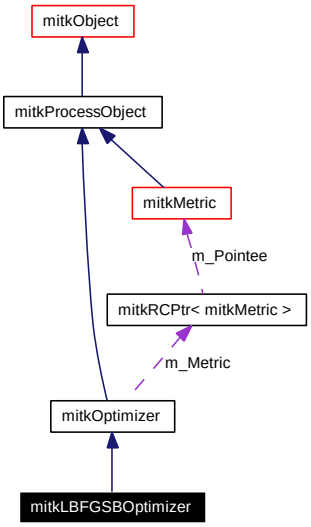
`#include <mitkLBFGSBOptimizer.h>`

Inherits mitkOptimizer.

Inheritance diagram for mitkLBFGSBOptimizer:



Collaboration diagram for mitkLBFGSBOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetLowerBound (const VectorParameterType &bound)
- void SetUpperBound (const VectorParameterType &bound)
- void SetBoundSelection (const VectorModeType &selection)
- void SetCostFunctionConvergenceFactor (double factor)
- void SetProjectedGradientTolerance (double tol)
- void SetMaximumEvaluations (unsigned int eval)
- void SetMaximumCorrections (unsigned int corr)

8.138.1 Detailed Description

mitkLBFGSBOptimizer -

mitkLBFGSBOptimizer

8.138.2 Member Function Documentation

8.138.2.1 virtual void mitkLBFGSBOptimizer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkOptimizer.

8.138.2.2 void mitkLBFGSBOptimizer::SetBoundSelection (const VectorModeType & *selection*) [inline]

Set the boundary condition for each variable, where select[i] = 0 if x[i] is unbounded, = 1 if x[i] has only a lower bound, = 2 if x[i] has both lower and upper bounds, and = 3 if x[1] has only an upper bound

8.138.2.3 void mitkLBFGSBOptimizer::SetCostFunctionConvergenceFactor (double *factor*) [inline]

Set/Get the CostFunctionConvergenceFactor. Algorithm terminates when the reduction in cost function is less than factor $*$ epsmcj where epsmch is the machine precision. Typical values for factor: 1e+12 for low accuracy; 1e+7 for moderate accuracy and 1e+1 for extremely high accuracy.

8.138.2.4 void mitkLBFGSBOptimizer::SetLowerBound (const VectorParameterType & *bound*) [inline]

Set the lower bound value for each variable.

8.138.2.5 void mitkLBFGSBOptimizer::SetMaximumCorrections (unsigned int *corr*) [inline]

Set/Get the MaximumNumberOfCorrections. Default is 5

8.138.2.6 void mitkLBFGSBOptimizer::SetMaximumEvaluations (unsigned int *eval*) [inline]

Set/Get the MaximumNumberOfEvaluations. Default is 500

8.138.2.7 void mitkLBFGSBOptimizer::SetProjectedGradientTolerance (double *tol*) [inline]

Set/Get the ProjectedGradientTolerance. Algorithm terminates when the project gradient is below the tolerance. Default value is 1e-5.

8.138.2.8 void mitkLBFGSBOptimizer::SetUpperBound (const VectorParameterType & *bound*) [inline]

Set the upper bound value for each variable.

The documentation for this class was generated from the following file:

- mitkLBFGSBOptimizer.h

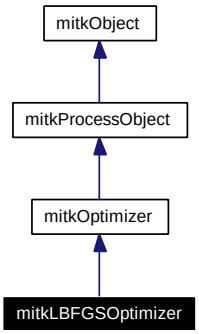
8.139 mitkLBFGSOptimizer Class Reference

mitkLBFGSOptimizer -

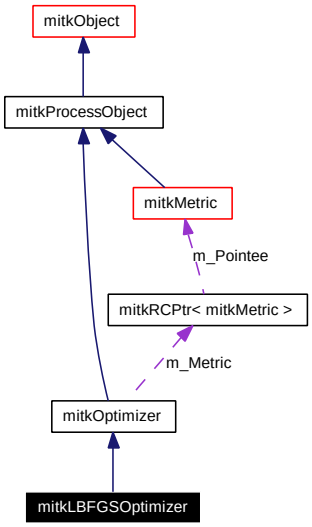
`#include <mitkLBFGSOptimizer.h>`

Inherits mitkOptimizer.

Inheritance diagram for mitkLBFGSOptimizer:



Collaboration diagram for mitkLBFGSOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.139.1 Detailed Description

mitkLBFGSOptimizer -

mitkLBFGSOptimizer

8.139.2 Member Function Documentation

8.139.2.1 virtual void mitkLBFGSOptimizer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkOptimizer.

The documentation for this class was generated from the following file:

- mitkLBFGSOptimizer.h

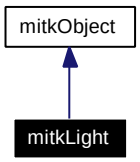
8.140 mitkLight Class Reference

mitkLight - a light object in 3D scene

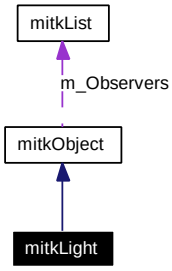
`#include <mitkLight.h>`

Inherits mitkObject.

Inheritance diagram for mitkLight:



Collaboration diagram for mitkLight:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.140.1 Detailed Description

mitkLight - a light object in 3D scene

mitkLight is a light object in 3D scene. The only way you can access it is to get a pointer to it by using the member function GetLight() of mitkScene. Then you can control it through its interface, and the change will affect the image rendered in the scene.

8.140.2 Member Function Documentation

8.140.2.1 virtual void mitkLight::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

The documentation for this class was generated from the following file:

- mitkLight.h

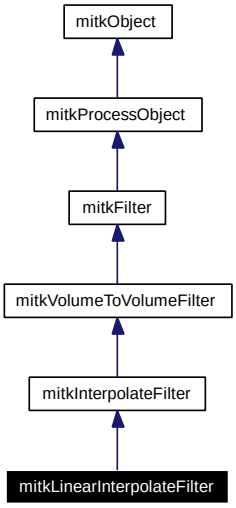
8.141 mitkLinearInterpolateFilter Class Reference

mitkLinearInterpolateFilter - a concrete class for bilinear and trilinear interpolation.

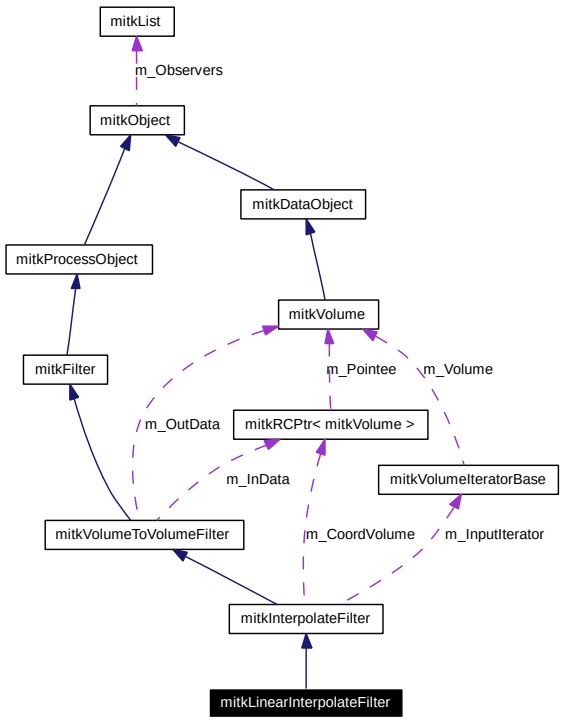
`#include <mitkLinearInterpolateFilter.h>`

Inherits mitkInterpolateFilter.

Inheritance diagram for mitkLinearInterpolateFilter:



Collaboration diagram for mitkLinearInterpolateFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkLinearInterpolateFilter ()

8.141.1 Detailed Description

mitkLinearInterpolateFilter - a concrete class for bilinear and trilinear interpolation.

mitkLinearInterpolateFilter is a concrete class for bilinear and trilinear interpolation.A bi-linear interpolation algorithem was implemented for 2D image which the intensity at a point is determined from the weighted sum of intensities at four pixels closest to it. Also a tri-linear method was developed for 3d volume interpolation that the intensity at a point is estimated from 2x2x2 neighborhood.

This filter can perform point interpolation by using InterpolatePoint() and perform volume interpolation by using Run().

User should specify the input volume and run Update() first before interpolation.

8.141.2 Constructor & Destructor Documentation

8.141.2.1 mitkLinearInterpolateFilter::mitkLinearInterpolateFilter ()

Constructor.

8.141.3 Member Function Documentation

8.141.3.1 virtual void mitkLinearInterpolateFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkInterpolateFilter.

The documentation for this class was generated from the following file:

- mitkLinearInterpolateFilter.h

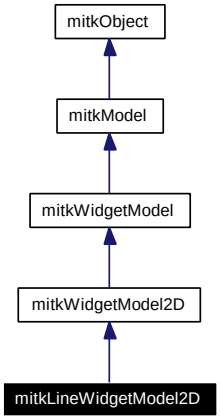
8.142 mitkLineWidgetModel2D Class Reference

mitkLineWidgetModel2D - a line widget used in 2D scene

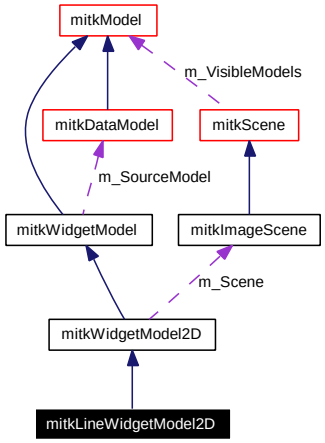
`#include <mitkLineWidgetModel2D.h>`

Inherits mitkWidgetModel2D.

Inheritance diagram for mitkLineWidgetModel2D:



Collaboration diagram for mitkLineWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkLineWidgetModel2D ()
- mitkLineWidgetModel2D (float startPoint[2])
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- float GetLineLength ()
- void SetUnitName (const string &name)

- const string & GetUnitName ()
- void SetStartPoint (float point[2])
- void SetStartPoint (float x, float y)
- void SetStartPoint (int sx, int sy)
- void SetMovePoint (float point[2])
- void SetMovePoint (float x, float y)
- void SetMovePoint (int sx, int sy)
- void SetEndPoint (float point[2])
- void SetEndPoint (float x, float y)
- void SetEndPoint (int sx, int sy)
- int GetPointsSetNum ()
- void GetStartPoint (float &tx, float &ty)
- void GetStartPoint (int &ix, int &iy)
- void GetEndPoint (float &tx, float &ty)
- void GetEndPoint (int &ix, int &iy)

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.142.1 Detailed Description

mitkLineWidgetModel2D - a line widget used in 2D scene

mitkLineWidgetModel2D is a line widget used in 2D scene which can respond the mouse events to change the line status and return the current length of the line. It is supposed to be attached to a 2D data model (e.g. mitkImageModel) and add to a 2D scene (e.g. mitkImageScene), and in other conditions the display could be improper.

8.142.2 Constructor & Destructor Documentation

8.142.2.1 mitkLineWidgetModel2D::mitkLineWidgetModel2D ()

The default constructor.

8.142.2.2 mitkLineWidgetModel2D::mitkLineWidgetModel2D (float *startPoint*[2])

One and only constructor of mitkLineWidgetModel3D.

Parameters:

startPoint[0] x-coordinate of start point of this line

startPoint[1] y-coordinate of start point of this line

Note:

The coordinates of the start point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.142.3 Member Function Documentation

8.142.3.1 virtual void mitkLineWidgetModel2D::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.142.3.2 virtual void mitkLineWidgetModel2D::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.142.3.3 virtual void mitkLineWidgetModel2D::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.142.3.4 void mitkLineWidgetModel2D::GetEndPoint (int & *ix*, int & *iy*)

Get the integral coordinates of the end point in the original image.

Parameters:
- ***ix*** return the x-coordinate
- ***iy*** return the y-coordinate

8.142.3.5 void mitkLineWidgetModel2D::GetEndPoint (float & *tx*, float & *ty*)

Get the physical coordinates in the object space of the end point.

Parameters:
- ***tx*** return the x-coordinate of the end point
- ***ty*** return the y-coordinate of the end point

8.142.3.6 float mitkLineWidgetModel2D::GetLineLength ()

Get length of this line.

Returns:
Return the length of the line.

8.142.3.7 int mitkLineWidgetModel2D::GetPointsSetNum () [inline]

Get how many points have been set.

Returns:
The number of points set properly. The value should be 0, 1 or 2.

8.142.3.8 void mitkLineWidgetModel2D::GetStartPoint (int & *ix*, int & *iy*)

Get the integral coordinates of the start point in the original image.

Parameters:
- ***ix*** return the x-coordinate
- ***iy*** return the y-coordinate

8.142.3.9 void mitkLineWidgetModel2D::GetStartPoint (float & *tx*, float & *ty*)

Get the physical coordinates in the object space of the start point.

Parameters:
- ***tx*** return the x-coordinate of the start point
- ***ty*** return the y-coordinate of the start point

8.142.3.10 const string& mitkLineWidgetModel2D::GetUnitName () [inline]

Get name string of unit.

Returns:
Return a constant reference to a string contains the name of the length unit this line uses

8.142.3.11 virtual void mitkLineWidgetModel2D::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:
names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.142.3.12 virtual void mitkLineWidgetModel2D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel2D.

8.142.3.13 virtual void mitkLineWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.142.3.14 virtual int mitkLineWidgetModel2D::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.142.3.15 void mitkLineWidgetModel2D::SetEndPoint (int *sx*, int *sy*) [inline]

Set position of the end point.

Parameters:
sx x-coordinate of the end point of this line on screen

sy y-coordinate of the end point of this line on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.142.3.16 void mitkLineWidgetModel2D::SetEndPoint (float *x*, float *y*) [inline]

Set position of the end point in the object space.

Parameters:
x x-coordinate of the end point of this line
y y-coordinate of the end point of this line

Note:
Because this widget is for 2D image, the z-coordinate will always be zero.

8.142.3.17 void mitkLineWidgetModel2D::SetEndPoint (float *point*[2]) [inline]

Set position of the end point in the object space.

Parameters:
point[0] x-coordinate of the end point of this line
point[1] y-coordinate of the end point of this line

Note:
Because this widget is for 2D image, the z-coordinate will always be zero.

8.142.3.18 void mitkLineWidgetModel2D::SetMovePoint (int *sx*, int *sy*) [inline]

Set position of the moving end point.

Parameters:
sx x-coordinate of the moving end point of this line on screen
sy y-coordinate of the moving end point of this line on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.142.3.19 void mitkLineWidgetModel2D::SetMovePoint (float *x*, float *y*) [inline]

Set position of the moving end point in the object space.

Parameters:

x x-coordinate of the moving end point of this line

y y-coordinate of the moving end point of this line

Note:

Because this widget is for 2D image, the z-coordinate will always be zero.

8.142.3.20 void mitkLineWidgetModel2D::SetMovePoint (float *point*[2]) [inline]

Set position of the moving end point in the object space.

Parameters:

point[0] x-coordinate of the moving end point of this line

point[1] y-coordinate of the moving end point of this line

Note:

Because this widget is for 2D image, the z-coordinate will always be zero.

8.142.3.21 void mitkLineWidgetModel2D::SetStartPoint (int *sx*, int *sy*) [inline]

Set position of the start point.

Parameters:

sx x-coordinate of the start point of this line on screen

sy y-coordinate of the start point of this line on screen

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.142.3.22 void mitkLineWidgetModel2D::SetStartPoint (float *x*, float *y*) [inline]

Set position of the start point in the object space.

Parameters:

x x-coordinate of the start point of this line

y y-coordinate of the start point of this line

Note:

Because this widget is for 2D image, the z-coordinate will always be zero.

8.142.3.23 void mitkLineWidgetModel2D::SetStartPoint (float *point*[2]) [inline]

Set position of the start point in the object space.

Parameters:

point[0] x-coordinate of the start point of this line

point[1] y-coordinate of the start point of this line

Note:

Because this widget is for 2D image, the z-coordinate will always be zero.

8.142.3.24 void mitkLineWidgetModel2D::SetUnitName (const string & *name*) [inline]

Set name string of unit.

Parameters:

name a constant reference to a string contains the name of the length unit (e.g. mm) this line uses

The documentation for this class was generated from the following file:

- mitkLineWidgetModel2D.h

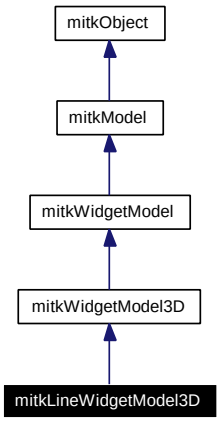
8.143 mitkLineWidgetModel3D Class Reference

mitkLineWidgetModel3D - a line widget used in a 3D scene

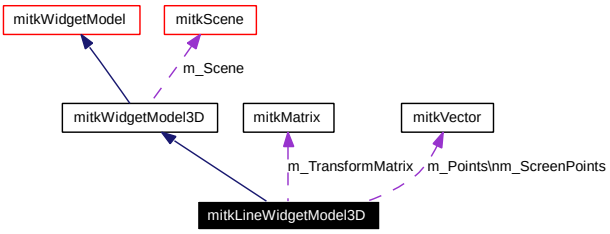
`#include <mitkLineWidgetModel3D.h>`

Inherits mitkWidgetModel3D.

Inheritance diagram for mitkLineWidgetModel3D:



Collaboration diagram for mitkLineWidgetModel3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkLineWidgetModel3D (float point0[3], float point1[3])
- virtual int Render (mitkScene *view)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- void SetUnits (float ux, float uy, float uz)
- void SetUnits (float units[3])
- float GetLineLength ()
- void SetUnitName (const string &name)
- const string & GetUnitName ()
- virtual void Update ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.143.1 Detailed Description

mitkLineWidgetModel3D - a line widget used in a 3D scene

mitkLineWidgetModel3D is a line widget used in a 3D scene which can respond the mouse events to change the line status and return the current length of the line. It is supposed to be attached to a 3D data model (e.g. mitkVolumeModel, mitkSurfaceModel) and add to a 3D view (e.g. mitkScene), and in other conditions the display could be improper.

8.143.2 Constructor & Destructor Documentation

8.143.2.1 mitkLineWidgetModel3D::mitkLineWidgetModel3D (float *point0*[3], float *point1*[3])

One and only constructor of mitkLineWidgetModel3D.

Parameters:

point0[0] x-coordinate of one end point of this line

point0[1] y-coordinate of one end point of this line

point0[2] z-coordinate of one end point of this line

point1[0] x-coordinate of the other end point of this line

point1[1] y-coordinate of the other end point of this line

point1[2] z-coordinate of the other end point of this line

8.143.3 Member Function Documentation

8.143.3.1 virtual void mitkLineWidgetModel3D::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.143.3.2 virtual void mitkLineWidgetModel3D::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.143.3.3 virtual void mitkLineWidgetModel3D::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.143.3.4 float mitkLineWidgetModel3D::GetLineLength ()

Get length of this line.

Returns:

Return the length of the line.

8.143.3.5 const string& mitkLineWidgetModel3D::GetUnitName () [inline]

Get name string of unit.

Returns:

Return a constant reference to a string contains the name

8.143.3.6 virtual void mitkLineWidgetModel3D::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:

names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.143.3.7 virtual void mitkLineWidgetModel3D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkWidgetModel3D.

8.143.3.8 virtual void mitkLineWidgetModel3D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.143.3.9 virtual int mitkLineWidgetModel3D::Render (mitkScene * *view*) [virtual]

Render this model.

Parameters:

view the pointer of an mitkScene in which this model will be shown

Returns:

Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.143.3.10 void mitkLineWidgetModel3D::SetUnitName (const string & *name*) [inline]

Set name string of unit.

Parameters:

name a constant reference to a string contains the name of the length unit (e.g. mm) this line uses

8.143.3.11 void mitkLineWidgetModel3D::SetUnits (float *units*[3]) [inline]

Set unit length on axes.

Parameters:

units[0] unit length in x-axis

units[1] unit length in y-axis

units[2] unit length in z-axis

8.143.3.12 void mitkLineWidgetModel3D::SetUnits (float *ux*, float *uy*, float *uz*) [inline]

Set unit length on axes.

Parameters:

ux unit length in x-axis

uy unit length in y-axis

uz unit length in z-axis

8.143.3.13 virtual void mitkLineWidgetModel3D::Update () [virtual]

Update the parameters of the widget.

Reimplemented from mitkWidgetModel3D.

The documentation for this class was generated from the following file:

- mitkLineWidgetModel3D.h

8.144 mitkList Class Reference

mitkList - a utility class for a list of mitkObject

`#include <mitkList.h>`

Public Member Functions

- void Add (mitkObject *itemAdding)
- void Insert (mitkObject *itemInserting, int pos)
- void Replace (int i, mitkObject *itemNew)
- void Remove (int i)
- void Remove (mitkObject *itemRemove)
- void RemoveAll ()
- int Find (mitkObject *itemFinding)
- mitkObject * GetItem (int i)
- int Count ()
- void InitTraversal ()

8.144.1 Detailed Description

mitkList - a utility class for a list of mitkObject

mitkList is a utility class for a list of mitkObject

8.144.2 Member Function Documentation

8.144.2.1 void mitkList::Add (mitkObject * *itemAdding*)

Add an object to the list.

8.144.2.2 int mitkList::Count () `[inline]`

Return the number of objects in the list.

8.144.2.3 int mitkList::Find (mitkObject * *itemFinding*)

Search for an object and return location in list. If location == -1, object was not found.

8.144.2.4 mitkObject* mitkList::GetItem (int *i*)

Get an object in location i

8.144.2.5 void mitkList::InitTraversal () `[inline]`

Traversal the list

8.144.2.6 void mitkList::Insert (mitkObject * *itemInserting*, int *pos*)

Insert an object to the list.

8.144.2.7 void mitkList::Remove (mitkObject * *itemRemove*)

Remove an object from the list.

8.144.2.8 void mitkList::Remove (int *i*)

Remove the i'th item in the list.

8.144.2.9 void mitkList::RemoveAll ()

Remove all objects from the list.

8.144.2.10 void mitkList::Replace (int *i*, mitkObject * *itemNew*)

Replace the i'th item

The documentation for this class was generated from the following file:

- mitkList.h

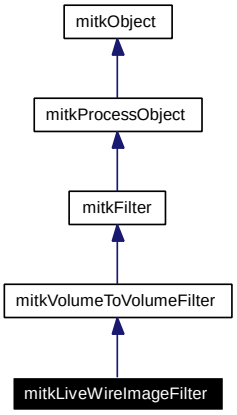
8.145 mitkLiveWireImageFilter Class Reference

mitkLiveWireImageFilter - A class that implement livewire segmentation arithmetic

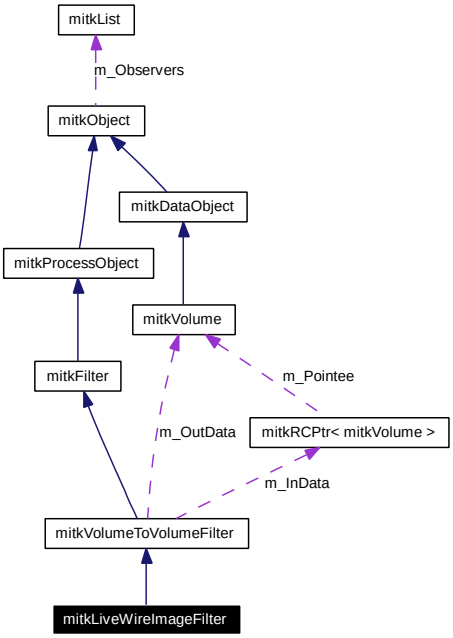
`#include <mitkLiveWireImageFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkLiveWireImageFilter:



Collaboration diagram for mitkLiveWireImageFilter:



Public Member Functions

- mitkLiveWireImageFilter ()
- virtual void PrintSelf (ostream &os)
- virtual ~mitkLiveWireImageFilter ()

8.145.1 Detailed Description

mitkLiveWireImageFilter - A class that implement livewire segmentation arithmetic

mitkLiveWireImageFilter - A class that implement livewire segmentation arithmetic

8.145.2 Constructor & Destructor Documentation

8.145.2.1 mitkLiveWireImageFilter::mitkLiveWireImageFilter ()

Constructor of the class.

8.145.2.2 virtual mitkLiveWireImageFilter::~mitkLiveWireImageFilter () [virtual]

Constructor of the class.

8.145.3 Member Function Documentation

8.145.3.1 virtual void mitkLiveWireImageFilter::PrintSelf (ostream & *os*) [inline, virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkLiveWireImageFilter.h

8.146 mitkLut1D Class Reference

mitkLut1D - A data class for 1D-LookUPTable,

`#include <mitkLut1D.h>`

8.146.1 Detailed Description

mitkLut1D - A data class for 1D-LookUPTable,

mitkLut1D is a data class for 1D-LookUPTable

The documentation for this class was generated from the following file:

- mitkLut1D.h

8.147 mitkManipulator Class Reference

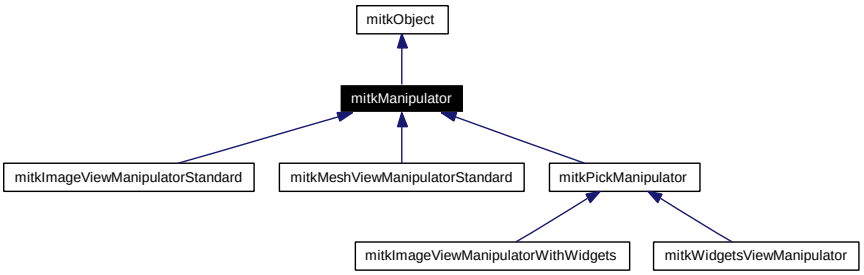
mitkManipulator - an abstract class to define an interface to implement mouse events processing in a scene

`#include <mitkManipulator.h>`

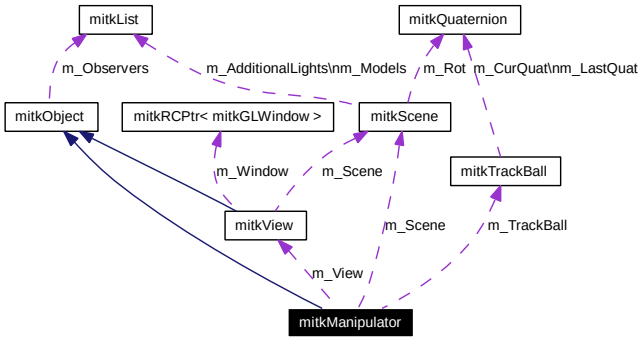
Inherits mitkObject.

Inherited by mitkImageViewManipulatorStandard, mitkMeshViewManipulatorStandard, and mitkPickManipulator.

Inheritance diagram for mitkManipulator:



Collaboration diagram for mitkManipulator:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void OnMouseWheel (bool ctrlDown, bool shiftDown, int xPos, int yPos, int delta)
- void SetScene (mitkScene *scene)
- void SetView (mitkView *view)
- void EnableLoD (bool enable=true)
- bool IsLoDEnabled ()

8.147.1 Detailed Description

mitkManipulator - an abstract class to define an interface to implement mouse events processing in a scene

mitkManipulator is an abstract class to define an interface to implement mouse events processing in a scene. Its purpose is to provide user a chance to override the default mouse events processing. All of its subclasses must implement OnMouseDown(), OnMouseUp(), OnMouseMove() functions to process mouse events in a scene properly.

8.147.2 Member Function Documentation

8.147.2.1 void mitkManipulator::EnableLoD (bool *enable* = `true`) `[inline]`

Enable or disable LoD mode.

Parameters:

enable ture = enable LoD mode; false = disable LoD mode

Note:

It is only a hint to rendering method and depends on actual implementation. Maybe it will not take any effect.

8.147.2.2 bool mitkManipulator::IsLoDEnabled () `[inline]`

Test if the LoD mode is enabled.

Returns:

Return true if the LoD mode is enabled, otherwise return false.

8.147.2.3 void mitkManipulator::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

8.147.2.4 void mitkManipulator::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

8.147.2.5 void mitkManipulator::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

8.147.2.6 void mitkManipulator::OnMouseWheel (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *delta*)

Deal with mouse wheel rotate event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

delta the wheel rotation

8.147.2.7 virtual void mitkManipulator::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkImageViewManipulatorStandard, mitkImageViewManipulatorWithWidgets, mitkMeshViewManipulatorStandard, mitkPickManipulator, and mitkWidgetsViewManipulator.

8.147.2.8 void mitkManipulator::SetScene (mitkScene * *scene*)

Set the associated scene. The mouse events come from this scene.

Parameters:

scene Specify the associated scene.

8.147.2.9 void mitkManipulator::SetView (mitkView * *view*)

Set the associated view. The mouse events come from this view

Parameters:

view Specify the associated view.

The documentation for this class was generated from the following file:

- mitkManipulator.h

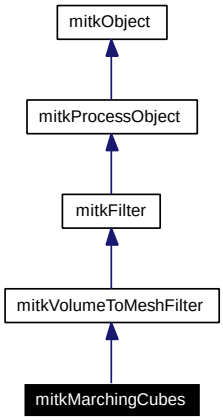
8.148 mitkMarchingCubes Class Reference

mitkMarchingCubes - implementing 3d reconstruction algorithm

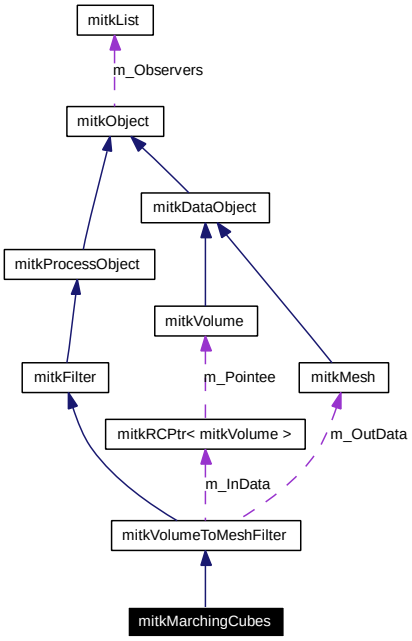
`#include <mitkMarchingCubes.h>`

Inherits mitkVolumeToMeshFilter.

Inheritance diagram for mitkMarchingCubes:



Collaboration diagram for mitkMarchingCubes:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkMarchingCubes ()

- void SetThreshold (float lowThreshold, float highThreshold)
- float GetLowThreshold ()
- float GetHighThreshold ()

8.148.1 Detailed Description

mitkMarchingCubes - implementing 3d reconstruction algorithm

mitkMarchingCubes is a process object that defines a set of function to process Volume Data to generate 3D Image using Marching Cubes algorithm.

Warning:
Marching Cubes algorithm may have the protect of United States patent, please use it at your own risk.

8.148.2 Constructor & Destructor Documentation

8.148.2.1 mitkMarchingCubes::mitkMarchingCubes ()

Default constructor.

8.148.3 Member Function Documentation

8.148.3.1 float mitkMarchingCubes::GetHighThreshold () [inline]

Get low and high threshold.

Returns:
Return the high threshold.

8.148.3.2 float mitkMarchingCubes::GetLowThreshold () [inline]

Get low threshold.

Returns:
Retrurn the low threshold.

8.148.3.3 virtual void mitkMarchingCubes::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToMeshFilter.

8.148.3.4 void mitkMarchingCubes::SetThreshold (float *lowThreshold*, float *highThreshold*)

Set low and high threshold.

Parameters:

lowThreshold the low threshold

highThreshold the high threshold

The documentation for this class was generated from the following file:

- mitkMarchingCubes.h

8.149 mitkMatrix Class Reference

mitkMatrix - an encapsulation of a matrix

`#include <mitkMatrix.h>`

Public Member Functions

- void Transpose ()
- float Inverse ()
- float Determinant ()
- void Adjoint ()
- float MinValue ()
- float MaxValue ()
- void ZeroMatrix ()
- void IdentityMatrix ()
- void TranslateMatrix (const float dx, const float dy, const float dz)
- void ScaleMatrix (const float a, const float b, const float c)
- void ScaleMatrix (const float a)
- void RotateXMatrix (const float angle)
- void RotateYMatrix (const float angle)
- void RotateZMatrix (const float angle)
- void Translate (const float dx, const float dy, const float dz)
- void Scale (const float a, const float b, const float c)
- void Rotate (const float angle, const float x, const float y, const float z)
- void RotateX (const float angle)
- void RotateY (const float angle)
- void RotateZ (const float angle)

8.149.1 Detailed Description

mitkMatrix - an encapsulation of a matrix

mitkMatrix provides an encapsulation of matrix operations. It is used in volume rendering algorithm, which requires intensive matrix calculations. The interface of mitkMatrix provides many common matrix operations.

8.149.2 Member Function Documentation

8.149.2.1 void mitkMatrix::Adjoint ()

Adjoint matrix

8.149.2.2 float mitkMatrix::Determinant ()

Returns the determinant

8.149.2.3 void mitkMatrix::IdentityMatrix () [inline]

Set the matrix to identity matrix

8.149.2.4 float mitkMatrix::Inverse ()

Inverses the matrix and returns the determinant

8.149.2.5 float mitkMatrix::MaxValue ()

Returns the maximum absolute value of the matrix

8.149.2.6 float mitkMatrix::MinValue ()

Returns the minimum absolute value of the matrix

8.149.2.7 void mitkMatrix::Rotate (const float *angle*, const float *x*, const float *y*, const float *z*)

Rotate current matrix around arbitrary axis

8.149.2.8 void mitkMatrix::RotateX (const float *angle*)

Rotate current matrix around x axis

8.149.2.9 void mitkMatrix::RotateXMatrix (const float *angle*)

Rotation around x axis

8.149.2.10 void mitkMatrix::RotateY (const float *angle*)

Rotate current matrix around y axis

8.149.2.11 void mitkMatrix::RotateYMatrix (const float *angle*)

Rotation around y axis

8.149.2.12 void mitkMatrix::RotateZ (const float *angle*)

Rotate current matrix around z axis

8.149.2.13 void mitkMatrix::RotateZMatrix (const float *angle*)

Rotation around z axis

8.149.2.14 void mitkMatrix::Scale (const float *a*, const float *b*, const float *c*)

Scale current matrix

8.149.2.15 void mitkMatrix::ScaleMatrix (const float *a*)

Uniform scale transformation

8.149.2.16 void mitkMatrix::ScaleMatrix (const float *a*, const float *b*, const float *c*)

Scale transformation

8.149.2.17 void mitkMatrix::Translate (const float *dx*, const float *dy*, const float *dz*)

Translate current matrix

8.149.2.18 void mitkMatrix::TranslateMatrix (const float *dx*, const float *dy*, const float *dz*)

Translate transformation

8.149.2.19 void mitkMatrix::Transpose ()

Transposes the matrix

8.149.2.20 void mitkMatrix::ZeroMatrix () [inline]

Clean the matrix to zero

The documentation for this class was generated from the following file:

- mitkMatrix.h

8.150 mitkMatrixD Class Reference

mitkMatrixD - an encapsulation of a matrix

`#include <mitkMatrixD.h>`

Public Member Functions

- void Transpose ()
- double Inverse ()
- double Determinant ()
- void Adjoint ()
- double MinValue ()
- double MaxValue ()
- void ZeroMatrix ()
- void IdentityMatrix ()
- void TranslateMatrix (const double dx, const double dy, const double dz)
- void ScaleMatrix (const double a, const double b, const double c)
- void ScaleMatrix (const double a)
- void RotateXMatrix (const double angle)
- void RotateYMatrix (const double angle)
- void RotateZMatrix (const double angle)
- void Translate (const double dx, const double dy, const double dz)
- void Scale (const double a, const double b, const double c)
- void Rotate (const double angle, const double x, const double y, const double z)
- void RotateX (const double angle)
- void RotateY (const double angle)
- void RotateZ (const double angle)

8.150.1 Detailed Description

mitkMatrixD - an encapsulation of a matrix

mitkMatrixD provides an encapsulation of matrix operations. It is used in volume rendering algorithm, which requires intensive matrix calculations. The interface of mitkMatrixD provides many common matrix operations.

8.150.2 Member Function Documentation

8.150.2.1 void mitkMatrixD::Adjoint ()

Adjoint matrix

8.150.2.2 double mitkMatrixD::Determinant ()

Returns the determinant

8.150.2.3 void mitkMatrixD::IdentityMatrix () `[inline]`

Set the matrix to identity matrix

8.150.2.4 double mitkMatrixD::Inverse ()

Inverses the matrix and returns the determinant

8.150.2.5 double mitkMatrixD::MaxValue ()

Returns the maximum absolute value of the matrix

8.150.2.6 double mitkMatrixD::MinValue ()

Returns the minimum absolute value of the matrix

8.150.2.7 void mitkMatrixD::Rotate (const double *angle*, const double *x*, const double *y*, const double *z*)

Rotate current matrix around arbitrary axis

8.150.2.8 void mitkMatrixD::RotateX (const double *angle*)

Rotate current matrix around x axis

8.150.2.9 void mitkMatrixD::RotateXMatrix (const double *angle*)

Rotation around x axis

8.150.2.10 void mitkMatrixD::RotateY (const double *angle*)

Rotate current matrix around y axis

8.150.2.11 void mitkMatrixD::RotateYMatrix (const double *angle*)

Rotation around y axis

8.150.2.12 void mitkMatrixD::RotateZ (const double *angle*)

Rotate current matrix around z axis

8.150.2.13 void mitkMatrixD::RotateZMatrix (const double *angle*)

Rotation around z axis

8.150.2.14 void mitkMatrixD::Scale (const double *a*, const double *b*, const double *c*)

Scale current matrix

8.150.2.15 void mitkMatrixD::ScaleMatrix (const double *a*)

Uniform scale transformation

8.150.2.16 void mitkMatrixD::ScaleMatrix (const double *a*, const double *b*, const double *c*)

Scale transformation

8.150.2.17 void mitkMatrixD::Translate (const double *dx*, const double *dy*, const double *dz*)

Translate current matrix

8.150.2.18 void mitkMatrixD::TranslateMatrix (const double *dx*, const double *dy*, const double *dz*)

Translate transformation

8.150.2.19 void mitkMatrixD::Transpose ()

Transposes the matrix

8.150.2.20 void mitkMatrixD::ZeroMatrix () [inline]

Clean the matrix to zero

The documentation for this class was generated from the following file:

- mitkMatrixD.h

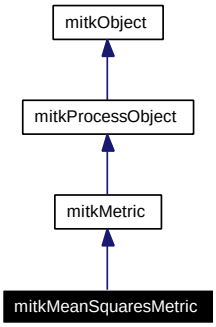
8.151 mitkMeanSquaresMetric Class Reference

mitkMeanSquaresMetric - a concrete class computing similarity between two objects to be registered

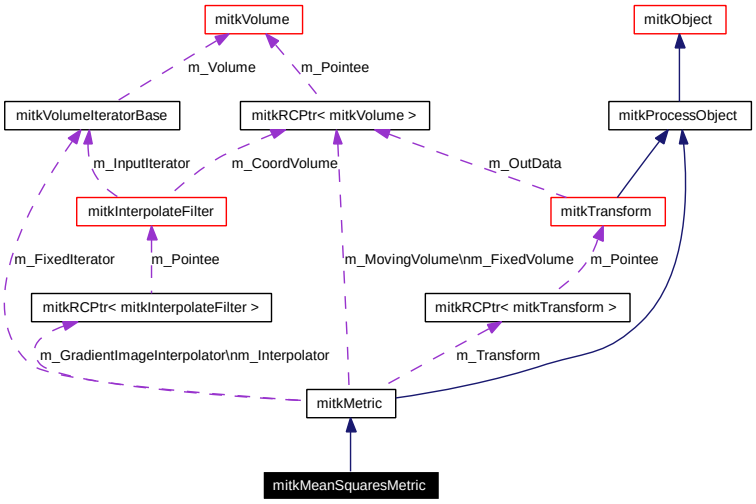
`#include <mitkMeanSquaresMetric.h>`

Inherits mitkMetric.

Inheritance diagram for mitkMeanSquaresMetric:



Collaboration diagram for mitkMeanSquaresMetric:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkMeanSquaresMetric ()
- virtual bool GetSimilarity (const VectorParameterType &parameters, ScalarParameterType &similarity)
- virtual bool GetSimilarityAndDerivative (const VectorParameterType &parameters, ScalarParameterType &similarity, VectorParameterType &derivative)

8.151.1 Detailed Description

mitkMeanSquaresMetric - a concrete class computing similarity between two objects to be registered

mitkMeanSquaresMetric computes the sum of squared differenced between pixels in the moving image and pixels in the fixed image. The spatial correspondance between both images is established through a Transform. Pixel values are taken from the Moving image. Their positions are mapped to the Fixed image and result in general in non-grid position on it. Values at these non-grid position of the Fixed image are interpolated using a user-selected Interpolator.

8.151.2 Constructor & Destructor Documentation

8.151.2.1 mitkMeanSquaresMetric::mitkMeanSquaresMetric ()

Constructor.

8.151.3 Member Function Documentation

8.151.3.1 virtual bool mitkMeanSquaresMetric::GetSimilarity (const VectorParameterType & *parameters*, ScalarParameterType & *similarity*) [virtual]

Get Similarity between regions of two volumes.

Parameters:

parameters The vector pointer to the transform parameters.

similarity The metric results.

Returns:

Return true if the metric computation is performed without error.

Reimplemented from mitkMetric.

8.151.3.2 virtual bool mitkMeanSquaresMetric::GetSimilarityAndDerivative (const VectorParameterType & *parameters*, ScalarParameterType & *similarity*, VectorParameterType & *derivative*) [virtual]

Get Similarity And Gradient Value between regions of two volumes.

Parameters:

transformParameters The vector pointer to the transform parameters.

similarity Return the similarity.

derivative Return the gradient value in all dimensions.

Returns:

Return true if the metric computation is performed without error.

Reimplemented from mitkMetric.

8.151.3.3 virtual void mitkMeanSquaresMetric::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMetric.

The documentation for this class was generated from the following file:

- mitkMeanSquaresMetric.h

8.152 mitkMesh Class Reference

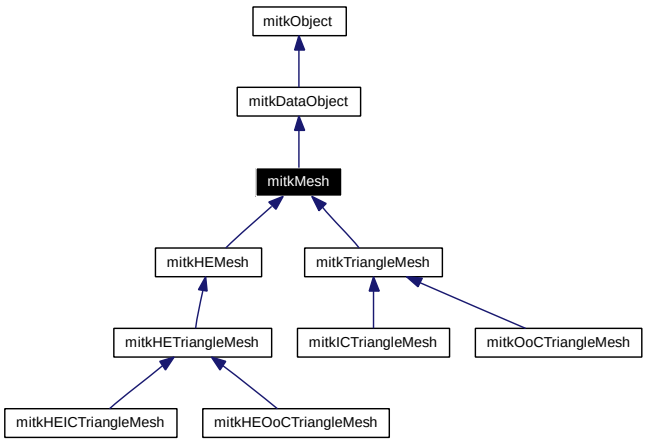
mitkMesh - an abstract class for mesh types

`#include <mitkMesh.h>`

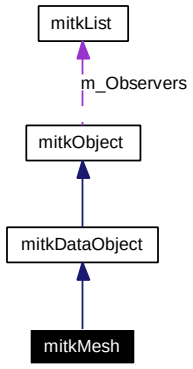
Inherits mitkDataObject.

Inherited by mitkHEMesh, and mitkTriangleMesh.

Inheritance diagram for mitkMesh:



Collaboration diagram for mitkMesh:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- bool CopyMesh (mitkMesh *src)
- virtual int GetDataObjectType () const
- virtual void SetVertexNumber (size_type number)=0
- virtual size_type GetVertexNumber () const =0

- virtual void SetFaceNumber (size_type number)=0
- virtual size_type GetFaceNumber () const =0
- virtual float * GetVertexData ()=0
- virtual index_type * GetFaceData ()=0
- float const * GetBoundingBox ()
- void GetBoundingBox (float &minX, float &maxX, float &minY, float &maxY, float &minZ, float &maxZ)
- void GetBoundingBox (float bounds[6])
- void SetBoundingBox (float minX, float maxX, float minY, float maxY, float minZ, float maxZ)
- void SetBoundingBox (float bounds[6])
- bool IsNormalReversed () const
- bool IsClockwise () const
- void SetClockwise (bool isClockwise=true)
- virtual bool TestClockwise ()=0
- virtual void ReverseNormals ()
- index_type AddVertex (Vertex3f &vert)
- template<typename IndexType, unsigned int indexNum> index_type AddFace (_face_type< IndexType, indexNum > &face)
- index_type AddFace (TriangleFace &face)
- bool GetVertex (index_type vertIdx, Vertex3f &vert)
- size_type GetVertices (index_type startIdx, size_type num, Vertex3f *verts)
- template<typename IndexType, unsigned int indexNum> bool GetFace (index_type faceIdx, _face_type< IndexType, indexNum > &face)
- bool GetFace (index_type faceIdx, TriangleFace &face)
- size_type GetTriangleFaces (index_type startIdx, size_type num, TriangleFace *faces)
- size_type GetTriangleFaces (index_type startIdx, size_type num, Vertex3f *verts)
- bool SetVertex (index_type vertIdx, Vertex3f const &vert)
- template<typename IndexType, unsigned int indexNum> bool SetFace (index_type faceIdx, _face_type< IndexType, indexNum > &face)
- bool SetFace (index_type faceIdx, TriangleFace &face)

8.152.1 Detailed Description

mitkMesh - an abstract class for mesh types

mitkMesh is an abstract class for mesh types which are used to represent 3D objects with their surfaces.

See also:
mitkTriangleMesh
mitkHEMesh
mitkHETriangleMesh

8.152.2 Member Function Documentation

8.152.2.1 index_type mitkMesh::AddFace (TriangleFace & *face*) [inline]

An explicit instantiation of AddFace().

Reimplemented in mitkTriangleMesh.

8.152.2.2 template<typename IndexType, unsigned int indexNum> index_type mitkMesh::AddFace (_face_type< IndexType, indexNum > & *face*) [inline]

A general interface for adding a face.

Parameters:
face the face to add

Returns:
Return the index of the face added.

Note:
The template struct _face_type is defined as follows in mitkGeometryTypes.h:

```
template <typename IndexType, unsigned int indexNum>
struct _face_type
{
    enum { vertNum = indexNum };
    IndexType verts[indexNum];
};
```

Warning:
Each index in the "verts" array of the face must be valid, i.e. represents a existent vertex in the mesh (has been added before).

See also:
mitkGeometryTypes.h

Reimplemented in mitkTriangleMesh.

8.152.2.3 index_type mitkMesh::AddVertex (Vertex3f & *vert*) [inline]

A general interface for adding a vertex.

Parameters:
vert the vertex to add

Returns:
Return the index of the vertex added.

Note:
The struct Vertex3f is equal to follows:

```
struct Point3f
{
    union
    {
        struct { float x, y, z; };
        float coord[3];
    };
};

struct Vertex3f
{
    Point3f point;
    Point3f normal;
};
```

See also:
mitkGeometryTypes.h

8.152.2.4 bool mitkMesh::CopyMesh (mitkMesh * *src*)

Copy the contents from another mesh. It is used to transfer between different concrete mesh type, i.e. in-core mesh <=> out-of-core mesh.

Parameters:
src the mesh copy from

Returns:
Return true if success, otherwise return false.

8.152.2.5 virtual void mitkMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Implements mitkDataObject.

Reimplemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkOoCTriangleMesh, and mitkTriangleMesh.

8.152.2.6 void mitkMesh::GetBoundingBox (float *bounds*[6]) [inline]

Get the bounds for this mesh data as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Parameters:
bounds[0] return minimum coordinate value in x-axis
bounds[1] return maximum coordinate value in x-axis
bounds[2] return minimum coordinate value in y-axis
bounds[3] return maximum coordinate value in y-axis
bounds[4] return minimum coordinate value in z-axis
bounds[5] return maximum coordinate value in z-axis

8.152.2.7 void mitkMesh::GetBoundingBox (float & *minX*, float & *maxX*, float & *minY*, float & *maxY*, float & *minZ*, float & *maxZ*) [inline]

Get the bounds for this mesh data as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Parameters:
minX return minimum coordinate value in x-axis
maxX return maximum coordinate value in x-axis
minY return minimum coordinate value in y-axis
maxY return maximum coordinate value in y-axis
minZ return minimum coordinate value in z-axis
maxZ return maximum coordinate value in z-axis

8.152.2.8 float const* mitkMesh::GetBoundingBox () [inline]

Get the bounds for this mesh data as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Returns:
Return a float array which contains Xmin, Xmax, Ymin, Ymax, Zmin and Zmax in turn.

8.152.2.9 virtual int mitkMesh::GetDataObjectType () const [inline, virtual]

Return what type of data object this is.

Returns:
Return the type of this data object.

Reimplemented from mitkDataObject.

Reimplemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkOoCTriangleMesh, and mitkTriangleMesh.

8.152.2.10 bool mitkMesh::GetFace (index_type *faceIdx*, TriangleFace & *face*) [inline]

An explicit instantiation of GetFace().

8.152.2.11 template<typename IndexType, unsigned int indexNum> bool mitkMesh::GetFace (index_type *faceIdx*, _face_type< IndexType, indexNum > & *face*) [inline]

A general interface for getting a face.

Parameters:
faceIdx the index of the face to get

face contain the returned face

Returns:
Return true if success, otherwise return false.

8.152.2.12 virtual index_type* mitkMesh::GetFaceData () [pure virtual]

Get data pointer of this face data.

Returns:
Return a unsigned int pointer to the face data (indices to vertices).

Warning:
This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes.

Implemented in mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.152.2.13 virtual size_type mitkMesh::GetFaceNumber () const [pure virtual]

Get the mesh's face number.

Returns:
Return the number of faces.

Implemented in mitkHEMesh, and mitkTriangleMesh.

8.152.2.14 size_type mitkMesh::GetTriangleFaces (index_type *startIdx*, size_type *num*, Vertex3f * *verts*) [inline]

A general interface for getting all the vertices of a set of triangle faces.

Parameters:
startIdx the index of the first face to get

num the number of faces to get

verts contain the vertices of the triangle faces to get (3 vertices for each face)

Returns:
Return the actual number of gotten faces.

8.152.2.15 size_type mitkMesh::GetTriangleFaces (index_type *startIdx*, size_type *num*, TriangleFace * *faces*) [inline]

A general interface for getting a set of triangle faces.

Parameters:
startIdx the index of the first face to get

num the number of faces to get

faces contain the returned faces (from startIdx to startIdx+num-1)

Returns:
Return the actual number of gotten faces.

8.152.2.16 bool mitkMesh::GetVertex (index_type *vertIdx*, Vertex3f & *vert*) [inline]

A general interface for getting a vertex.

Parameters:
vertIdx the index of the vertex to get

vert contain the returned vertex

Returns:
Return true if success, otherwise return false.

8.152.2.17 virtual float* mitkMesh::GetVertexData () [pure virtual]

Get data pointer of this vertex data.

Returns:
Return a float pointer to the vertex data.

Warning:
This function is obsolete. It is provided to keep old codes working. We strongly advise against using it in new codes.

Implemented in mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.152.2.18 virtual size_type mitkMesh::GetVertexNumber () const [pure virtual]

Get the mesh's vertex number.

Returns:
Return the number of vertices.

Implemented in mitkHEMesh, and mitkTriangleMesh.

8.152.2.19 size_type mitkMesh::GetVertices (index_type *startIdx*, size_type *num*, Vertex3f * *verts*) [inline]

A general interface for getting a set of vertices.

Parameters:
startIdx the index of the first vertex to get

num the number of vertices to get

verts contain the returned vertices (from startIdx to startIdx+num-1)

Returns:
Return the actual number of gotten vertices.

8.152.2.20 virtual void mitkMesh::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Implements mitkDataObject.

Reimplemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkHETriangleMesh, mitkICTriangleMesh, mitkOoCTriangleMesh, and mitkTriangleMesh.

8.152.2.21 bool mitkMesh::IsClockwise () const [inline]

Get the orientation of front-facing polygons.

Returns:
Return true if selects clockwise polygons as front-facing.

8.152.2.22 bool mitkMesh::IsNormalReversed () const [inline]

Indicate if the normals are reversed.

Returns:
Return true if the normals are reversed, otherwise return false.

8.152.2.23 virtual void mitkMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkDataObject.

Reimplemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkHETriangleMesh, mitkICTriangleMesh, mitkOoCTriangleMesh, and mitkTriangleMesh.

8.152.2.24 virtual void mitkMesh::ReverseNormals () [inline, virtual]

Reverse normals.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.152.2.25 void mitkMesh::SetBoundingBox (float *bounds*[6]) [inline]

Set the bounds for this mesh data as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Parameters:
bounds[0] minimum coordinate value in x-axis

bounds[1] maximum coordinate value in x-axis

bounds[2] minimum coordinate value in y-axis

bounds[3] maximum coordinate value in y-axis

bounds[4] minimum coordinate value in z-axis

bounds[5] maximum coordinate value in z-axis

8.152.2.26 void mitkMesh::SetBoundingBox (float *minX*, float *maxX*, float *minY*, float *maxY*, float *minZ*, float *maxZ*) [inline]

Set the bounds for this mesh data as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Parameters:

minX minimum coordinate value in x-axis

maxX maximum coordinate value in x-axis

minY minimum coordinate value in y-axis

maxY maximum coordinate value in y-axis

minZ minimum coordinate value in z-axis

maxZ maximum coordinate value in z-axis

8.152.2.27 void mitkMesh::SetClockwise (bool *isClockwise* = true) [inline]

Set the orientation of front-facing polygons to isClockwise.

Parameters:

isClockwise if selects clockwise polygons as front-facing

8.152.2.28 bool mitkMesh::SetFace (index_type *faceIdx*, TriangleFace & *face*) [inline]

An explicit instantiation of SetFace().

8.152.2.29 template<typename IndexType, unsigned int indexNum> bool mitkMesh::SetFace (index_type *faceIdx*, _face_type< IndexType, indexNum > & *face*) [inline]

A general interface for setting a face.

Parameters:

faceIdx the index of the face to set

face the face to set

Returns:

Return true if success, otherwise return false.

8.152.2.30 virtual void mitkMesh::SetFaceNumber (size_type *number*) [pure virtual]

Set the mesh's faces' number and allocate memory.

Parameters:

number the number of faces

Implemented in mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.152.2.31 bool mitkMesh::SetVertex (index_type *vertIdx*, Vertex3f const & *vert*) [inline]

A general interface for setting a vertex.

Parameters:
vertIdx the index of the vertex to set

vert the vertex to set

Returns:
Return true if success, otherwise return false.

8.152.2.32 virtual void mitkMesh::SetVertexNumber (size_type *number*) [pure virtual]

Set the mesh's vertices' number and allocate memory.

Parameters:
number the number of vertices

Implemented in mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.152.2.33 virtual void mitkMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Implements mitkDataObject.

Reimplemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkOoCTriangleMesh, and mitkTriangleMesh.

8.152.2.34 virtual bool mitkMesh::TestClockwise () [pure virtual]

Test the orientation of front-facing polygons.

Returns:
Return true if the orientation of front-facing polygons is clockwise, otherwise return false.

Implemented in mitkHEMesh, mitkICTriangleMesh, and mitkOoCTriangleMesh.

The documentation for this class was generated from the following file:

- mitkMesh.h

8.153 mitkMeshReader Class Reference

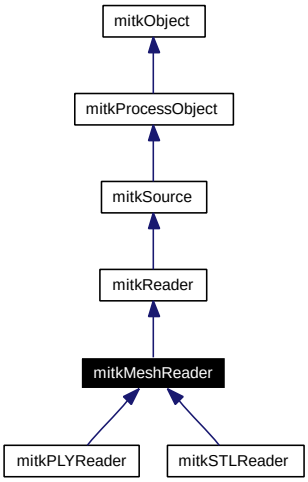
mitkMeshReader - an abstract class represents a mesh reader to read mesh files

`#include <mitkMeshReader.h>`

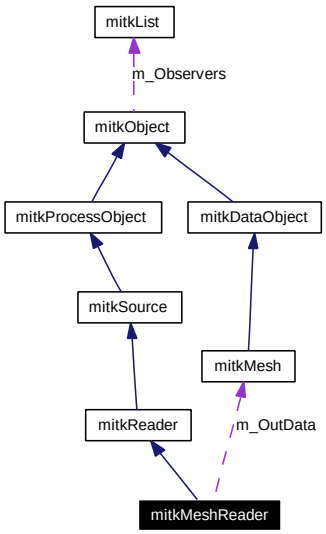
Inherits mitkReader.

Inherited by mitkPLYReader, and mitkSTLReader.

Inheritance diagram for mitkMeshReader:



Collaboration diagram for mitkMeshReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetOoCSupport (char const *diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufBlockNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)

- mitkMesh * GetOutput ()

8.153.1 Detailed Description

mitkMeshReader - an abstract class represents a mesh reader to read mesh files

mitkMeshReader is an abstract class represents a mesh reader to read mesh files. To use a concrete mesh reader, the code snippet is:

```
mitkSomeMeshReader *aReader = new mitkSomeMeshReader;
aReader->AddFileName(filename); //Only require one file name
if (aReader->Run())
{
    mitkMesh *aMesh = aReader->GetOutput();
    Using aMesh
}
```

8.153.2 Member Function Documentation

8.153.2.1 mitkMesh* mitkMeshReader::GetOutput ()

Get the output mesh the reader has read.

Returns:
the output volume.

8.153.2.2 virtual void mitkMeshReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkReader.

Reimplemented in mitkPLYReader, and mitkSTLReader.

8.153.2.3 void mitkMeshReader::SetOoCSupport (char const * *diskPath* = mitkOoCGlobalSettings::DiskCachePath, unsigned int *bufBlockNum* = mitkOoCGlobalSettings::BufferedSliceNumber, bool *supportOoC* = true)

Let the reader support out-of-core mesh data.

Parameters:
diskPath the path in the disk to cache the mesh data

bufSliceNum the number of slices to cache in the main memory

supportOoC whether to turn on out-of-core support

Note:
The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

The documentation for this class was generated from the following file:

- mitkMeshReader.h

8.154 mitkMeshToMeshFilter Class Reference

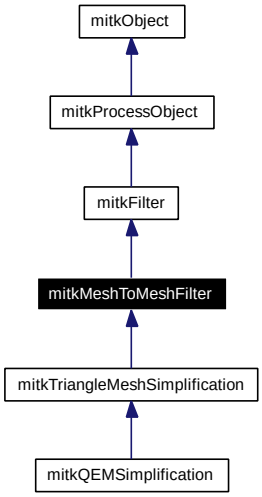
mitkMeshToMeshFilter - abstract class specifies interface for mesh to mesh filter

`#include <mitkMeshToMeshFilter.h>`

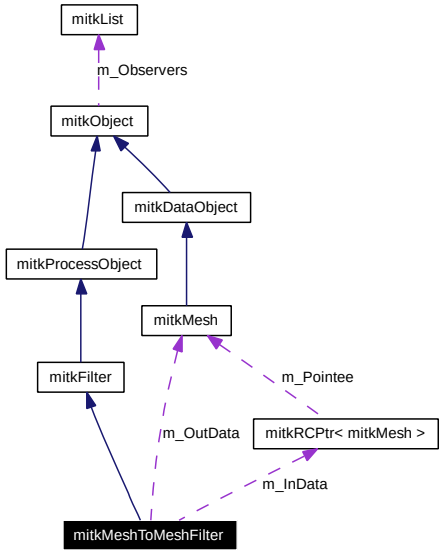
Inherits mitkFilter.

Inherited by mitkMeshTransform, and mitkTriangleMeshSimplification.

Inheritance diagram for mitkMeshToMeshFilter:



Collaboration diagram for mitkMeshToMeshFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- void SetInput (mitkMesh ∗mesh)
- mitkMesh ∗ GetInput ()
- void SetOoCSupport (char const ∗diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufBlockNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)
- mitkMesh ∗ GetOutput ()
- void SetOutputMesh (mitkMesh ∗mesh)

8.154.1 Detailed Description

mitkMeshToMeshFilter - abstract class specifies interface for mesh to mesh filter

mitkMeshToMeshFilter is an abstract class specifies interface for mesh to mesh filter. This type of filter has a mesh input and generates a mesh as output.

8.154.2 Member Function Documentation

8.154.2.1 mitkMesh∗ mitkMeshToMeshFilter::GetInput () [inline]

Get the input mesh

Returns:
Return the input mesh.

8.154.2.2 mitkMesh∗ mitkMeshToMeshFilter::GetOutput () [inline]

Get the output mesh

Returns:
Return the output mesh.

8.154.2.3 virtual void mitkMeshToMeshFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFilter.

Reimplemented in mitkQEMSimplification, and mitkTriangleMeshSimplification.

8.154.2.4 void mitkMeshToMeshFilter::SetInput (mitkMesh ∗ *mesh*) [inline]

Set the input mesh.

Parameters:
mesh specify the input mesh

8.154.2.5 void mitkMeshToMeshFilter::SetOoCSupport (char const * *diskPath* = `mitkOoCGlobalSettings::DiskCachePath`, unsigned int *bufBlockNum* = `mitkOoCGlobalSettings::BufferedSliceNumber`, bool *supportOoC* = `true`)

Let the reader support out-of-core mesh data.

Parameters:

diskPath the path in the disk to cache the mesh data

bufSliceNum the number of slices to cache in the main memory

supportOoC whether to turn on out-of-core support

Note:

The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

8.154.2.6 void mitkMeshToMeshFilter::SetOutputMesh (mitkMesh * *mesh*)

Set user defined output mesh to replace the default generated mesh.

Parameters:

mesh the new output mesh object to be set

The documentation for this class was generated from the following file:

- mitkMeshToMeshFilter.h

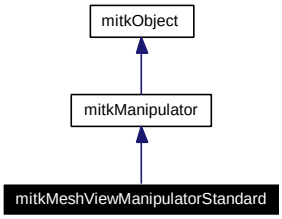
8.155 mitkMeshViewManipulatorStandard Class Reference

mitkMeshViewManipulatorStandard - a concrete manipulator to process mouse events in an 3D scene

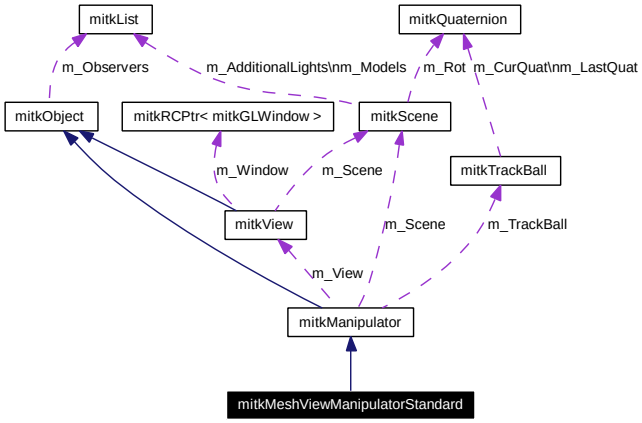
`#include <mitkMeshViewManipulatorStandard.h>`

Inherits mitkManipulator.

Inheritance diagram for mitkMeshViewManipulatorStandard:



Collaboration diagram for mitkMeshViewManipulatorStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)

8.155.1 Detailed Description

mitkMeshViewManipulatorStandard - a concrete manipulator to process mouse events in an 3D scene

mitkMeshViewManipulatorStandard is a concrete manipulator to process mouse events in an 3D scene. It is the default manipulator of a mitkScene. The behavior of this manipulator is shown as follows:

Mouse Left Key — Rotation

Mouse Middle Key — Translation

Mouse Right Key — Zoom in / out

8.155.2 Member Function Documentation

8.155.2.1 virtual void mitkMeshViewManipulatorStandard::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkManipulator.

8.155.2.2 virtual void mitkMeshViewManipulatorStandard::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

Implements mitkManipulator.

8.155.2.3 virtual void mitkMeshViewManipulatorStandard::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkManipulator.

8.155.2.4 virtual void mitkMeshViewManipulatorStandard::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkManipulator.

The documentation for this class was generated from the following file:

- mitkMeshViewManipulatorStandard.h

8.156 mitkMeshWriter Class Reference

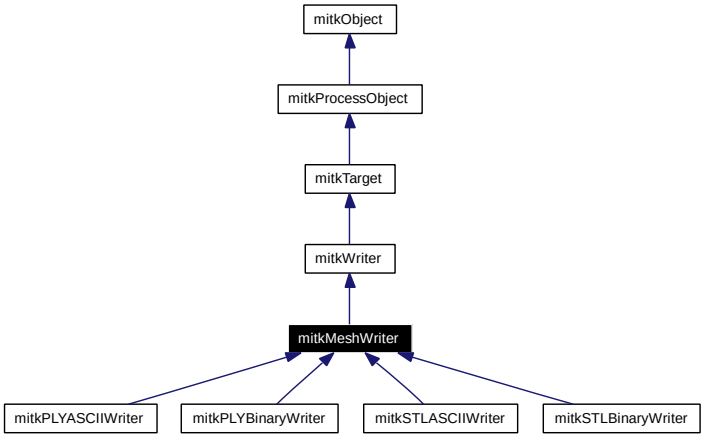
mitkMeshWriter - an abstract class represents a mesh writer for writing mesh data to disk

`#include <mitkMeshWriter.h>`

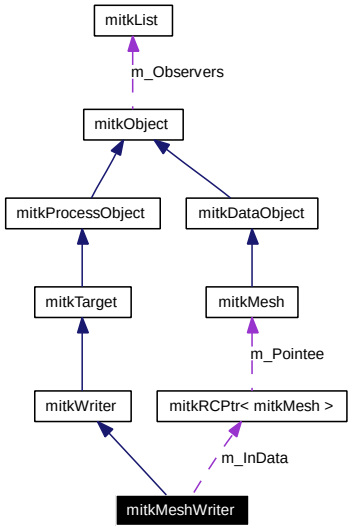
Inherits mitkWriter.

Inherited by mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkSTLASCIIWriter, and mitkSTLBinaryWriter.

Inheritance diagram for mitkMeshWriter:



Collaboration diagram for mitkMeshWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInput (mitkMesh *inData)
- mitkMesh * GetInput ()

8.156.1 Detailed Description

mitkMeshWriter - an abstract class represents a mesh writer for writing mesh data to disk

mitkMeshWriter is an abstract class represents a mesh writer for writing mesh data to disk. To use a concrete mesh reader, the code snippet is:

```
mitkSomeMeshWriter *aWriter = new mitkSomeMeshWriter;
aWriter->SetInput(aMesh);
aWriter->AddFileName(filename); //Only require one file name
aWriter->Run()
```

8.156.2 Member Function Documentation

8.156.2.1 mitkMesh* mitkMeshWriter::GetInput () [inline]

Get input mesh.

Returns:
Return the input volume

8.156.2.2 virtual void mitkMeshWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWriter.

Reimplemented in mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkSTLASCIIWriter, and mitkSTLBinaryWriter.

8.156.2.3 void mitkMeshWriter::SetInput (mitkMesh * *inData*) [inline]

Set input mesh to write to disk file.

Parameters:
inData Input volume

The documentation for this class was generated from the following file:

- mitkMeshWriter.h

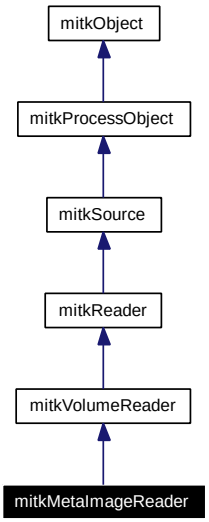
8.157 mitkMetaImageReader Class Reference

mitkMetaImageReader -

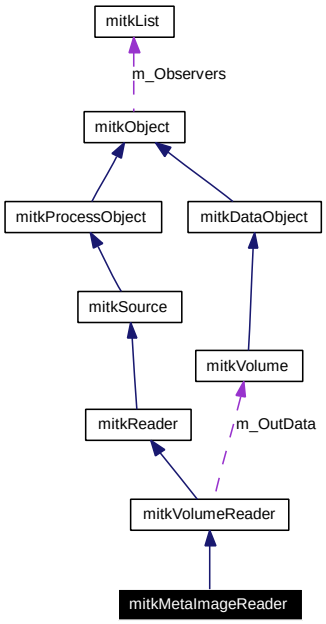
`#include <mitkMetaImageReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkMetaImageReader:



Collaboration diagram for mitkMetaImageReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.157.1 Detailed Description

mitkMetaImageReader -

mitkMetaImageReader

8.157.2 Member Function Documentation

8.157.2.1 virtual void mitkMetaImageReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

The documentation for this class was generated from the following file:

- mitkMetaImageReader.h

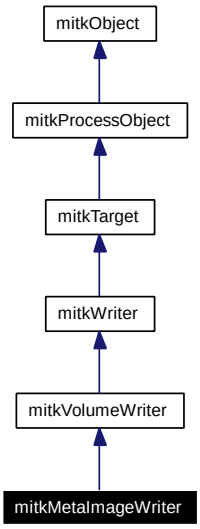
8.158 mitkMetaImageWriter Class Reference

mitkMetaImageWriter -

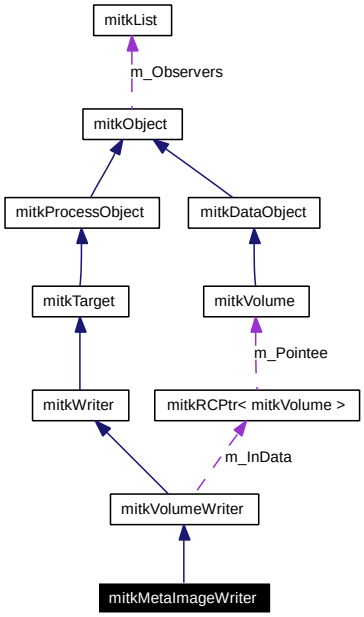
`#include <mitkMetaImageWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkMetaImageWriter:



Collaboration diagram for mitkMetaImageWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.158.1 Detailed Description

mitkMetaImageWriter -

mitkMetaImageWriter

8.158.2 Member Function Documentation

8.158.2.1 virtual void mitkMetaImageWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

The documentation for this class was generated from the following file:

- mitkMetaImageWriter.h

8.159 mitkMetric Class Reference

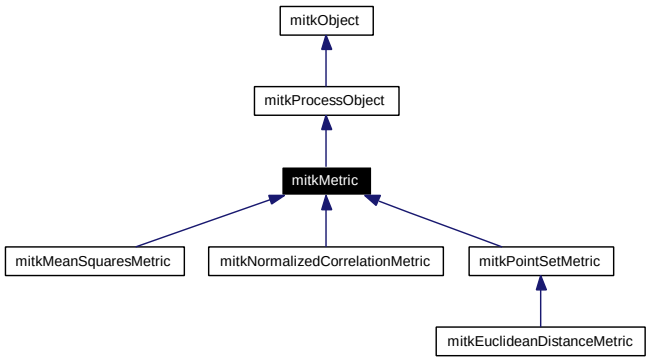
mitkMetric - an abstract class specifies interface for computing similarity between regions of two volumes

`#include <mitkMetric.h>`

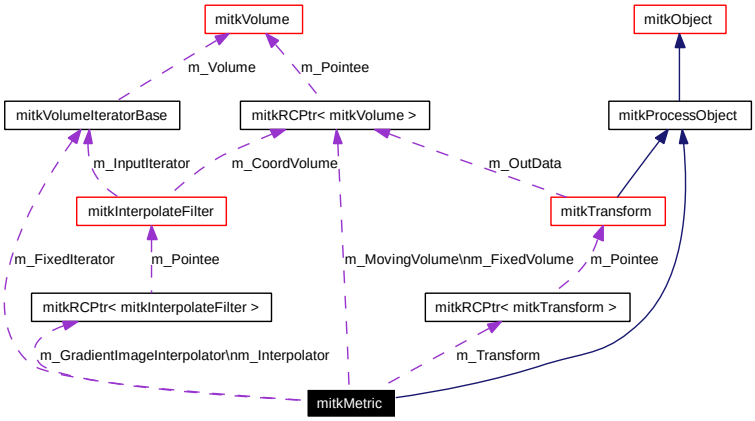
Inherits mitkProcessObject.

Inherited by mitkMeanSquaresMetric, mitkNormalizedCorrelationMetric, and mitkPointSetMetric.

Inheritance diagram for mitkMetric:



Collaboration diagram for mitkMetric:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetFixedVolume (mitkVolume *fixedVolume)
- mitkVolume * GetFixedVolume ()
- void SetMovingVolume (mitkVolume *movingVolume)
- mitkVolume * GetMovingVolume ()
- void SetInterpolator (mitkInterpolateFilter *interpolator)
- mitkInterpolateFilter * GetInterpolator ()
- void SetTransform (mitkTransform *transform)
- mitkTransform * GetTransform ()

- void SetTransformParameters (const VectorParameterType &parameters)
- VectorParameterType & GetTransformParameters ()
- virtual void Update ()
- ScalarParameterType GetSimilarity ()
- virtual bool GetSimilarity (const mitkMatrixType *matrix, ScalarParameterType &similarity)
- virtual bool GetSimilarity (const VectorParameterType &parameters, ScalarParameterType &similarity)
- virtual bool GetSimilarityAndDerivative (const VectorParameterType &parameters, ScalarParameterType &similarity, VectorParameterType &derivative)
- mitkInterpolateFilter * GetGradientImageInterpolator ()
- void SetComputeImageGradientFlag (bool flag)
- void SetInitialTransformMatrix ()
- void SetInitialTransformMatrix (mitkMatrixScalarType *matrix)

8.159.1 Detailed Description

mitkMetric - an abstract class specifies interface for computing similarity between regions of two volumes

mitkMetric is an abstract class specifies interface for computing similarity between regions of two volumes. This Class expects a Transform and an Interpolator to be plugged in. This particular class is the base class for a hierarchy of similarity metrics.

This class computes a value that measures the similarity between the Fixed image and the transformed Moving image. The Interpolator is used to compute intensity values on non-grid positions resulting from mapping points through the Transform.

8.159.2 Member Function Documentation

8.159.2.1 mitkVolume* mitkMetric::GetFixedVolume () [inline]

Get the Fixed Volume.

Returns:
Return the pointer to the fixed volume.

8.159.2.2 mitkInterpolateFilter* mitkMetric::GetGradientImageInterpolator ()

Get the Gaussian Derivative Interpolator.

Returns:
Return the pointer to the Gaussian Derivative Interpolator.

8.159.2.3 mitkInterpolateFilter* mitkMetric::GetInterpolator ()

Get the Interpolator.

Returns:
Return the pointer to the Interpolator.

8.159.2.4 mitkVolume* mitkMetric::GetMovingVolume () [inline]

Get the Moving Volume.

Returns:
Return the pointer to the moving volume.

8.159.2.5 virtual bool mitkMetric::GetSimilarity (const VectorParameterType & *parameters*, ScalarParameterType & *similarity*) [virtual]

Get Similarity between regions of two volumes.

Parameters:
parameters The vector pointer to the transform parameters.

similarity The metric results.

Returns:
Return true if the metric computation is performed without error.

Reimplemented in mitkMeanSquaresMetric, and mitkNormalizedCorrelationMetric.

8.159.2.6 virtual bool mitkMetric::GetSimilarity (const mitkMatrixType * *matrix*, ScalarParameterType & *similarity*) [virtual]

Get Similarity between fixed input and moving input.

Parameters:
matrix The vector pointer to the transform matrix.

similarity The metric results.

Returns:
Return true if the metric computation is performed without error.

8.159.2.7 ScalarParameterType mitkMetric::GetSimilarity () [inline]

Get Similarity between fixed input and moving input.

Returns:
Return the similarity.

8.159.2.8 virtual bool mitkMetric::GetSimilarityAndDerivative (const VectorParameterType & *parameters*, ScalarParameterType & *similarity*, VectorParameterType & *derivative*) [virtual]

Get Similarity And Gradient Value between regions of two volumes.

Parameters:
transformParameters The vector pointer to the transform parameters.

similarity Return the similarity.

derivative Return the gradient value in all dimensions.

Returns:
Return true if the metric computation is performed without error.

Reimplemented in mitkMeanSquaresMetric, and mitkNormalizedCorrelationMetric.

8.159.2.9 mitkTransform* mitkMetric::GetTransform ()

Get the Transform.

Returns:
Return the pointer to the Transform.

8.159.2.10 VectorParameterType& mitkMetric::GetTransformParameters ()

Get the transform parameters.

Returns:
Return the vector pointer to the transform parameters.

8.159.2.11 virtual void mitkMetric::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkProcessObject.

Reimplemented in mitkEuclideanDistanceMetric, mitkMeanSquaresMetric, mitkNormalizedCorrelation-Metric, and mitkPointSetMetric.

8.159.2.12 void mitkMetric::SetComputeImageGradientFlag (bool *flag*)

Set the Compute Derivative flag.

Parameters:
flag Perform derivative computing if the value is true.

8.159.2.13 void mitkMetric::SetFixedVolume (mitkVolume * *fixedVolume*) [inline]

Set the Fixed Volume.

Parameters:
fixedVolume The pointer to the fixed volume.

8.159.2.14 void mitkMetric::SetInitialTransformMatrix (mitkMatrixScalarType * *matrix*)

Set the initial transform matrix.

Parameters:
matrix The pointer to initial transform matrix.

8.159.2.15 void mitkMetric::SetInitialTransformMatrix ()

Initial the transform matrix to identity matrix.

8.159.2.16 void mitkMetric::SetInterpolator (mitkInterpolateFilter * *interpolator*) [inline]

Set the Interpolator.

Parameters:
interpolator The pointer to the Interpolator.

8.159.2.17 void mitkMetric::SetMovingVolume (mitkVolume * *movingVolume*) [inline]

Set the Moving Volume.

Parameters:
movingVolume The pointer to the moving volume.

8.159.2.18 void mitkMetric::SetTransform (mitkTransform * *transform*) [inline]

Set the Transform.

Parameters:
transform The pointer to the Transform.

8.159.2.19 void mitkMetric::SetTransformParameters (const VectorParameterType & *parameters*)

Set the Transform parameters.

Parameters:
parameters The vector pointer to the transform parameters.

8.159.2.20 virtual void mitkMetric::Update () [virtual]

Initialize the Metric by making sure that all the components are present and plugged together correctly. Perform before Run() function.

Reimplemented in mitkPointSetMetric.

The documentation for this class was generated from the following file:

- mitkMetric.h

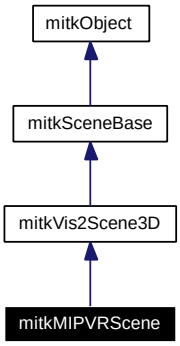
8.160 mitkMIPVRScene Class Reference

mitkMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Optimized for multicore CPU.

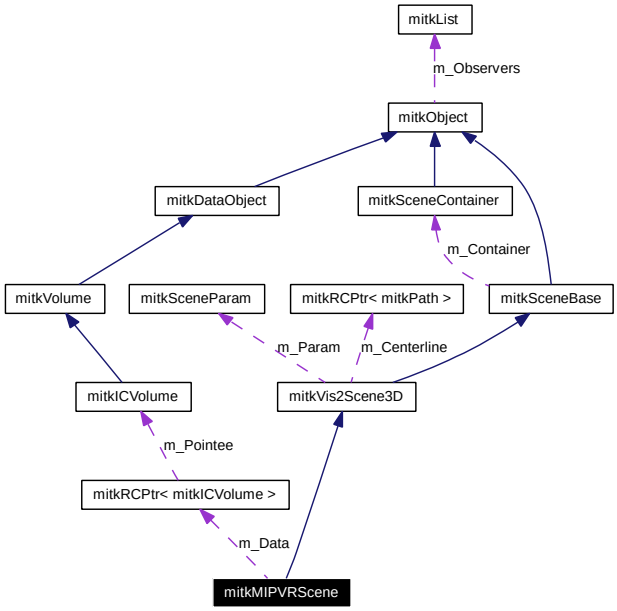
`#include <mitkMIPVRScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkMIPVRScene:



Collaboration diagram for mitkMIPVRScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data)
- void SetDataModified ()
- void SetCoherentData (mitkCoherentVolume *CVolume)
- void SetCoherentDataModified ()

- void SetWidthCenter (float windowWidth, float windowCenter)
- void SetWindowWidth (float windowWidth)
- void SetWindowCenter (float windowCenter)
- float GetWindowWidth ()
- float GetWindowCenter ()
- void SetCroppingBounds (const float CroppingBounds[6])
- void GetCroppingBounds (float CroppingBounds[6])

8.160.1 Detailed Description

mitkMIPVRScene - a 3D scene for MIP Rendering from volume datasets. Optimized for multicore CPU.

mitkMIPVRScene is a 3D scene for MIP Rendering from volume datasets. Optimized for multicore CPU. mitkCUDAMIPVRScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container. Two kinds of data source are supported, ICVolume and CohrerentVolume.

8.160.2 Member Function Documentation

8.160.2.1 void mitkMIPVRScene::GetCroppingBounds (float *CroppingBounds*[6])

Get the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.160.2.2 float mitkMIPVRScene::GetWindowCenter ()

Get the window center for the MIP

Returns:
Return the window center of the MIP

8.160.2.3 float mitkMIPVRScene::GetWindowWidth ()

Get the window width for the MIP

Returns:
Return the window width of the MIP

8.160.2.4 virtual void mitkMIPVRScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.160.2.5 void mitkMIPVRScene::SetCoherentData (mitkCoherentVolume * *CVolume*)

Set the data object(CoherentVolume) to be rendered.

Parameters:
data The data object to be rendered.

8.160.2.6 void mitkMIPVRScene::SetCoherentDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.160.2.7 void mitkMIPVRScene::SetCroppingBounds (const float *CroppingBounds*[6])

Set the cropping boundary

Parameters:
CroppingBounds,: the cropping boundary

8.160.2.8 void mitkMIPVRScene::SetData (mitkICVolume * *data*)

Set the data object(ICVolume) to be rendered.

Parameters:
data The data object to be rendered.

8.160.2.9 void mitkMIPVRScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.160.2.10 void mitkMIPVRScene::SetWidthCenter (float *windowWidth*, float *windowCenter*)

Set the window width and windows center for the MIP

Parameters:
windowWidth The window width of the MIP

windowCenter The window center of the MIP

8.160.2.11 void mitkMIPVRScene::SetWindowCenter (float *windowCenter*)

Set the window center for the MIP

Parameters:
windowCenter The window center of the MIP

8.160.2.12 void mitkMIPVRScene::SetWindowWidth (float *windowWidth*)

Set the window width for the MIP

Parameters:
windowWidth The window width of the MIP

The documentation for this class was generated from the following file:

- mitkMIPVRScene.h

8.161 mitkModel Class Reference

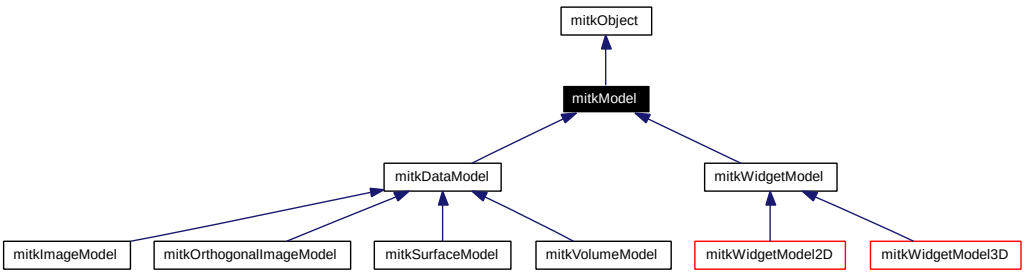
mitkModel - abstract class used to represent an entity in a rendering scene

`#include <mitkModel.h>`

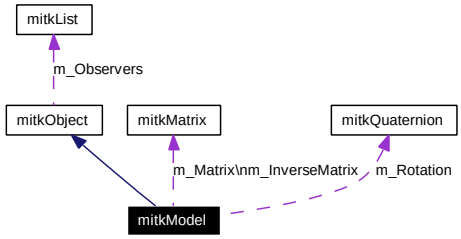
Inherits mitkObject.

Inherited by mitkDataModel, and mitkWidgetModel.

Inheritance diagram for mitkModel:



Collaboration diagram for mitkModel:



Public Types

- enum RenderMode { Rough, Medium, Refined }

Public Member Functions

- virtual void PrintSelf (ostream &os)
- void VisibilityOn (void)
- void VisibilityOff (void)
- void SetVisibility (int isVisible)
- int GetVisibility (void) const
- virtual int Render (mitkScene *scene)
- virtual void Select (mitkScene *scene)
- void SetOrigin (float x, float y, float z)
- void SetOrigin (float origin[3])
- float const * GetOrigin (void) const
- void GetOrigin (float origin[3]) const
- void SetTranslation (float x, float y, float z)

- void SetTranslation (float trans[3])
- float const * GetTranslation (void) const
- void GetTranslation (float trans[3]) const
- void SetRotation (float x, float y, float z)
- void SetRotation (float rot[3])
- void SetRotation (const mitkQuaternion &q)
- void SetRotation (float ax, float ay, float az, float angle)
- mitkQuaternion const * GetRotation (void) const
- void GetRotation (float rot[3]) const
- void GetRotation (float &ax, float &ay, float &az, float &angle) const
- void SetScale (float sx, float sy, float sz)
- void SetScale (float scale[3])
- void SetScale (float scale)
- float const * GetScale (void) const
- void GetScale (float scale[3]) const
- void GetModelMatrix (mitkMatrix *m)
- void GetModelMatrix (float m[16])
- mitkMatrix const * GetModelMatrix ()
- void GetInverseOfModelMatrix (mitkMatrix *m)
- void GetInverseOfModelMatrix (float m[16])
- mitkMatrix const * GetInverseOfModelMatrix ()
- float const * GetBounds ()
- void GetBounds (float bounds[6])
- void GetBounds (float &xMin, float &xMax, float &yMin, float &yMax, float &zMin, float &zMax)
- float const * GetCenter ()
- void GetCenter (float c[3])
- float GetLength ()
- void Reset ()
- void ModelToWorld (float const modelPoint[4], float worldPoint[4])
- void WorldToModel (float const worldPoint[4], float modelPoint[4])
- virtual bool IsOpaque ()=0
- bool GetDataModifyStatus () const
- void SetDataModifyStatus (bool isModify)
- void SetRenderMode (RenderMode mode)
- void SetRenderModeToRough ()
- void SetRenderModeToMedium ()
- void SetRenderModeToRefined ()
- RenderMode GetRenderMode () const

8.161.1 Detailed Description

mitkModel - abstract class used to represent an entity in a rendering scene

mitkModel is an abstract class used to represent an entity in a rendering scene.

8.161.2 Member Enumeration Documentation

8.161.2.1 enum mitkModel::RenderMode

The enumeration for render mode.

Enumerator:

Rough render model in rough mode to achieve a faster rendering process

Medium render model in medium mode to get a medium result at a medium rendering speed

Refined render model in refined mode, but result in a slower rendering speed

8.161.3 Member Function Documentation

8.161.3.1 void mitkModel::GetBounds (float & *xMin*, float & *xMax*, float & *yMin*, float & *yMax*, float & *zMin*, float & *zMax*)

Get the bounds for this Prop3D as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Parameters:

xMin minimum value in x-axis

xMax maximum value in x-axis

yMin minimum value in y-axis

yMax maximum value in y-axis

zMin minimum value in z-axis

zMax maximum value in z-axis

Note:

This function will call the virtual function GetBounds() to get the proper bounds according to the actual type of object.

8.161.3.2 void mitkModel::GetBounds (float *bounds*[6])

Get the bounds for this Prop3D as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Parameters:

bounds[6] a float array returns the Xmin, Xmax, Ymin, Ymax, Zmin and Zmax in turn.

Note:

This function will call the virtual function GetBounds() to get the proper bounds according to the actual type of object.

8.161.3.3 float const* mitkModel::GetBounds ()

Get the bounds for this Prop3D as (Xmin,Xmax,Ymin,Ymax,Zmin,Zmax).

Returns:

Return a float array which contains Xmin, Xmax, Ymin, Ymax, Zmin and Zmax in turn.

Note:

This is a pure virtual function. It must be implemented in subclasses.

8.161.3.4 void mitkModel::GetCenter (float *c*[3])

Get the center of the bounding box in world coordinates.

Parameters:
c[3] return a float array which contains Cx, Cy, Cz in turn

8.161.3.5 float const* mitkModel::GetCenter ()

Get the center of the bounding box in world coordinates.

Returns:
Return a float array which contains Cx, Cy, Cz in turn.

8.161.3.6 bool mitkModel::GetDataModifyStatus () const [inline]

Get the status of source data.

Returns:
Return true if the source data is modified. Otherwise return false.

8.161.3.7 mitkMatrix const* mitkModel::GetInverseOfModelMatrix ()

Get the inverse of model matrix.

Returns:
Return a pointer to a mitkMatrix contains the inverse of model matrix.

8.161.3.8 void mitkModel::GetInverseOfModelMatrix (float *m*[16])

Get the inverse of model matrix.

Parameters:
m[16] a float array returns the inverse of model matrix

Warning:
The matrix elements in the array are arranged as follows:

```
m[0]   m[4]   m[8]   m[12]
m[1]   m[5]   m[9]   m[13]
m[2]   m[6]   m[10]  m[14]
m[3]   m[7]   m[11]  m[15]
```

8.161.3.9 void mitkModel::GetInverseOfModelMatrix (mitkMatrix * *m*)

Get the inverse of model matrix.

Parameters:
m pointer to a mitkMatrix returns the inverse of model matrix

8.161.3.10 float mitkModel::GetLength ()

Get the length of the diagonal of the bounding box.

Returns:
the length of the diagonal of the bounding box

8.161.3.11 mitkMatrix const* mitkModel::GetModelMatrix ()

Return a reference to the model's 4x4 composite matrix. Get the matrix from the position, origin, scale and orientation this matrix is cached, so multiple GetMatrix() calls will be efficient.

Returns:
Return a pointer to a mitkMatrix contains the model matrix.

8.161.3.12 void mitkModel::GetModelMatrix (float *m*[16])

Return a reference to the model's 4x4 composite matrix. Get the matrix from the position, origin, scale and orientation this matrix is cached, so multiple GetMatrix() calls will be efficient.

Parameters:
m[16] a float array returns the model's 4x4 composite matrix

Warning:
The matrix elements in the array are arranged as follows:

```
m[0]   m[4]   m[8]   m[12]
m[1]   m[5]   m[9]   m[13]
m[2]   m[6]   m[10]  m[14]
m[3]   m[7]   m[11]  m[15]
```

8.161.3.13 void mitkModel::GetModelMatrix (mitkMatrix * *m*)

Return a reference to the model's 4x4 composite matrix. Get the matrix from the position, origin, scale and orientation this matrix is cached, so multiple GetMatrix() calls will be efficient.

Parameters:
m pointer to a mitkMatrix returns the model's 4x4 composite matrix

8.161.3.14 void mitkModel::GetOrigin (float *origin*[3]) const [inline]

Get the origin of the model. This is the point about which all rotations take place.

Parameters:
origin[0] return x-coordinate of the origin

origin[1] return y-coordinate of the origin

origin[2] return z-coordinate of the origin

8.161.3.15 float const* mitkModel::GetOrigin (void) const [inline]

Get the origin of the model. This is the point about which all rotations take place.

Returns:
Return a float array contains x, y and z coordinate in turn.

8.161.3.16 RenderMode mitkModel::GetRenderMode () const [inline]

Get the render mode of this model.

See also:
RenderMode

8.161.3.17 void mitkModel::GetRotation (float & *ax*, float & *ay*, float & *az*, float & *angle*) const [inline]

Get the rotation of the model.

Parameters:
ax return x-coordinate of rotation axis
ay return y-coordinate of rotation axis
az return z-coordinate of rotation axis
angle return the angle of rotation in degrees

8.161.3.18 void mitkModel::GetRotation (float *rot*[3]) const [inline]

Get the rotation of the model. (obsolete, just provided for convenience.)

Parameters:
rot[0] return rotation around x-axis in degrees
rot[1] return rotation around y-axis in degrees
rot[2] return rotation around z-axis in degrees

8.161.3.19 mitkQuaternion const* mitkModel::GetRotation (void) const [inline]

Get the rotation of the model.

Returns:
Return a float array contains rotations around x, y and z axes.

8.161.3.20 void mitkModel::GetScale (float *scale*[3]) const [inline]

Get the scale of the model.

Parameters:
scale[0] return scale in x-axis
scale[1] return scale in y-axis
scale[2] return scale in z-axis

8.161.3.21 float const* mitkModel::GetScale (void) const [inline]

Get the scales of the model.

Returns:
Return a float array contains scales in x, y and z axes.

8.161.3.22 void mitkModel::GetTranslation (float *trans*[3]) const [inline]

Get the translation of the model.

Parameters:
trans[0] return translation in x-axis
trans[1] return translation in y-axis
trans[2] return translation in z-axis

8.161.3.23 float const* mitkModel::GetTranslation (void) const [inline]

Get the translation of the model.

Returns:
Return a float array contains translations in x, y and z axes.

8.161.3.24 int mitkModel::GetVisibility (void) const [inline]

Get the visibility of this model.

Returns:
Return 1 if the model is visible. Otherwise return 0.

8.161.3.25 virtual bool mitkModel::IsOpaque () [pure virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Note:
This is a pure virtual function. It must be implemented in subclasses.

Implemented in mitkImageModel, mitkOrthogonalImageModel, mitkSurfaceModel, mitkVolumeModel, and mitkWidgetModel.

8.161.3.26 void mitkModel::ModelToWorld (float const *modelPoint*[4], float *worldPoint*[4]) [inline]

Transform a point from object(model) coordinate to world coordinate.

Parameters:
modelPoint[4] a float array contains coordinates of the model point
worldPoint[4] a float array to return coordinates of the world point corresponding to the model point

8.161.3.27 virtual void mitkModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkDataModel, mitkEllipseWidgetModel2D, mitkImageModel, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkOrthogonalImageModel, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, mitkReslicePlaneWidgetModel, mitkSurfaceModel, mitkVolumeModel, mitkWidgetModel, mitkWidgetModel2D, and mitkWidgetModel3D.

8.161.3.28 virtual int mitkModel::Render (mitkScene * *scene*) [inline, virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkEllipseWidgetModel2D, mitkImageModel, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkOrthogonalImageModel, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, mitkReslicePlaneWidgetModel, mitkSurfaceModel, and mitkVolumeModel.

8.161.3.29 void mitkModel::Reset ()

Reset all the geometric transformation

8.161.3.30 virtual void mitkModel::Select (mitkScene * *scene*) [inline, virtual]

Selecting object in interactive mode using widgets.

Parameters:
scene the pointer of an mitkScene in which this model is contained.

Warning:
Class inherited from mitkModel should rewrite this method only if itself can be picked. If not, ensure doing nothing in this method.

Reimplemented in mitkWidgetModel.

8.161.3.31 void mitkModel::SetDataModifyStatus (bool *isModify*) [inline]

Set the status of source data.

Parameters:
isModify if the status of source data is modified

8.161.3.32 void mitkModel::SetOrigin (float *origin*[3]) [inline]

Set the origin of the model. This is the point about which all rotations take place.

Parameters:
origin[0] x-coordinate of the origin

origin[1] y-coordinate of the origin

origin[2] z-coordinate of the origin

8.161.3.33 void mitkModel::SetOrigin (float *x*, float *y*, float *z*) [inline]

Set the origin of the model. This is the point about which all rotations take place.

Parameters:
x x-coordinate of the origin

y y-coordinate of the origin

z z-coordinate of the origin

8.161.3.34 void mitkModel::SetRenderMode (RenderMode *mode*) [inline]

Set render mode. It's just a hint to the renderer. The final result of rendering relies on the actual implementation of render algorithm.

Parameters:
mode the render mode

See also:
RenderMode

8.161.3.35 void mitkModel::SetRenderModeToMedium () [inline]

Set render mode to medium mode. It's just a hint to the renderer. The final result of rendering relies on the actual implementation of render algorithm.

See also:
RenderMode

8.161.3.36 void mitkModel::SetRenderModeToRefined () [inline]

Set render mode to refined mode. It's just a hint to the renderer. The final result of rendering relies on the actual implementation of render algorithm.

See also:
RenderMode

8.161.3.37 void mitkModel::SetRenderModeToRough () [inline]

Set render mode to rough mode. It's just a hint to the renderer. The final result of rendering relies on the actual implementation of render algorithm.

See also:
RenderMode

8.161.3.38 void mitkModel::SetRotation (float *ax*, float *ay*, float *az*, float *angle*) [inline]

Set the rotation of the model.

Parameters:
ax x-coordinate of rotation axis
ay y-coordinate of rotation axis
az z-coordinate of rotation axis
angle the angle of rotation in degrees

8.161.3.39 void mitkModel::SetRotation (const mitkQuaternion & *q*) [inline]

Set the rotation of the model.

Parameters:
q the quaternion of rotation

8.161.3.40 void mitkModel::SetRotation (float *rot*[3]) [inline]

Set the rotation of the model.

Parameters:
rot[0] rotation around x-axis in degrees
rot[1] rotation around y-axis in degrees
rot[2] rotation around z-axis in degrees

8.161.3.41 void mitkModel::SetRotation (float *x*, float *y*, float *z*) [inline]

Set the rotation of the model.

Parameters:
x rotation around x-axis in degrees
y rotation around y-axis in degrees
z rotation around z-axis in degrees

8.161.3.42 void mitkModel::SetScale (float *scale*) [inline]

Set the scales in x, y and z axes of the model to the same value.

Parameters:
scale scale in x, y and z axes

8.161.3.43 void mitkModel::SetScale (float *scale*[3]) [inline]

Set the scale of the model.

Parameters:
scale[0] scale in x-axis

scale[1] scale in y-axis

scale[2] scale in z-axis

8.161.3.44 void mitkModel::SetScale (float *sx*, float *sy*, float *sz*) [inline]

Set the scale of the model.

Parameters:
sx scale in x-axis

sy scale in y-axis

sz scale in z-axis

8.161.3.45 void mitkModel::SetTranslation (float *trans*[3]) [inline]

Set the translation of the model.

Parameters:
trans[0] translation in x-axis

trans[1] translation in y-axis

trans[2] translation in z-axis

8.161.3.46 void mitkModel::SetTranslation (float *x*, float *y*, float *z*) [inline]

Set the translation of the model.

Parameters:
x translation in x-axis

y translation in y-axis

z translation in z-axis

8.161.3.47 void mitkModel::SetVisibility (int *isVisible*) [inline]

Set the visibility of this model.

Parameters:

isVisible if this model is visible (0 - invisible; 1 - visible)

8.161.3.48 void mitkModel::VisibilityOff (void) [inline]

Set the visibility of this model off.

8.161.3.49 void mitkModel::VisibilityOn (void) [inline]

Set the visibility of this model on.

8.161.3.50 void mitkModel::WorldToModel (float const *worldPoint*[4], float *modelPoint*[4]) [inline]

Transform a point from world coordinate to object(model) coordinate

Parameters:

worldPoint[4] a float array contains coordinates of the world point

modelPoint[4] a float array to return coordinates of the model point corresponding to the world point

The documentation for this class was generated from the following file:

- mitkModel.h

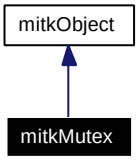
8.162 mitkMutex Class Reference

mitkMutex - a class that encapsulates a mutex

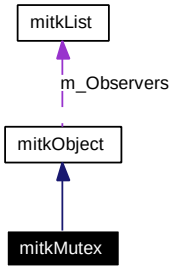
`#include <mitkMutex.h>`

Inherits mitkObject.

Inheritance diagram for mitkMutex:



Collaboration diagram for mitkMutex:



Public Member Functions

- void Lock ()
- void Unlock ()

8.162.1 Detailed Description

mitkMutex - a class that encapsulates a mutex

mitkMutex encapsulates a mutex.

8.162.2 Member Function Documentation

8.162.2.1 void mitkMutex::Lock () `[inline]`

Lock the mutex object

8.162.2.2 void mitkMutex::Unlock () `[inline]`

Unlock the mutex object

The documentation for this class was generated from the following file:

- mitkMutex.h

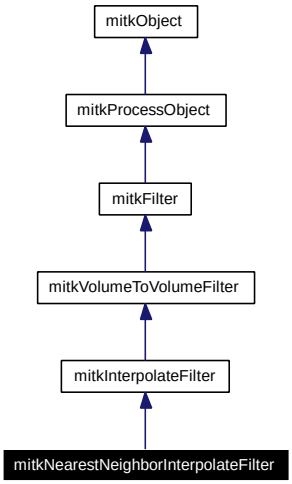
8.163 mitkNearestNeighborInterpolateFilter Class Reference

mitkNearestNeighborInterpolateFilter - a concrete interpolator

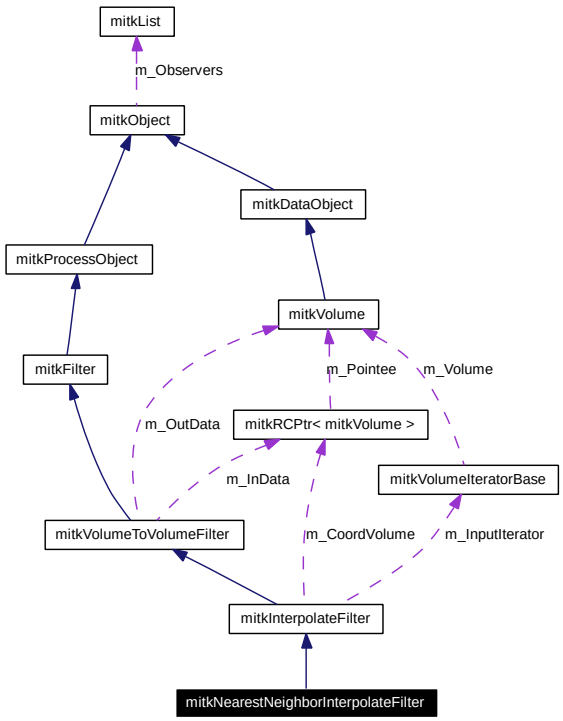
`#include <mitkNearestNeighborInterpolateFilter.h>`

Inherits mitkInterpolateFilter.

Inheritance diagram for mitkNearestNeighborInterpolateFilter:



Collaboration diagram for mitkNearestNeighborInterpolateFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkNearestNeighborInterpolateFilter ()

8.163.1 Detailed Description

mitkNearestNeighborInterpolateFilter - a concrete interpolator

mitkNearestNeighborInterpolateFilter is a concrete interpolator, which interpolates image intensity at a non-integer pixel position by copying the intensity for the nearest neighbor.

8.163.2 Constructor & Destructor Documentation

8.163.2.1 mitkNearestNeighborInterpolateFilter::mitkNearestNeighborInterpolateFilter ()

Constructor.

8.163.3 Member Function Documentation

8.163.3.1 virtual void mitkNearestNeighborInterpolateFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkInterpolateFilter.

The documentation for this class was generated from the following file:

- mitkNearestNeighborInterpolateFilter.h

8.164 mitkNeighborhood Class Reference

A light-weight container object for storing an N-dimensional neighborhood of values.

`#include <mitkNeighborhood.h>`

Public Types

- typedef mitkNeighborhood Self

Public Member Functions

- ~mitkNeighborhood ()
- mitkNeighborhood (const Self &other)
- void operator= (const Self &other)
- void SetDataType (int data_type)
- int GetDataType () const

8.164.1 Detailed Description

A light-weight container object for storing an N-dimensional neighborhood of values.

A mitkNeighborhood has an N-dimensional *radius*. The radius is defined separately for each dimension as the number of pixels that the neighborhood extends outward from the center pixel. For example, a 2D Neighborhood object with a radius of 2x3 has sides of length 5x7. Neighborhood objects always have an unambiguous center because their side lengths are always odd.

8.164.2 Member Typedef Documentation

8.164.2.1 typedef mitkNeighborhood mitkNeighborhood::Self

Standard class typedefs.

8.164.3 Constructor & Destructor Documentation

8.164.3.1 mitkNeighborhood::~mitkNeighborhood () `[inline]`

Default constructor.

8.164.3.2 mitkNeighborhood::mitkNeighborhood (const Self & *other*)

Copy constructor.

8.164.4 Member Function Documentation

8.164.4.1 int mitkNeighborhood::GetDataType () const `[inline]`

Get data type of this Neighborhood. MITK supports various data type.

Returns:

All of the return values and their meaning are shown as follows:
MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

8.164.4.2 void mitkNeighborhood::operator= (const Self & *other*)

Assignment operator.

8.164.4.3 void mitkNeighborhood::SetDataType (int *data_type*)

Set data type of this Neighborhood. MITK supports various data type.

Parameters:

data_type Its valid value and meaning is shown as follows:
MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

The documentation for this class was generated from the following file:

- mitkNeighborhood.h

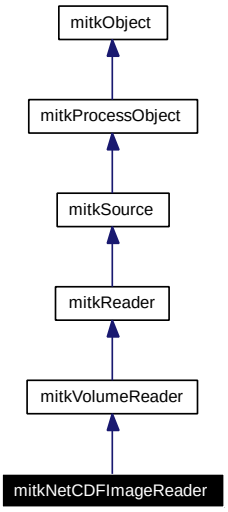
8.165 mitkNetCDFImageReader Class Reference

mitkNetCDFImageReader -

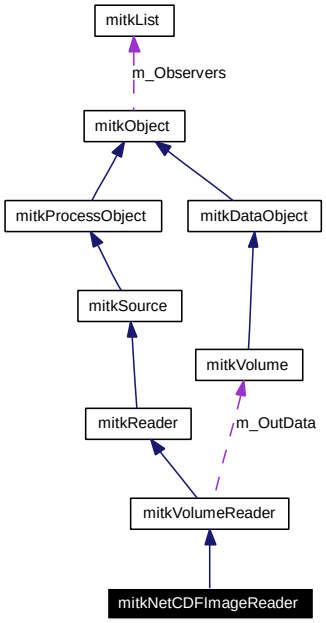
`#include <mitkNetCDFImageReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkNetCDFImageReader:



Collaboration diagram for mitkNetCDFImageReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.165.1 Detailed Description

mitkNetCDFImageReader -

mitkNetCDFImageReader

8.165.2 Member Function Documentation

8.165.2.1 virtual void mitkNetCDFImageReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

The documentation for this class was generated from the following file:

- mitkNetCDFImageReader.h

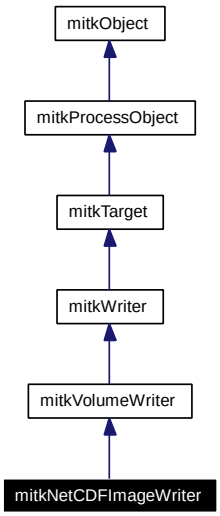
8.166 mitkNetCDFImageWriter Class Reference

mitkNetCDFImageWriter -

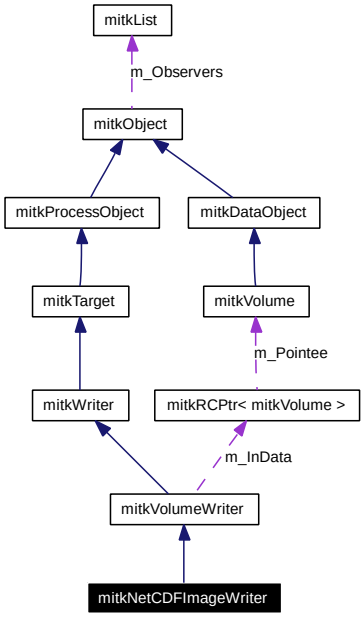
`#include <mitkNetCDFImageWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkNetCDFImageWriter:



Collaboration diagram for mitkNetCDFImageWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.166.1 Detailed Description

mitkNetCDFImageWriter -

mitkNetCDFImageWriter

8.166.2 Member Function Documentation

8.166.2.1 virtual void mitkNetCDFImageWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

The documentation for this class was generated from the following file:

- mitkNetCDFImageWriter.h

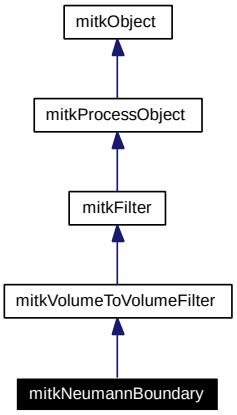
8.167 mitkNeumannBoundary Class Reference

A function object that determines a neighborhood of values at an image boundary according to a Neumann boundary condition where first, upwind derivatives on the boundary are zero. This is a useful condition in solving some classes of differential equations.

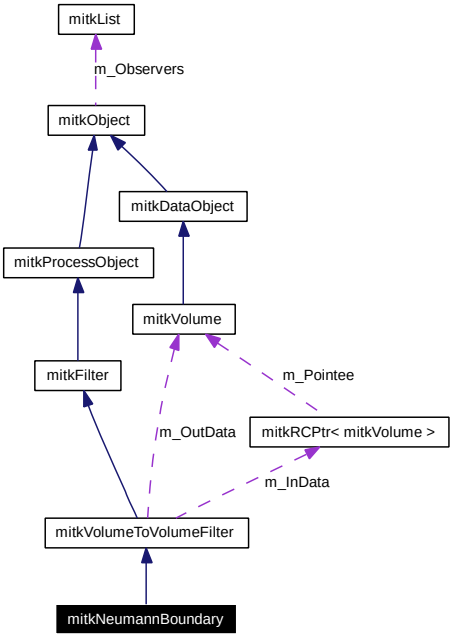
`#include <mitkNeumannBoundary.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkNeumannBoundary:



Collaboration diagram for mitkNeumannBoundary:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- mitkNeumannBoundary ()

8.167.1 Detailed Description

A function object that determines a neighborhood of values at an image boundary according to a Neumann boundary condition where first, upwind derivatives on the boundary are zero. This is a useful condition in solving some classes of differential equations.

For example, invoking this function object on a 7x5 iterator that masks a region at an image corner (iterator is centered on the 2):

```
* * * * * * *
* * * * * * *
* * 1 2 3 4 5  (where * denotes pixels that lie
* * 3 3 5 5 6          outside of the image boundary)
* * 4 4 6 7 8
```

returns the following neighborhood of values:

```
1 1 1 2 3 4 5
1 1 1 2 3 4 5
1 1 1 2 3 4 5
3 3 3 3 5 5 6   (note the corner values)
4 4 4 4 6 7 8
```

8.167.2 Constructor & Destructor Documentation

8.167.2.1 mitkNeumannBoundary::mitkNeumannBoundary () [inline]

Default constructor.

8.167.3 Member Function Documentation

8.167.3.1 virtual void mitkNeumannBoundary::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkNeumannBoundary.h

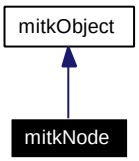
8.168 mitkNode Class Reference

mitkNode - a class that define NodeType used by mitkFastMarchingImageFilter

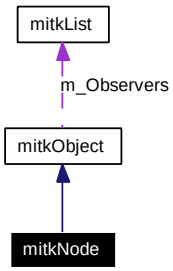
`#include <mitkNode.h>`

Inherits mitkObject.

Inheritance diagram for mitkNode:



Collaboration diagram for mitkNode:



Public Member Functions

- void SetIndex (int x, int y, int z)
- void SetIndex (Index index)
- void GetIndex (Index index) const
- void SetValue (double value)
- double GetValue () const
- bool operator> (const mitkNode &right)
- bool operator< (const mitkNode &right)
- mitkNode & operator= (const mitkNode &right)
- mitkNode (const mitkNode &node)

8.168.1 Detailed Description

mitkNode - a class that define NodeType used by mitkFastMarchingImageFilter

mitkNode defines several node types that represent the nodes' positions.

8.168.2 Constructor & Destructor Documentation

8.168.2.1 mitkNode::mitkNode (const mitkNode & *node*) [inline]

Constructor of this class

Parameters:
node The object according which to construct this class

8.168.3 Member Function Documentation

8.168.3.1 void mitkNode::GetIndex (Index *index*) const [inline]

Get the x, y, z coordinates of the node.

Returns:
index[0] Return the x coor of the node.
index[1] Return the y coor of the node.
index[2] Return the z coor of the node.

8.168.3.2 double mitkNode::GetValue () const [inline]

Get the pix value of the node.

Returns:
Return the pix value of the node.

8.168.3.3 bool mitkNode::operator< (const mitkNode & *right*) [inline]

Over load operator < for this class

Parameters:
right Represent the right side of the operator

Returns:
Return true if the left side < the right side, otherwize return false.

8.168.3.4 mitkNode& mitkNode::operator= (const mitkNode & *right*) [inline]

Over load operator = for this class

Parameters:
right Represent the right side of the operator

Returns:
Return an object equal to the right side.

8.168.3.5 bool mitkNode::operator> (const mitkNode & *right*) [inline]

Over load operator > for this class

Parameters:
right Represent the right side of the operator

Returns:
Return true if the left side > the right side, otherwize return false.

8.168.3.6 void mitkNode::SetIndex (Index *index*) [inline]

Set the x, y, z coordinates of the node.

Parameters:

index[0] Represent the x coor of the node.

index[1] Represent the y coor of the node.

index[2] Represent the z coor of the node.

8.168.3.7 void mitkNode::SetIndex (int *x*, int *y*, int *z*) [inline]

Set the x, y, z coordinates of the node.

Parameters:

x Represent the x coor of the node.

y Represent the y coor of the node.

z Represent the z coor of the node.

8.168.3.8 void mitkNode::SetValue (double *value*) [inline]

Set the pix value of the node.

Parameters:

value Represent the pix value of the node.

The documentation for this class was generated from the following file:

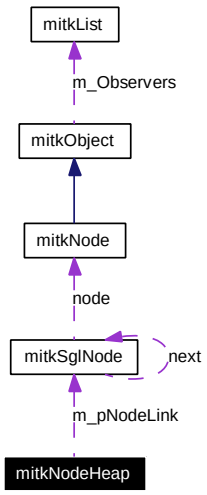
- mitkNode.h

8.169 mitkNodeHeap Class Reference

mitkNodeHeap - a class that define mitkNodeHeap used by mitkFastMarchingImageFilter

`#include <mitkNodeHeap.h>`

Collaboration diagram for mitkNodeHeap:



Public Member Functions

- mitkNodeHeap ()
- ~mitkNodeHeap ()
- mitkNode top ()
- void pop ()
- void push (mitkNode node)
- bool empty ()

8.169.1 Detailed Description

mitkNodeHeap - a class that define mitkNodeHeap used by mitkFastMarchingImageFilter

mitkNodeHeap defines a min-nodeHeap. The top of the heap is always the minimal of the elements in the heap.

8.169.2 Constructor & Destructor Documentation

8.169.2.1 mitkNodeHeap::mitkNodeHeap ()

Constructor of the class

8.169.2.2 mitkNodeHeap::~mitkNodeHeap ()

Destructor of the class

8.169.3 Member Function Documentation

8.169.3.1 bool mitkNodeHeap::empty ()

Check if the heap is empty

Returns:
Return true if the heap is empty otherwise return false.

8.169.3.2 void mitkNodeHeap::pop ()

remove the top node of the heap.

8.169.3.3 void mitkNodeHeap::push (mitkNode *node*)

Push a node into the heap.

Parameters:
node Represent the node to put into the heap

8.169.3.4 mitkNode mitkNodeHeap::top ()

Get the top node of the heap.

Returns:
Return the top node of the heap which is also the minimal element of the heap.

The documentation for this class was generated from the following file:

- mitkNodeHeap.h

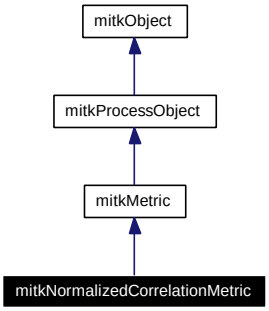
8.170 mitkNormalizedCorrelationMetric Class Reference

mitkNormalizedCorrelationMetric - a concrete class computing similarity between two objects to be registered

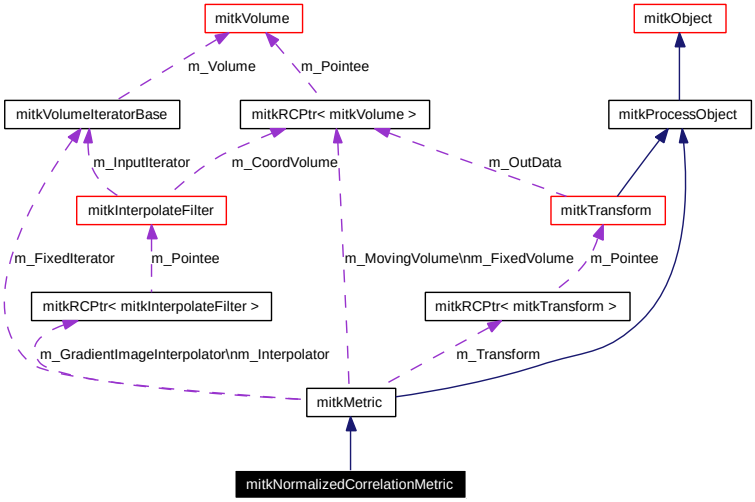
`#include <mitkNormalizedCorrelationMetric.h>`

Inherits mitkMetric.

Inheritance diagram for mitkNormalizedCorrelationMetric:



Collaboration diagram for mitkNormalizedCorrelationMetric:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkNormalizedCorrelationMetric ()
- virtual bool GetSimilarity (const VectorParameterType &parameters, ScalarParameterType &similarity)
- virtual bool GetSimilarityAndDerivative (const VectorParameterType &parameters, ScalarParameterType &similarity, VectorParameterType &derivative)
- void SetSubtractMeanFlag (bool flag)

8.170.1 Detailed Description

mitkNormalizedCorrelationMetric - a concrete class computing similarity between two objects to be registered

mitkNormalizedCorrelationMetric computes the correlation between pixels in the fixed image and pixels in the moving image. The spatial correspondance between fixed and moving image is established through a Transform. Pixel values are taken from the fixed image, their positions are mapped to the moving image and result in general in non-grid position on it. Values at these non-grid position of the moving image are interpolated using a user-selected Interpolator. The correlation is normalized by the autocorrelations of both the fixed and moving images.

8.170.2 Constructor & Destructor Documentation

8.170.2.1 mitkNormalizedCorrelationMetric::mitkNormalizedCorrelationMetric ()

Constructor.

8.170.3 Member Function Documentation

8.170.3.1 virtual bool mitkNormalizedCorrelationMetric::GetSimilarity (const VectorParameterType & *parameters*, ScalarParameterType & *similarity*) [virtual]

Get Similarity between regions of two volumes.

Parameters:

parameters The vector pointer to the transform parameters.

similarity The metric results.

Returns:

Return true if the metric computation is performed without error.

Reimplemented from mitkMetric.

8.170.3.2 virtual bool mitkNormalizedCorrelationMetric::GetSimilarityAndDerivative (const VectorParameterType & *parameters*, ScalarParameterType & *similarity*, VectorParameterType & *derivative*) [virtual]

Get Similarity And Gradient Value between regions of two volumes.

Parameters:

transformParameters The vector pointer to the transform parameters.

similarity Return the similarity.

derivative Return the gradient value in all dimensions.

Returns:

Return true if the metric computation is performed without error.

Reimplemented from mitkMetric.

8.170.3.3 virtual void mitkNormalizedCorrelationMetric::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkMetric.

8.170.3.4 void mitkNormalizedCorrelationMetric::SetSubtractMeanFlag (bool *flag*) [inline]

Set the Subtract Mean mode.Default is false.

Parameters:
flag Specify the subtract mean flag.

The documentation for this class was generated from the following file:

- mitkNormalizedCorrelationMetric.h

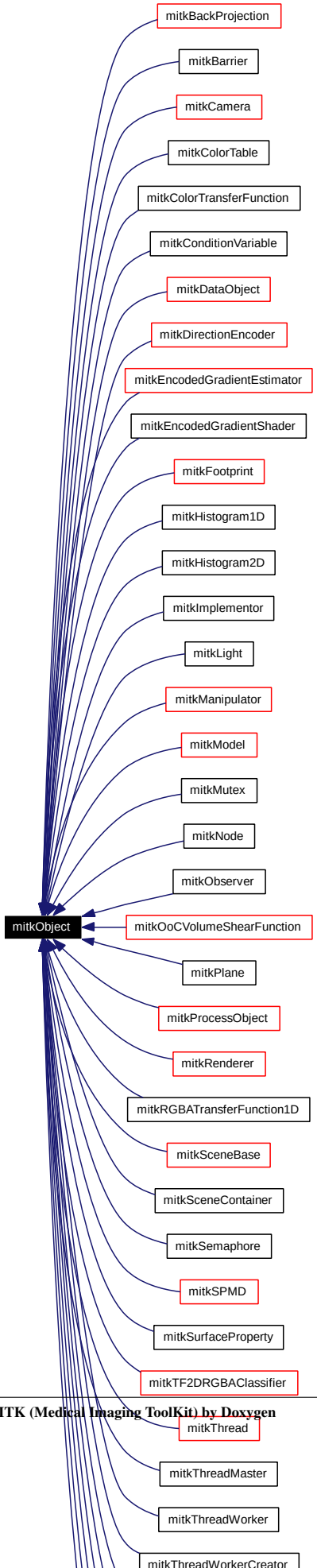
8.171 mitkObject Class Reference

mitkObject - abstract base class for most objects in MITK

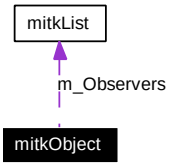
`#include <mitkObject.h>`

Inherited by mitkBackProjection, mitkBarrier, mitkCamera, mitkColorTable, mitkColorTransferFunction, mitkConditionVariable, mitkDataObject, mitkDirectionEncoder, mitkEncodedGradientEstimator, mitkEncodedGradientShader, mitkFootprint, mitkGeneratePathNetwork, mitkGLSLProgram, mitkGLSLShader, mitkHistogram1D, mitkHistogram2D, mitkImplementor, mitkIntNode, mitkLight, mitkMainPathExtraction, mitkManipulator, mitkModel, mitkMutex, mitkNode, mitkObserver, mitkOoCVolumeShearFunction, mitkPath, mitkPathNetwork, mitkPhantom, mitkPhantomElement, mitkPlane, mitkProcessObject, mitkRenderer, mitkRGBATransferFunction1D, mitkSceneBase, mitkSceneContainer, mitkSemaphore, mitkSPMD, mitkSurfaceProperty, mitkTF2DRGBAClassifier, mitkThread, mitkThreadMaster, mitkThreadWorker, mitkThreadWorkerCreator, mitkTransferFunction, mitkView, mitkVolumeDivider, mitkVolumeProperty, mitkVolumeRayCastFunction, and mitkVolumeSplatFunction.

Inheritance diagram for mitkObject:


mitkBackProjection
mitkBarrier
mitkCamera
mitkColorTable
mitkColorTransferFunction
mitkConditionVariable
mitkDataObject
mitkDirectionEncoder
mitkEncodedGradientEstimator
mitkEncodedGradientShader
mitkFootprint
mitkHistogram1D
mitkHistogram2D
mitkImplementor
mitkLight
mitkManipulator
mitkModel
mitkMutex
mitkNode
mitkObserver
mitkObject
mitkOoCVolumeShearFunction
mitkPlane
mitkProcessObject
mitkRenderer
mitkRGBATransferFunction1D
mitkSceneBase
mitkSceneContainer
mitkSemaphore
mitkSPMD
mitkSurfaceProperty
mitkTF2DRGBAClassifier
mitkThread
mitkThreadMaster
mitkThreadWorker
mitkThreadWorkerCreator

Collaboration diagram for mitkObject:



Public Member Functions

- virtual const char * GetClassname () const
- virtual int IsA (const char *name)
- virtual void DebugOn ()
- virtual void DebugOff ()
- unsigned char GetDebug ()
- void SetDebug (unsigned char debugFlag)
- void Print (ostream &os)
- virtual void PrintSelf (ostream &os)
- void AddObserver (mitkObserver *observer)
- void RemoveObserver (mitkObserver *observer)
- void RemoveAllObservers ()
- void AddReference ()
- void RemoveReference ()
- int GetReferenceCount ()
- void Delete ()

Static Public Member Functions

- static int IsTypeOf (const char *name)
- static mitkObject * SafeDownCast (mitkObject *o)
- static void BreakOnError ()

8.171.1 Detailed Description

mitkObject - abstract base class for most objects in MITK

mitkObject is the base class for most objects in MITK. It provides several base services for all of MITK objects.

The first base service is the RTTI(Run Time Type Information). Please see the GetClassname(), IsTypeOf(), IsA(), SafeDownCast()

The second base service is the debugging information. Please see the DebugOn(), DebugOff(), GetDebug(), SetDebug(), BreakOnError(), PrintSelf()

The third base service is the memory management, including Reference Counting and Garbage Collection. Please see the AddReference(), RemoveReference(), GetReferenceCount(), Delete()

The fourth base service is the support of the observer design pattern, which makes the updating user interface to reflect the internal status of a MITK object possible. Please see the AddObserver(), RemoveObserver(), RemoveAllObservers()

Note:
The deconstructor of mitkObject and all of its subclasses is protected. This means that a object in MITK must be allocated in the heap using new operator. If you define a local MITK object in the stack, the compiler will generate an error information.

8.171.2 Member Function Documentation

8.171.2.1 void mitkObject::AddObserver (mitkObserver * *observer*)

Attach an observer to this object.

Parameters:
observer pointer to an mitkObserver to attach

8.171.2.2 void mitkObject::AddReference ()

Add 1 to the referenct count. Only when the reference count of a MITK object is equal to 0, it can be deleted.

8.171.2.3 static void mitkObject::BreakOnError () [static]

This method is called when mitkErrorMessage executes.

8.171.2.4 virtual void mitkObject::DebugOff () [virtual]

Turn debugging output off.

8.171.2.5 virtual void mitkObject::DebugOn () [virtual]

Turn debugging output on.

8.171.2.6 void mitkObject::Delete ()

If current reference count is equal to 0, delete this object, otherwise, nothing happens.

Note:
Usually a MITK object can only be deleted through two ways. One is to call RemoveReference(), and the other is to call Delete(). But RemoveReference() is usually called internally to implement the reference counting design pattern, so Delete() is the correct function to call if you want to delete a MITK object.

8.171.2.7 virtual const char* mitkObject::GetClassname () const [inline, virtual]

Get the class name as a string. The purpose is to support RTTI.

Returns:
Return the class name of this object. For mitkObject, it always return "mitkObject"

8.171.2.8 unsigned char mitkObject::GetDebug ()

Get the value of the debug flag.

Returns:
Return zero, the debug flag is off, otherwise the debug flag is on.

8.171.2.9 int mitkObject::GetReferenceCount () [inline]

Get current referenct count of this object. Only when the reference count of a MITK object is equal to 0, it can be deleted.

Returns:
Return the reference count of this object.

8.171.2.10 virtual int mitkObject::IsA (const char * *name*) [virtual]

Decide if this class is one type of the specified class

Parameters:
name The name of specified class

Returns:
Return 1 if this class is the same type of (or a subclass of) the specified class. Returns 0 otherwise.

8.171.2.11 static int mitkObject::IsTypeOf (const char * *name*) [static]

Decide if this class is one type of the specified class

Parameters:
name The name of specified class

Returns:
Return 1 if this class is the same type of (or a subclass of) the specified class. Returns 0 otherwise.

8.171.2.12 void mitkObject::Print (ostream & *os*)

Print this object to an ostream.

Parameters:
os The specified ostream to output information.

8.171.2.13 virtual void mitkObject::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented in mitkAnalyzeImageReader, mitkAnalyzeImageWriter, mitkBinaryFilter, mitkBMPReader, mitkBMPWriter, mitkCacheVolumeReader, mitkCacheVolumeWriter, mitkDataObject, mitkDICOMInfoReader, mitkDICOMReader, mitkDICOMWriter, mitkDiffusionFilter, mitkFFTFilter, mitkFilter, mitkGaussianDerivativeImageFilter, mitkGaussianFilter, mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkHETriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkImageIOFactory, mitkInfoReader, mitkJPEGReader, mitkJPEGWriter, mitkMesh, mitkMeshReader, mitkMeshToMeshFilter, mitkMeshWriter, mitkMetaImageReader, mitkMetaImageWriter, mitkNetCDFImageReader, mitkNetCDFImageWriter, mitkNeumannBoundary, mitkObserver, mitkOoCTriangleMesh, mitkOoCVolume, mitkPlane, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkPLYReader, mitkProcessObject, mitkQEMSimplification, mitkRawFilesReader, mitkRawFilesWriter, mitkRawReader, mitkRawWriter, mitkReader, mitkRGBToGrayFilter, mitkSheetFilter, mitkSobelEdgeDetectFilter, mitkSource, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkSTLReader, mitkTarget, mitkTIFFReader, mitkTIFFWriter, mitkTriangleMesh, mitkTriangleMeshSimplification, mitkVolume, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeReader, mitkVolumeResizeFilter, mitkVolumeResliceFilter, mitkVolumeToMeshFilter, mitkVolumeToVolumeFilter, mitkVolumeWriter, mitkWriter, mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitk3DUSPixelBasedMT, mitk3DUSReconstructionFilter, mitkBackProjection, mitkCBFDKReconCF, mitkCBFDKReconCFCPU, mitkCBTFDKReconCF, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, mitkConeBeamProjectData, mitkConeBeamRealTimePD, mitkConeBeamReconFromHelix, mitkCTProjectionData, mitkCTReconstructionFilter, mitkFanBeamProjectData, mitkFanBeamReconFromCirle, mitkFbFbp, mitkFBFBPReconCF, mitkFBFBPReconCFCPU, mitkFbFbpSpeedUsingCpu, mitkFDK, mitkFDKSpeedUsingCpu, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkParallelBeamReconFromCirle, mitkParallelProjectData, mitkPbFbp, mitkPBFBPReconCF, mitkProjectData, mitkProjectDataToVolumeFilter, mitkRawConeBeamProjectData, mitkRawData, mitkRawDataReader, mitkRawDataToVolumeFilter, mitkRawDataWriter, mitkRawFilesReader, mitkTFDK, mitkTrackedBscanData, mitkTrackedBscanDataReader, mitkTrackedBscanDataWriter, mitkTrackedBscanICData, mitkTrackedBscanOoCData, mitkAffineTransform, mitkAmoebaOptimizer, mitkBSplineInterpolateFilter, mitkCubicInterpolateFilter, mitkEuclideanDistanceMetric, mitkGradientDescentOptimizer, mitkInterpolateFilter, mitkIterativeClosestPointOptimizer, mitkLandmarkRegistrationFilter, mitkLBFGSBOptimizer, mitkLBFGSOptimizer, mitkLinearInterpolateFilter, mitkMeanSquaresMetric, mitkMetric, mitkNearestNeighborInterpolateFilter, mitkNormalizedCorrelationMetric, mitkOptimizer, mitkPointSetEstimator, mitkPointSetKernelTransformEstimator, mitkPointSetMetric, mitkPointSetQuaternionEstimator, mitkPointSetRegistrationFilter, mitkPowellOptimizer, mitkRegistrationFilter, mitkRigidTransform, mitkScaleTransform, mitkSimilarityTransform, mitkSubtractImageFilter, mitkThinPlateSplineTransform, mitkTransform, mitkTranslationTransform, mitkVersorRigid3DTransform, mitkVersorSimilarity3DTransform, mitkConnectivityLabelFilter, mitkLiveWireImageFilter, mitkOtsuSegmentationFilter, mitkRegionGrowImageFilter, mitkSeedFillFilter, mitkThresholdSegmentationFilter, mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkBinMarchingCubes, mitkCamera, mitkClippingPlaneWidgetModel, mitkColorTable, mitkColorTransferFunction, mitkDataModel, mitkEllipseWidgetModel2D, mitkFootprint, mitkFootprint1D, mitkFootprint1DGaussian, mitkFootprint2D, mitkFootprint2DGaussian, mitkImageModel, mitkImageView, mitkImageViewManipulatorStandard, mitkImageViewManipulatorWithWidgets, mitkLight, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkManipulator, mitkMarchingCubes, mitkMeshViewManipulatorStandard, mitkModel, mitkOoCSurfaceRendererStandard, mitkOoCSurfaceRendererUseVA, mitkOoCVolumeRayCastCompositeFunction, mitkOoCVolumeRayCastCompositeFunctionEx, mitkOoCVolumeRendererRayCasting, mitkOoCVolumeRendererRayCastingEx, mitkOoCVolumeRendererShearWarp, mitkOoCVolumeRendererTexture3D, mitkOoCVolumeRendererTexture3DEx, mitkOoCVolumeShearFunction, mitkOoCVolumeShearParallel, mitkOoCVolumeShearPerspective, mitkOrthogonalImageModel, mitkPickManipulator, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, mitkRenderer, mitkReslicePlaneWidgetModel, mitkSplatCamera, mitkSurfaceModel, mitkSurfaceProperty, mitkSurfaceRenderer, mitkSurfaceRendererStandard, mitkSurfaceRendererUseVA, mitkSurfaceRendererUseVBO, mitkTransferFunction, mitkTransferFunction1D, mitkView, mitkVolumeDivider, mitkVolumeModel, mitkVolumeProperty, mitkVolumeRayCastCompositeFunction, mitkVolumeRayCastFunction, mitk-

VolumeRenderer, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, mitkVolumeRendererTexture3D, mitkVolumeSplatFunction, mitkVolumeSplatParallel, mitkVolumeSplatPerspective, mitkWidgetModel, mitkWidgetModel2D, mitkWidgetModel3D, mitkWidgetsViewManipulator, mitk2DBackProjectionUsingGPU, mitk3DBackProjectionUsingGPU, mitkBackProjectionUsingGPU, mitkCBFDKReconCFGPU, mitkFBFBPReconCFGPU, mitkFbFbpSpeedUsingGpu, mitkFDKSpeedUsingGpu, and mitk3DUSPixelBasedGPU.

8.171.2.14 void mitkObject::RemoveAllObservers ()

Detach all observers from this object.

8.171.2.15 void mitkObject::RemoveObserver (mitkObserver * *observer*)

Detach an observer from this object.

Parameters:
observer pointer to an mitkObserver to detach

8.171.2.16 void mitkObject::RemoveReference ()

Remove 1 to the referenct count. If reference count is equal to zero after remove 1, then this object is deleted automatically.

8.171.2.17 static mitkObject* mitkObject::SafeDownCast (mitkObject * *o*) [static]

Safely cast the specified object to mitkObject*

Parameters:
o The specified object

Returns:
If success, return the casted o, otherwise return NULL.

8.171.2.18 void mitkObject::SetDebug (unsigned char *debugFlag*)

Set the value of the debug flag.

Parameters:
debugFlag If debugFlag is zero, set the debug flag to off, otherwise set the debug flag to on.

The documentation for this class was generated from the following file:

- mitkObject.h

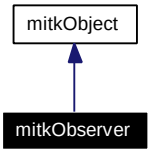
8.172 mitkObserver Class Reference

mitkObserver - abstract base class for observers

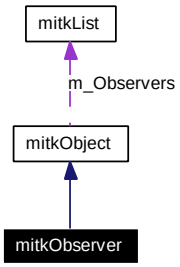
`#include <mitkObserver.h>`

Inherits mitkObject.

Inheritance diagram for mitkObserver:



Collaboration diagram for mitkObserver:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Update ()=0

8.172.1 Detailed Description

mitkObserver - abstract base class for observers

mitkObserver is an abstract class for observers. All mitkObjects can attach observers. You can derive from this class to get an actual observer processing the interaction between an mitkObject and the user interface.

8.172.2 Member Function Documentation

8.172.2.1 virtual void mitkObserver::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

8.172.2.2 virtual void mitkObserver::Update () [pure virtual]

Update the observer according to the changes of the mitkObject it is attached to. A working observer should implement this pure virtual function.

The documentation for this class was generated from the following file:

- mitkObserver.h

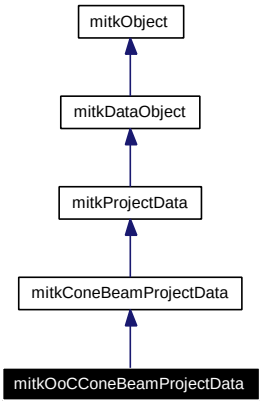
8.173 mitkOoCConeBeamProjectData Class Reference

mitkOoCConeBeamProjectData - concret class to represent cone-beam project data.

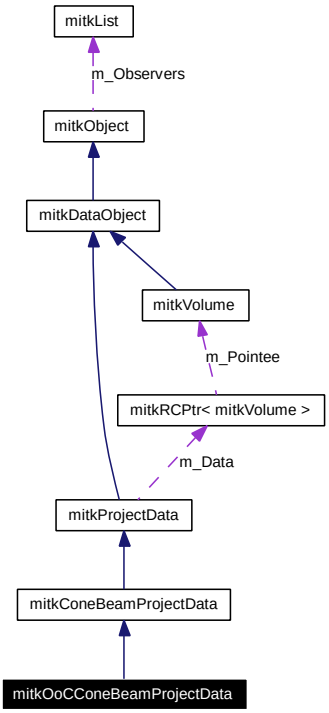
`#include <mitkOoCConeBeamProjectData.h>`

Inherits mitkConeBeamProjectData.

Inheritance diagram for mitkOoCConeBeamProjectData:



Collaboration diagram for mitkOoCConeBeamProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject ∗src)
- virtual void DeepCopy (mitkDataObject ∗src)
- virtual void const ∗ GetAngleSliceForRead (int sliceIdx)
- virtual void Initialize ()

8.173.1 Detailed Description

mitkOoCConeBeamProjectData - concret class to represent cone-beam project data.

mitkOoCConeBeamProjectData - concret class to represent cone-beam project data.

8.173.2 Member Function Documentation

8.173.2.1 virtual void mitkOoCConeBeamProjectData::DeepCopy (mitkDataObject ∗ *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamProjectData.

8.173.2.2 virtual unsigned long long mitkOoCConeBeamProjectData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkConeBeamProjectData.

8.173.2.3 virtual void const∗ mitkOoCConeBeamProjectData::GetAngleSliceForRead (int *sliceIdx*) [virtual]

GetAngleSliceForRead GetAngleSliceForWrite and GetAngleSliceForReadWrite

Implements mitkConeBeamProjectData.

8.173.2.4 virtual void mitkOoCConeBeamProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkConeBeamProjectData.

8.173.2.5 virtual void mitkOoCConeBeamProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkConeBeamProjectData.

8.173.2.6 virtual void mitkOoCConeBeamProjectData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamProjectData.

The documentation for this class was generated from the following file:

- mitkOoCConeBeamProjectData.h

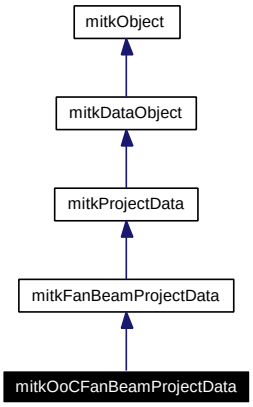
8.174 mitkOoCFanBeamProjectData Class Reference

mitkOoCFanBeamProjectData - concret class to represent fan-beam project data.

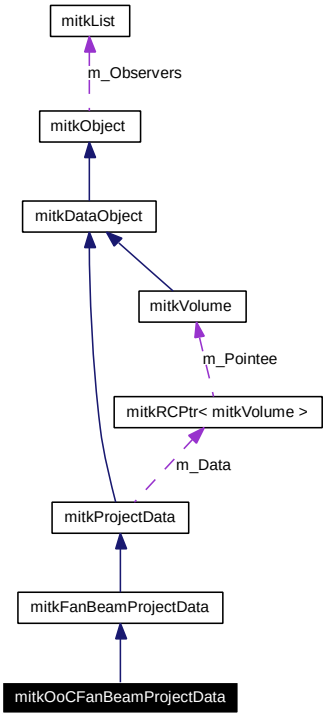
`#include <mitkOoCFanBeamProjectData.h>`

Inherits mitkFanBeamProjectData.

Inheritance diagram for mitkOoCFanBeamProjectData:



Collaboration diagram for mitkOoCFanBeamProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject ∗src)
- virtual void DeepCopy (mitkDataObject ∗src)
- virtual void Initialize ()
- virtual void const ∗ GetSliceForRead (int sliceIdx)

8.174.1 Detailed Description

mitkOoCFanBeamProjectData - concret class to represent fan-beam project data.

mitkOoCFanBeamProjectData - concret class to represent fan-beam project data.

8.174.2 Member Function Documentation

8.174.2.1 virtual void mitkOoCFanBeamProjectData::DeepCopy (mitkDataObject ∗ *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkFanBeamProjectData.

8.174.2.2 virtual unsigned long long mitkOoCFanBeamProjectData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkFanBeamProjectData.

8.174.2.3 virtual void const∗ mitkOoCFanBeamProjectData::GetSliceForRead (int *sliceIdx*) [virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implements mitkFanBeamProjectData.

8.174.2.4 virtual void mitkOoCFanBeamProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkFanBeamProjectData.

8.174.2.5 virtual void mitkOoCFanBeamProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFanBeamProjectData.

8.174.2.6 virtual void mitkOoCFanBeamProjectData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkFanBeamProjectData.

The documentation for this class was generated from the following file:

- mitkOoCFanBeamProjectData.h

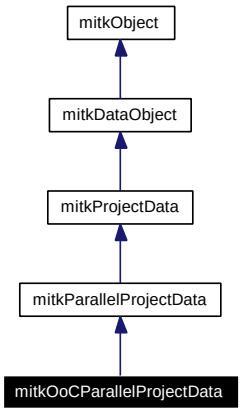
8.175 mitkOoCParallelProjectData Class Reference

mitkOoCParallelProjectData - concret class to represent Parallel project data.

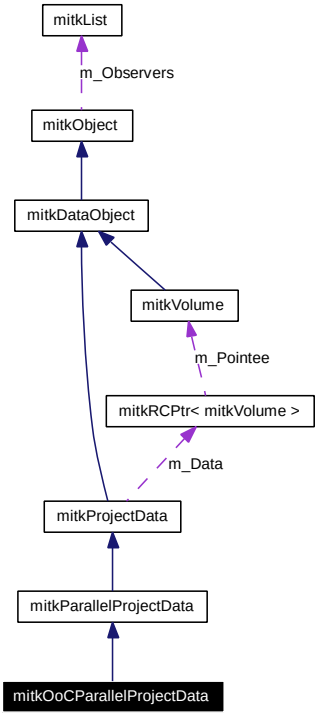
`#include <mitkOoCParallelProjectData.h>`

Inherits mitkParallelProjectData.

Inheritance diagram for mitkOoCParallelProjectData:



Collaboration diagram for mitkOoCParallelProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int GetDataObjectType () const
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void Initialize ()
- virtual void const * GetSliceForRead (int sliceIdx)

8.175.1 Detailed Description

mitkOoCParallelProjectData - concret class to represent Parallel project data.

mitkOoCParallelProjectData - concret class to represent Parallel project data.

8.175.2 Member Function Documentation

8.175.2.1 virtual void mitkOoCParallelProjectData::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkParallelProjectData.

8.175.2.2 virtual unsigned long long mitkOoCParallelProjectData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkDataObject.

8.175.2.3 virtual int mitkOoCParallelProjectData::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_DATA_OBJECT

Note:
Pure virtual function. Its concrete subclass must implement this function and return its data object type.

Reimplemented from mitkParallelProjectData.

8.175.2.4 virtual void const* mitkOoCParallelProjectData::GetSliceForRead (int *sliceIdx*) [virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implements mitkParallelProjectData.

8.175.2.5 virtual void mitkOoCParallelProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkParallelProjectData.

8.175.2.6 virtual void mitkOoCParallelProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkParallelProjectData.

8.175.2.7 virtual void mitkOoCParallelProjectData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkParallelProjectData.

The documentation for this class was generated from the following file:

- mitkOoCParallelProjectData.h

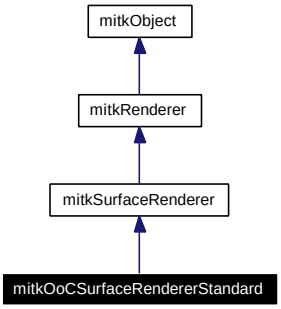
8.176 mitkOoCSurfaceRendererStandard Class Reference

mitkOoCSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

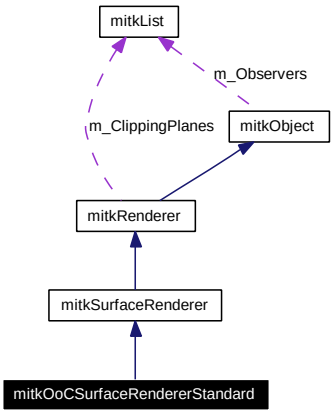
`#include <mitkOoCSurfaceRendererStandard.h>`

Inherits mitkSurfaceRenderer.

Inheritance diagram for mitkOoCSurfaceRendererStandard:



Collaboration diagram for mitkOoCSurfaceRendererStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.176.1 Detailed Description

mitkOoCSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

mitkOoCSurfaceRendererStandard is a standard renderer object for mitkSurfaceModel using general OpenGL.

8.176.2 Member Function Documentation

8.176.2.1 virtual void mitkOoCSurfaceRendererStandard::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkSurfaceRenderer.

The documentation for this class was generated from the following file:

- mitkOoCSurfaceRendererStandard.h

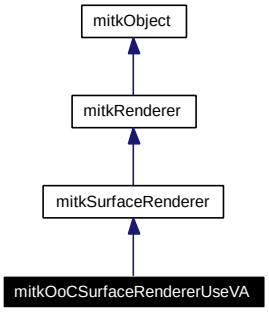
8.177 mitkOoCSurfaceRendererUseVA Class Reference

mitkOoCSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

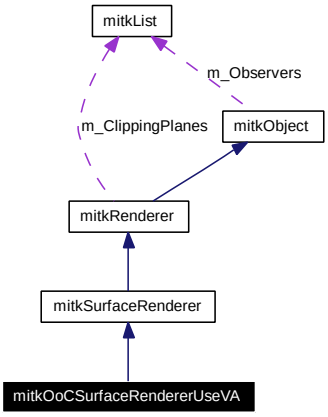
`#include <mitkOoCSurfaceRendererUseVA.h>`

Inherits mitkSurfaceRenderer.

Inheritance diagram for mitkOoCSurfaceRendererUseVA:



Collaboration diagram for mitkOoCSurfaceRendererUseVA:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.177.1 Detailed Description

mitkOoCSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

mitkOoCSurfaceRendererUseVA is a renderer for mitkSurfaceModel using vertex array

8.177.2 Member Function Documentation

8.177.2.1 virtual void mitkOoCSurfaceRendererUseVA::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkSurfaceRenderer.

The documentation for this class was generated from the following file:

- mitkOoCSurfaceRendererUseVA.h

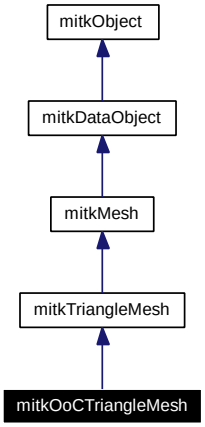
8.178 mitkOoCTriangleMesh Class Reference

mitkOoCTriangleMesh - a concrete class for out-of-core triangle meshes

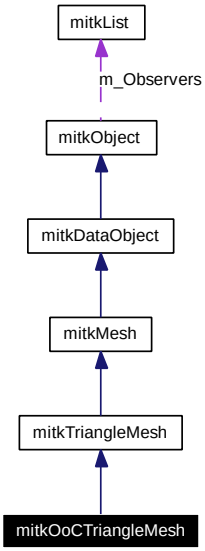
`#include <mitkOoCTriangleMesh.h>`

Inherits mitkTriangleMesh.

Inheritance diagram for mitkOoCTriangleMesh:



Collaboration diagram for mitkOoCTriangleMesh:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCTriangleMesh ()
- void SetVertexBlockSize (unsigned int vn)
- void SetFaceBlockSize (unsigned int fn)

- virtual int GetDataObjectType () const
- virtual void Initialize ()
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void SetVertexNumber (size_type number)
- virtual void SetFaceNumber (size_type number)
- virtual float * GetVertexData ()
- virtual index_type * GetFaceData ()
- virtual void ReverseNormals ()
- virtual bool TestClockwise ()
- void SetVertexBufferSize (size_type s)
- void SetFaceBufferSize (size_type s)
- void SetBufferedVertexBlockNum (unsigned int n)
- unsigned int GetBufferedVertexBlockNum ()
- void SetBufferedFaceBlockNum (unsigned int n)
- unsigned int GetBufferedFaceBlockNum ()
- void SetPathOfDiskBuffer (char const *path)
- char const * GetPathOfDiskBuffer ()

8.178.1 Detailed Description

mitkOoCTriangleMesh - a concrete class for out-of-core triangle meshes

mitkOoCTriangleMesh is a concrete implementation of triangle mesh, for representation of a very large (out-of-core) 3D object (can NOT be loaded to the main memory entirely).

See also:
mitkICTriangleMesh for in-core implementation of triangle mesh

8.178.2 Constructor & Destructor Documentation

8.178.2.1 mitkOoCTriangleMesh::mitkOoCTriangleMesh ()

Default constructor.

8.178.3 Member Function Documentation

8.178.3.1 virtual void mitkOoCTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

8.178.3.2 virtual unsigned long long mitkOoCTriangleMesh::GetActualMemorySize () const [virtual]

Get the actual size of the data in bytes.

Returns:
Return the actual size of the data in bytes.

Implements mitkDataObject.

8.178.3.3 unsigned int mitkOoCTriangleMesh::GetBufferedFaceBlockNum ()

Get the number of buffered face blocks in memory.

Returns:
Return the number of buffered face blocks.

8.178.3.4 unsigned int mitkOoCTriangleMesh::GetBufferedVertexBlockNum ()

Get the number of buffered vertex blocks in memory.

Returns:
Return the number of buffered vertex blocks.

8.178.3.5 virtual int mitkOoCTriangleMesh::GetDataObjectType () const [inline, virtual]

Return what type of data object this is.

Returns:
Return the type of this data object.

Reimplemented from mitkTriangleMesh.

8.178.3.6 virtual index_type* mitkOoCTriangleMesh::GetFaceData () [inline, virtual]

You can not get a permanent pointer to the entire face data since it can not be loaded to the main memory all together, so do not call this function when using a mitkOoCTriangleMesh object, and it will always return NULL.

Returns:
Return NULL.

Warning:
Do not call this function to get face data. Use mitkMesh::GetFace() or mitkMesh::GetTriangleFaces() instead.

Implements mitkMesh.

8.178.3.7 char const* mitkOoCTriangleMesh::GetPathOfDiskBuffer ()

Get the full path of the disk buffer which contains all the data of the mesh.

Returns:
Return the full path of the disk buffer.

8.178.3.8 virtual float* mitkOoCTriangleMesh::GetVertexData () [inline, virtual]

You can not get a permanent pointer to the entire vertex data since it can not be loaded to the main memory all together, so do not call this function when using a mitkOoCTriangleMesh object, and it will always return NULL.

Returns:
Return NULL.

Warning:
Do not call this function to get vertex data. Use mitkMesh::GetVertex() or mitkMesh::GetVertices() instead.

Implements mitkMesh.

8.178.3.9 virtual void mitkOoCTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkTriangleMesh.

8.178.3.10 virtual void mitkOoCTriangleMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTriangleMesh.

8.178.3.11 virtual void mitkOoCTriangleMesh::ReverseNormals () [virtual]

Reverse normals.

Reimplemented from mitkMesh.

8.178.3.12 void mitkOoCTriangleMesh::SetBufferedFaceBlockNum (unsigned int *n*)

Set the number of buffered face blocks in memory.

Parameters:
n the number of buffered face blocks

Note:
When calling this function, the buffer size is also calculated by (face block size) * (face block number), so it is unnecessary to call SetFaceBufferSize() again.

8.178.3.13 void mitkOoCTriangleMesh::SetBufferedVertexBlockNum (unsigned int *n*)

Set the number of buffered vertex blocks in memory.

Parameters:
n the number of buffered vertex blocks

Note:
When calling this function, the buffer size is also calculated by (vertex block size) * (vertex block number), so it is unnecessary to call SetVertexBufferSize() again.

8.178.3.14 void mitkOoCTriangleMesh::SetFaceBlockSize (unsigned int *fn*)

Set the size of the face block (number of triangles of one block).

Parameters:
fn the number of triangles of one block

8.178.3.15 void mitkOoCTriangleMesh::SetFaceBufferSize (size_type *s*)

Set the size of the memory buffer for containing cached face blocks.

Parameters:
s the size of the buffer in bytes

Note:
When calling this function, the number of buffered slice is also calculated by (buffer size) / (face block size), so it is unnecessary to call SetBufferedFaceBlockNum() again.

8.178.3.16 virtual void mitkOoCTriangleMesh::SetFaceNumber (size_type *number*) [inline, virtual]

Set the mesh's faces' number and allocate memory.

Parameters:
number the number of faces

Implements mitkMesh.

8.178.3.17 void mitkOoCTriangleMesh::SetPathOfDiskBuffer (char const * *path*)

Set the full path of the disk buffer to contain all the data of the mesh.

Parameters:
path the full path of the disk buffer

8.178.3.18 void mitkOoCTriangleMesh::SetVertexBlockSize (unsigned int *vn*)

Set the size of the vertex block (number of vertices of one block).

Parameters:
vn the number of vertices of one block

8.178.3.19 void mitkOoCTriangleMesh::SetVertexBufferSize (size_type *s*)

Set the size of the memory buffer for containing cached vertex blocks.

Parameters:
s the size of the buffer in bytes

Note:
When calling this function, the number of buffered vertex blocks is also calculated by (buffer size) / (vertex block size), so it is unnecessary to call SetBufferedVertexBlockNum() again.

8.178.3.20 virtual void mitkOoCTriangleMesh::SetVertexNumber (size_type *number*) [inline, virtual]

Set the mesh's vertices' number and allocate memory.

Parameters:
number the number of vertices

Implements mitkMesh.

8.178.3.21 virtual void mitkOoCTriangleMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

8.178.3.22 virtual bool mitkOoCTriangleMesh::TestClockwise () [virtual]

Test the orientation of front-facing triangles.

Returns:
Return true if the orientation of front-facing triangles is clockwise, otherwise return false.

Implements mitkMesh.

The documentation for this class was generated from the following file:

- mitkOoCTriangleMesh.h

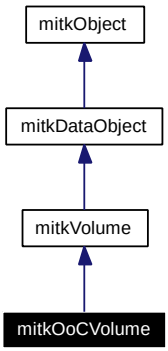
8.179 mitkOoCVolume Class Reference

mitkOoCVolume - a concrete data object to represent a out-of-core multi-dimensional medical image dataset

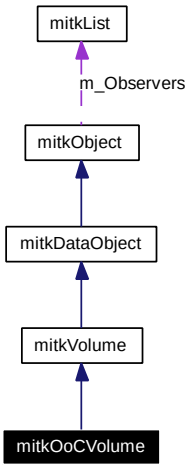
`#include <mitkOoCVolume.h>`

Inherits mitkVolume.

Inheritance diagram for mitkOoCVolume:



Collaboration diagram for mitkOoCVolume:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolume ()
- virtual int GetDataObjectType () const
- virtual void const * GetData () const
- virtual void * GetData ()
- virtual void FreezeSlice (int sliceIdx)
- virtual void UnFreezeSlice (int sliceIdx)
- virtual void const * GetSliceForRead (int sliceIdx)

- virtual void * GetSliceForWrite (int sliceIdx)
- virtual void * GetSliceForReadWrite (int sliceIdx)
- virtual bool ReadSliceData (int sliceIdx, void *dst)
- virtual bool ReadYZSliceData (int xIdx, void *dst)
- virtual bool ReadXZSliceData (int yIdx, void *dst)
- virtual bool GetArbitrarySlice (int w, int h, double o[3], double ux[3], double uy[3], void *dst)
- virtual bool WriteSliceData (int sliceIdx, void const *src)
- virtual bool ReadSubVolume (int x, int y, int z, int w, int h, int d, int &tw, int &th, int &td, void *dst)
- virtual bool WriteSubVolume (int x, int y, int z, int w, int h, int d, int &tw, int &th, int &td, void const *src)
- virtual bool Allocate ()
- virtual unsigned long long GetActualMemorySize () const
- virtual void Initialize ()
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- void SetSliceDataModified (int sliceIdx, bool modified=true)
- void SetMemoryBufferSize (size_type s)
- void SetBufferedSliceNum (unsigned int n)
- unsigned int GetBufferedSliceNum ()
- void SetPathOfDiskBuffer (char const *path)
- char const * GetPathOfDiskBuffer ()
- void SetNamePrefix (char const *prefix)
- char const * GetNamePrefix ()
- void SetKeepCacheFiles (bool keep=true)

8.179.1 Detailed Description

mitkOoCVolume - a concrete data object to represent a out-of-core multi-dimensional medical image dataset

mitkOoCVolume is a concrete data object to represent a out-of-core multi-dimensional medical image dataset.

8.179.2 Constructor & Destructor Documentation

8.179.2.1 mitkOoCVolume::mitkOoCVolume ()

Default constructor.

8.179.3 Member Function Documentation

8.179.3.1 virtual bool mitkOoCVolume::Allocate () [virtual]

Generate the buffer for the volume data on disk and allocate necessary space in memory for holding the buffered image data. The previous allocated data will be deleted if exists.

Returns:
Return true if successful, otherwise return false.

Implements mitkVolume.

8.179.3.2 virtual void mitkOoCVolume::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Reimplemented from mitkVolume.

8.179.3.3 virtual void mitkOoCVolume::FreezeSlice (int *sliceIdx*) [virtual]

Lock the physical memory containing sliceIdx'th slice data. (only useful for out-of-core volume data (mitkOoCVolume) to avoid swapping some slice data from memory buffer to disk buffer, i.e. to keep the returned pointer of some 'Get' functions, such as GetSliceData(), GetSliceForRead() and so on, always be valid until call UnFreezeSlice().

Parameters:
sliceIdx the index of the slice to freeze.

Reimplemented from mitkVolume.

8.179.3.4 virtual unsigned long long mitkOoCVolume::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by the buffered volume data. The unit is BYTE.

Returns:
Return the actual memory size occupied by this volume. The unit is BYTE.

Implements mitkDataObject.

8.179.3.5 virtual bool mitkOoCVolume::GetArbitrarySlice (int *w*, int *h*, double *o*[3], double *ux*[3], double *uy*[3], void * *dst*) [virtual]

Get arbitrary directional slice from the volume to a specified memory buffer (i.e. re-slicing).

Parameters:
w width of the new slice (in pixels)
h height of the new slice (in pixels)
o start position (left-bottom) of the new slice in the volume space
ux the step (in pixels) vector of moving to the next pixel along the x-direction of the new slice in the volume space
uy the step (in pixels) vector of moving to the next pixel along the y-direction of the new slice in the volume space
dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be w * h * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.179.3.6 unsigned int mitkOoCVolume::GetBufferedSliceNum () [inline]

Get the number of buffered slice in memory.

Returns:
Return the number of buffered slice.

8.179.3.7 virtual void* mitkOoCVolume::GetData () [inline, virtual]

You can not get a permanent pointer to the entire volume data since it can not be loaded to the main memory all together, so do not call this function when using a mitkOoCVolume object, and it will always return NULL.

Returns:
Return NULL.

Warning:
Do not call this function to get volume data. Use GetSliceData() or ReadSliceData() instead.

Implements mitkVolume.

8.179.3.8 virtual void const* mitkOoCVolume::GetData () const [inline, virtual]

You can not get a permanent pointer to the entire volume data since it can not be loaded to the main memory all together, so do not call this function when using a mitkOoCVolume object, and it will always return NULL.

Returns:
Return NULL.

Warning:
Do not call this function to get volume data. Use GetSliceData() or ReadSliceData() instead.

Implements mitkVolume.

8.179.3.9 virtual int mitkOoCVolume::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_OUT_OF_CORE_VOLUME

Reimplemented from mitkVolume.

8.179.3.10 char const* mitkOoCVolume::GetNamePrefix ()

Get the name prefix of the volume cache files. For quick restore of an out-of-core volume.

Returns:
Return the name prefix

8.179.3.11 char const* mitkOoCVolume::GetPathOfDiskBuffer ()

Get the full path of the disk buffer which contains all the slices of the volume.

Returns:
Return the full path of the disk buffer.

8.179.3.12 virtual void const* mitkOoCVolume::GetSliceForRead (int *sliceIdx*) [virtual]

Get data pointer of a specified slice in the volume (for read only).

Parameters:
sliceIdx the index of the slice to get (in [0, GetImageNum()-1])

Returns:
Return a const void pointer to data.

Note:
The returned type is void const *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function. The returned value is not a permanently valid pointer. After the next call of this function, the previous returned pointer could be invalid or point to another slice. Use this function carefully.

Implements mitkVolume.

8.179.3.13 virtual void* mitkOoCVolume::GetSliceForReadWrite (int *sliceIdx*) [virtual]

Get data pointer of a specified slice in the volume (for read & write).

Parameters:
sliceIdx the index of the slice to get (in [0, GetImageNum()-1])

Returns:
Return a const void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function. The returned value is not a permanently valid pointer. After the next call of this function, the previous returned pointer could be invalid or point to another slice. Use this function carefully.

Implements mitkVolume.

8.179.3.14 virtual void* mitkOoCVolume::GetSliceForWrite (int *sliceIdx*) [virtual]

Get data pointer of a specified slice in the volume (for write only).

Parameters:
sliceIdx the index of the slice to get (in [0, GetImageNum()-1])

Returns:
Return a const void pointer to data.

Note:
The returned type is void *, it should be converted to some useful data type according to the return value of GetDataType(). This function returns an empty buffer for writing the specified slice at FIRST time, and it will not load any data from the disk buffer even if the specified slice has already contained data in the disk buffer, so don't use it if you really want GetSliceForReadWrite().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function. The returned value is not a permanently valid pointer. After the next call of this function, the previous returned pointer could be invalid or point to another slice. Use this function carefully.

Implements mitkVolume.

8.179.3.15 virtual void mitkOoCVolume::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Reimplemented from mitkVolume.

8.179.3.16 virtual void mitkOoCVolume::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolume.

8.179.3.17 virtual bool mitkOoCVolume::ReadSliceData (int *sliceIdx*, void * *dst*) [virtual]

Copy slice data from the volume to a specified memory buffer.

Parameters:
sliceIdx the index of the slice to read (in [0, GetImageNum()-1])
dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.179.3.18 virtual bool mitkOoCVolume::ReadSubVolume (int *x*, int *y*, int *z*, int *w*, int *h*, int *d*, int & *tw*, int & *th*, int & *td*, void * *dst*) [virtual]

Copy a sub-volume (start position = (x, y, z), size = w * h * d) from the volume to a specified memory buffer.

Parameters:

x the x-coordinate of the start point

y the y-coordinate of the start point

z the z-coordinate of the start point

w the width of the sub-volume

h the height of the sub-volume

d the depth of the sub-volume

tw return the true width of the sub-volume copied (in case that x+w > GetWidth())

th return the true height of the sub-volume copied (in case that y+h > GetHeight())

td return the true depth of the sub-volume copied (in case that z+d > GetSliceNum())

dst the pointer to the destination memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of destination memory buffer should be w * h * d * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.179.3.19 virtual bool mitkOoCVolume::ReadXZSliceData (int *yIdx*, void * *dst*) [virtual]

Copy y-directional slice from the volume to a specified memory buffer.

Parameters:

yIdx the position of the y-directional slice to read (in [0, GetHeight()-1])

dst the pointer to the destination memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of destination memory buffer should be GetWidth() * GetImageNum() * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.179.3.20 virtual bool mitkOoCVolume::ReadYZSliceData (int *xIdx*, void * *dst*) [virtual]

Copy x-directional slice from the volume to a specified memory buffer.

Parameters:

xIdx the position of the x-directional slice to read (in [0, GetWidth()-1])

dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be GetHeight() ∗ GetImageNum() ∗ GetChannelNum() ∗ GetDataTypeSize().

Implements mitkVolume.

8.179.3.21 void mitkOoCVolume::SetBufferedSliceNum (unsigned int *n*)

Set the number of buffered slice in memory.

Parameters:
n the number of buffered slice

Note:
When calling this function, the buffer size is also calculated by (slice size) ∗ (slice number), so it is unnecessary to call SetMemoryBufferSize() again.

8.179.3.22 void mitkOoCVolume::SetKeepCacheFiles (bool *keep* = `true`) `[inline]`

Set whether or not to keep the cache files. To set it to true for quick restore of an out-of-core volume.

Parameters:
keep true for keep, false for delete

8.179.3.23 void mitkOoCVolume::SetMemoryBufferSize (size_type *s*)

Set the size of the memory buffer for containing cached slices.

Parameters:
s the size of the buffer in bytes

Note:
When calling this function, the number of buffered slice is also calculated by (buffer size) / (slice size), so it is unnecessary to call SetBufferedSliceNum() again.

8.179.3.24 void mitkOoCVolume::SetNamePrefix (char const ∗ *prefix*)

Set the name prefix of the volume cache files. For quick restore of an out-of-core volume.

Parameters:
prefix the name prefix to be set

8.179.3.25 void mitkOoCVolume::SetPathOfDiskBuffer (char const * *path*)

Set the full path of the disk buffer to contain all the slices of the volume.

Parameters:
path the full path of the disk buffer

8.179.3.26 void mitkOoCVolume::SetSliceDataModified (int *sliceIdx*, bool *modified* = `true`)

Set the modified flag of the sliceIdx'th slice.

Parameters:
sliceIdx the index of the slice to set

modified whether the slice data is modified

Note:
Call this function to ensure the specified slice written back to disk, when you use GetSliceData() to get the pointer to a slice data and change its content subsequently.

8.179.3.27 virtual void mitkOoCVolume::ShallowCopy (mitkDataObject * *src*) `[virtual]`

Warning:
Don't call this function directly.

Reimplemented from mitkVolume.

8.179.3.28 virtual void mitkOoCVolume::UnFreezeSlice (int *sliceIdx*) `[virtual]`

Unlock the physical memory containing sliceIdx'th slice data.

Parameters:
sliceIdx the index of the slice to un-freeze.

See also:
FreezeSlice()

Reimplemented from mitkVolume.

8.179.3.29 virtual bool mitkOoCVolume::WriteSliceData (int *sliceIdx*, void const * *src*) `[virtual]`

Copy slice data from a specified memory buffer to the volume.

Parameters:
sliceIdx the index of the slice to write (in [0, GetImageNum()-1])

src the pointer to the source memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of source memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

8.179.3.30 virtual bool mitkOoCVolume::WriteSubVolume (int *x*, int *y*, int *z*, int *w*, int *h*, int *d*, int & *tw*, int & *th*, int & *td*, void const * *src*) [virtual]

Copy a sub-volume (start position = (x, y, z), size = w * h * d) from a specified memory buffer to the volume.

Parameters:
- ***x*** the x-coordinate of the start point
- ***y*** the y-coordinate of the start point
- ***z*** the z-coordinate of the start point
- ***w*** the width of the sub-volume
- ***h*** the height of the sub-volume
- ***d*** the depth of the sub-volume
- ***tw*** return the true width of the sub-volume copied (in case that x+w > GetWidth())
- ***th*** return the true height of the sub-volume copied (in case that y+h > GetHeight())
- ***td*** return the true depth of the sub-volume copied (in case that z+d > GetSliceNum())
- ***src*** the pointer to the source memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of source memory buffer should be w * h * d * GetChannelNum() * GetDataTypeSize().

Implements mitkVolume.

The documentation for this class was generated from the following file:

- mitkOoCVolume.h

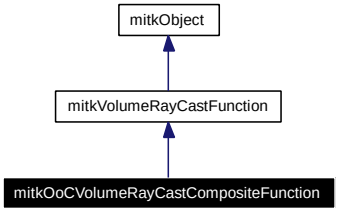
8.180 mitkOoCVolumeRayCastCompositeFunction Class Reference

mitkOoCVolumeRayCastCompositeFunction - a concrete ray cast function for compositing

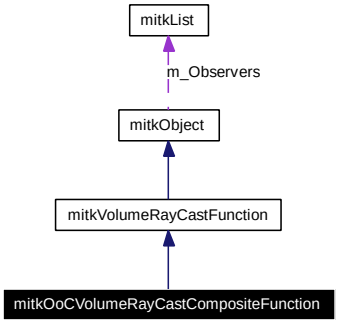
`#include <mitkOoCVolumeRayCastCompositeFunction.h>`

Inherits mitkVolumeRayCastFunction.

Inheritance diagram for mitkOoCVolumeRayCastCompositeFunction:



Collaboration diagram for mitkOoCVolumeRayCastCompositeFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRayCastCompositeFunction ()
- virtual void CastRay (mitkRay *rayInfo)

8.180.1 Detailed Description

mitkOoCVolumeRayCastCompositeFunction - a concrete ray cast function for compositing

mitkOoCVolumeRayCastCompositeFunction is a concrete ray cast function by using the composition. It implements the CastRay() function defined in base class and support out-of-core data sets.

8.180.2 Constructor & Destructor Documentation

8.180.2.1 mitkOoCVolumeRayCastCompositeFunction::mitkOoCVolumeRayCastComposite-Function ()

Default constructor.

8.180.3 Member Function Documentation

8.180.3.1 virtual void mitkOoCVolumeRayCastCompositeFunction::CastRay (mitkRay * *rayInfo*) [virtual]

Calculate the ray cast function using composition.

Parameters:

rayInfo Specify the ray information required by the calcualtion.

Implements mitkVolumeRayCastFunction.

8.180.3.2 virtual void mitkOoCVolumeRayCastCompositeFunction::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeRayCastFunction.

The documentation for this class was generated from the following file:

- mitkOoCVolumeRayCastCompositeFunction.h

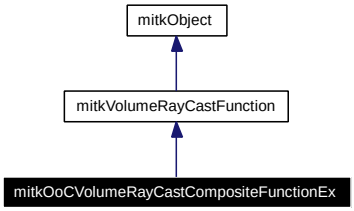
8.181 mitkOoCVolumeRayCastCompositeFunctionEx Class Reference

mitkOoCVolumeRayCastCompositeFunctionEx - a concrete ray cast function for compositing

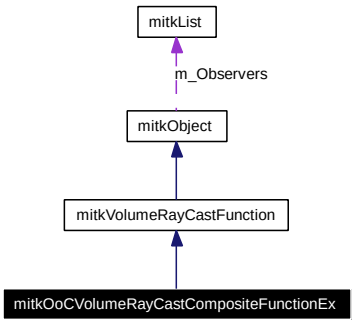
`#include <mitkOoCVolumeRayCastCompositeFunctionEx.h>`

Inherits mitkVolumeRayCastFunction.

Inheritance diagram for mitkOoCVolumeRayCastCompositeFunctionEx:



Collaboration diagram for mitkOoCVolumeRayCastCompositeFunctionEx:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRayCastCompositeFunctionEx ()
- virtual void CastRay (mitkRay *rayInfo)

8.181.1 Detailed Description

mitkOoCVolumeRayCastCompositeFunctionEx - a concrete ray cast function for compositing

mitkOoCVolumeRayCastCompositeFunctionEx is a concrete ray cast function by using the composition. It implements the CastRay() function defined in base class and support out-of-core data sets.

8.181.2 Constructor & Destructor Documentation

8.181.2.1 mitkOoCVolumeRayCastCompositeFunctionEx::mitkOoCVolumeRayCastCompositeFunctionEx ()

Default constructor.

8.181.3 Member Function Documentation

8.181.3.1 virtual void mitkOoCVolumeRayCastCompositeFunctionEx::CastRay (mitkRay * *rayInfo*) [virtual]

Calculate the ray cast function using composition.

Parameters:

rayInfo Specify the ray information required by the calcualtion.

Implements mitkVolumeRayCastFunction.

8.181.3.2 virtual void mitkOoCVolumeRayCastCompositeFunctionEx::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeRayCastFunction.

The documentation for this class was generated from the following file:

- mitkOoCVolumeRayCastCompositeFunctionEx.h

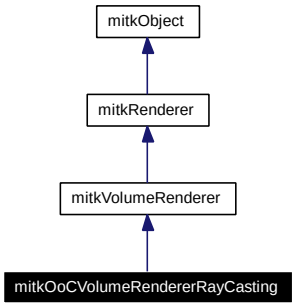
8.182 mitkOoCVolumeRendererRayCasting Class Reference

mitkOoCVolumeRendererRayCasting - a concrete volume renderer for rendering an out-of-core volume

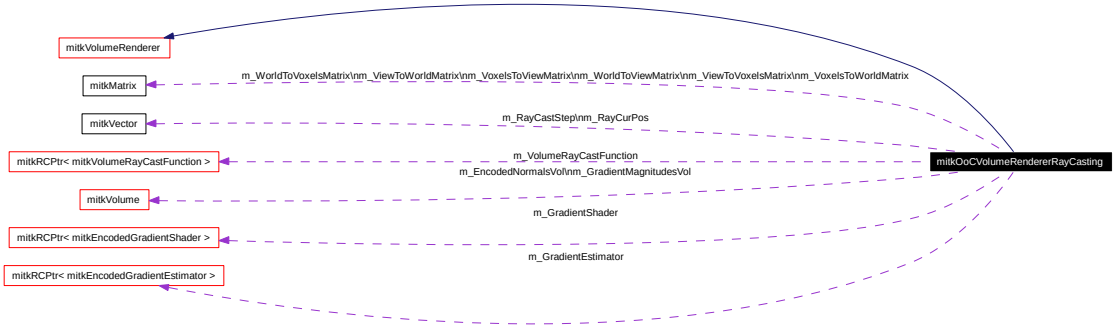
`#include <mitkOoCVolumeRendererRayCasting.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkOoCVolumeRendererRayCasting:



Collaboration diagram for mitkOoCVolumeRendererRayCasting:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRendererRayCasting ()
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetGradientEstimator (mitkEncodedGradientEstimator *gradest)
- mitkEncodedGradientEstimator * GetGradientEstimator ()

8.182.1 Detailed Description

mitkOoCVolumeRendererRayCasting - a concrete volume renderer for rendering an out-of-core volume

mitkOoCVolumeRendererRayCasting is a concrete volume renderer for rendering an out-of-core volume which cannot be loaded into the main memory entirely. It provides an improved algorithm for rendering

such large data sets and gets much faster rendering speed than directly applying conventional volume ray casting method on the out-of-core data sets.

8.182.2 Constructor & Destructor Documentation

8.182.2.1 mitkOoCVolumeRendererRayCasting::mitkOoCVolumeRendererRayCasting ()

Default constructor.

8.182.3 Member Function Documentation

8.182.3.1 mitkEncodedGradientEstimator* mitkOoCVolumeRendererRayCasting::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals

Returns:
Return the gradient estimator used to estimate normals.

8.182.3.2 virtual void mitkOoCVolumeRendererRayCasting::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.182.3.3 virtual int mitkOoCVolumeRendererRayCasting::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.182.3.4 void mitkOoCVolumeRendererRayCasting::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals

Parameters:
gradest Represent the gradient estimator used to estimate normals

The documentation for this class was generated from the following file:

- mitkOoCVolumeRendererRayCasting.h

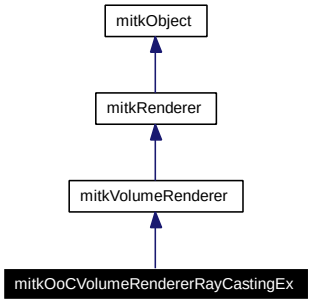
8.183 mitkOoCVolumeRendererRayCastingEx Class Reference

mitkOoCVolumeRendererRayCastingEx - a concrete volume renderer for rendering an out-of-core volume

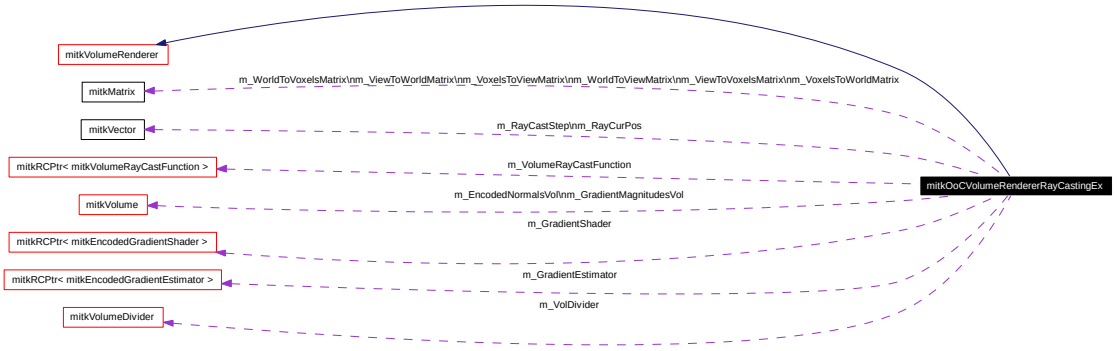
`#include <mitkOoCVolumeRendererRayCastingEx.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkOoCVolumeRendererRayCastingEx:



Collaboration diagram for mitkOoCVolumeRendererRayCastingEx:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRendererRayCastingEx ()
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetSubVolumeSize (int wmin, int wmax, int hmin, int hmax, int dmin, int dmax)
- void SetSubVolumeSize (int sizemin[3], int sizemax[3])

8.183.1 Detailed Description

mitkOoCVolumeRendererRayCastingEx - a concrete volume renderer for rendering an out-of-core volume

mitkOoCVolumeRendererRayCastingEx is a concrete volume renderer for rendering an out-of-core volume which cannot be loaded into the main memory entirely. It provides an improved algorithm for rendering

such large data sets and gets much faster rendering speed than directly applying conventional volume ray casting method on the out-of-core data sets.

8.183.2 Constructor & Destructor Documentation

8.183.2.1 mitkOoCVolumeRendererRayCastingEx::mitkOoCVolumeRendererRayCastingEx ()

Default constructor.

8.183.3 Member Function Documentation

8.183.3.1 virtual void mitkOoCVolumeRendererRayCastingEx::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.183.3.2 virtual int mitkOoCVolumeRendererRayCastingEx::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.183.3.3 void mitkOoCVolumeRendererRayCastingEx::SetSubVolumeSize (int *sizemin*[3], int *sizemax*[3])

Set the sub-volume size for divider.

Parameters:

sizemin[0] the minimum size of sub-volume's width

sizemax[0] the maximum size of sub-volume's width

sizemin[1] the minimum size of sub-volume's height

sizemax[1] the maximum size of sub-volume's height

sizemin[2] the minimum size of sub-volume's depth

sizemax[2] the maximum size of sub-volume's depth

8.183.3.4 void mitkOoCVolumeRendererRayCastingEx::SetSubVolumeSize (int *wmin*, int *wmax*, int *hmin*, int *hmax*, int *dmin*, int *dmax*)

Set the sub-volume size for divider.

Parameters:

wmin the minimum size of sub-volume's width

wmax the maximum size of sub-volume's width

hmin the minimum size of sub-volume's height

hmax the maximum size of sub-volume's height

dmin the minimum size of sub-volume's depth

dmax the maximum size of sub-volume's depth

The documentation for this class was generated from the following file:

- mitkOoCVolumeRendererRayCastingEx.h

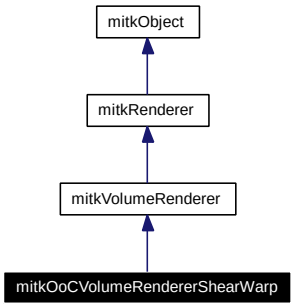
8.184 mitkOoCVolumeRendererShearWarp Class Reference

mitkVolumeRendererShearWarp - a concrete volume renderer for rendering a volume

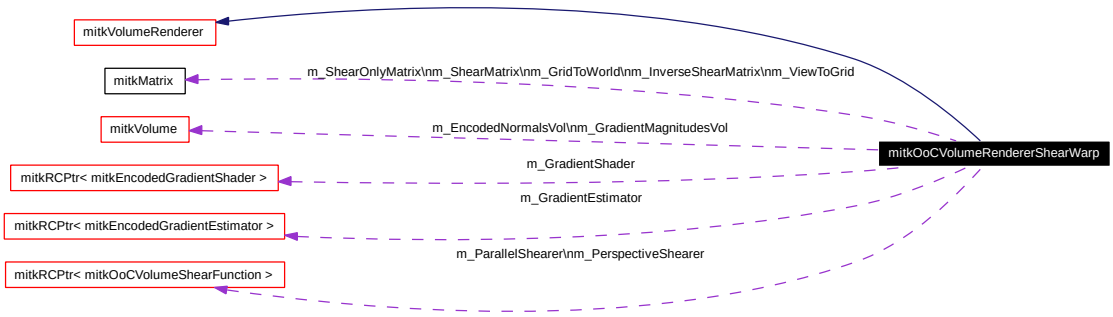
`#include <mitkOoCVolumeRendererShearWarp.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkOoCVolumeRendererShearWarp:



Collaboration diagram for mitkOoCVolumeRendererShearWarp:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetGradientEstimator (mitkEncodedGradientEstimator *gradest)
- mitkEncodedGradientEstimator * GetGradientEstimator ()
- mitkEncodedGradientShader * GetEncodedGradientShader ()
- void SetPerspectiveShearer (mitkOoCVolumeShearFunction *shearer)
- void SetParallelShearer (mitkOoCVolumeShearFunction *shearer)
- mitkOoCVolumeShearFunction * GetPerspectiveShearer ()
- mitkOoCVolumeShearFunction * GetParallelShearer ()
- void SetImageSampleDistance (float val)
- float GetImageSampleDistance ()
- float GetMinimumImageSampleDistance ()
- float GetMaximumImageSampleDistance ()
- bool GetAutoAdjustImageSampleDistances ()

- void AutoAdjustImageSampleDistancesOn ()
- void AutoAdjustImageSampleDistancesOff ()
- virtual float GetGradientMagnitudeScale ()
- virtual float GetGradientMagnitudeBias ()
- void SetModeIntegral ()
- void SetModeMop ()
- int GetMode ()
- void SetLevelOfDetail (bool lod)

8.184.1 Detailed Description

mitkVolumeRendererShearWarp - a concrete volume renderer for rendering a volume

mitkVolumeRendererShearWarp is a concrete volume renderer for rendering a volume using shear-warp technique.

> Our implemented algorithm references to:
> [1]Lacroute P, Levoy M. Fast volume rendering using a shear-warp factorization of the viewing transformation. Computer Graphics Proceedings, 1994.
> [2]Lacroute P. Fast volume rendering using a shear-warp factorization of the viewing transformation[R]. CSL-TR-95- 678,Stanford University,1995.

> However, this class just implements brute-force shear-warp algorithm, that is, it does not use run-length encoding technique as Lacroute.

Its rendering speed is generally faster than mitkVolumeRendererRayCasting and mitkVolumeRendererSplatting, and the rendering effect is between them.

The default perspective shearer is the object of mitkVolumeShearPerspective, and the default parallel shearer is the object of mitkVolumeShearParallel. Particular perspective and parallel shearers are also allowed by calling SetPerspectiveShearer() and SetParallelShearer().

Two rendering modes are alternative, one is integral mode, and the other is MOP mode, you can switch the mode using SetModeIntegral() and SetModeMop().

8.184.2 Member Function Documentation

8.184.2.1 void mitkOoCVolumeRendererShearWarp::AutoAdjustImageSampleDistancesOff () [inline]

Set image sample distances auto-adjusting off.

8.184.2.2 void mitkOoCVolumeRendererShearWarp::AutoAdjustImageSampleDistancesOn () [inline]

Set image sample distances auto-adjusting on.

8.184.2.3 bool mitkOoCVolumeRendererShearWarp::GetAutoAdjustImageSampleDistances () [inline]

Get whether to auto-adjust the image sample distances.

Returns:
Return a value which indicates whether to auto-adjust the sample distances (1 means true, 0 means false).

8.184.2.4 mitkEncodedGradientShader* mitkOoCVolumeRendererShearWarp::GetEncodedGradientShader () [inline]

Get the gradient shader.

Returns:
Return the gradient shader.

8.184.2.5 mitkEncodedGradientEstimator* mitkOoCVolumeRendererShearWarp::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals.

Returns:
Return the gradient estimator used to estimate normals.

8.184.2.6 virtual float mitkOoCVolumeRendererShearWarp::GetGradientMagnitudeBias () [virtual]

Get gradient magnitude bias.

Returns:
Return the gradient magnitude bias.

Reimplemented from mitkVolumeRenderer.

8.184.2.7 virtual float mitkOoCVolumeRendererShearWarp::GetGradientMagnitudeScale () [virtual]

Get gradient magnitude scale.

Returns:
Return the gradient magnitude scale.

Reimplemented from mitkVolumeRenderer.

8.184.2.8 float mitkOoCVolumeRendererShearWarp::GetImageSampleDistance () [inline]

Get the sample distance in the image plane.

Returns:
Return the image sample distance in the image plane.

8.184.2.9 float mitkOoCVolumeRendererShearWarp::GetMaximumImageSampleDistance () [inline]

Get the maximum image sample distance in the image plane.

Returns:
Return the maximum sample distance in the image plane.

8.184.2.10 float mitkOoCVolumeRendererShearWarp::GetMinimumImageSampleDistance () [inline]

Get the minimum image sample distance in the image plane.

Returns:
Return the minimum sample distance in the image plane.

8.184.2.11 int mitkOoCVolumeRendererShearWarp::GetMode () [inline]

Get the volume rendering mode.

Returns:
Return 0 if the volume rendering mode is integral projection. Return 1 if the volume rendering mode is MOP(maximum opacity projection).

8.184.2.12 mitkOoCVolumeShearFunction* mitkOoCVolumeRendererShearWarp::GetParallelShearer ()

Get the Parallel shearer.

Returns:
Return the current Parallel shearer.

8.184.2.13 mitkOoCVolumeShearFunction* mitkOoCVolumeRendererShearWarp::GetPerspectiveShearer ()

Get the perspective shearer.

Returns:
Return the current perspective shearer.

8.184.2.14 virtual void mitkOoCVolumeRendererShearWarp::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.184.2.15 virtual int mitkOoCVolumeRendererShearWarp::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.184.2.16 void mitkOoCVolumeRendererShearWarp::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*) [inline]

Set the gradient estimator used to estimate normals.

Parameters:

gradest Represent the gradient estimator used to estimate normals.

8.184.2.17 void mitkOoCVolumeRendererShearWarp::SetImageSampleDistance (float *val*) [inline]

Set the sample distance in the image plane.

Parameters:

val Specify the sample distance in the image plane.

8.184.2.18 void mitkOoCVolumeRendererShearWarp::SetLevelOfDetail (bool *lod*) [inline]

Set level-of-detail display.

Parameters:

lod The status of level-of-detail. If lod is true, it will display level-of-detail volume rendering image, that is, the image sampling rate is enlarged when the scene is scaled so that you will get a more detailed image. however, the rendering speed will be slowed accordingly. If lod is false, it will display volume rendering image with the same detail, that is, the image sampling rate will not be changed.

Note:

In default, it renders volume without level-of-detail.

8.184.2.19 void mitkOoCVolumeRendererShearWarp::SetModeIntegral () [inline]

Set volume rendering mode to be integral projection.

8.184.2.20 void mitkOoCVolumeRendererShearWarp::SetModeMop () [inline]

Set volume rendering mode to be MOP(maximum opacity projection).

Note:

Each image pixel stands for maximum opacity element in the ray direction, and when opacity is linear with intensity, MOP is equal to MIP(maximum intensity projection).

8.184.2.21 void mitkOoCVolumeRendererShearWarp::SetParallelShearer (mitkOoCVolumeShearFunction * *shearer*)

Set Parallel shearer to shear and composite the volume.

Parameters:
shearer Represent the Parallel shearer.

8.184.2.22 void mitkOoCVolumeRendererShearWarp::SetPerspectiveShearer (mitkOoCVolumeShearFunction * *shearer*)

Set perspective shearer to shear and composite the volume.

Parameters:
shearer Represent the perspective shearer.

The documentation for this class was generated from the following file:

- mitkOoCVolumeRendererShearWarp.h

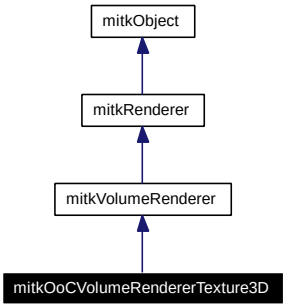
8.185 mitkOoCVolumeRendererTexture3D Class Reference

mitkOoCVolumeRendererTexture3D - a concrete volume renderer for rendering an out-of-core volume

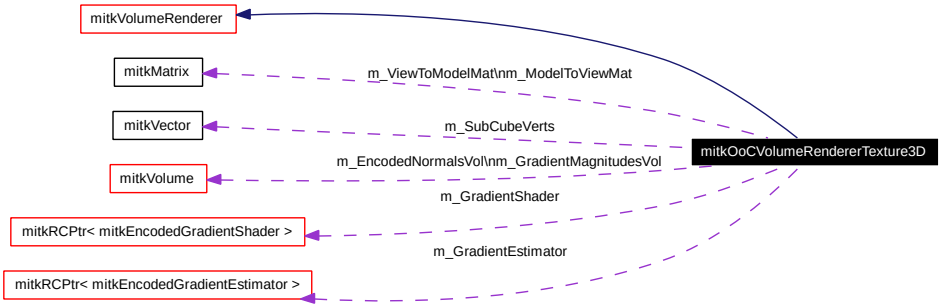
`#include <mitkOoCVolumeRendererTexture3D.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkOoCVolumeRendererTexture3D:



Collaboration diagram for mitkOoCVolumeRendererTexture3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRendererTexture3D ()
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetSampleDistance (float sd)
- void SetSubVolumeSize (int w, int h, int d)
- void SetGradientEstimator (mitkEncodedGradientEstimator *gradest)
- mitkEncodedGradientEstimator * GetGradientEstimator ()

8.185.1 Detailed Description

mitkOoCVolumeRendererTexture3D - a concrete volume renderer for rendering an out-of-core volume

mitkOoCVolumeRendererTexture3D is a concrete volume renderer for rendering an out-of-core volume by using 3D texture acceleration.

Note:
To use this class, your graphics card should support 3D texture.

8.185.2 Constructor & Destructor Documentation

8.185.2.1 mitkOoCVolumeRendererTexture3D::mitkOoCVolumeRendererTexture3D ()

Default constructor.

8.185.3 Member Function Documentation

8.185.3.1 mitkEncodedGradientEstimator* mitkOoCVolumeRendererTexture3D::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals

Returns:
Return the gradient estimator used to estimate normals.

8.185.3.2 virtual void mitkOoCVolumeRendererTexture3D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.185.3.3 virtual int mitkOoCVolumeRendererTexture3D::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.185.3.4 void mitkOoCVolumeRendererTexture3D::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals

Parameters:
gradest Represent the gradient estimator used to estimate normals

8.185.3.5 void mitkOoCVolumeRendererTexture3D::SetSampleDistance (float *sd*) [inline]

Set sample distance.

Parameters:
sd sample distance

8.185.3.6 void mitkOoCVolumeRendererTexture3D::SetSubVolumeSize (int *w*, int *h*, int *d*)

Set the size of the sub-volume. It must be smaller than the max 3D texture size supported by your 3D graphics accelerator card. To get a faster render speed, a 'd' <= buffered slice number of the out-of-core volume is strongly recommended.

Parameters:

w the width of the sub-volume

h the height of the sub-volume

d the depth of the sub-volume

The documentation for this class was generated from the following file:

- mitkOoCVolumeRendererTexture3D.h

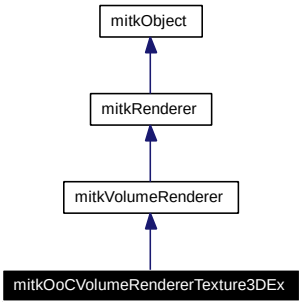
8.186 mitkOoCVolumeRendererTexture3DEx Class Reference

mitkOoCVolumeRendererTexture3DEx - a concrete volume renderer for rendering an out-of-core volume

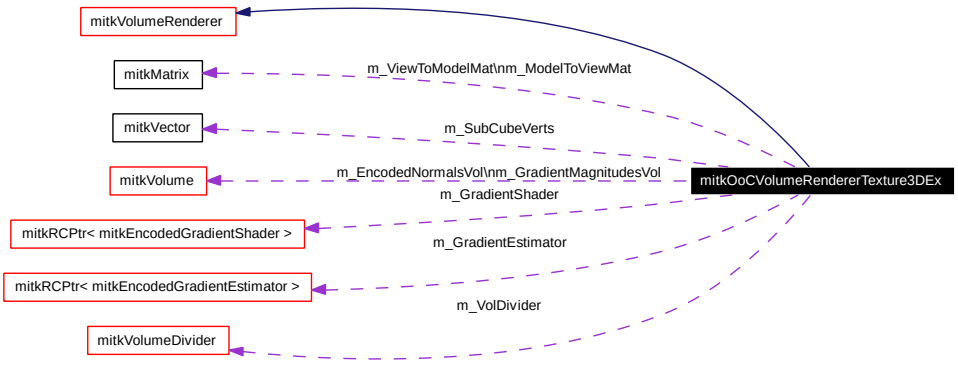
`#include <mitkOoCVolumeRendererTexture3DEx.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkOoCVolumeRendererTexture3DEx:



Collaboration diagram for mitkOoCVolumeRendererTexture3DEx:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRendererTexture3DEx ()
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetSampleDistance (float sd)
- void SetSubVolumeSize (int wmin, int wmax, int hmin, int hmax, int dmin, int dmax)
- void SetSubVolumeSize (int sizemin[3], int sizemax[3])

8.186.1 Detailed Description

mitkOoCVolumeRendererTexture3DEx - a concrete volume renderer for rendering an out-of-core volume

mitkOoCVolumeRendererTexture3DEx is a concrete volume renderer for rendering an out-of-core volume by using 3D texture acceleration.

Note:
To use this class, your graphics card should support 3D texture.

8.186.2 Constructor & Destructor Documentation

8.186.2.1 mitkOoCVolumeRendererTexture3DEx::mitkOoCVolumeRendererTexture3DEx ()

Default constructor.

8.186.3 Member Function Documentation

8.186.3.1 virtual void mitkOoCVolumeRendererTexture3DEx::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.186.3.2 virtual int mitkOoCVolumeRendererTexture3DEx::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.186.3.3 void mitkOoCVolumeRendererTexture3DEx::SetSampleDistance (float *sd*) [inline]

Set sample distance.

Parameters:
sd sample distance

8.186.3.4 void mitkOoCVolumeRendererTexture3DEx::SetSubVolumeSize (int *sizemin*[3], int *sizemax*[3])

Set the sub-volume size for divider.

Parameters:
sizemin[0] the minimum size of sub-volume's width
sizemax[0] the maximum size of sub-volume's width
sizemin[1] the minimum size of sub-volume's height
sizemax[1] the maximum size of sub-volume's height
sizemin[2] the minimum size of sub-volume's depth
sizemax[2] the maximum size of sub-volume's depth

8.186.3.5 void mitkOoCVolumeRendererTexture3DEx::SetSubVolumeSize (int *wmin*, int *wmax*, int *hmin*, int *hmax*, int *dmin*, int *dmax*)

Set the sub-volume size for divider.

Parameters:

wmin the minimum size of sub-volume's width

wmax the maximum size of sub-volume's width

hmin the minimum size of sub-volume's height

hmax the maximum size of sub-volume's height

dmin the minimum size of sub-volume's depth

dmax the maximum size of sub-volume's depth

The documentation for this class was generated from the following file:

- mitkOoCVolumeRendererTexture3DEx.h

8.187 mitkOoCVolumeShearFunction Class Reference

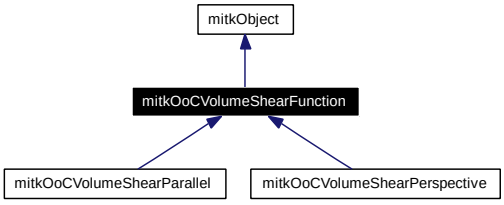
mitkOoCVolumeShearFunction - abstract class defines interface for volume shear

`#include <mitkOoCVolumeShearFunction.h>`

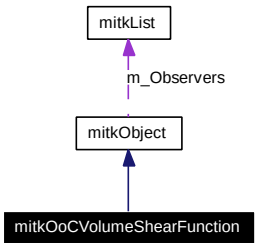
Inherits mitkObject.

Inherited by mitkOoCVolumeShearParallel, and mitkOoCVolumeShearPerspective.

Inheritance diagram for mitkOoCVolumeShearFunction:



Collaboration diagram for mitkOoCVolumeShearFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeShearFunction ()
- virtual bool IsParallel ()
- virtual void Shear (mitkShear *shearInform)=0

8.187.1 Detailed Description

mitkOoCVolumeShearFunction - abstract class defines interface for volume shear

mitkOoCVolumeShearFunction is an abstract class that defines interface for both parallel and perspective shear. The mitkOoCVolumeRendererShearWarp class provides common information for shear, and then calls "Shear()" to implement volume shear progress concretely.

8.187.2 Constructor & Destructor Documentation

8.187.2.1 mitkOoCVolumeShearFunction::mitkOoCVolumeShearFunction ()

The default constructor.

8.187.3 Member Function Documentation

8.187.3.1 virtual bool mitkOoCVolumeShearFunction::IsParallel () [inline, virtual]

Get the projection status of the class.

Returns:
Return true, means that this class is used for parallel shear-warp. Return false, means that this class is used for perspective shear-warp.

Note:
All of its subclasses must implement this virtual to indicate its status.

Reimplemented in mitkOoCVolumeShearParallel, and mitkOoCVolumeShearPerspective.

8.187.3.2 virtual void mitkOoCVolumeShearFunction::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkOoCVolumeShearParallel, and mitkOoCVolumeShearPerspective.

8.187.3.3 virtual void mitkOoCVolumeShearFunction::Shear (mitkShear * *shearInform*) [pure virtual]

The real function to shear the volume.

Note:
All of its subclasses must implement this pure virtual to shear the volume.

Implemented in mitkOoCVolumeShearParallel, and mitkOoCVolumeShearPerspective.

The documentation for this class was generated from the following file:

- mitkOoCVolumeShearFunction.h

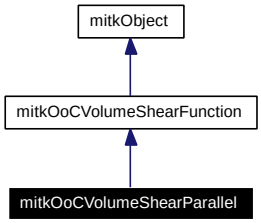
8.188 mitkOoCVolumeShearParallel Class Reference

mitkOoCVolumeShearParallel -

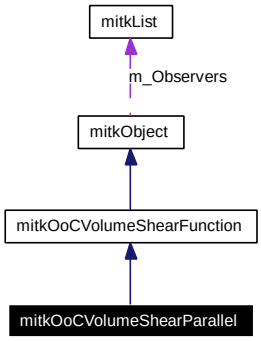
`#include <mitkOoCVolumeShearParallel.h>`

Inherits mitkOoCVolumeShearFunction.

Inheritance diagram for mitkOoCVolumeShearParallel:



Collaboration diagram for mitkOoCVolumeShearParallel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual bool IsParallel ()
- virtual void Shear (mitkShear *shearInform)

8.188.1 Detailed Description

mitkOoCVolumeShearParallel -

mitkOoCVolumeShearParallel

8.188.2 Member Function Documentation

8.188.2.1 virtual bool mitkOoCVolumeShearParallel::IsParallel () [inline, virtual]

Get the projection status of the class.

Returns:
Return true, means that this class is used for parallel shear-warp. Return false, means that this class is used for perspective shear-warp.

Note:
All of its subclasses must implement this virtual to indicate its status.

Reimplemented from mitkOoCVolumeShearFunction.

8.188.2.2 virtual void mitkOoCVolumeShearParallel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkOoCVolumeShearFunction.

8.188.2.3 virtual void mitkOoCVolumeShearParallel::Shear (mitkShear * *shearInform*) [virtual]

The real function to shear the volume.

Note:
All of its subclasses must implement this pure virtual to shear the volume.

Implements mitkOoCVolumeShearFunction.

The documentation for this class was generated from the following file:

- mitkOoCVolumeShearParallel.h

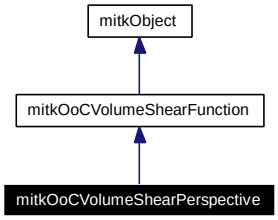
8.189 mitkOoCVolumeShearPerspective Class Reference

mitkOoCVolumeShearPerspective -

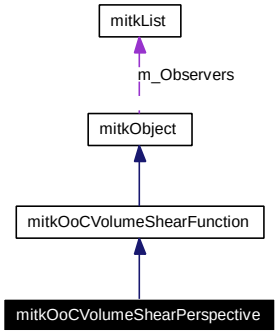
`#include <mitkOoCVolumeShearPerspective.h>`

Inherits mitkOoCVolumeShearFunction.

Inheritance diagram for mitkOoCVolumeShearPerspective:



Collaboration diagram for mitkOoCVolumeShearPerspective:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual bool IsParallel ()
- virtual void Shear (mitkShear *shearInform)

8.189.1 Detailed Description

mitkOoCVolumeShearPerspective -

mitkOoCVolumeShearPerspective

8.189.2 Member Function Documentation

8.189.2.1 virtual bool mitkOoCVolumeShearPerspective::IsParallel () [inline, virtual]

Get the projection status of the class.

Returns:
Return true, means that this class is used for parallel shear. Return false, means that this class is used for perspective shear.

Note:
All of its subclasses must implement this virtual to indicate its status.

Reimplemented from mitkOoCVolumeShearFunction.

8.189.2.2 virtual void mitkOoCVolumeShearPerspective::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkOoCVolumeShearFunction.

8.189.2.3 virtual void mitkOoCVolumeShearPerspective::Shear (mitkShear * *shearInform*) [virtual]

The real function to splat the volume.

Note:
All of its subclasses must implement this pure virtual to splat the volume.

Implements mitkOoCVolumeShearFunction.

The documentation for this class was generated from the following file:

- mitkOoCVolumeShearPerspective.h

8.190 mitkOptimizer Class Reference

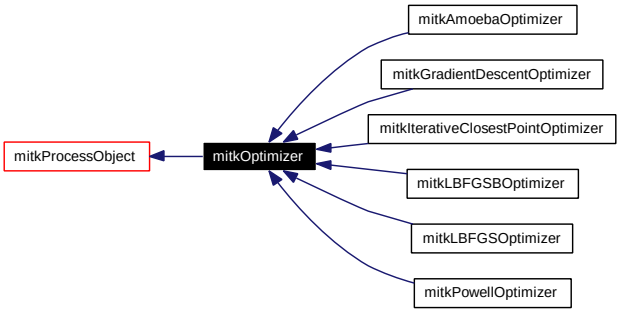
mitkOptimizer - an abstract class specifies interface for an optimization method

`#include <mitkOptimizer.h>`

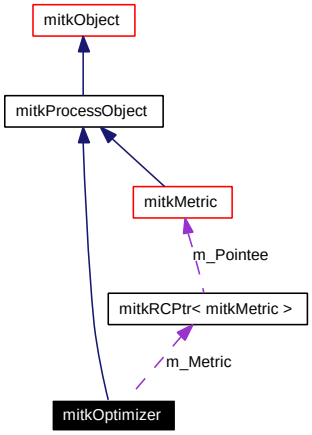
Inherits mitkProcessObject.

Inherited by mitkAmoebaOptimizer, mitkGradientDescentOptimizer, mitkIterativeClosestPointOptimizer, mitkLBFGSBOptimizer, mitkLBFGSOptimizer, and mitkPowellOptimizer.

Inheritance diagram for mitkOptimizer:



Collaboration diagram for mitkOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void SetInitialParameters (VectorParameterType &param)
- virtual VectorParameterType & GetInitialParameters ()
- void SetScales (VectorParameterType &scales)
- void SetScales (ScalarParameterType *scales, unsigned int length)
- VectorParameterType & GetScales ()
- virtual VectorParameterType & GetLastParameters ()
- void SetMetric (mitkMetric *metric)
- mitkMetric * GetMetric ()

- void SetMaxIterations (unsigned int n)
- unsigned int GetMaxIterationNumber ()
- virtual void Update ()
- void SetNumberOfParameters (unsigned int num)
- unsigned int GetNumberOfParameters ()
- void SetSpaceDimension (unsigned int dim)
- unsigned int GetSpaceDimension ()
- void SetCurrentIterationNumber (unsigned int iterations)
- unsigned int GetCurrentIterationNumber ()
- VectorParameterType & GetCurrentPosition ()
- void GetCurrentPosition (double *currentPosition)
- void SetMaximizeFlag (bool flag)
- bool GetMaximizeFlag ()
- void SetStopFlag (bool flag)
- bool GetStopFlag ()
- void StopOptimization (StopCondition condition)

8.190.1 Detailed Description

mitkOptimizer - an abstract class specifies interface for an optimization method

mitkOptimizer is an abstract class specifies interface for an optimization method. It is not intended to be instantiated. This class is a base for a hierarchy of optimizers.

8.190.2 Member Function Documentation

8.190.2.1 unsigned int mitkOptimizer::GetCurrentIterationNumber () [inline]

Get current iteration number.

Returns:
Current iteration number.

8.190.2.2 void mitkOptimizer::GetCurrentPosition (double * *currentPosition*)

Get parameters in current iteration. currentPosition An array to store the parameters.

8.190.2.3 VectorParameterType& mitkOptimizer::GetCurrentPosition () [inline]

Get the pointer of parameters vector in current iteration.

Returns:
The pointer to parameters vector.

8.190.2.4 virtual VectorParameterType& mitkOptimizer::GetInitialParameters () [virtual]

Get the position to initialize the optimization.

Returns:
Return the vector pointer to the initial position for optimization.

8.190.2.5 virtual VectorParameterType& mitkOptimizer::GetLastParameters () [virtual]

Get the last position of optimization.

Returns:
Return the vector pointer to the last position.

8.190.2.6 bool mitkOptimizer::GetMaximizeFlag () [inline]

Get the maximize flag.

Returns:
The maximize flag.

8.190.2.7 unsigned int mitkOptimizer::GetMaxIterationNumber () [inline]

Get the maximum number of iterations.

Returns:
The maximum number of iterations.

8.190.2.8 mitkMetric* mitkOptimizer::GetMetric ()

Get the Metric

Returns:
Return the pointer to the Metric.

8.190.2.9 unsigned int mitkOptimizer::GetNumberOfParameters () [inline]

Get the number of parameters.

Returns:
The number of parameters.

8.190.2.10 VectorParameterType& mitkOptimizer::GetScales ()

Get current parameters scaling.

Returns:
Return the pointer to scale vector.

8.190.2.11 unsigned int mitkOptimizer::GetSpaceDimension () [inline]

Get the dimensions of optimization space.

Returns:
The dimension of optimization space.

8.190.2.12 bool mitkOptimizer::GetStopFlag () [inline]

Get the stop flag.

Returns:
The stop flag.

8.190.2.13 virtual void mitkOptimizer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkProcessObject.

Reimplemented in mitkAmoebaOptimizer, mitkGradientDescentOptimizer, mitkIterativeClosestPoint-Optimizer, mitkLBFGSBOptimizer, mitkLBFGSOptimizer, and mitkPowellOptimizer.

8.190.2.14 void mitkOptimizer::SetCurrentIterationNumber (unsigned int *iterations*) [inline]

Set current iteration number.

Parameters:
iterations Current iteration number.

8.190.2.15 virtual void mitkOptimizer::SetInitialParameters (VectorParameterType & *param*) [inline, virtual]

Set the position to initialize the optimization.

Parameters:
param The vector pointer to the initial position for optimization.

8.190.2.16 void mitkOptimizer::SetMaximizeFlag (bool *flag*) [inline]

Define whether the cost function will be maximized or minimized. flag Maximizing the cost function if flag is true.

8.190.2.17 void mitkOptimizer::SetMaxIterations (unsigned int *n*) [inline]

Set the maximum number of iterations.

Parameters:
n The maximum number of iterations.

8.190.2.18 void mitkOptimizer::SetMetric (mitkMetric * *metric*) [inline]

Set the similarity function

Parameters:
metric The pointer to the Metric.

8.190.2.19 void mitkOptimizer::SetNumberOfParameters (unsigned int *num*) [inline]

Set the number of parameters.

Parameters:
num The number of parameters.

8.190.2.20 void mitkOptimizer::SetScales (ScalarParameterType * *scales*, unsigned int *length*)

Set the scalings of current parameters.

Parameters:
scales The array pointer to the scales for every elements of parameters.

numberOfParameters The length of input scaling array.

8.190.2.21 void mitkOptimizer::SetScales (VectorParameterType & *scales*)

Set the scalings of current parameters.

Parameters:
scales The vector pointer to the scales for every elements of parameters.

8.190.2.22 void mitkOptimizer::SetSpaceDimension (unsigned int *dim*) [inline]

Set the dimensions of optimization space.

Parameters:
num The dimensions of optimization space.

8.190.2.23 void mitkOptimizer::SetStopFlag (bool *flag*) [inline]

Set the stop flag, it will terminate the optimization process if the value is true. flag The stop flag.

8.190.2.24 void mitkOptimizer::StopOptimization (StopCondition *condition*)

General stop conditions handling function call by concrete optimize classes. condition The stop condition.

8.190.2.25 virtual void mitkOptimizer::Update () [virtual]

Initialize the Optimizer and make sure the components are present.

The documentation for this class was generated from the following file:

- mitkOptimizer.h

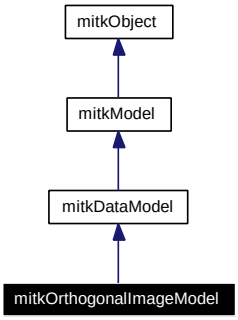
8.191 mitkOrthogonalImageModel Class Reference

mitkOrthogonalImageModel

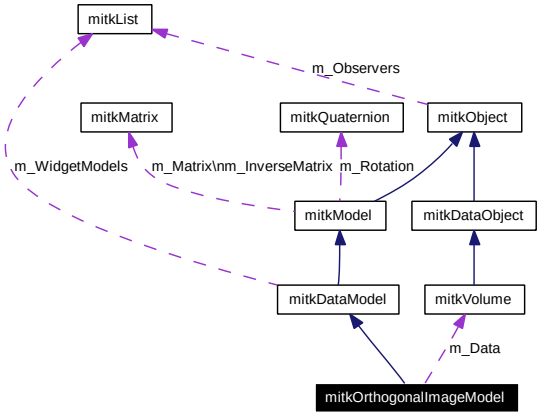
`#include <mitkOrthogonalImageModel.h>`

Inherits mitkDataModel.

Inheritance diagram for mitkOrthogonalImageModel:



Collaboration diagram for mitkOrthogonalImageModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOrthogonalImageModel ()
- void CleanUp ()
- void SetData (mitkVolume *data)
- mitkVolume * GetData (void)
- virtual int Render (mitkScene *scene)
- virtual bool IsOpaque ()

8.191.1 Detailed Description

mitkOrthogonalImageModel

8.191.2 Constructor & Destructor Documentation

8.191.2.1 mitkOrthogonalImageModel::mitkOrthogonalImageModel ()

The default constructor.

8.191.3 Member Function Documentation

8.191.3.1 void mitkOrthogonalImageModel::CleanUp ()

Internal function, Don't call it directly.

8.191.3.2 mitkVolume* mitkOrthogonalImageModel::GetData (void)

Get the volume data.

Returns:
Return pointer to the volume data.

8.191.3.3 virtual bool mitkOrthogonalImageModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Implements mitkModel.

8.191.3.4 virtual void mitkOrthogonalImageModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkDataModel.

8.191.3.5 virtual int mitkOrthogonalImageModel::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.191.3.6 void mitkOrthogonalImageModel::SetData (mitkVolume * *data*)

Set the volume data.

Parameters:
data pointer to an mitkVolume

The documentation for this class was generated from the following file:

- mitkOrthogonalImageModel.h

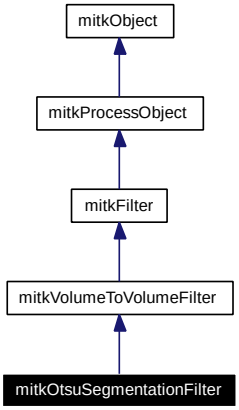
8.192 mitkOtsuSegmentationFilter Class Reference

mitkOtsuSegmentationFilter - a class implement Ostu segmentation arithmetic

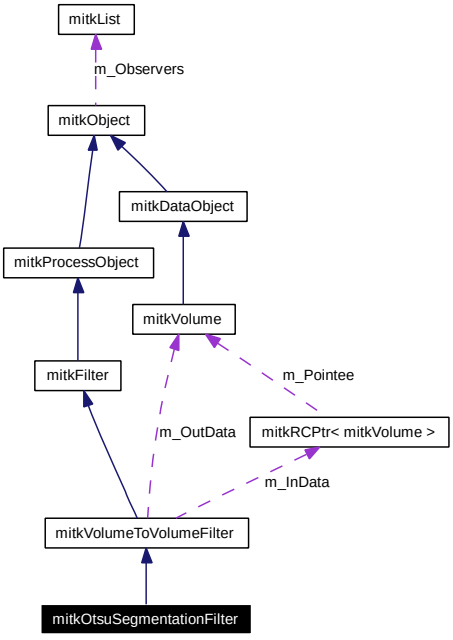
`#include <mitkOtsuSegmentationFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkOtsuSegmentationFilter:



Collaboration diagram for mitkOtsuSegmentationFilter:



Public Member Functions

- mitkOtsuSegmentationFilter ()
- virtual void PrintSelf (ostream &os)

8.192.1 Detailed Description

mitkOtsuSegmentationFilter - a class implement Ostu segmentation arithmetic

mitkOtsuSegmentationFilter implement Ostu segmentation arithmetic. The output is of the same datatype of the input.

8.192.2 Constructor & Destructor Documentation

8.192.2.1 mitkOtsuSegmentationFilter::mitkOtsuSegmentationFilter ()

Constructor of the class

8.192.3 Member Function Documentation

8.192.3.1 virtual void mitkOtsuSegmentationFilter::PrintSelf (ostream & *os*) [inline, virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkOtsuSegmentationFilter.h

8.193 mitkParallelBeamReconFromCirle Class Reference

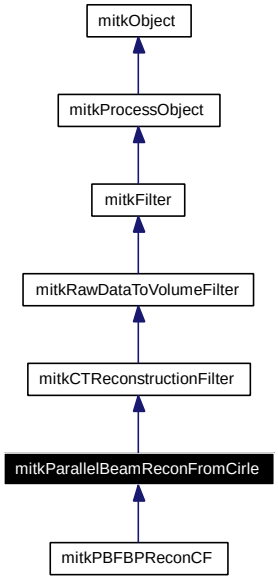
mitkParallelBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm.

`#include <mitkParallelBeamReconFromCirle.h>`

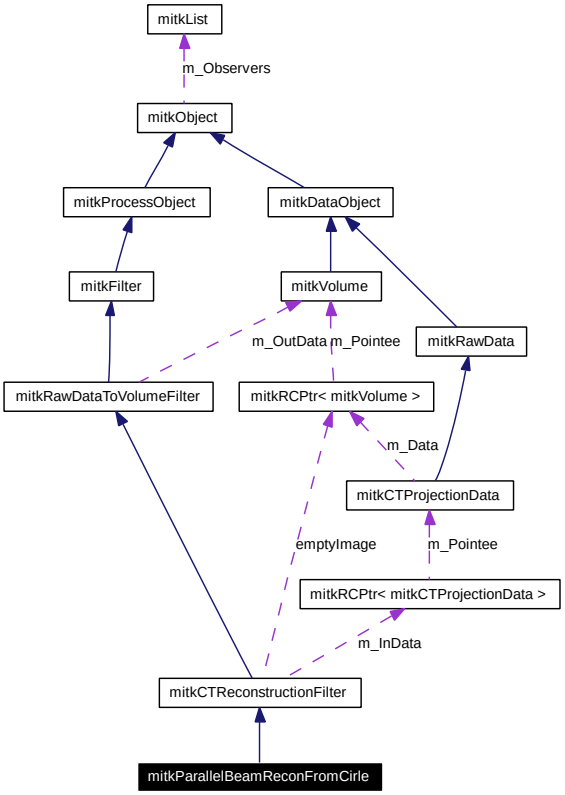
Inherits mitkCTReconstructionFilter.

Inherited by mitkPBFBPReconCF.

Inheritance diagram for mitkParallelBeamReconFromCirle:



Collaboration diagram for mitkParallelBeamReconFromCirle:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetDetectorLength (double m)
- double GetDetectorLength ()
- void SetFov (double *m)
- double * GetFov ()

8.193.1 Detailed Description

mitkParallelBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm.

mitkParallelBeamReconFromCirle - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm.

8.193.2 Member Function Documentation

8.193.2.1 double mitkParallelBeamReconFromCirle::GetDetectorLength () [inline]

// Get the length of detector

8.193.2.2 double* mitkParallelBeamReconFromCirle::GetFov () [inline]

Get field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.193.2.3 virtual void mitkParallelBeamReconFromCirle::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkCTReconstructionFilter.

Reimplemented in mitkPBFBPReconCF.

8.193.2.4 void mitkParallelBeamReconFromCirle::SetDetectorLength (double *m*) [inline]

Set the length of detector

8.193.2.5 void mitkParallelBeamReconFromCirle::SetFov (double * *m*) [inline]

Set field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

The documentation for this class was generated from the following file:

- mitkParallelBeamReconFromCirle.h

8.194 mitkParallelProjectData Class Reference

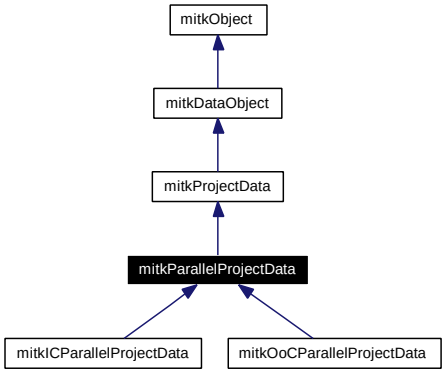
mitkParallelProjectData - abstract class to represent Parallel project data.

`#include <mitkParallelProjectData.h>`

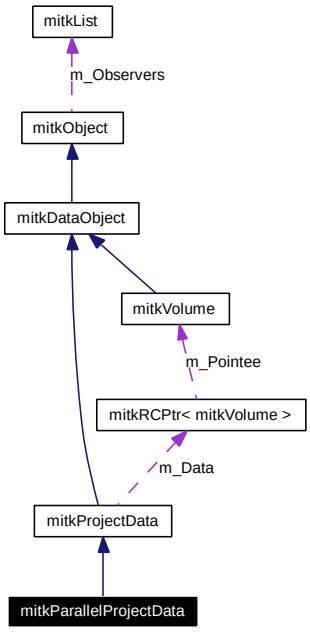
Inherits mitkProjectData.

Inherited by mitkICParallelProjectData, and mitkOoCParallelProjectData.

Inheritance diagram for mitkParallelProjectData:



Collaboration diagram for mitkParallelProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int GetDataObjectType () const

- virtual void const * GetData () const
- virtual void * GetData ()
- void SetData (mitkVolume *m)
- int GetDetectorMove ()
- void SetDetectorMove (int m)
- void SetDetectorLength (double m)
- double GetDetectorLength ()
- void SetFov (double *m)
- double * GetFov ()
- void SetXrayNum (int m)
- int GetXrayNum () const
- void SetSliceNum (int m)

  Set Slice Number.

- int GetSliceNum () const
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0
- virtual void const * GetSliceForRead (int sliceIdx)=0
- virtual void Initialize ()

8.194.1 Detailed Description

mitkParallelProjectData - abstract class to represent Parallel project data.

mitkParallelProjectData - abstract class to represent Parallel project data.

8.194.2 Member Function Documentation

8.194.2.1 virtual void mitkParallelProjectData::DeepCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkICParallelProjectData, and mitkOoCParallelProjectData.

8.194.2.2 virtual void* mitkParallelProjectData::GetData () [inline, virtual]

get volume pointer

Implements mitkProjectData.

8.194.2.3 virtual void const* mitkParallelProjectData::GetData () const [inline, virtual]

get a const volume pointer

Implements mitkProjectData.

8.194.2.4 virtual int mitkParallelProjectData::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_DATA_OBJECT

Note:
Pure virtual function. Its concrete subclass must implement this function and return its data object type.

Reimplemented from mitkDataObject.

Reimplemented in mitkOoCParallelProjectData.

8.194.2.5 double mitkParallelProjectData::GetDetectorLength () [inline]

Get the length of detector

8.194.2.6 int mitkParallelProjectData::GetDetectorMove () [inline]

??

8.194.2.7 double* mitkParallelProjectData::GetFov () [inline]

Get field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.194.2.8 virtual void const* mitkParallelProjectData::GetSliceForRead (int *sliceIdx*) [pure virtual]

GetSliceForRead GetSliceForWrite and GetSliceForReadWrite are for multilayer

Implemented in mitkICParallelProjectData, and mitkOoCParallelProjectData.

8.194.2.9 int mitkParallelProjectData::GetSliceNum () const [inline]

Get Slice Number

8.194.2.10 int mitkParallelProjectData::GetXrayNum () const [inline]

Get number of Xray

8.194.2.11 virtual void mitkParallelProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkProjectData.

Reimplemented in mitkICParallelProjectData, and mitkOoCParallelProjectData.

8.194.2.12 virtual void mitkParallelProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkProjectData.

Reimplemented in mitkICParallelProjectData, and mitkOoCParallelProjectData.

8.194.2.13 void mitkParallelProjectData::SetData (mitkVolume * *m*)

Set volume pointer

8.194.2.14 void mitkParallelProjectData::SetDetectorLength (double *m*) [inline]

Set the length of detector

8.194.2.15 void mitkParallelProjectData::SetDetectorMove (int *m*) [inline]

??

8.194.2.16 void mitkParallelProjectData::SetFov (double * *m*) [inline]

Set field of view [0] left of x axis [1] right of x axis [2] bottom of y axis [3] top of y axis

8.194.2.17 void mitkParallelProjectData::SetXrayNum (int *m*) [inline]

Set number of Xray

8.194.2.18 virtual void mitkParallelProjectData::ShallowCopy (mitkDataObject * *src*) [pure virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkDataObject.

Implemented in mitkICParallelProjectData, and mitkOoCParallelProjectData.

The documentation for this class was generated from the following file:

- mitkParallelProjectData.h

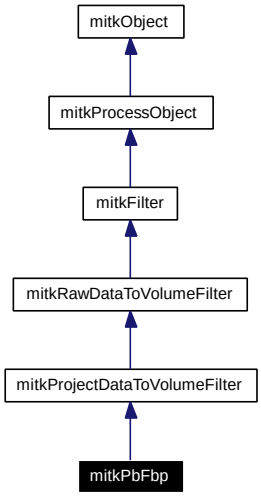
8.195 mitkPbFbp Class Reference

mitkPbFbp - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm.

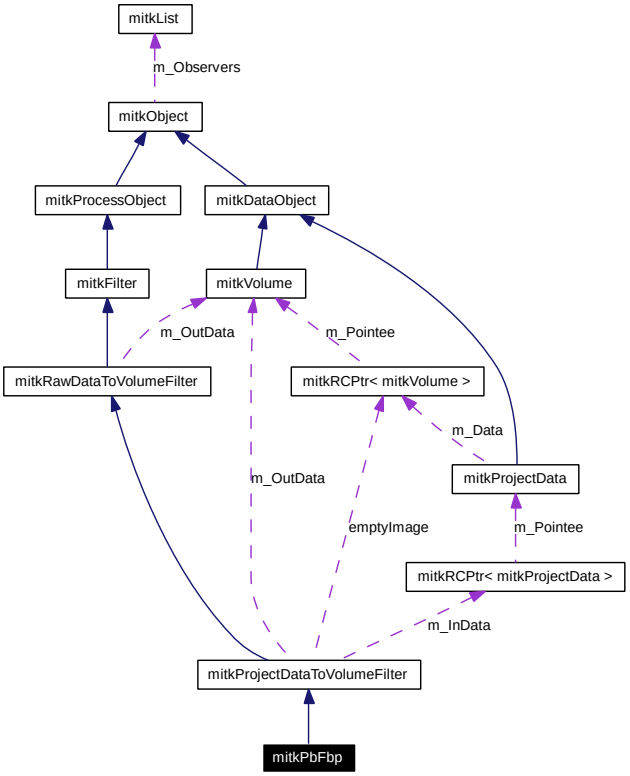
`#include <mitkPbFbp.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkPbFbp:



Collaboration diagram for mitkPbFbp:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.195.1 Detailed Description

mitkPbFbp - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm.

mitkPbFbp - concret class implement parallel-beam reconstruction using filtered backprojection Algorithm.

8.195.2 Member Function Documentation

8.195.2.1 virtual void mitkPbFbp::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkPbFbp.h

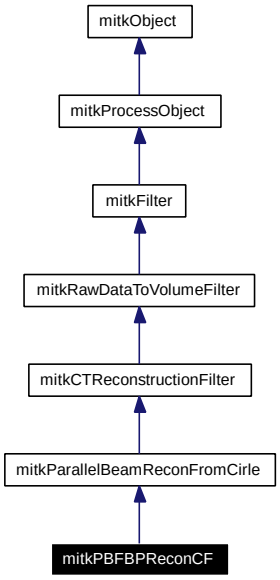
8.196 mitkPBFBPReconCF Class Reference

mitkPBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

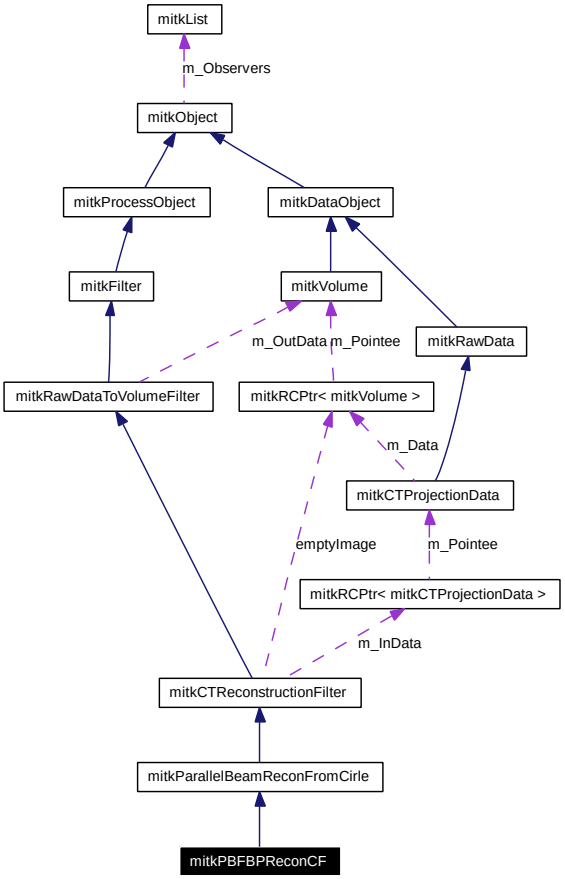
`#include <mitkPBFBPReconCF.h>`

Inherits mitkParallelBeamReconFromCirle.

Inheritance diagram for mitkPBFBPReconCF:



Collaboration diagram for mitkPBFBPReconCF:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.196.1 Detailed Description

mitkPBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm.

mitkPBFBPReconCF - concret class implement fan-beam reconstruction using filtered backprojection Algorithm, this implemention just using C++ without any acceleration

8.196.2 Member Function Documentation

8.196.2.1 virtual void mitkPBFBPReconCF::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkParallelBeamReconFromCirle.

The documentation for this class was generated from the following file:

- mitkPBFBPReconCF.h

8.197 mitkPickManipulator Class Reference

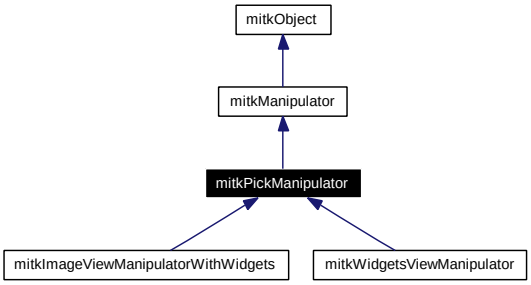
mitkPickManipulator - a manipulator with picking function enabled

`#include <mitkPickManipulator.h>`

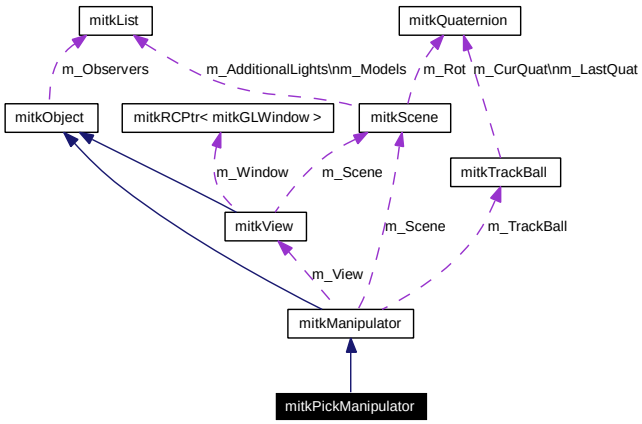
Inherits mitkManipulator.

Inherited by mitkImageViewManipulatorWithWidgets, and mitkWidgetsViewManipulator.

Inheritance diagram for mitkPickManipulator:



Collaboration diagram for mitkPickManipulator:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void ClearPickResult ()
- mitkWidgetModel * GetPickedObject ()

8.197.1 Detailed Description

mitkPickManipulator - a manipulator with picking function enabled

mitkPickManipulator is a manipulator with picking function enabled. Manipulators who need to pick objects in the scene should derive from this class. In your derived class, use protected function _pick() to do the picking operation. If it returns a "true", something was picked and the information of the picked

object was put into a WidgetNames struct member : m_PickedWidgets. The protected member m_PickBox indicates the precision of the picking operation. Smaller means preciser. The protected member m_EnableDepthTest indicates if the picking function deal with the depth information when multiple objects are at the same position on screen. The default value is "true", which means the picking function will test the depth of all the objects and return the most front object. In your derived class, you can set it to "false", which means no depth test will be done, and picked object will be the one rendered at last.

The WidgetNames struct is defined as follows:

```
typedef struct _widget_names
{
    mitkWidgetModel *owner; // a pointer to the widget which was picked
    GLuint name1;           // name1 to name3 indicate the picked parts of the widget
    GLuint name2;
    GLuint name3;
} WidgetNames;
```

Note:

The name1, name2 and name3 fields are used by mitkWidgetModel. In your derived class of mitkPickManipulator, you can pass the struct to picked mitkWidgetModel by calling m_PickedWidgets.owner->Pick() after picking. And in the samemanner, you can transfer the control to the picked widget through m_PickedWidgets.owner.

8.197.2 Member Function Documentation

8.197.2.1 void mitkPickManipulator::ClearPickResult ()

Clear picked results.

8.197.2.2 mitkWidgetModel* mitkPickManipulator::GetPickedObject () [inline]

Get current picked widget.

Returns:

Return the pointer to a mitkWidgetModel currently picked.

8.197.2.3 virtual void mitkPickManipulator::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkManipulator.

Reimplemented in mitkImageViewManipulatorWithWidgets, and mitkWidgetsViewManipulator.

The documentation for this class was generated from the following file:

- mitkPickManipulator.h

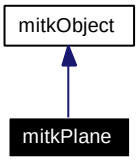
8.198 mitkPlane Class Reference

mitkPlane - a class to represent a plane

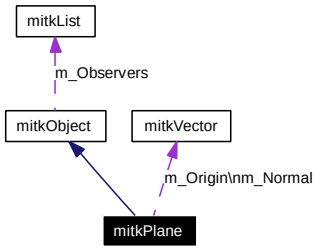
`#include <mitkPlane.h>`

Inherits mitkObject.

Inheritance diagram for mitkPlane:



Collaboration diagram for mitkPlane:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.198.1 Detailed Description

mitkPlane - a class to represent a plane

mitkPlane is a class to represenet a plane

8.198.2 Member Function Documentation

8.198.2.1 virtual void mitkPlane::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

The documentation for this class was generated from the following file:

- mitkPlane.h

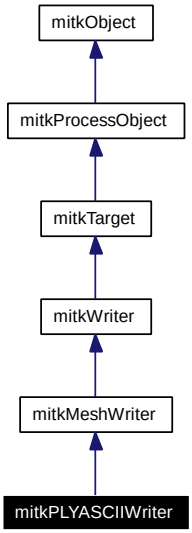
8.199 mitkPLYASCIIWriter Class Reference

mitkPLYASCIIWriter - a concrete writer for writing a mesh to PLY files with ASCII format

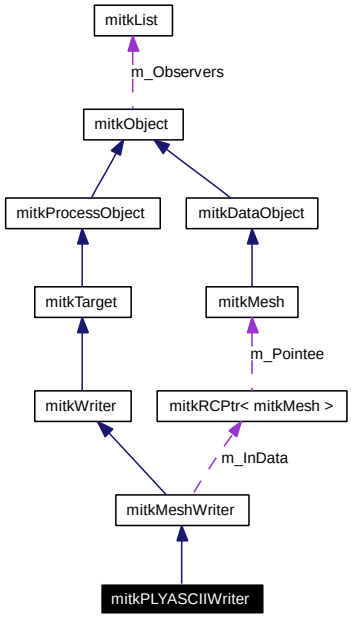
`#include <mitkPLYASCIIWriter.h>`

Inherits mitkMeshWriter.

Inheritance diagram for mitkPLYASCIIWriter:



Collaboration diagram for mitkPLYASCIIWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.199.1 Detailed Description

mitkPLYASCIIWriter - a concrete writer for writing a mesh to PLY files with ASCII format

mitkPLYASCIIWriter is a concrete writer for writing a mesh to PLY files with ASCII format. PLY is a file format developed by Stanford University for exchanging data like meshes and 3D scans. To use this writer, the code snippet is:

```
mitkPLYASCIIWriter *aWriter = new mitkPLYASCIIWriter;
aWriter->SetInput(aMesh);
aWriter->AddFileName(filename);
aWriter->Run();
```

8.199.2 Member Function Documentation

8.199.2.1 virtual void mitkPLYASCIIWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMeshWriter.

The documentation for this class was generated from the following file:

- mitkPLYASCIIWriter.h

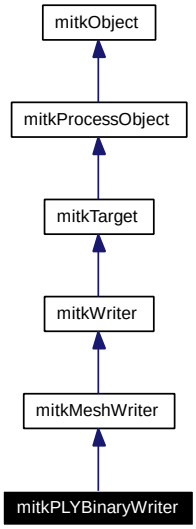
8.200 mitkPLYBinaryWriter Class Reference

mitkPLYBinaryWriter - a concrete writer for writing a mesh to PLY files with binary format

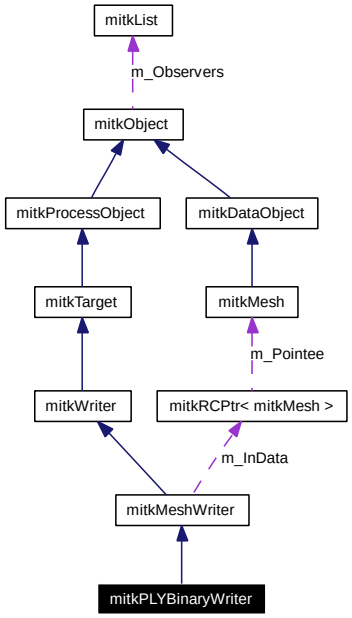
`#include <mitkPLYBinaryWriter.h>`

Inherits mitkMeshWriter.

Inheritance diagram for mitkPLYBinaryWriter:



Collaboration diagram for mitkPLYBinaryWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.200.1 Detailed Description

mitkPLYBinaryWriter - a concrete writer for writing a mesh to PLY files with binary format

mitkPLYBinaryWriter is a concrete writer for writing a mesh to PLY files with binary format. PLY is a file format developed by Stanford University for exchanging data like meshes and 3D scans. To use this writer, the code snippet is:

```
mitkPLYASCIIWriter *aWriter = new mitkPLYASCIIWriter;
aWriter->SetInput(aMesh);
aWriter->AddFileName(filename);
aWriter->SetBigEndian(true); //Default is little endian
aWriter->Run();
```

8.200.2 Member Function Documentation

8.200.2.1 virtual void mitkPLYBinaryWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMeshWriter.

The documentation for this class was generated from the following file:

- mitkPLYBinaryWriter.h

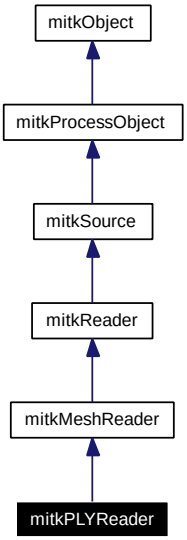
8.201 mitkPLYReader Class Reference

mitkPLYReader - a concrete reader for reading a PLY file

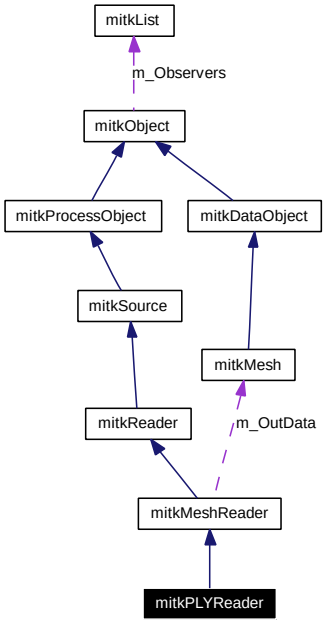
`#include <mitkPLYReader.h>`

Inherits mitkMeshReader.

Inheritance diagram for mitkPLYReader:



Collaboration diagram for mitkPLYReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.201.1 Detailed Description

mitkPLYReader - a concrete reader for reading a PLY file

mitkPLYReader is a concrete reader for reading a PLY file. PLY is a file format developed by Stanford University for exchanging data like meshes and 3D scans. To use this writer, the code snippet is:

```
mitkPLYReader *aReader = new mitkPLYReader;
aReader->AddFileName(filename);
if (aReader->Run())
{
    mitkMesh *aMesh = aReader->GetOutput();
    Using aMesh
}
```

8.201.2 Member Function Documentation

8.201.2.1 virtual void mitkPLYReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMeshReader.

The documentation for this class was generated from the following file:

- mitkPLYReader.h

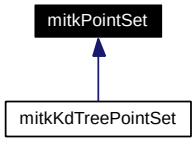
8.202 mitkPointSet Class Reference

mitkPointSet - a class to represent geometry in the form of a set of points in two or three dimensional space.

`#include <mitkPointSet.h>`

Inherited by mitkKdTreePointSet.

Inheritance diagram for mitkPointSet:



Public Member Functions

- mitkPointSet ()
- virtual ~mitkPointSet ()
- mitkPointSet (unsigned int numberOfPoints, unsigned int pointSetDimension, double fillElement=0)
- mitkPointSet (mitkPointSet *pointSet)
- void GetPoint (unsigned int i, ScalarParameterType *p)
- double GetPoint (unsigned int i, unsigned int dim)
- void SetPoint (unsigned int i, ScalarParameterType *p)
- vector< double > * GetPointsInSequence (vector< int > *sequence)
- vector< double > * GetPointData ()
- double * GetPointDataPtr ()
- void SetPointData (vector< double > *pointSet, unsigned int dimension)
- void SetPointData (double *pointSet, unsigned int dimension, unsigned int numberOfPoints)
- vector< double > * GetPointsInDimension (unsigned int sDimension)
- void SetPointsInDimension (vector< double > *points, unsigned int sDimension)
- void SetPointsInDimension (double fillElement, unsigned int sDimension)
- void SetPointData (mitkMesh *mesh)
- void GetPointData (mitkMesh *mesh)
- void GetWeight (unsigned int i, vector< double > *w)
- void SetWeight (unsigned int i, vector< double > *w)
- vector< double > * GetWeightsInSequence (vector< int > *sequence)
- void SetWeightData (vector< double > *weights, unsigned int dimension)
- void SetWeightData (double *weights, unsigned int dimension, unsigned int numberOfPoints)
- vector< double > * GetWeightData ()
- double * GetWeightDataPtr ()
- void SetPointSetMask (vector< int > *pointSetMask)
- vector< int > * GetPointSetMask ()
- void SetPointSetMaskFlag (bool maskFlag)
- bool GetPointSetMaskFlag ()
- void ReadPointDataFromFile (char *inFileName, int pointSetDimension, int dataType)
- void WritePointDataToFile (char *outFileName)
- unsigned int GetNumberOfPoints ()
- void SetNumberOfPoints (unsigned int num)
- int GetPointSetDimension ()

- void SetPointSetDimension (unsigned int pointSetDimension)
- int GetWeightsDimension ()
- void GetBoundingBox (double b[6])
- void GetBoundingBox (float b[6])
- void GetCentroid (double p[3])
- virtual void GetNearestPoint (double *query_point, double *result_point, int &i)
- virtual void GetKNearestPoints (double *query_point, unsigned int k, vector< int > &indices)
- virtual void GetKNearestPoints (vector< double > *query_point, unsigned int k, vector< int > &indices)
- virtual void GetNearestPoints (mitkPointSet *query_pointset, vector< int > &indices)
- virtual void GetPointsInBoundingBox (mitkBoundingBox< double > &box, vector< int > &indices)
- virtual void GetPointsInRadius (double *query_point, double radius, vector< int > &indices)
- void Fill (double fillElement)
- bool RemoveDuplicates ()
- void ConvertPointSet2Dto3D ()
- virtual void Initialize ()
- virtual void Clear ()
- bool IsPointSetEmpty ()
- bool IsWeightsEmpty ()
- bool IsInitialized ()
- void ShallowCopy (mitkPointSet *pointSet)
- void DeepCopy (mitkPointSet *pointSet)
- void Merge (mitkPointSet *pointSet)
- void Reserve (unsigned int s, unsigned int d=3)
- void SetOriginAndDirection (double o[3], int d[3])

8.202.1 Detailed Description

mitkPointSet - a class to represent geometry in the form of a set of points in two or three dimensional space.

mitkPointSet is a class to represent geometry in the form of a set of points in two or three dimensional space. It provides lots of basic point set handling methods and several k nearest neighbor searching methods.

8.202.2 Constructor & Destructor Documentation

8.202.2.1 mitkPointSet::mitkPointSet ()

Constructor.

8.202.2.2 virtual mitkPointSet::~mitkPointSet () [virtual]

De-constructor.

8.202.2.3 mitkPointSet::mitkPointSet (unsigned int *numberOfPoints*, unsigned int *pointSetDimension*, double *fillElement* = 0)

Constructor filling with initial elements

Parameters:

numberOfPoints The number of points.

pointSetDimension The point set dimension.

fillElement The element value to fill.

8.202.2.4 mitkPointSet::mitkPointSet (mitkPointSet * *pointSet*)

Constructor with a point set

Parameters:

pointSet The input point set.

8.202.3 Member Function Documentation

8.202.3.1 virtual void mitkPointSet::Clear () [virtual]

Deconstruct the point set object.

8.202.3.2 void mitkPointSet::ConvertPointSet2Dto3D ()

Convert point set's dimension from 2D to 3D.

8.202.3.3 void mitkPointSet::DeepCopy (mitkPointSet * *pointSet*)

Initialize the point set data by another point set.(copy data)

Parameters:

pointSet The pointer to input point set.

8.202.3.4 void mitkPointSet::Fill (double *fillElement*)

Fill the point set by a single element.

Parameters:

fillElement The filling element

8.202.3.5 void mitkPointSet::GetBoundingBox (float *b*[6])

Get bounding box of the point set.

Parameters:

b[0] return minimum coordinate value in x-axis

b[1] return maximum coordinate value in x-axis

b[2] return minimum coordinate value in y-axis

b[3] return maximum coordinate value in y-axis

b[4] return minimum coordinate value in z-axis

b[5] return maximum coordinate value in z-axis

8.202.3.6 void mitkPointSet::GetBoundingBox (double *b*[6])

Get bounding box of the point set.

Parameters:

b[0] return minimum coordinate value in x-axis

b[1] return maximum coordinate value in x-axis

b[2] return minimum coordinate value in y-axis

b[3] return maximum coordinate value in y-axis

b[4] return minimum coordinate value in z-axis

b[5] return maximum coordinate value in z-axis

8.202.3.7 void mitkPointSet::GetCentroid (double *p*[3])

Get centroid of the point set.

Parameters:

p[0] return centroid coordinate value in x-axis

p[1] return centroid coordinate value in y-axis

p[2] return centroid coordinate value in z-axis

8.202.3.8 virtual void mitkPointSet::GetKNearestPoints (vector< double > * *query_point*, unsigned int *k*, vector< int > & *indices*) [virtual]

Get k nearest points to the query point.

Parameters:

query_point The query point stored in a vector

k The number of points to search

indices Return a vector contains result point's indices

Reimplemented in mitkKdTreePointSet.

8.202.3.9 virtual void mitkPointSet::GetKNearestPoints (double * *query_point*, unsigned int *k*, vector< int > & *indices*) [virtual]

Get k nearest point to the query point.

Parameters:

query_point The query point stored in an array

k The number of points to search

indices Return a vector contains result point's indices

Reimplemented in mitkKdTreePointSet.

8.202.3.10 virtual void mitkPointSet::GetNearestPoint (double * *query_point*, double * *result_point*, int & *i*) [virtual]

Get a nearest point to the query point.

Parameters:

query_point The query point

result_point Return the pointer to result point

i Return the index to result point

Reimplemented in mitkKdTreePointSet.

8.202.3.11 virtual void mitkPointSet::GetNearestPoints (mitkPointSet * *query_pointset*, vector< int > & *indices*) [virtual]

Get nearest point to a set of query points.

Parameters:

query_pointset The query points stored in a mitkPointSet

indices Return a vector contains result point's indices

Reimplemented in mitkKdTreePointSet.

8.202.3.12 unsigned int mitkPointSet::GetNumberOfPoints ()

Get number of points in the point set.

Returns:

Return the number of points.

8.202.3.13 double mitkPointSet::GetPoint (unsigned int *i*, unsigned int *dim*)

Get a point from the point set.

Parameters:

i The index of the point.

p An array to store the returned point.

8.202.3.14 void mitkPointSet::GetPoint (unsigned int *i*, ScalarParameterType * *p*)

Get a point from the point set.

Parameters:

i The index of the point.

p An array to store the returned point.

8.202.3.15 void mitkPointSet::GetPointData (mitkMesh * *mesh*)

Get points data to a mesh object.

Parameters:
mesh A mesh object to store the points data.

8.202.3.16 vector<double>* mitkPointSet::GetPointData ()

Get all the points in the point set.

Returns:
Return a vector contains the points.

8.202.3.17 double* mitkPointSet::GetPointDataPtr () [inline]

Get the pointer of point data.

Returns:
Return a pointer to the points data.

8.202.3.18 int mitkPointSet::GetPointSetDimension () [inline]

Get the point set's dimension.

Returns:
Return the point set's dimension.

8.202.3.19 vector<int>* mitkPointSet::GetPointSetMask () [inline]

Get the mask of point set.

Returns:
Return the pointer to the mask vector.

8.202.3.20 bool mitkPointSet::GetPointSetMaskFlag () [inline]

Get using mask flag.

Returns:
Return the mask flag.

8.202.3.21 virtual void mitkPointSet::GetPointsInBoundingBox (mitkBoundingBox< double > & *box*, vector< int > & *indices*) [virtual]

Get points in a bounding box.

Parameters:

box The query bounding box

indices Return a vector contains result point's indices

Reimplemented in mitkKdTreePointSet.

8.202.3.22 vector<double>* mitkPointSet::GetPointsInDimension (unsigned int *sDimension*)

Get points in a specified dimension.

Parameters:

sDimension A specified dimension.

Returns:

Return a vector contains the points.

8.202.3.23 virtual void mitkPointSet::GetPointsInRadius (double * *query_point*, double *radius*, vector< int > & *indices*) [virtual]

Get points in a given circle/sphere.

Parameters:

query_point The query point

radius The radius of the given circle

indices Return a vector contains result point's indices

Reimplemented in mitkKdTreePointSet.

8.202.3.24 vector<double>* mitkPointSet::GetPointsInSequence (vector< int > * *sequence*)

Get points using a sequence mask .

Parameters:

sequence The mask with points' index.

Returns:

Return a vector contains mask-specified points.

8.202.3.25 void mitkPointSet::GetWeight (unsigned int *i*, vector< double > * *w*)

Get a point's weight vector.

Parameters:

i The index to specified point(weight).

w A vector to contain the weight vector.

8.202.3.26 vector<double>* mitkPointSet::GetWeightData ()

Get all the weights.

Returns:
Return a vector contains the weights.

8.202.3.27 double* mitkPointSet::GetWeightDataPtr () [inline]

Get weights data's pointer.

Returns:
Return the pointer to the weights data.

8.202.3.28 int mitkPointSet::GetWeightsDimension () [inline]

Get weights set's dimension.

Returns:
Return the weights set's dimension.

8.202.3.29 vector<double>* mitkPointSet::GetWeightsInSequence (vector< int > * *sequence*)

Get weights using a sequence mask .

Parameters:
sequence The mask with points' index.

Returns:
Return a vector contains mask-specified weights.

8.202.3.30 virtual void mitkPointSet::Initialize () [virtual]

Initialize the point set.(calculate centroid, bounding box, build tree)

Reimplemented in mitkKdTreePointSet.

8.202.3.31 bool mitkPointSet::IsInitialized () [inline]

Check if the point set is initialized.

Returns:
Return true if the point set is initialized.

8.202.3.32 bool mitkPointSet::IsPointSetEmpty ()

Check if the point set is empty.

Returns:
Return true if the point set is empty.

8.202.3.33 bool mitkPointSet::IsWeightsEmpty ()

Check if the weight set is empty.

Returns:
Return true if the weight set is empty.

8.202.3.34 void mitkPointSet::Merge (mitkPointSet * *pointSet*)

Merge another point set to the point set.

Parameters:
pointSet The pointer to input point set.

8.202.3.35 void mitkPointSet::ReadPointDataFromFile (char * *inFileName*, int *pointSetDimension*, int *dataType*)

Read points data form a txt file.

Parameters:
inFileName File name.
pointSetDimension Point set dimension.
dataType Date type.

8.202.3.36 bool mitkPointSet::RemoveDuplicates ()

Remove duplicate points in the point set.

8.202.3.37 void mitkPointSet::Reserve (unsigned int *s*, unsigned int *d* = 3)

Reserve a memory space for the point set. (resize the point set) If s is smaller than numberOfPoints, do noting.

Parameters:
s The size of reserved space.
d The point set dimension.

8.202.3.38 void mitkPointSet::SetNumberOfPoints (unsigned int *num*) [inline]

Set number of points in the point set.

Parameters:
num The number of points.

8.202.3.39 void mitkPointSet::SetOriginAndDirection (double *o*[3], int *d*[3])

mitkPointSet represents points in physical space. This function just convert from one to another.

8.202.3.40 void mitkPointSet::SetPoint (unsigned int *i*, ScalarParameterType * *p*)

Set a point in the point set.

Parameters:

i The index of the point.

p An array with input point data.

8.202.3.41 void mitkPointSet::SetPointData (mitkMesh * *mesh*)

Set data of the point set by a mesh object.

Parameters:

mesh A mesh object to provide points data.

8.202.3.42 void mitkPointSet::SetPointData (double * *pointSet*, unsigned int *dimension*, unsigned int *numberOfPoints*)

Set data of the point set

Parameters:

pointSet The pointer to an array contains the points data.

dimension The point set's dimension.

numberOfPoints The number of points in the array.

8.202.3.43 void mitkPointSet::SetPointData (vector< double > * *pointSet*, unsigned int *dimension*)

Set data of the point set

Parameters:

pointSet The pointer to a vector contains the points data.

dimension The point set's dimension.

8.202.3.44 void mitkPointSet::SetPointSetDimension (unsigned int *pointSetDimension*) [inline]

Set point set's dimension.

Parameters:

pointSetDimension The point set's dimension.

8.202.3.45 void mitkPointSet::SetPointSetMask (vector< int > * *pointSetMask*)

Set the mask of point set.

Parameters:

pointSetMask The pointer to a vector contains point index data.

8.202.3.46 void mitkPointSet::SetPointSetMaskFlag (bool *maskFlag*) [inline]

Set using mask flag.

Parameters:

maskFlag Use mask in point set get/set method if maskFlag is true.

8.202.3.47 void mitkPointSet::SetPointsInDimension (double *fillElement*, unsigned int *sDimension*)

Set data of the point set in a specified dimension by filling an element.

Parameters:

fillElement The element value to fill.

sDimension A specified dimension of the point set.

8.202.3.48 void mitkPointSet::SetPointsInDimension (vector< double > * *points*, unsigned int *sDimension*)

Set data of the point set in a specified dimension.

Parameters:

points The pointer to a vector contains the points data.

sDimension A specified dimension of the point set.

8.202.3.49 void mitkPointSet::SetWeight (unsigned int *i*, vector< double > * *w*)

Set a point's weight vector

Parameters:

i The index to specified point(weight).

w A vector contains the weight vector.

8.202.3.50 void mitkPointSet::SetWeightData (double * *weights*, unsigned int *dimension*, unsigned int *numberOfPoints*)

Set weights data by an array

Parameters:

weights The pointer to an array contains the weights data.

dimension The weight set's dimension.

numberOfPoints The number of weights in the array.

8.202.3.51 void mitkPointSet::SetWeightData (vector< double > * *weights*, unsigned int *dimension*)

Set weights data by a vector

Parameters:

weights The pointer to a vector contains the weights data.

dimension The weight set's dimension.

8.202.3.52 void mitkPointSet::ShallowCopy (mitkPointSet * *pointSet*)

Initialize the point set data by another point set.(copy pointer, don't copy data)

Parameters:

pointSet The pointer to input point set.

8.202.3.53 void mitkPointSet::WritePointDataToFile (char * *outFileName*)

Write the points data to a txt file.

Parameters:

outFileName File name.

The documentation for this class was generated from the following file:

- mitkPointSet.h

8.203 mitkPointSetEstimator Class Reference

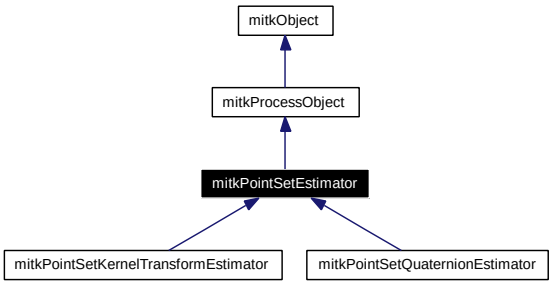
mitkPointSetEstimator - an abstract class specifies interface for an estimation method

`#include <mitkPointSetEstimator.h>`

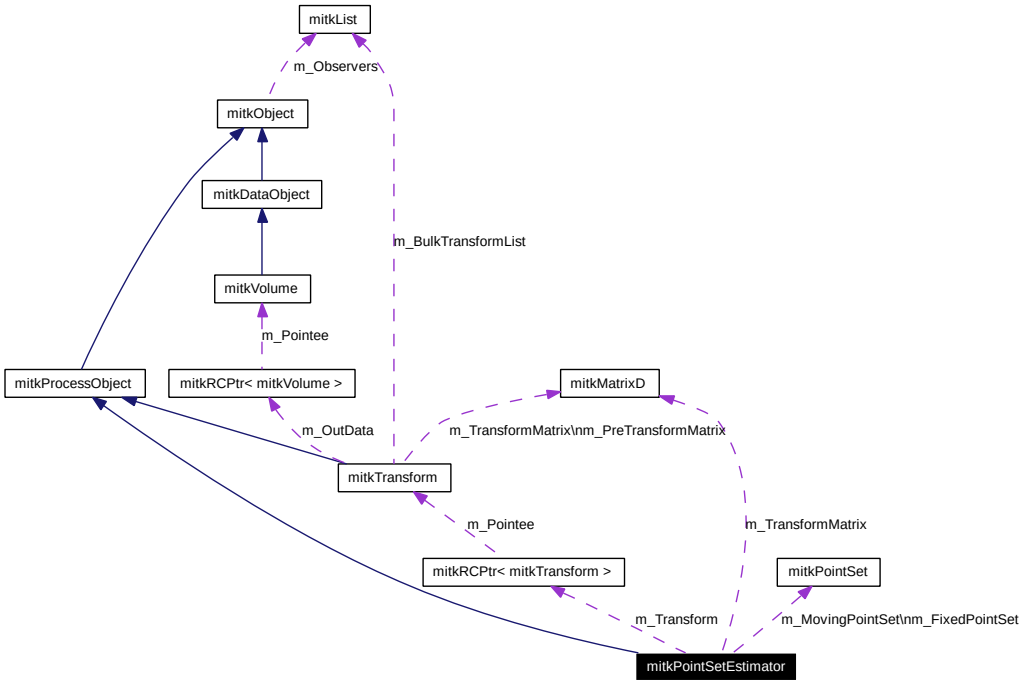
Inherits mitkProcessObject.

Inherited by mitkPointSetKernelTransformEstimator, and mitkPointSetQuaternionEstimator.

Inheritance diagram for mitkPointSetEstimator:



Collaboration diagram for mitkPointSetEstimator:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkPointSetEstimator ()
- virtual ~mitkPointSetEstimator ()
- virtual void SetFixedPointSet (mitkPointSet *fixedPointSet)

- virtual void SetMovingPointSet (mitkPointSet *movingPointSet)
- unsigned int GetNumberOfPoints ()
- void SetPointSetMaskFlag (bool flag)
- void SetTransform (mitkTransform *transform)
- mitkTransform * GetTransform ()
- mitkMatrixType * GetTransformMatrix ()
- const VectorParameterType & GetTransformParameters () const

8.203.1 Detailed Description

mitkPointSetEstimator - an abstract class specifies interface for an estimation method

mitkPointSetEstimator is an abstract class specifies interface for an estimation method. An estimator creates a transform from a set of matches.

8.203.2 Constructor & Destructor Documentation

8.203.2.1 mitkPointSetEstimator::mitkPointSetEstimator ()

Constructor.

8.203.2.2 virtual mitkPointSetEstimator::~mitkPointSetEstimator () [virtual]

Deconstructor.

8.203.3 Member Function Documentation

8.203.3.1 unsigned int mitkPointSetEstimator::GetNumberOfPoints () [inline]

Get the Number of Points.

Returns:
Return the number of points.

8.203.3.2 mitkTransform* mitkPointSetEstimator::GetTransform ()

Get the transform method.

Returns:
Return the pointer to transform object.

8.203.3.3 mitkMatrixType* mitkPointSetEstimator::GetTransformMatrix ()

Get the transform matrix.

Returns:
Return the pointer to the transform matrix.

8.203.3.4 const VectorParameterType& mitkPointSetEstimator::GetTransformParameters () const

Get the transform parameters.

Returns:
Return the vector pointer to the transform parameters.

8.203.3.5 virtual void mitkPointSetEstimator::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkProcessObject.

Reimplemented in mitkPointSetKernelTransformEstimator, and mitkPointSetQuaternionEstimator.

8.203.3.6 virtual void mitkPointSetEstimator::SetFixedPointSet (mitkPointSet * *fixedPointSet*) [inline, virtual]

Set the Fixed Point Set.

Parameters:
fixedPointSet The pointer to the fixed point set.

8.203.3.7 virtual void mitkPointSetEstimator::SetMovingPointSet (mitkPointSet * *movingPointSet*) [inline, virtual]

Set the Moving Point Set.

Parameters:
movingPointSet The pointer to the moving point set.

Reimplemented in mitkPointSetKernelTransformEstimator.

8.203.3.8 void mitkPointSetEstimator::SetPointSetMaskFlag (bool *flag*)

Set the flag of moving point set's mask.

Parameters:
flag Use point set mask in point get/set method if flag is true.

8.203.3.9 void mitkPointSetEstimator::SetTransform (mitkTransform * *transform*) [inline]

Set the transform method.

Parameters:
transform The pointer to transform object.

The documentation for this class was generated from the following file:

- mitkPointSetEstimator.h

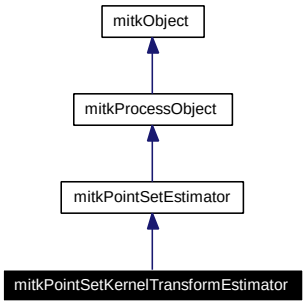
8.204 mitkPointSetKernelTransformEstimator Class Reference

mitkPointSetKernelTransformEstimator -

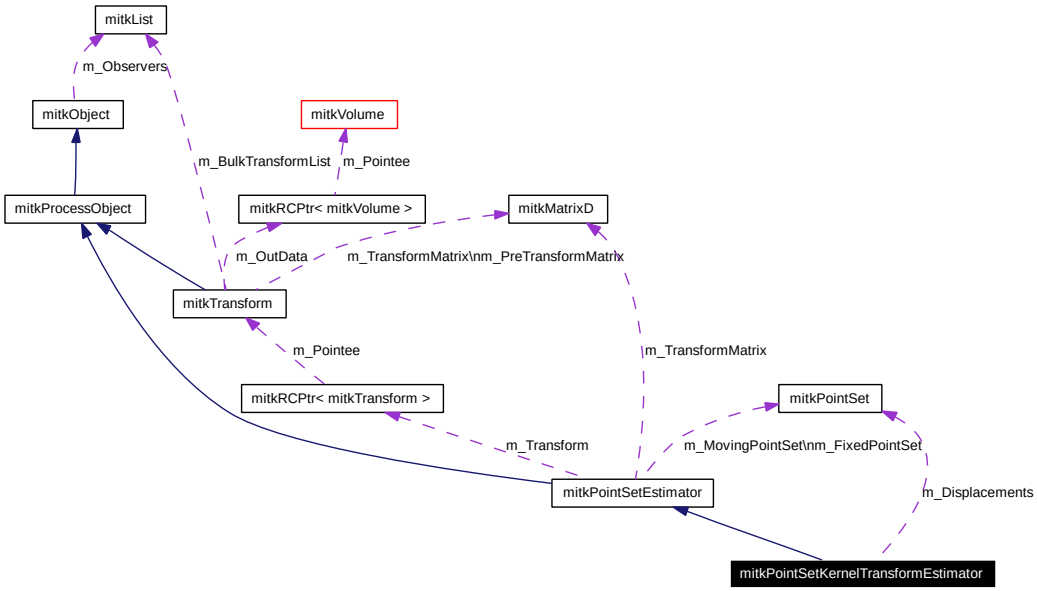
`#include <mitkPointSetKernelTransformEstimator.h>`

Inherits mitkPointSetEstimator.

Inheritance diagram for mitkPointSetKernelTransformEstimator:



Collaboration diagram for mitkPointSetKernelTransformEstimator:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkPointSetKernelTransformEstimator ()
- void SetMovingPointSet (mitkPointSet ∗movingPointSet)

8.204.1 Detailed Description

mitkPointSetKernelTransformEstimator -

8.204.2 Constructor & Destructor Documentation

8.204.2.1 mitkPointSetKernelTransformEstimator::mitkPointSetKernelTransformEstimator ()

Constructor.

8.204.3 Member Function Documentation

8.204.3.1 virtual void mitkPointSetKernelTransformEstimator::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkPointSetEstimator.

8.204.3.2 void mitkPointSetKernelTransformEstimator::SetMovingPointSet (mitkPointSet * *movingPointSet*) [virtual]

Set the Moving Point Set.

Parameters:

movingPointSet The pointer to the moving point set.

Reimplemented from mitkPointSetEstimator.

The documentation for this class was generated from the following file:

- mitkPointSetKernelTransformEstimator.h

8.205 mitkPointSetMetric Class Reference

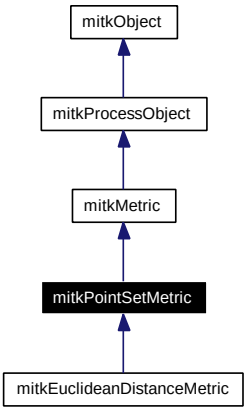
mitkPointSetMetric - an abstract class specifies interface for computing similarity between regions of two point sets

`#include <mitkPointSetMetric.h>`

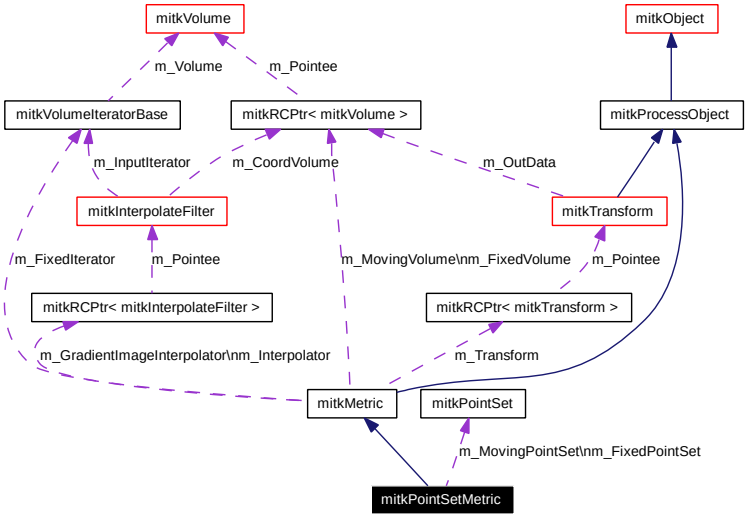
Inherits mitkMetric.

Inherited by mitkEuclideanDistanceMetric.

Inheritance diagram for mitkPointSetMetric:



Collaboration diagram for mitkPointSetMetric:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void SetFixedPointSet (mitkPointSet ∗fixedPointSet)
- virtual void SetMovingPointSet (mitkPointSet ∗movingPointSet)
- virtual mitkPointSet ∗ GetFixedPointSet ()

- virtual mitkPointSet * GetMovingPointSet ()
- unsigned int GetNumberOfFixedPoints ()
- unsigned int GetNumberOfMovingPoints ()
- virtual void SetNumberOfFixedPoints (unsigned int number)
- virtual void SetNumberOfMovingPoints (unsigned int number)
- virtual void Update ()
- virtual void GetClosestPointPairs (vector< int > *closestPointMask)
- virtual double GetPointSetSimilarity (mitkPointSet *p1, mitkPointSet *p2)
- virtual double GetPointSetSimilarity ()

8.205.1 Detailed Description

mitkPointSetMetric - an abstract class specifies interface for computing similarity between regions of two point sets

mitkPointSetMetric is an abstract class specifies interface for computing similarity between two point sets. This Class mainly handles FixedPointSet and MovingPointSet, provide Get & Set method for them. This particular class is the base class for a hierarchy of point set similarity metrics.

8.205.2 Member Function Documentation

8.205.2.1 virtual void mitkPointSetMetric::GetClosestPointPairs (vector< int > * *closestPointMask*) [virtual]

Find correspondent point matches from moving point set to fixed point set.

Parameters:
closestPointMask A mask contains matched points' index from moving point set

Reimplemented in mitkEuclideanDistanceMetric.

8.205.2.2 virtual mitkPointSet* mitkPointSetMetric::GetFixedPointSet () [inline, virtual]

Get fixed point set.

Returns:
Return the pointer to fixed point set.

8.205.2.3 virtual mitkPointSet* mitkPointSetMetric::GetMovingPointSet () [inline, virtual]

Get moving point set.

Returns:
Return the pointer to moving point set.

8.205.2.4 unsigned int mitkPointSetMetric::GetNumberOfFixedPoints () [inline]

Get the Number of Points in Fixed Point Set.

Returns:
Return the number of points.

8.205.2.5 unsigned int mitkPointSetMetric::GetNumberOfMovingPoints () [inline]

Get the Number of Points in Moving Point Set.

Returns:
Return the number of points.

8.205.2.6 virtual double mitkPointSetMetric::GetPointSetSimilarity () [virtual]

Get similarity between fixed point set and moving point set using current transform configure. (with the correspondence : point set mask)

Returns:
The similarity value between two point sets.

Reimplemented in mitkEuclideanDistanceMetric.

8.205.2.7 virtual double mitkPointSetMetric::GetPointSetSimilarity (mitkPointSet * *p1*, mitkPointSet * *p2*) [virtual]

Get similarity between two input point sets p1 and p2 using current transform configure. (without knowing the correspondence)

Parameters:
p1 An input point set.

p2 Another input point set.

Returns:
The similarity value between two point sets.

8.205.2.8 virtual void mitkPointSetMetric::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkMetric.

Reimplemented in mitkEuclideanDistanceMetric.

8.205.2.9 virtual void mitkPointSetMetric::SetFixedPointSet (mitkPointSet * *fixedPointSet*) [inline, virtual]

Set the Fixed Point Set using vector as input.

Parameters:
fixedPointSet The pointer to the fixed point set.

8.205.2.10 virtual void mitkPointSetMetric::SetMovingPointSet (mitkPointSet * *movingPointSet*) [inline, virtual]

Set the Moving Point Set using vector as input.

Parameters:
movingPointSet The pointer to the moving point set.

8.205.2.11 virtual void mitkPointSetMetric::SetNumberOfFixedPoints (unsigned int *number*) [inline, virtual]

Set the Number of Points in Fixed Point Set.

Parameters:
number Specify the number of points in Fixed Point Set.

8.205.2.12 virtual void mitkPointSetMetric::SetNumberOfMovingPoints (unsigned int *number*) [inline, virtual]

Set the Number of Points in Moving Point Set.

Parameters:
number Specify the number of points in Moving Point Set.

8.205.2.13 virtual void mitkPointSetMetric::Update () [virtual]

Initialize the Metric by making sure that all the components are present and plugged together correctly. Perform before Run() function.

Reimplemented from mitkMetric.

The documentation for this class was generated from the following file:

- mitkPointSetMetric.h

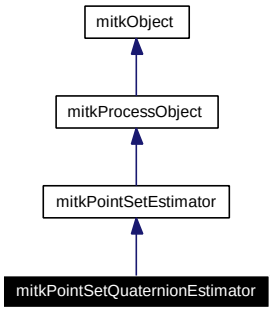
8.206 mitkPointSetQuaternionEstimator Class Reference

mitkPointSetQuaternionEstimator - a concrete class for transform estimation based on Unit Quaternion method.

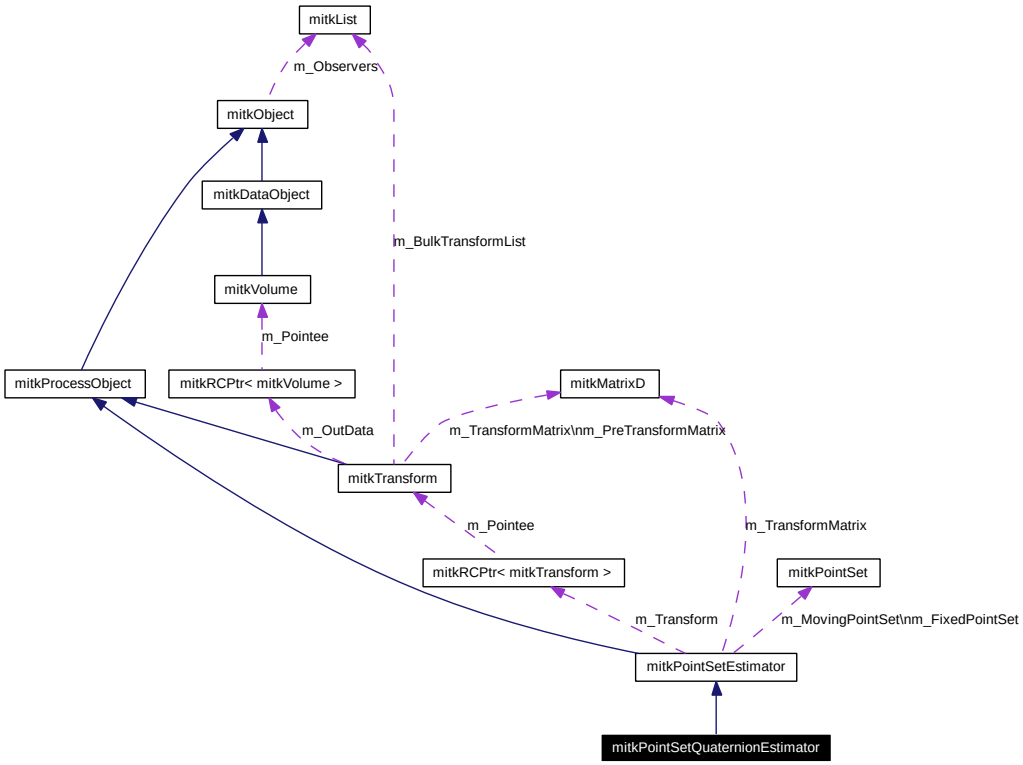
`#include <mitkPointSetQuaternionEstimator.h>`

Inherits mitkPointSetEstimator.

Inheritance diagram for mitkPointSetQuaternionEstimator:



Collaboration diagram for mitkPointSetQuaternionEstimator:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- mitkPointSetQuaternionEstimator ()

8.206.1 Detailed Description

mitkPointSetQuaternionEstimator - a concrete class for transform estimation based on Unit Quaternion method.

mitkPointSetQuaternionEstimator is a concrete class for transform estimation based on Unit Quaternion method. The solution is based on Berthold K. P. Horn (1987), "Closed-form solution of absolute orientation using unit quaternions," Journal of the Optical Society of America A, 4:629-642

8.206.2 Constructor & Destructor Documentation

8.206.2.1 mitkPointSetQuaternionEstimator::mitkPointSetQuaternionEstimator ()

Constructor.

8.206.3 Member Function Documentation

8.206.3.1 virtual void mitkPointSetQuaternionEstimator::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkPointSetEstimator.

The documentation for this class was generated from the following file:

- mitkPointSetQuaternionEstimator.h

8.207 mitkPointSetRegistrationFilter Class Reference

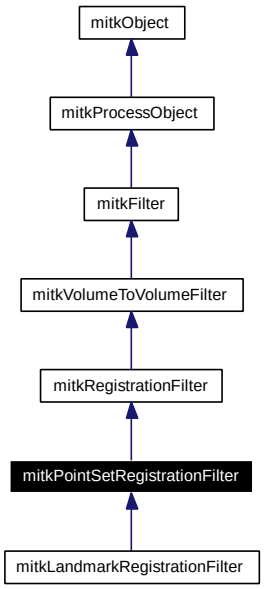
mitkPointSetRegistrationFilter - a class for the implementation of point sets based registration framework.

`#include <mitkPointSetRegistrationFilter.h>`

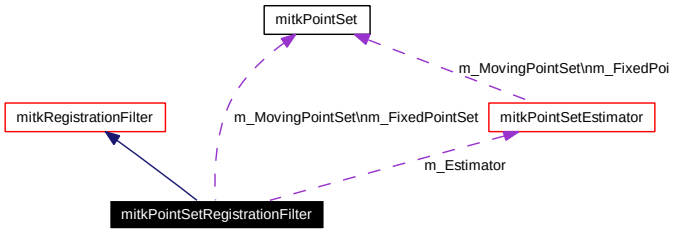
Inherits mitkRegistrationFilter.

Inherited by mitkLandmarkRegistrationFilter.

Inheritance diagram for mitkPointSetRegistrationFilter:



Collaboration diagram for mitkPointSetRegistrationFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkPointSetRegistrationFilter ()
- void SetFixedPointSet (mitkPointSet *fixedPointSet)
- void SetMovingPointSet (mitkPointSet *movingPointSet)
- void SetFixedPointSet (vector< double > *fixedPointSet, unsigned int dim=3)
- void SetMovingPointSet (vector< double > *movingPointSet, unsigned int dim=3)
- void SetFixedPointSet (double *fixedPointSet, unsigned int arraylength, unsigned int dim=3)
- void SetMovingPointSet (double *movingPointSet, unsigned int arraylength, unsigned int dim=3)

- unsigned int GetNumberOfFixedPoints ()
- unsigned int GetNumberOfMovingPoints ()
- virtual mitkTransform * GetTransform ()
- virtual mitkMetric * GetMetric ()
- virtual mitkOptimizer * GetOptimizer ()
- mitkPointSetEstimator * GetEstimator ()
- void SetEstimator (mitkPointSetEstimator *estimator)

8.207.1 Detailed Description

mitkPointSetRegistrationFilter - a class for the implementation of point sets based registration framework.

mitkPointSetRegistrationFilter is a class for the implementation of point sets based registration framework. There are four generic components in point set registration framework. A optimizer matches two point sets and gets their correspondences. A estimator computes transform parameters directly based on the matches. A transform performs the transform to moving point set. A metric gets the similarity between the two point sets.

8.207.2 Constructor & Destructor Documentation

8.207.2.1 mitkPointSetRegistrationFilter::mitkPointSetRegistrationFilter ()

Constructor.

8.207.3 Member Function Documentation

8.207.3.1 mitkPointSetEstimator* mitkPointSetRegistrationFilter::GetEstimator ()

Get the estimation method.

Returns:
Return the pointer to the estimator object.

8.207.3.2 virtual mitkMetric* mitkPointSetRegistrationFilter::GetMetric () [virtual]

Get the metric method.

Returns:
Return the pointer to the metric object.

Reimplemented from mitkRegistrationFilter.

8.207.3.3 unsigned int mitkPointSetRegistrationFilter::GetNumberOfFixedPoints () [inline]

Get the Number of Points in Fixed Point Set.

Returns:
Return the number of points.

8.207.3.4 unsigned int mitkPointSetRegistrationFilter::GetNumberOfMovingPoints () [inline]

Get the Number of Points in Moving Point Set.

Returns:
Return the number of points.

8.207.3.5 virtual mitkOptimizer* mitkPointSetRegistrationFilter::GetOptimizer () [virtual]

Get the optimization method.

Returns:
Return the pointer to the optimizer object.

Reimplemented from mitkRegistrationFilter.

8.207.3.6 virtual mitkTransform* mitkPointSetRegistrationFilter::GetTransform () [virtual]

Get the transform method.

Returns:
Return the pointer to the transform object.

Reimplemented from mitkRegistrationFilter.

8.207.3.7 virtual void mitkPointSetRegistrationFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRegistrationFilter.

Reimplemented in mitkLandmarkRegistrationFilter.

8.207.3.8 void mitkPointSetRegistrationFilter::SetEstimator (mitkPointSetEstimator * *estimator*) [inline]

Set the estimation method

Parameters:
estimator The pointer to the estimator.

8.207.3.9 void mitkPointSetRegistrationFilter::SetFixedPointSet (double * *fixedPointSet*, unsigned int *arraylength*, unsigned int *dim* = 3)

Set the Fixed Point Set using 1d array as input.

Parameters:

fixedPointSet The pointer to the fixed point set.

arraylength The size of point set array.

dim The point set dimension.

8.207.3.10 void mitkPointSetRegistrationFilter::SetFixedPointSet (vector< double > * *fixedPointSet*, unsigned int *dim* = 3)

Set the Fixed Point Set using vector as input.

Parameters:

fixedPointSet The pointer to the fixed point set.

dim The point set dimension.

8.207.3.11 void mitkPointSetRegistrationFilter::SetFixedPointSet (mitkPointSet * *fixedPointSet*)

Set the Fixed Point Set using vector as input.

Parameters:

fixedPointSet The pointer to the fixed point set.

8.207.3.12 void mitkPointSetRegistrationFilter::SetMovingPointSet (double * *movingPointSet*, unsigned int *arraylength*, unsigned int *dim* = 3)

Set the Moving Point Set using 1d array as input.

Parameters:

movingPointSet The pointer to the moving point set.

arraylength The size of point set array.

dim The point set dimension.

8.207.3.13 void mitkPointSetRegistrationFilter::SetMovingPointSet (vector< double > * *movingPointSet*, unsigned int *dim* = 3)

Set the Moving Point Set using vector as input.

Parameters:

movingPointSet The pointer to the moving point set.

dim The point set dimension.

8.207.3.14 void mitkPointSetRegistrationFilter::SetMovingPointSet (mitkPointSet * *movingPointSet*)

Set the Moving Point Set using vector as input.

Parameters:

movingPointSet The pointer to the moving point set.

The documentation for this class was generated from the following file:

- mitkPointSetRegistrationFilter.h

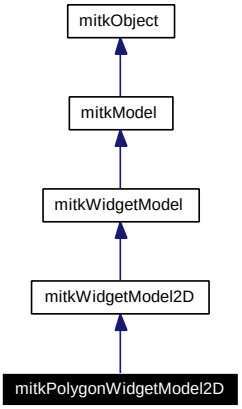
8.208 mitkPolygonWidgetModel2D Class Reference

mitkPolygonWidgetModel2D - a 2D widget for displaying an arbitrary polygon in an image scene

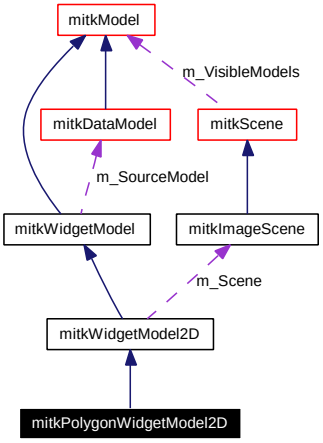
`#include <mitkPolygonWidgetModel2D.h>`

Inherits mitkWidgetModel2D.

Inheritance diagram for mitkPolygonWidgetModel2D:



Collaboration diagram for mitkPolygonWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- void AddPoint (int sx, int sy)
- void AddPoint (float x, float y)
- void AddPoint (float point[2])
- void MoveCurrentPoint (int sx, int sy)

- void MoveCurrentPoint (float x, float y)
- void MoveCurrentPoint (float point[2])
- void SetUnitName (const string &name)
- const string & GetUnitName ()
- float GetArea ()
- float GetPerimeter ()
- bool GetCurrentPoint (float &x, float &y)
- bool GetCurrentPoint (int &ix, int &iy)
- int GetPointsNumber ()
- bool GetPoint (int idx, float &x, float &y)
- bool GetPoint (int idx, int &ix, int &iy)
- virtual mitkVolume * GetRegionMask ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.208.1 Detailed Description

mitkPolygonWidgetModel2D - a 2D widget for displaying an arbitrary polygon in an image scene

mitkPolygonWidgetModel2D a 2D widget for displaying an arbitrary polygon in an image scene. It can respond the mouse events and return the current area of this arbitrary polygon. It is supposed to be attached to a 2D data model (e.g. mitkImageModel) and add to a 2D scene (e.g. mitkImageScene), and in other conditions the display could be improper.

8.208.2 Member Function Documentation

8.208.2.1 virtual void mitkPolygonWidgetModel2D::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.208.2.2 virtual void mitkPolygonWidgetModel2D::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.208.2.3 virtual void mitkPolygonWidgetModel2D::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.208.2.4 void mitkPolygonWidgetModel2D::AddPoint (float *point*[2]) [inline]

Add point.

Parameters:

point[0] x-coordinate of the point in object space

point[1] y-coordinate of the point in object space

8.208.2.5 void mitkPolygonWidgetModel2D::AddPoint (float *x*, float *y*)

Add point.

Parameters:

x x-coordinate of the point in object space

y y-coordinate of the point in object space

8.208.2.6 void mitkPolygonWidgetModel2D::AddPoint (int *sx*, int *sy*)

Add point.

Parameters:

sx x-coordinate of the point in screen space

sy y-coordinate of the point in screen space

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.208.2.7 float mitkPolygonWidgetModel2D::GetArea ()

Get area of this polygon.

Returns:

Return the area of this polygon.

8.208.2.8 bool mitkPolygonWidgetModel2D::GetCurrentPoint (int & *ix*, int & *iy*)

Get currently picked point in the original image.

Parameters:

ix return x-coordinate of the picked point

iy return y-coordinate of the picked point

Returns:

Return false if no point was picked. Otherwise return true.

8.208.2.9 bool mitkPolygonWidgetModel2D::GetCurrentPoint (float & *x*, float & *y*)

Get currently picked point in object space (affected by pixel spacing).

Parameters:

x return x-coordinate of the picked point

y return y-coordinate of the picked point

Returns:

Return false if no point was picked. Otherwise return true.

8.208.2.10 float mitkPolygonWidgetModel2D::GetPerimeter ()

Get perimeter of this polygon.

Returns:
Return the perimeter of this polygon.

8.208.2.11 bool mitkPolygonWidgetModel2D::GetPoint (int *idx*, int & *ix*, int & *iy*)

Get the integral coordinates of the ith point in the original image.

Parameters:
idx the index of the point to get
ix return the x-coordinate
iy return the y-coordinate

Returns:
Return false if no such point was found, otherwise return true.

8.208.2.12 bool mitkPolygonWidgetModel2D::GetPoint (int *idx*, float & *x*, float & *y*)

Get the coordinates of the ith point in object space (affected by pixel spacing).

Parameters:
idx the index of the point to get
x return the x-coordinate
y return the y-coordinate

Returns:
Return false if no such point was found, otherwise return true.

8.208.2.13 int mitkPolygonWidgetModel2D::GetPointsNumber ()

Get the number of points.

Returns:
Return the number of points.

8.208.2.14 virtual mitkVolume* mitkPolygonWidgetModel2D::GetRegionMask () [virtual]

Get mask image where the value of widget region is 255 and the rest is 0.

Returns:
Return a pointer of mitkVolume object contains the mask image.

Note:
The returned object pointer should be deleted properly by yourself.

Reimplemented from mitkWidgetModel2D.

8.208.2.15 const string& mitkPolygonWidgetModel2D::GetUnitName () [inline]

Get name string of unit.

Returns:
Return a constant reference to a string contains the name of the length unit this line uses

8.208.2.16 void mitkPolygonWidgetModel2D::MoveCurrentPoint (float *point*[2]) [inline]

Move current point.

Parameters:
point[0] x-coordinate of the point in object space
point[1] y-coordinate of the point in object space

8.208.2.17 void mitkPolygonWidgetModel2D::MoveCurrentPoint (float *x*, float *y*)

Move current point.

Parameters:
x x-coordinate of the point in object space
y y-coordinate of the point in object space

8.208.2.18 void mitkPolygonWidgetModel2D::MoveCurrentPoint (int *sx*, int *sy*)

Move current point.

Parameters:
sx x-coordinate of the point in screen space
sy y-coordinate of the point in screen space

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.208.2.19 virtual void mitkPolygonWidgetModel2D::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:
names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.208.2.20 virtual void mitkPolygonWidgetModel2D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel2D.

8.208.2.21 virtual void mitkPolygonWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.208.2.22 virtual int mitkPolygonWidgetModel2D::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.208.2.23 void mitkPolygonWidgetModel2D::SetUnitName (const string & *name*) [inline]

Set name string of unit.

Parameters:
name a constant reference to a string contains the name of the length unit (e.g. mm) this line uses

The documentation for this class was generated from the following file:

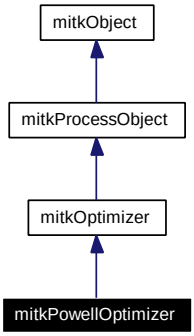
- mitkPolygonWidgetModel2D.h

8.209 mitkPowellOptimizer Class Reference

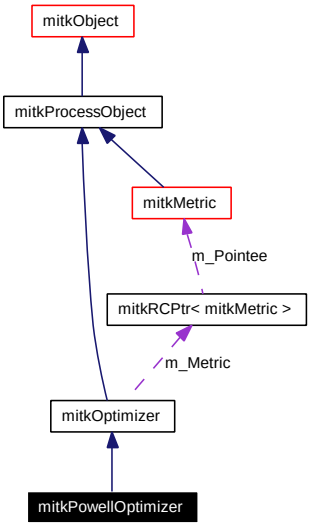
`#include <mitkPowellOptimizer.h>`

Inherits mitkOptimizer.

Inheritance diagram for mitkPowellOptimizer:



Collaboration diagram for mitkPowellOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkPowellOptimizer ()
- void SetStepTolerance (double tolerance)
- void SetValueTolerance (double tolerance)
- void SetMaximumLineIterations (unsigned int mli)
- virtual void SetCurrentStepLength (double length)

8.209.1 Detailed Description

mitkPowellOptimizer - a concrete class for implementation of the Powell optimization algorithm.

mitkPowellOptimizer is a concrete class for implementation of the Powell optimization algorithm. For an N-dimensional parameter space, each Powell iteration minimizes(maximizes) the function in N (initially orthogonal) directions. Typically only 2-5 iterations are required.

8.209.2 Constructor & Destructor Documentation

8.209.2.1 mitkPowellOptimizer::mitkPowellOptimizer ()

Constructor.

8.209.3 Member Function Documentation

8.209.3.1 virtual void mitkPowellOptimizer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkOptimizer.

8.209.3.2 virtual void mitkPowellOptimizer::SetCurrentStepLength (double *length*) [inline, virtual]

Set the stepLength for the (scaled) spacing of the sampling of parameter space.

Parameters:
length The step length value.

8.209.3.3 void mitkPowellOptimizer::SetMaximumLineIterations (unsigned int *mli*) [inline]

Set the maximum number of line search iterations.

Parameters:
mli The maximum line search iterations value.

8.209.3.4 void mitkPowellOptimizer::SetStepTolerance (double *tolerance*) [inline]

Set the step tolerance. Once the local extreme is known to be within this distance of the current parameter values, optimization terminates.

Parameters:
tolerance The step tolerance value.

8.209.3.5 void mitkPowellOptimizer::SetValueTolerance (double *tolerance*) [inline]

Set the function tolerance. Once this current cost function value is known to be within this tolerance of the cost function value at the local extreme, optimization terminates.

Parameters:
tolerance The function tolerance value.

The documentation for this class was generated from the following file:

- mitkPowellOptimizer.h

8.210 mitkProcessObject Class Reference

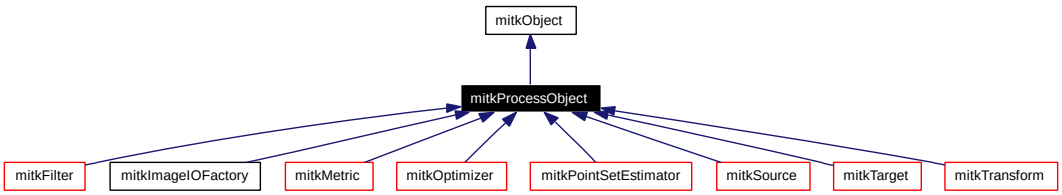
mitkProcessObject - abstract base class for source, filter(algorithm) and mapper

`#include <mitkProcessObject.h>`

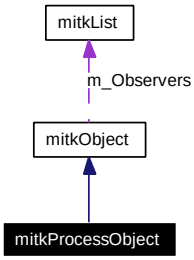
Inherits mitkObject.

Inherited by mitkFilter, mitkImageIOFactory, mitkMetric, mitkOptimizer, mitkPointSetEstimator, mitkSource, mitkTarget, and mitkTransform.

Inheritance diagram for mitkProcessObject:



Collaboration diagram for mitkProcessObject:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- unsigned long GetProgressRate ()
- void SetProgressRateMax (unsigned long m)

8.210.1 Detailed Description

mitkProcessObject - abstract base class for source, filter(algorithm) and mapper

mitkProcessObject is an abstract object that specifies behavior and interface of source, filter and mapper.

8.210.2 Member Function Documentation

8.210.2.1 unsigned long mitkProcessObject::GetProgressRate () `[inline]`

Get current rate of process of the running process.

Returns:
Return current rate of process.

8.210.2.2 virtual void mitkProcessObject::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkAnalyzeImageReader, mitkAnalyzeImageWriter, mitkBinaryFilter, mitkBMPReader, mitkBMPWriter, mitkCacheVolumeReader, mitkCacheVolumeWriter, mitkDICOMInfoReader, mitkDICOMReader, mitkDICOMWriter, mitkDiffusionFilter, mitkFFTFilter, mitkFilter, mitkGaussianDerivativeImageFilter, mitkGaussianFilter, mitkImageIOFactory, mitkInfoReader, mitkJPEGReader, mitkJPEGWriter, mitkMeshReader, mitkMeshToMeshFilter, mitkMeshWriter, mitkMetaImageReader, mitkMetaImageWriter, mitkNetCDFImageReader, mitkNetCDFImageWriter, mitkNeumannBoundary, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkPLYReader, mitkQEMSimplification, mitkRawFilesReader, mitkRawFilesWriter, mitkRawReader, mitkRawWriter, mitkReader, mitkRGBToGrayFilter, mitkSheetFilter, mitkSobelEdgeDetectFilter, mitkSource, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkSTLReader, mitkTarget, mitkTIFFReader, mitkTIFFWriter, mitkTriangleMeshSimplification, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeReader, mitkVolumeResizeFilter, mitkVolumeResliceFilter, mitkVolumeToMeshFilter, mitkVolumeToVolumeFilter, mitkVolumeWriter, mitkWriter, mitk3DUSPixelBasedMT, mitk3DUSReconstructionFilter, mitkCBFDKReconCF, mitkCBFDKReconCFCPU, mitkCBTFDKReconCF, mitkConeBeamReconFromHelix, mitkCTReconstructionFilter, mitkFanBeamReconFromCirle, mitkFbFbp, mitkFBFBPReconCF, mitkFBFBPReconCFCPU, mitkFbFbpSpeedUsingCpu, mitkFDK, mitkFDKSpeedUsingCpu, mitkParallelBeamReconFromCirle, mitkPbFbp, mitkPBFBPReconCF, mitkProjectDataToVolumeFilter, mitkRawDataReader, mitkRawDataToVolumeFilter, mitkRawDataWriter, mitkRawFilesReader, mitkTFDK, mitkTrackedBscanDataReader, mitkTrackedBscanDataWriter, mitkAffineTransform, mitkAmoebaOptimizer, mitkBSplineInterpolateFilter, mitkCubicInterpolateFilter, mitkEuclideanDistanceMetric, mitkGradientDescentOptimizer, mitkInterpolateFilter, mitkIterativeClosestPointOptimizer, mitkLandmarkRegistrationFilter, mitkLBFGSBOptimizer, mitkLBFGSOptimizer, mitkLinearInterpolateFilter, mitkMeanSquaresMetric, mitkMetric, mitkNearestNeighborInterpolateFilter, mitkNormalizedCorrelationMetric, mitkOptimizer, mitkPointSetEstimator, mitkPointSetKernelTransformEstimator, mitkPointSetMetric, mitkPointSetQuaternionEstimator, mitkPointSetRegistrationFilter, mitkPowellOptimizer, mitkRegistrationFilter, mitkRigidTransform, mitkScaleTransform, mitkSimilarityTransform, mitkSubtractImageFilter, mitkThinPlateSplineTransform, mitkTransform, mitkTranslationTransform, mitkVersorRigid3DTransform, mitkVersorSimilarity3DTransform, mitkConnectivityLabelFilter, mitkLiveWireImageFilter, mitkOtsuSegmentationFilter, mitkRegionGrowImageFilter, mitkSeedFillFilter, mitkThresholdSegmentationFilter, mitkBinMarchingCubes, mitkMarchingCubes, mitkCBFDKReconCFGPU, mitkFBFBPReconCFGPU, mitkFbFbpSpeedUsingGpu, mitkFDKSpeedUsingGpu, and mitk3DUSPixelBasedGPU.

8.210.2.3 void mitkProcessObject::SetProgressRateMax (unsigned long *m*) [inline]

Set maximum value the rate of process can reach.

Parameters:
m the maximum value of the rate of process

The documentation for this class was generated from the following file:

- mitkProcessObject.h

8.211 mitkProjectData Class Reference

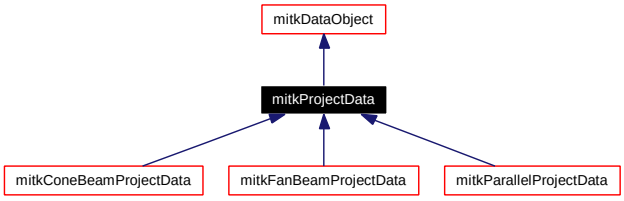
mitkProjectData - abstract class to represent project data.

`#include <mitkProjectData.h>`

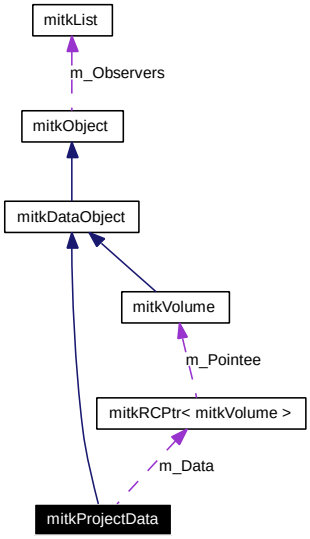
Inherits mitkDataObject.

Inherited by mitkConeBeamProjectData, mitkFanBeamProjectData, and mitkParallelProjectData.

Inheritance diagram for mitkProjectData:



Collaboration diagram for mitkProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void const * GetData () const =0
- virtual void * GetData ()=0
- int GetDataTypeSize () const
- void SetDataType (int data_type)
- int GetDataType () const
- void SetBeginAngle (double angle)
- double GetBeginAngle () const
- void SetEndAngle (double angle)
- double GetEndAngle () const

- void SetAngleNum (int length)
- int GetAngleNum () const
- float GetPitch () const
- void SetPitch (float m)
- float GetInitZPostion () const
- void SetInitZPosition (float m)
- virtual void Initialize ()

8.211.1 Detailed Description

mitkProjectData - abstract class to represent project data.

mitkProjectData - abstract class to represent project data.

8.211.2 Member Function Documentation

8.211.2.1 int mitkProjectData::GetAngleNum () const [inline]

Get number of rotate angle

8.211.2.2 double mitkProjectData::GetBeginAngle () const [inline]

get beginning rotate angle which is the angle between detector and +x axis. unit are degree

8.211.2.3 virtual void* mitkProjectData::GetData () [pure virtual]

Get data pointer of the project data (changeable).

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type

Implemented in mitkConeBeamProjectData, mitkFanBeamProjectData, and mitkParallelProjectData.

8.211.2.4 virtual void const* mitkProjectData::GetData () const [pure virtual]

Get data pointer of the project data (unchangeable).

Returns:
Return a void pointer to const data.

Note:
The returned type is void const *, it must be converted to some useful data type

Implemented in mitkConeBeamProjectData, mitkFanBeamProjectData, and mitkParallelProjectData.

8.211.2.5 int mitkProjectData::GetDataType () const [inline]

Get data type of this volume. MITK supports various data type.

8.211.2.6 int mitkProjectData::GetDataTypeSize () const [inline]

Get the size of the data type in bytes.

Returns:
Return the size of the data type in bytes.

8.211.2.7 double mitkProjectData::GetEndAngle () const [inline]

get end rotate angle which is the angle between detector and +x axis. unit are degree

8.211.2.8 float mitkProjectData::GetInitZPostion () const [inline]

only available for helical,else z0 =0

8.211.2.9 float mitkProjectData::GetPitch () const [inline]

only available for helical,else pitch=0

8.211.2.10 virtual void mitkProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Implements mitkDataObject.

Reimplemented in mitkConeBeamProjectData, mitkFanBeamProjectData, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkParallelProjectData, and mitkRawConeBeamProjectData.

8.211.2.11 virtual void mitkProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkDataObject.

Reimplemented in mitkConeBeamProjectData, mitkFanBeamProjectData, mitkICConeBeamProjectData, mitkICFanBeamProjectData, mitkICParallelProjectData, mitkOoCConeBeamProjectData, mitkOoCFanBeamProjectData, mitkOoCParallelProjectData, mitkParallelProjectData, and mitkRawConeBeamProjectData.

8.211.2.12 void mitkProjectData::SetAngleNum (int *length*) [inline]

Set number of rotate angle

8.211.2.13 void mitkProjectData::SetBeginAngle (double *angle*) [inline]

Set beginning rotate angle which is the angle between detector and +x axis. unit must be degree

8.211.2.14 void mitkProjectData::SetDataType (int *data_type*)

Set data type of this volume. MITK supports various data type.

Parameters:

data_type Its valid value and meaning is shown as follows:

MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

8.211.2.15 void mitkProjectData::SetEndAngle (double *angle*) [inline]

Set end rotate angle which is the angle between detector and +x axis. unit must be degree

8.211.2.16 void mitkProjectData::SetInitZPosition (float *m*) [inline]

only available for helical,else z0=0

8.211.2.17 void mitkProjectData::SetPitch (float *m*) [inline]

only available for helical,else pitch=0

The documentation for this class was generated from the following file:

- mitkProjectData.h

8.212 mitkProjectDataToVolumeFilter Class Reference

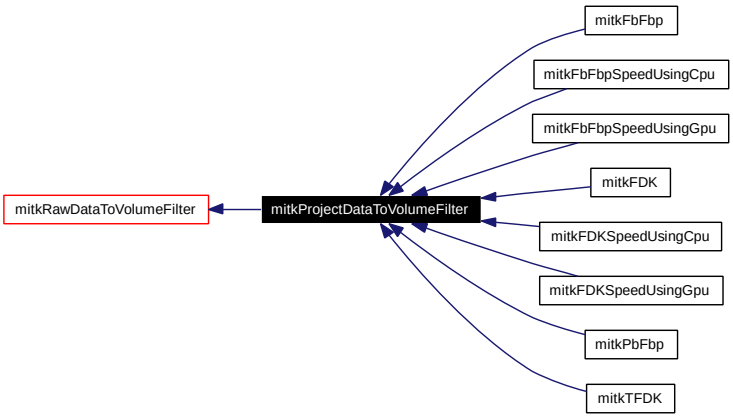
mitkProjectDataToVolumeFilter - abstract class for forward construction

`#include <mitkProjectDataToVolumeFilter.h>`

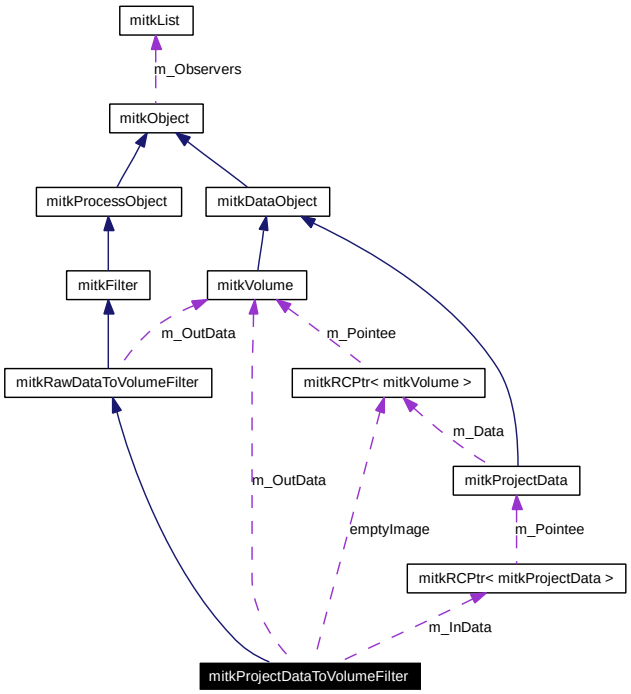
Inherits mitkRawDataToVolumeFilter.

Inherited by mitkFbFbp, mitkFbFbpSpeedUsingCpu, mitkFbFbpSpeedUsingGpu, mitkFDK, mitkFDKSpeedUsingCpu, mitkFDKSpeedUsingGpu, mitkPbFbp, and mitkTFDK.

Inheritance diagram for mitkProjectDataToVolumeFilter:



Collaboration diagram for mitkProjectDataToVolumeFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInterpolationModel (int m)
- int GetInterpolationModel ()
- void SetInput (mitkProjectData *inData)
- mitkProjectData * GetInput ()
- mitkVolume * GetOutput ()
- void SetOutput (mitkVolume *m)
- void SetIsLogOperation (bool m)
- bool GetIsLogOperation ()
- void SetEmptyScanImage (mitkVolume *m)
- mitkVolume * GetEmptyScanImage ()
- void SetOoCSupport (char const *diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufSliceNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)

8.212.1 Detailed Description

mitkProjectDataToVolumeFilter - abstract class for forward construction

mitkProjectDataToVolumeFilter - abstract class for forward construction

8.212.2 Member Function Documentation

8.212.2.1 mitkVolume* mitkProjectDataToVolumeFilter::GetEmptyScanImage () [inline]

Get empty scan image usefull when isLogOperation is false

8.212.2.2 mitkProjectData* mitkProjectDataToVolumeFilter::GetInput () [inline]

Get the input ProjectData

Returns:
Return the input ProjectData

8.212.2.3 int mitkProjectDataToVolumeFilter::GetInterpolationModel () [inline]

Get the interpolation model

8.212.2.4 bool mitkProjectDataToVolumeFilter::GetIsLogOperation () [inline]

Get whether log operation is done

Returns:
Return the output volume

8.212.2.5 mitkVolume* mitkProjectDataToVolumeFilter::GetOutput () [virtual]

Get the output volume

Returns:
Return the output volume

Implements mitkRawDataToVolumeFilter.

8.212.2.6 virtual void mitkProjectDataToVolumeFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRawDataToVolumeFilter.

Reimplemented in mitkFbFbp, mitkFbFbpSpeedUsingCpu, mitkFDK, mitkFDKSpeedUsingCpu, mitkPbFbp, mitkTFDK, mitkFbFbpSpeedUsingGpu, and mitkFDKSpeedUsingGpu.

8.212.2.7 void mitkProjectDataToVolumeFilter::SetEmptyScanImage (mitkVolume * *m*) [inline]

Set empty scan image usefull when isLogOperation is false

8.212.2.8 void mitkProjectDataToVolumeFilter::SetInput (mitkProjectData * *inData*) [inline]

Set the input ProjectData

Parameters:
inData Specify the input ProjectData

8.212.2.9 void mitkProjectDataToVolumeFilter::SetInterpolationModel (int *m*) [inline]

Set the interpolation model

8.212.2.10 void mitkProjectDataToVolumeFilter::SetIsLogOperation (bool *m*) [inline]

Set whether log operation is done

Returns:
Return the output volume

8.212.2.11 void mitkProjectDataToVolumeFilter::SetOoCSupport (char const * *diskPath* = mitkOoCGlobalSettings::DiskCachePath, unsigned int *bufSliceNum* = mitkOoCGlobalSettings::BufferedSliceNumber, bool *supportOoC* = true)

Let the reader support out-of-core data.

Parameters:

diskPath the path in the disk to cache the volume data

bufSliceNum the number of slices to cache in the main memory

supportOoC whether to turn on out-of-core support

Note:

The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

Reimplemented from mitkRawDataToVolumeFilter.

8.212.2.12 void mitkProjectDataToVolumeFilter::SetOutput (mitkVolume * *m*) [inline]

Set the output volume

The documentation for this class was generated from the following file:

- mitkProjectDataToVolumeFilter.h

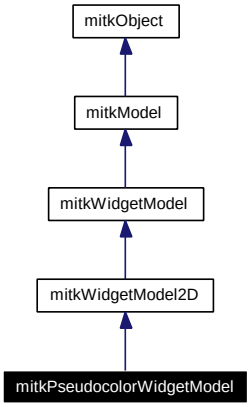
8.213 mitkPseudocolorWidgetModel Class Reference

mitkPseudocolorWidgetModel - a 2D widget for displaying pseudocolor image

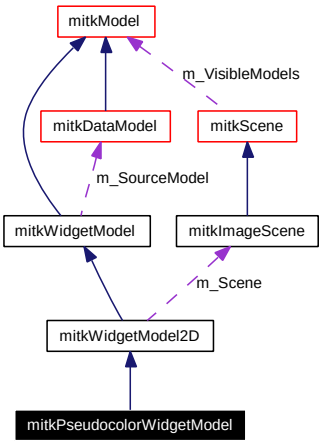
`#include <mitkPseudocolorWidgetModel.h>`

Inherits mitkWidgetModel2D.

Inheritance diagram for mitkPseudocolorWidgetModel:



Collaboration diagram for mitkPseudocolorWidgetModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- void SetStartPoint (int x, int y)
- void SetMovePoint (int x, int y)
- void SetEndPoint (int x, int y)
- int GetWidth ()

- int GetHeight ()
- int GetLeft ()
- int GetRight ()
- int GetBottom ()
- int GetTop ()
- void GetStartPoint (int &sx, int &sy)
- void GetStartPoint (int p[2])
- void GetEndPoint (int &ex, int &ey)
- void GetEndPoint (int p[2])
- bool IsValid ()
- int SetColorTable (unsigned char table[ ][3], int colorNum=COLOR_TABLE_SIZE)
- void ResetColorTable ()
- int GetColorTable (unsigned char table[ ][3], int colorNum=COLOR_TABLE_SIZE)
- virtual mitkVolume * GetRegionMask ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.213.1 Detailed Description

mitkPseudocolorWidgetModel - a 2D widget for displaying pseudocolor image

mitkPseudocolorWidgetModel is a 2D widget for displaying pseudocolor image. It can respond the mouse events and display the pseudocolor image of current rectangle part and return the width and height. It is supposed to be attached to a 2D data model (e.g. mitkImageModel) and add to a 2D scene (e.g. mitkImageScene), and in other conditions the display could be improper.

8.213.2 Member Function Documentation

8.213.2.1 virtual void mitkPseudocolorWidgetModel::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.213.2.2 virtual void mitkPseudocolorWidgetModel::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.213.2.3 virtual void mitkPseudocolorWidgetModel::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.213.2.4 int mitkPseudocolorWidgetModel::GetBottom () [inline]

Get the bottom of the pseudocolor rectangle.

Returns:

Return the bottom of the pseudocolor rectangle.

8.213.2.5 int mitkPseudocolorWidgetModel::GetColorTable (unsigned char *table*[ ][3], int *colorNum* = COLOR_TABLE_SIZE)

Get color table.

Parameters:

table an unsigned char array to contain the color table and the arrangement of the color components is 'RGBRGBRGB...'

colorNum the number of colors the buffer table[][3] can contain

Returns:

Return the number of colors gotten actually.

8.213.2.6 void mitkPseudocolorWidgetModel::GetEndPoint (int *p*[2]) [inline]

Get the end point of the pseudocolor rectangle.

Parameters:

p[0] return the x-coordinate of the start point

p[1] return the y-coordinate of the start point

8.213.2.7 void mitkPseudocolorWidgetModel::GetEndPoint (int & *ex*, int & *ey*) [inline]

Get the end point of the pseudocolor rectangle.

Parameters:

ex return the x-coordinate of the end point

ey return the y-coordinate of the end point

8.213.2.8 int mitkPseudocolorWidgetModel::GetHeight () [inline]

Get the height of the pseudocolor rectangle.

Returns:

Return the height of the pseudocolor rectangle.

8.213.2.9 int mitkPseudocolorWidgetModel::GetLeft () [inline]

Get the left of the pseudocolor rectangle.

Returns:

Return the left of the pseudocolor rectangle.

8.213.2.10 virtual mitkVolume* mitkPseudocolorWidgetModel::GetRegionMask () [virtual]

Get mask image where the value of widget region is 255 and the rest is 0.

Returns:

Return a pointer of mitkVolume object contains the mask image.

Note:

The returned object pointer should be deleted properly by yourself.

Reimplemented from mitkWidgetModel2D.

8.213.2.11 int mitkPseudocolorWidgetModel::GetRight () [inline]

Get the right of the pseudocolor rectangle.

Returns:

Return the right of the pseudocolor rectangle.

8.213.2.12 void mitkPseudocolorWidgetModel::GetStartPoint (int *p*[2]) [inline]

Get the start point of the pseudocolor rectangle.

Parameters:

p[0] return the x-coordinate of the start point

p[1] return the y-coordinate of the start point

8.213.2.13 void mitkPseudocolorWidgetModel::GetStartPoint (int & *sx*, int & *sy*) [inline]

Get the start point of the pseudocolor rectangle.

Parameters:

sx return the x-coordinate of the start point

sy return the y-coordinate of the start point

8.213.2.14 int mitkPseudocolorWidgetModel::GetTop () [inline]

Get the top of the pseudocolor rectangle.

Returns:

Return the left of the pseudocolor rectangle.

8.213.2.15 int mitkPseudocolorWidgetModel::GetWidth () [inline]

Get the width of the pseudocolor rectangle.

Returns:

Return the width of the pseudocolor rectangle.

8.213.2.16 bool mitkPseudocolorWidgetModel::IsValid () [inline]

If the widget model is valid.

Returns:

Return if the widget model is valid (i.e. can be shown in the scene).

8.213.2.17 virtual void mitkPseudocolorWidgetModel::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:

names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.213.2.18 virtual void mitkPseudocolorWidgetModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel2D.

8.213.2.19 virtual void mitkPseudocolorWidgetModel::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.213.2.20 virtual int mitkPseudocolorWidgetModel::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.213.2.21 void mitkPseudocolorWidgetModel::ResetColorTable ()

Reset the color table to the default values.

8.213.2.22 int mitkPseudocolorWidgetModel::SetColorTable (unsigned char *table*[ ][3], int *colorNum* = COLOR_TABLE_SIZE)

Set color table.

Parameters:
table an unsigned char array contains the color table and the arrangement of the color components is 'RGBRGBRGB...'

colorNum the number of colors in the table to set

Returns:
Return the number of colors set to the table.

Note:
The total number of colors in the table must be less than mitkPseudocolorWidgetModel::COLOR_-TABLE_SIZE. At present, this class can support a color table whose number of colors is up to 256 (i.e. mitkPseudocolorWidgetModel::COLOR_TABLE_SIZE = 256).

8.213.2.23 void mitkPseudocolorWidgetModel::SetEndPoint (int *x*, int *y*)

Set the end point of the pseudocolor image rectangle draged by mouse.

Parameters:

x x-coordinate of this point

y y-coordinate of this point

8.213.2.24 void mitkPseudocolorWidgetModel::SetMovePoint (int *x*, int *y*)

Set position of the moving end point.

Parameters:

x x-coordinate of the moving end point of this line

y y-coordinate of the moving end point of this line

8.213.2.25 void mitkPseudocolorWidgetModel::SetStartPoint (int *x*, int *y*)

Set the start point of the pseudocolor image rectangle draged by mouse.

Parameters:

x x-coordinate of this point

y y-coordinate of this point

The documentation for this class was generated from the following file:

- mitkPseudocolorWidgetModel.h

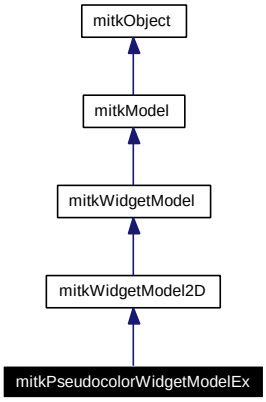
8.214 mitkPseudocolorWidgetModelEx Class Reference

mitkPseudocolorWidgetModelEx - a 2D widget for displaying pseudocolor image

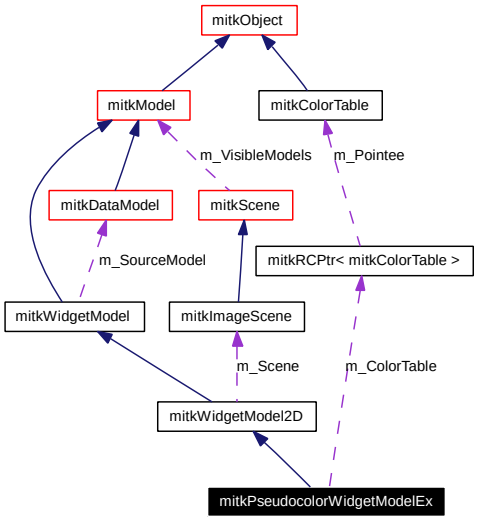
`#include <mitkPseudocolorWidgetModelEx.h>`

Inherits mitkWidgetModel2D.

Inheritance diagram for mitkPseudocolorWidgetModelEx:



Collaboration diagram for mitkPseudocolorWidgetModelEx:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- virtual void SetSourceModel (mitkDataModel *model)
- bool IsValid ()

- void SetStartPoint (float x, float y)
- void SetStartPoint (int sx, int sy)
- void SetMovePoint (float x, float y)
- void SetMovePoint (int sx, int sy)
- void SetEndPoint (float x, float y)
- void SetEndPoint (int sx, int sy)
- void SetColorTable (mitkColorTable *ct)
- mitkColorTable * GetColorTable ()
- void GetLeftBottom (int &l, int &b)
- int GetRectWidthInImage ()
- int GetRectHeightInImage ()
- bool UpdatePseudocolorRect ()
- void SetRectChanged ()
- virtual mitkVolume * GetRegionMask ()
- virtual void Update ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.214.1 Detailed Description

mitkPseudocolorWidgetModelEx - a 2D widget for displaying pseudocolor image

mitkPseudocolorWidgetModelEx is a 2D widget for displaying pseudocolor image. It can respond the mouse events and display the pseudocolor image of current rectangle part and return the width and height. It is supposed to be attached to a 2D data model (e.g. mitkImageModel) and add to a 2D scene (e.g. mitkImageScene), and in other conditions the display could be improper. Being different with mitkPseudocolorWidgetModel, it directly maps original pixel values (including 8-bit, 16-bit, 32-bit integers and float point values) to RGB colors, and use mitkColorTable to set color table freely.

See also:
mitkColorTable

8.214.2 Member Function Documentation

8.214.2.1 virtual void mitkPseudocolorWidgetModelEx::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.214.2.2 virtual void mitkPseudocolorWidgetModelEx::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.214.2.3 virtual void mitkPseudocolorWidgetModelEx::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.214.2.4 mitkColorTable* mitkPseudocolorWidgetModelEx::GetColorTable ()

Get color table.

Returns:

Return the pointer to the mitkColorTable object.

8.214.2.5 void mitkPseudocolorWidgetModelEx::GetLeftBottom (int & *l*, int & *b*) [inline]

Get left-top point (in image space) of the pseudo color window.

Parameters:

l return the left value

t return the top value

8.214.2.6 int mitkPseudocolorWidgetModelEx::GetRectHeightInImage () [inline]

Get the height (in image space) of the pseudo color window.

Returns:

Return the height value in number of pixels.

8.214.2.7 int mitkPseudocolorWidgetModelEx::GetRectWidthInImage () [inline]

Get the width (in image space) of the pseudo color window.

Returns:
Return the width value in number of pixels.

8.214.2.8 virtual mitkVolume* mitkPseudocolorWidgetModelEx::GetRegionMask () [virtual]

Get mask image where the value of widget region is 255 and the rest is 0.

Returns:
Return a pointer of mitkVolume object contains the mask image.

Note:
The returned object pointer should be deleted properly by yourself.

Reimplemented from mitkWidgetModel2D.

8.214.2.9 bool mitkPseudocolorWidgetModelEx::IsValid () [inline]

If the widget model is valid.

Returns:
Return if the widget model is valid (i.e. can be shown in the scene).

8.214.2.10 virtual void mitkPseudocolorWidgetModelEx::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:
names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.214.2.11 virtual void mitkPseudocolorWidgetModelEx::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel2D.

8.214.2.12 virtual void mitkPseudocolorWidgetModelEx::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.214.2.13 virtual int mitkPseudocolorWidgetModelEx::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:

scene the pointer of an mitkScene in which this model will be shown

Returns:

Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.214.2.14 void mitkPseudocolorWidgetModelEx::SetColorTable (mitkColorTable * *ct*) [inline]

Set color table.

Parameters:

ct the pointer to a mitkColorTable object

8.214.2.15 void mitkPseudocolorWidgetModelEx::SetEndPoint (int *sx*, int *sy*) [inline]

Set position of the end point.

Parameters:

sx x-coordinate of the end point of mouse dragging on screen

sy y-coordinate of the end point of mouse dragging on screen

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.214.2.16 void mitkPseudocolorWidgetModelEx::SetEndPoint (float *x*, float *y*) [inline]

Set position of the end point in the object space.

Parameters:

x x-coordinate of the end point of mouse dragging

y y-coordinate of the end point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.214.2.17 void mitkPseudocolorWidgetModelEx::SetMovePoint (int *sx*, int *sy*)

Set position of the moving end point.

Parameters:

sx x-coordinate of the moving end point of mouse dragging on screen

sy y-coordinate of the moving end point of mouse dragging on screen

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.214.2.18 void mitkPseudocolorWidgetModelEx::SetMovePoint (float *x*, float *y*)

Set position of the moving end point in the object space.

Parameters:

x x-coordinate of the moving end point of mouse dragging

y y-coordinate of the moving end point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.214.2.19 void mitkPseudocolorWidgetModelEx::SetRectChanged () `[inline]`

Set that the pseudo-color region is changed.

8.214.2.20 virtual void mitkPseudocolorWidgetModelEx::SetSourceModel (mitkDataModel * *model*) `[virtual]`

Associate this widget with a model.

Parameters:

model pointer to an miekDataModel to which this widget attach

Reimplemented from mitkWidgetModel2D.

8.214.2.21 void mitkPseudocolorWidgetModelEx::SetStartPoint (int *sx*, int *sy*)

Set position of the start point.

Parameters:

sx x-coordinate of the start point of mouse dragging on screen

sy y-coordinate of the start point of mouse dragging on screen

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.214.2.22 void mitkPseudocolorWidgetModelEx::SetStartPoint (float *x*, float *y*)

Set position of the start point in the object space.

Parameters:

x x-coordinate of the start point of mouse dragging

y y-coordinate of the start point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.214.2.23 virtual void mitkPseudocolorWidgetModelEx::Update () [virtual]

Update the parameters of the widget.

Reimplemented from mitkWidgetModel2D.

8.214.2.24 bool mitkPseudocolorWidgetModelEx::UpdatePseudocolorRect ()

Update the pseudo-color region.

The documentation for this class was generated from the following file:

- mitkPseudocolorWidgetModelEx.h

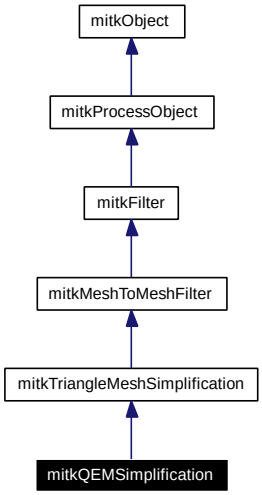
8.215 mitkQEMSimplification Class Reference

mitkQEMSimplification - an implementation for the simplification algorithm using Quadric Error Metrics (QEM)

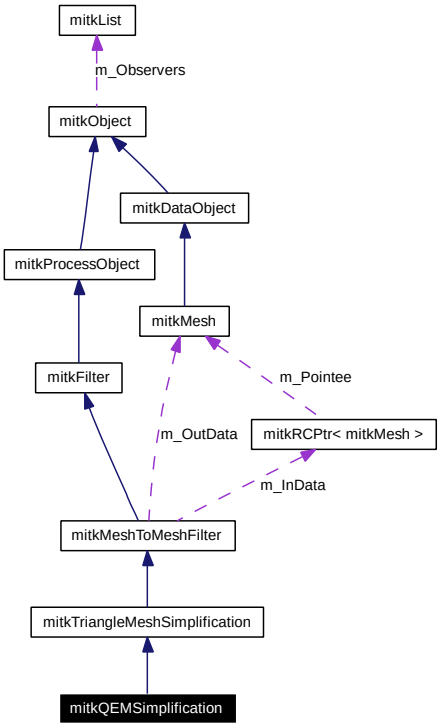
`#include <mitkQEMSimplification.h>`

Inherits mitkTriangleMeshSimplification.

Inheritance diagram for mitkQEMSimplification:



Collaboration diagram for mitkQEMSimplification:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetMeshingPenalty (double mp)
- double GetMeshingPenalty ()
- void SetVertexDegreeLimit (size_type vdl)
- size_type GetVertexDegreeLimit ()
- void SetCompactnessRatio (double cr)
- double GetCompactnessRatio ()
- void SetLocalValidityThreshold (double lvt)
- double GetLocalValidityThreshold ()

8.215.1 Detailed Description

mitkQEMSimplification - an implementation for the simplification algorithm using Quadric Error Metrics (QEM)

mitkQEMSimplification is an implementation for the simplification algorithm using Quadric Error Metrics (QEM). The implementation is based on the algorithm presented in following papers written by Michael Garland and Paul S. Heckbert and their implementation in QSlim Simplification Software:

- Michael Garland and Paul S. Heckbert, Surface Simplification Using Quadric Error Metrics, in *Proceedings of SIGGARPH'97*, pp. 209-216, 1997.
- Michael Garland and Paul S. Heckbert, Simplifying Surfaces with Color and Texture using Quadric Error Metrics, in *Proceedings of IEEE Visualization'98*, pp. 263-269, 1998.

Thanks for their excellent works.

8.215.2 Member Function Documentation

8.215.2.1 double mitkQEMSimplification::GetCompactnessRatio () [inline]

Get the compactness ratio for triangle compactness check.

Returns:
Return the compactness ratio.

8.215.2.2 double mitkQEMSimplification::GetLocalValidityThreshold () [inline]

Get the local validity threshold for local validity check.

Returns:
Return the local validity threshold.

8.215.2.3 double mitkQEMSimplification::GetMeshingPenalty () [inline]

Get meshing penalty.

Returns:
Return the meshing penalty.

8.215.2.4 size_type mitkQEMSimplification::GetVertexDegreeLimit () [inline]

Set the limit of the vertex degree.

Returns:
Return the limit of the vertex degree.

8.215.2.5 virtual void mitkQEMSimplification::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTriangleMeshSimplification.

8.215.2.6 void mitkQEMSimplification::SetCompactnessRatio (double *cr*) [inline]

Set the compactness ratio for triangle compactness check. The default value is 0.0.

Parameters:
cr the compactness ratio

8.215.2.7 void mitkQEMSimplification::SetLocalValidityThreshold (double *lvt*) [inline]

Set the local validity threshold for local validity check. The default value is 0.0.

Parameters:
lvt the local validity threshold

8.215.2.8 void mitkQEMSimplification::SetMeshingPenalty (double *mp*) [inline]

Set meshing penalty. The default value is 1.0.

Parameters:
mp meshing penalty (>1.0 will enable checks for the mesh penalties)

8.215.2.9 void mitkQEMSimplification::SetVertexDegreeLimit (size_type *vdl*) [inline]

Set the limit of the vertex degree after simplification. The default value is 24.

Parameters:
vdl the vertex degree limit

The documentation for this class was generated from the following file:

- mitkQEMSimplification.h

8.216 mitkQuaternion Class Reference

mitkQuaternion - it's used to implement 3d rotation

`#include <mitkQuaternion.h>`

Public Member Functions

- mitkQuaternion ()
- mitkQuaternion (const float x, const float y, const float z, const float w)
- mitkQuaternion (const float x, const float y, const float z)
- mitkQuaternion & operator= (const mitkQuaternion &q)
- mitkQuaternion (const mitkQuaternion &q)
- void Identity (void)
- void Normalize (void)
- void GetValues (float &x, float &y, float &z, float &w) const
- void AxisRadToQuat (const float ax, const float ay, const float az, const float angle)
- void AxisDegToQuat (const float ax, const float ay, const float az, const float angle)
- void GetAxisRad (float &ax, float &ay, float &az, float &angle) const
- void GetAxisDeg (float &ax, float &ay, float &az, float &angle) const
- void EulerRadToQuat (const float xr, const float yr, const float zr)
- void EulerDegToQuat (const float xd, const float yd, const float zd)
- void GetEulerRad (float &xr, float &yr, float &zr) const
- void GetEulerDeg (float &xd, float &yd, float &zd) const
- void BuildMatrix (mitkMatrix &mat) const

Friends

- mitkQuaternion operator * (const mitkQuaternion &p, const mitkQuaternion &q)

8.216.1 Detailed Description

mitkQuaternion - it's used to implement 3d rotation

mitkQuaternion is used to implement 3d rotation

8.216.2 Constructor & Destructor Documentation

8.216.2.1 mitkQuaternion::mitkQuaternion ()

Default constructor. Construct a quaternion with zero rotation.

8.216.2.2 mitkQuaternion::mitkQuaternion (const float *x*, const float *y*, const float *z*, const float *w*)

Construct a quaternion from given values. Quat will be normalized. Will perform a valid check.

8.216.2.3 mitkQuaternion::mitkQuaternion (const float *x*, const float *y*, const float *z*)

Construct a quaternion from euler angles in degrees.

Parameters:

x rotation around x-axis in degrees

y rotation around y-axis in degrees

z rotation around z-axis in degrees

8.216.2.4 mitkQuaternion::mitkQuaternion (const mitkQuaternion & *q*) [inline]

Copy constructor.

8.216.3 Member Function Documentation

8.216.3.1 void mitkQuaternion::AxisDegToQuat (const float *ax*, const float *ay*, const float *az*, const float *angle*)

Convert an axis angle to quaternion, angle is in degrees.

Parameters:

ax the x-coordinate of rotation axis

ay the y-coordinate of rotation axis

az the z-coordinate of rotation axis

angle the angle of rotation in degrees

8.216.3.2 void mitkQuaternion::AxisRadToQuat (const float *ax*, const float *ay*, const float *az*, const float *angle*)

Convert an axis angle to quaternion, angle is in radians.

Parameters:

ax the x-coordinate of rotation axis

ay the y-coordinate of rotation axis

az the z-coordinate of rotation axis

angle the angle of rotation in radians

8.216.3.3 void mitkQuaternion::BuildMatrix (mitkMatrix & *mat*) const

Get the 4x4 rotation matrix representation of this quaternion.

Parameters:

mat return the rotation matrix.

8.216.3.4 void mitkQuaternion::EulerDegToQuat (const float *xd*, const float *yd*, const float *zd*)

Convert euler angles to quaternion, euler angles are in degrees.

Parameters:

xd rotation around x-axis in degrees

yd rotation around y-axis in degrees

zd rotation around z-axis in degrees

8.216.3.5 void mitkQuaternion::EulerRadToQuat (const float *xr*, const float *yr*, const float *zr*)

Convert euler angles to quaternion, euler angles are in radians.

Parameters:

xr rotation around x-axis in radians

yr rotation around y-axis in radians

zr rotation around z-axis in radians

8.216.3.6 void mitkQuaternion::GetAxisDeg (float & *ax*, float & *ay*, float & *az*, float & *angle*) const

Get an axis angle from this quaternion, angle is returned in degrees.

Parameters:

ax return the x-coordinate of rotation axis

ay return the y-coordinate of rotation axis

az return the z-coordinate of rotation axis

angle return the angle of rotation in degrees

8.216.3.7 void mitkQuaternion::GetAxisRad (float & *ax*, float & *ay*, float & *az*, float & *angle*) const

Get an axis angle from this quaternion, angle is returned in radians.

Parameters:

ax return the x-coordinate of rotation axis

ay return the y-coordinate of rotation axis

az return the z-coordinate of rotation axis

angle return the angle of rotation in radians

8.216.3.8 void mitkQuaternion::GetEulerDeg (float & *xd*, float & *yd*, float & *zd*) const

Get euler angles from this quaternion, angle is returned in degrees.

Parameters:

xd return the rotation around x-axis in degrees

yd return the rotation around y-axis in degrees

zd return the rotation around z-axis in degrees

8.216.3.9 void mitkQuaternion::GetEulerRad (float & *xr*, float & *yr*, float & *zr*) const

Get euler angles from this quaternion, angle is returned in radians.

Parameters:

xr return the rotation around x-axis in radians

yr return the rotation around y-axis in radians

zr return the rotation around z-axis in radians

8.216.3.10 void mitkQuaternion::GetValues (float & *x*, float & *y*, float & *z*, float & *w*) const

Get the values of this quaternion.

8.216.3.11 void mitkQuaternion::Identity (void)

Set this quaternion to the identity quaternion.

8.216.3.12 void mitkQuaternion::Normalize (void)

Normalize this quaternion.

8.216.3.13 mitkQuaternion& mitkQuaternion::operator= (const mitkQuaternion & *q*) [inline]

"=" operator.

8.216.4 Friends And Related Function Documentation

8.216.4.1 mitkQuaternion operator * (const mitkQuaternion & *p*, const mitkQuaternion & *q*) [friend]

Multiplication

The documentation for this class was generated from the following file:

- mitkQuaternion.h

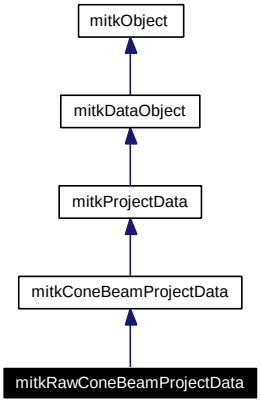
8.217 mitkRawConeBeamProjectData Class Reference

mitkRawConeBeamProjectData - concret class to represent cone-beam project data.

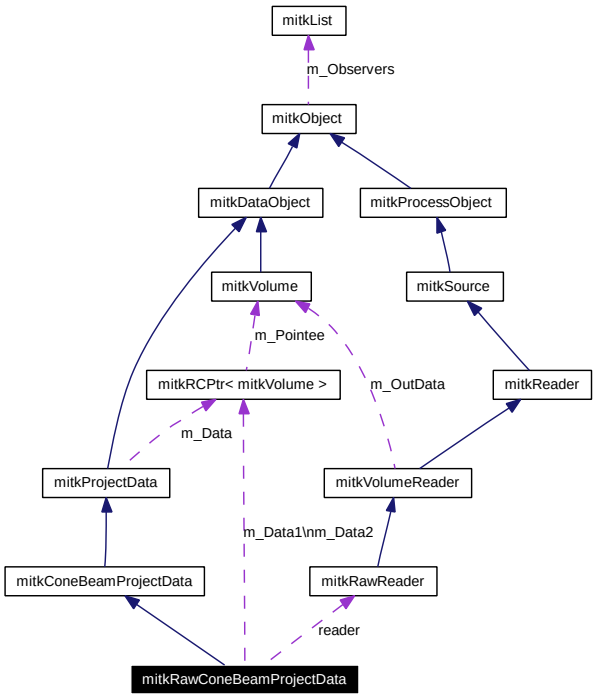
`#include <mitkRawConeBeamProjectData.h>`

Inherits mitkConeBeamProjectData.

Inheritance diagram for mitkRawConeBeamProjectData:



Collaboration diagram for mitkRawConeBeamProjectData:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject ∗src)
- virtual void DeepCopy (mitkDataObject ∗src)
- void SetBlockSize (int m)
- bool SetFileName (const char ∗m)
- void SetIsOneFile (bool onefile)
- void SetFilePostfixNum (int m)
- virtual void const ∗ GetAngleSliceForRead (int sliceIdx)
- virtual void ∗ GetAngleSliceForWrite (int sliceIdx)
- virtual void ∗ GetAngleSliceForReadWrite (int sliceIdx)
- int GetGpuBlockNum () const
- void SetGpuBlockNum (int m)
- virtual void Initialize ()

8.217.1 Detailed Description

mitkRawConeBeamProjectData - concret class to represent cone-beam project data.

mitkRawConeBeamProjectData - concret class to represent cone-beam project data.

8.217.2 Member Function Documentation

8.217.2.1 virtual void mitkRawConeBeamProjectData::DeepCopy (mitkDataObject ∗ *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamProjectData.

8.217.2.2 virtual unsigned long long mitkRawConeBeamProjectData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by this data object. The unit is BYTE.

Returns:
Return the actual memory size occupied by this data object. The unit is BYTE.

Implements mitkConeBeamProjectData.

8.217.2.3 virtual void const∗ mitkRawConeBeamProjectData::GetAngleSliceForRead (int *sliceIdx*) [virtual]

GetAngleSliceForRead

Implements mitkConeBeamProjectData.

8.217.2.4 virtual void* mitkRawConeBeamProjectData::GetAngleSliceForReadWrite (int *sliceIdx*) [virtual]

GetAngleSliceForReadWrite dosn't write to harddisk

Implements mitkConeBeamProjectData.

8.217.2.5 virtual void* mitkRawConeBeamProjectData::GetAngleSliceForWrite (int *sliceIdx*) [inline, virtual]

Never call it

Implements mitkConeBeamProjectData.

8.217.2.6 int mitkRawConeBeamProjectData::GetGpuBlockNum () const [inline]

Get number of Gpu Block

8.217.2.7 virtual void mitkRawConeBeamProjectData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Note:
Pure virtual function. Its concrete subclass must implement this function.

Reimplemented from mitkConeBeamProjectData.

8.217.2.8 virtual void mitkRawConeBeamProjectData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkConeBeamProjectData.

8.217.2.9 void mitkRawConeBeamProjectData::SetBlockSize (int *m*) [inline]

Set how many slices one block include and slice numbers should be devide by m

8.217.2.10 bool mitkRawConeBeamProjectData::SetFileName (const char * *m*)

Set file name

8.217.2.11 void mitkRawConeBeamProjectData::SetFilePostfixNum (int *m*) [inline]

Set File Postfix Num.

8.217.2.12 void mitkRawConeBeamProjectData::SetGpuBlockNum (int *m*) [inline]

Set number of Gpu Block

8.217.2.13 void mitkRawConeBeamProjectData::SetIsOneFile (bool *onefile*) [inline]

Set is one file

8.217.2.14 virtual void mitkRawConeBeamProjectData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Internal function. Don't call it directly.

Implements mitkConeBeamProjectData.

The documentation for this class was generated from the following file:

- mitkRawConeBeamProjectData.h

8.218 mitkRawData Class Reference

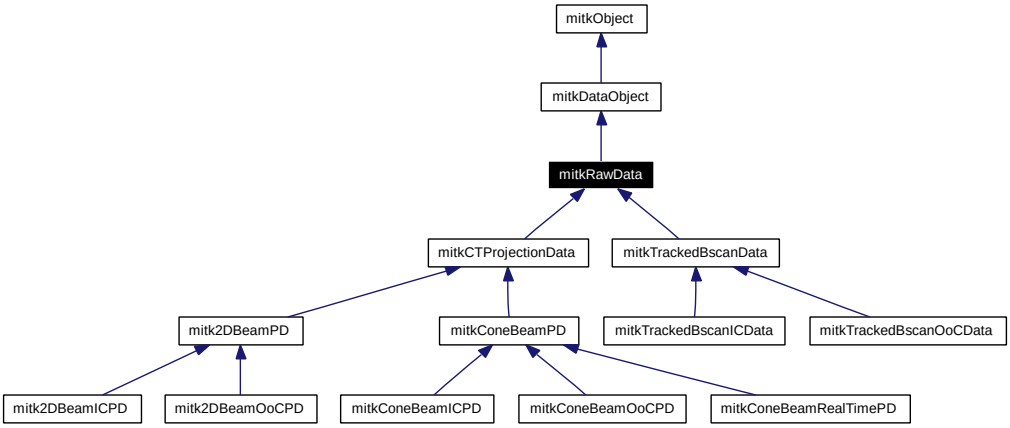
mitkRawData - an abstract class representing Raw Data.

`#include <mitkRawData.h>`

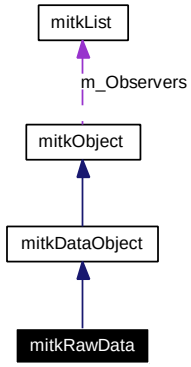
Inherits mitkDataObject.

Inherited by mitkCTProjectionData, and mitkTrackedBscanData.

Inheritance diagram for mitkRawData:



Collaboration diagram for mitkRawData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int GetDataObjectType () const
- virtual int GetDataTypeSize () const
- virtual void SetDataType (int data_type)
- virtual int GetDataType () const

8.218.1 Detailed Description

mitkRawData - an abstract class representing Raw Data.

mitkRawData is an abstract class representing Raw Data for Volume Reconstruction. It is inherited by all kinds of concrete classes representing Raw Data for dedicated Reconstruction (CT, MRI, or 3D ultrasound). Concrete mitkRawDataToVolumeFilter classes encapsulating dedicated Reconstruction (CT, MRI, or 3D ultrasound) algorithms reconstruct Raw Data to Volume.

8.218.2 Member Function Documentation

8.218.2.1 virtual int mitkRawData::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_DATA_OBJECT

Note:
Pure virtual function. Its concrete subclass must implement this function and return its data object type.

Reimplemented from mitkDataObject.

Reimplemented in mitkTrackedBscanData, mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.218.2.2 virtual int mitkRawData::GetDataType () const [inline, virtual]

Get data type of this volume. MITK supports various data type.

Returns:
All of the return values and their meaning are shown as follows:
MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

8.218.2.3 virtual int mitkRawData::GetDataTypeSize () const [inline, virtual]

Get the size of the data type in bytes.

Returns:
Return the size of the data type in bytes.

8.218.2.4 virtual void mitkRawData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkDataObject.

Reimplemented in mitk2DBeamICPD, mitk2DBeamOoCPD, mitk2DBeamPD, mitkConeBeamICPD, mitkConeBeamOoCPD, mitkConeBeamPD, mitkConeBeamRealTimePD, mitkCTProjectionData, mitkTrackedBscanData, mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.218.2.5 virtual void mitkRawData::SetDataType (int *data_type*) [virtual]

Set data type of this volume. MITK supports various data type.

Parameters:

data_type Its valid value and meaning is shown as follows:

MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

The documentation for this class was generated from the following file:

- mitkRawData.h

8.219 mitkRawDataReader Class Reference

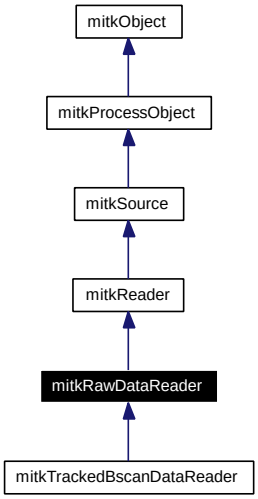
mitkRawDataReader - An abstract class of reading Raw Data.

`#include <mitkRawDataReader.h>`

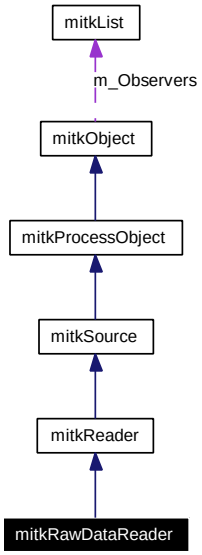
Inherits mitkReader.

Inherited by mitkTrackedBscanDataReader.

Inheritance diagram for mitkRawDataReader:



Collaboration diagram for mitkRawDataReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- void SetOoCSupport (char const *diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufSliceNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)

8.219.1 Detailed Description

mitkRawDataReader - An abstract class of reading Raw Data.

mitkRawDataReader is an abstract class of reading Raw Data for Volume reconstruction. It is inherited by all kinds of concrete classes (e.g. mitkTrackedBscanDataReader) reading particular Raw Data (e.g. mitkTrackedBscanData) for dedicated Volume Reconstruction (CT, MRI, or 3D ultrasound).

8.219.2 Member Function Documentation

8.219.2.1 virtual void mitkRawDataReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkReader.

Reimplemented in mitkTrackedBscanDataReader.

8.219.2.2 void mitkRawDataReader::SetOoCSupport (char const * *diskPath* = mitkOoCGlobalSettings::DiskCachePath, unsigned int *bufSliceNum* = mitkOoCGlobalSettings::BufferedSliceNumber, bool *supportOoC* = true)

Let the reader support out-of-core volume data.

Parameters:

diskPath the path in the disk to cache the volume data

bufSliceNum the number of slices to cache in the main memory

supportOoC whether to turn on out-of-core support

Note:

The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

The documentation for this class was generated from the following file:

- mitkRawDataReader.h

8.220 mitkRawDataToVolumeFilter Class Reference

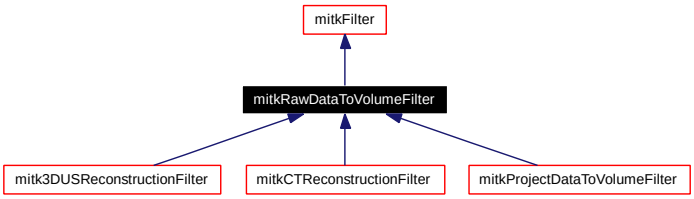
mitkRawDataToVolumeFilter - an abstract class for volume reconstruction

`#include <mitkRawDataToVolumeFilter.h>`

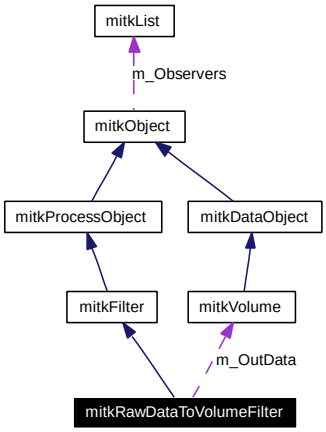
Inherits mitkFilter.

Inherited by mitk3DUSReconstructionFilter, mitkCTReconstructionFilter, and mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkRawDataToVolumeFilter:



Collaboration diagram for mitkRawDataToVolumeFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual mitkVolume * GetOutput ()=0
- void SetOoCSupport (char const *diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufSliceNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)

8.220.1 Detailed Description

mitkRawDataToVolumeFilter - an abstract class for volume reconstruction

mitkRawDataToVolumeFilter is an abstract class specifying interfaces for volume reconstruction classes. Concrete volume reconstruction classes process an input mitkRawData object and generate an output mitkVolume object. Here only the output interface is defined, because different imaging modalities have differ-

ent concrete input mitkRawData. Particular input interfaces for particular imaging modalities (CT, 3DUS) are defined in subclasses (e.g. mitk3DUSReconstructionFilter) of the mitkRawDataToVolumeFilter.

8.220.2 Member Function Documentation

8.220.2.1 virtual mitkVolume* mitkRawDataToVolumeFilter::GetOutput () [pure virtual]

Get the output volume

Returns:
Return the output volume

Implemented in mitk3DUSPixelBasedMT, mitkCTReconstructionFilter, mitkProjectDataToVolumeFilter, and mitk3DUSPixelBasedGPU.

8.220.2.2 virtual void mitkRawDataToVolumeFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFilter.

Reimplemented in mitk3DUSPixelBasedMT, mitk3DUSReconstructionFilter, mitkCBFDKReconCF, mitkCBFDKReconCFCPU, mitkCBTFDKReconCF, mitkConeBeamReconFromHelix, mitkCTReconstructionFilter, mitkFanBeamReconFromCirle, mitkFbFbp, mitkFBFBPReconCF, mitkFBFBPReconCFCPU, mitkFbFbpSpeedUsingCpu, mitkFDK, mitkFDKSpeedUsingCpu, mitkParallelBeamReconFromCirle, mitkPbFbp, mitkPBFBPReconCF, mitkProjectDataToVolumeFilter, mitkTFDK, mitkCBFDKReconCFGPU, mitkFBFBPReconCFGPU, mitkFbFbpSpeedUsingGpu, mitkFDKSpeedUsingGpu, and mitk3DUSPixelBasedGPU.

8.220.2.3 void mitkRawDataToVolumeFilter::SetOoCSupport (char const * *diskPath* = mitkOoCGlobalSettings::DiskCachePath, unsigned int *bufSliceNum* = mitkOoCGlobalSettings::BufferedSliceNumber, bool *supportOoC* = true)

Let the reader support out-of-core data.

Parameters:
diskPath the path in the disk to cache the volume data
bufSliceNum the number of slices to cache in the main memory
supportOoC whether to turn on out-of-core support

Note:
The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

Reimplemented in mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkRawDataToVolumeFilter.h

8.221 mitkRawDataWriter Class Reference

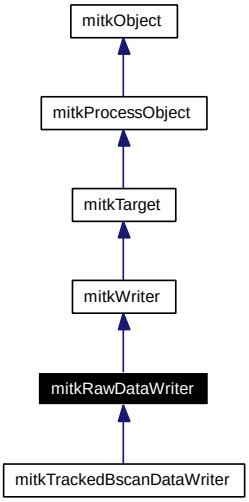
mitkRawDataWriter - an abstract class of writing Raw Data.

`#include <mitkRawDataWriter.h>`

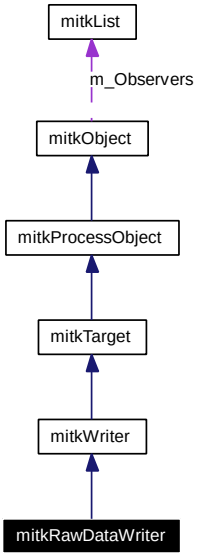
Inherits mitkWriter.

Inherited by mitkTrackedBscanDataWriter.

Inheritance diagram for mitkRawDataWriter:



Collaboration diagram for mitkRawDataWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.221.1 Detailed Description

mitkRawDataWriter - an abstract class of writing Raw Data.

mitkRawDataWriter is an abstract class of writing Raw Data to Disk Files. It is inherited by all kinds of concrete classes (e.g. mitkTrackedBscanDataWriter) writing particular Raw Data (e.g. mitkTrackedBscanData) to dedicated formatted files (e.g. PAI) relative to CT, MRI, or 3D ultrasound.

8.221.2 Member Function Documentation

8.221.2.1 virtual void mitkRawDataWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkWriter.

Reimplemented in mitkTrackedBscanDataWriter.

The documentation for this class was generated from the following file:

- mitkRawDataWriter.h

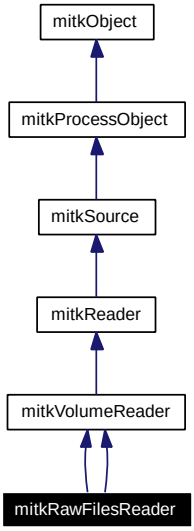
8.222 mitkRawFilesReader Class Reference

mitkRawFilesReader - a concrete reader for reading a series of files contain raw data (no header information).

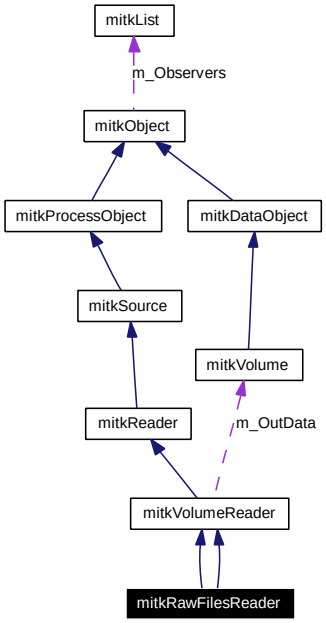
`#include <mitkRawFilesReader.h>`

Inherits mitkVolumeReader, and mitkVolumeReader.

Inheritance diagram for mitkRawFilesReader:



Collaboration diagram for mitkRawFilesReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkRawFilesReader ()
- void SetWidth (int w)
- void SetHeight (int h)
- void SetSpacingX (float px)
- void SetSpacingY (float py)
- void SetSpacingZ (float pz)
- void SetChannelNum (int n)
- void SetTitleBytes (int n)
- void SetEndian (bool isBig=false)
- void SetBigEndian (bool isBig=true)
- void SetLittleEndian (bool isBig=false)
- void SetPlanarCfg (bool isColorByPlane)
- void SetDataType (int dataType)
- void SetDataTypeToFloat ()
- void SetDataTypeToDouble ()
- void SetDataTypeToInt ()
- void SetDataTypeToUnsignedInt ()
- void SetDataTypeToLong ()
- void SetDataTypeToUnsignedLong ()
- void SetDataTypeToShort ()
- void SetDataTypeToUnsignedShort ()
- void SetDataTypeToUnsignedChar ()
- void SetDataTypeToChar ()
- virtual void PrintSelf (ostream &os)
- mitkRawFilesReader ()
- void SetWidth (int w)
- void SetHeight (int h)
- void SetSpacingX (float px)
- void SetSpacingY (float py)
- void SetSpacingZ (float pz)
- void SetChannelNum (int n)
- void SetTitleBytes (int n)
- void SetEndian (bool isBig=false)
- void SetBigEndian (bool isBig=true)
- void SetLittleEndian (bool isBig=false)
- void SetPlanarCfg (bool isColorByPlane)
- void SetDataType (int dataType)
- void SetDataTypeToFloat ()
- void SetDataTypeToDouble ()
- void SetDataTypeToInt ()
- void SetDataTypeToUnsignedInt ()
- void SetDataTypeToLong ()
- void SetDataTypeToUnsignedLong ()
- void SetDataTypeToShort ()
- void SetDataTypeToUnsignedShort ()
- void SetDataTypeToUnsignedChar ()
- void SetDataTypeToChar ()

8.222.1 Detailed Description

mitkRawFilesReader - a concrete reader for reading a series of files contain raw data (no header information).

mitkRawReader reads a a series of raw data files (no any header information) to a volume, each file contains the data of one slice. Because raw file doesn't has any header information, including width, height, image number, spacing information, etc. , you must set enough information using the series of Set∗ functions. To use this reader, the code snippet is:

```
mitkRawReader *aReader = new mitkRawReader;
aReader->SetWidth(w);
aReader->SetHeight(h);
aReader->SetImageNum(d);
aReader->SetSpacingX(sx);
aReader->SetSpacingY(sy);
aReader->SetSpacingZ(sz);
aReader->SetDataType(dt);
aReader->SetChannelNum(cn);
aReader->AddFileName(file1);
aReader->AddFileName(file2);
...  ...  ...
//the above is the required information
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

8.222.2 Constructor & Destructor Documentation

8.222.2.1 mitkRawFilesReader::mitkRawFilesReader ()

Default constructor.

8.222.2.2 mitkRawFilesReader::mitkRawFilesReader ()

Default constructor.

8.222.3 Member Function Documentation

8.222.3.1 virtual void mitkRawFilesReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

8.222.3.2 virtual void mitkRawFilesReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

8.222.3.3 void mitkRawFilesReader::SetBigEndian (bool *isBig* = `true`) `[inline]`

Provided for convenience, just the same as SetEndian().

8.222.3.4 void mitkRawFilesReader::SetBigEndian (bool *isBig* = `true`) `[inline]`

Provided for convenience, just the same as SetEndian().

8.222.3.5 void mitkRawFilesReader::SetChannelNum (int *n*) `[inline]`

Set number of channels.

Parameters:
n number of channel 1—gray image 3—RGB image 4—RGBA image

8.222.3.6 void mitkRawFilesReader::SetChannelNum (int *n*) `[inline]`

Set number of channels.

Parameters:
n number of channel 1—gray image 3—RGB image 4—RGBA image

8.222.3.7 void mitkRawFilesReader::SetDataType (int *dataType*) `[inline]`

Set the data type of these images.

Parameters:
dataType the data type of these images, the value should be one of follows:
MITK_CHAR (char)
MITK_UNSIGNED_CHAR (unsigned char)
MITK_SHORT (short)
MITK_UNSIGNED_SHORT (unsigned short)
MITK_INT (int)
MITK_UNSIGNED_INT (unsigned int)
MITK_LONG (long)
MITK_UNSIGNED_LONG (unsigned long)
MITK_FLOAT (float)
MITK_DOUBLE (double)

8.222.3.8 void mitkRawFilesReader::SetDataType (int *dataType*) [inline]

Set the data type of these images.

Parameters:

dataType the data type of these images, the value should be one of follows:

MITK_CHAR (char)
MITK_UNSIGNED_CHAR (unsigned char)
MITK_SHORT (short)
MITK_UNSIGNED_SHORT (unsigned short)
MITK_INT (int)
MITK_UNSIGNED_INT (unsigned int)
MITK_LONG (long)
MITK_UNSIGNED_LONG (unsigned long)
MITK_FLOAT (float)
MITK_DOUBLE (double)

8.222.3.9 void mitkRawFilesReader::SetDataTypeToChar () [inline]

Set data type to char.

8.222.3.10 void mitkRawFilesReader::SetDataTypeToChar () [inline]

Set data type to char.

8.222.3.11 void mitkRawFilesReader::SetDataTypeToDouble () [inline]

Set data type to double.

8.222.3.12 void mitkRawFilesReader::SetDataTypeToDouble () [inline]

Set data type to double.

8.222.3.13 void mitkRawFilesReader::SetDataTypeToFloat () [inline]

Set data type to float.

8.222.3.14 void mitkRawFilesReader::SetDataTypeToFloat () [inline]

Set data type to float.

8.222.3.15 void mitkRawFilesReader::SetDataTypeToInt () [inline]

Set data type to int.

8.222.3.16 void mitkRawFilesReader::SetDataTypeToInt () [inline]

Set data type to int.

8.222.3.17 void mitkRawFilesReader::SetDataTypeToLong () [inline]

Set data type to long.

8.222.3.18 void mitkRawFilesReader::SetDataTypeToLong () [inline]

Set data type to long.

8.222.3.19 void mitkRawFilesReader::SetDataTypeToShort () [inline]

Set data type to short.

8.222.3.20 void mitkRawFilesReader::SetDataTypeToShort () [inline]

Set data type to short.

8.222.3.21 void mitkRawFilesReader::SetDataTypeToUnsignedChar () [inline]

Set data type to unsigned char.

8.222.3.22 void mitkRawFilesReader::SetDataTypeToUnsignedChar () [inline]

Set data type to unsigned char.

8.222.3.23 void mitkRawFilesReader::SetDataTypeToUnsignedInt () [inline]

Set data type to unsigned int.

8.222.3.24 void mitkRawFilesReader::SetDataTypeToUnsignedInt () [inline]

Set data type to unsigned int.

8.222.3.25 void mitkRawFilesReader::SetDataTypeToUnsignedLong () [inline]

Set data type to unsigned long.

8.222.3.26 void mitkRawFilesReader::SetDataTypeToUnsignedLong () [inline]

Set data type to unsigned long.

8.222.3.27 void mitkRawFilesReader::SetDataTypeToUnsignedShort () [inline]

Set data type to unsigned short.

8.222.3.28 void mitkRawFilesReader::SetDataTypeToUnsignedShort () [inline]

Set data type to unsigned short.

8.222.3.29 void mitkRawFilesReader::SetEndian (bool *isBig* = false) [inline]

Set endian if depth per pixel is bigger than 8.

Parameters:
isBig if the endian configuration is big endian (Mac)

8.222.3.30 void mitkRawFilesReader::SetEndian (bool *isBig* = false) [inline]

Set endian if depth per pixel is bigger than 8.

Parameters:
isBig if the endian configuration is big endian (Mac)

8.222.3.31 void mitkRawFilesReader::SetHeight (int *h*) [inline]

Set image height.

Parameters:
h image height

8.222.3.32 void mitkRawFilesReader::SetHeight (int *h*) [inline]

Set image height.

Parameters:
h image height

8.222.3.33 void mitkRawFilesReader::SetLittleEndian (bool *isBig* = false) [inline]

Provided for convenience, just the same as SetEndian().

8.222.3.34 void mitkRawFilesReader::SetLittleEndian (bool *isBig* = false) [inline]

Provided for convenience, just the same as SetEndian().

8.222.3.35 void mitkRawFilesReader::SetPlanarCfg (bool *isColorByPlane*) [inline]

Set planar configuration.

Parameters:
isColorByPlane if the planar configuration is color-by-plane

Note:
Only used when channel number is bigger than 1. For RGB images, if it is color-by-plane, it's means the pixels should be stored as "R1 R2 R3 ... G1 G2 G3 ... B1 B2 B3 ...", otherwise "R1 G1 B1 R2 G2 B2 ..."

8.222.3.36 void mitkRawFilesReader::SetPlanarCfg (bool *isColorByPlane*) [inline]

Set planar configuration.

Parameters:
isColorByPlane if the planar configuration is color-by-plane

Note:
Only used when channel number is bigger than 1. For RGB images, if it is color-by-plane, it's means the pixels should be stored as "R1 R2 R3 ... G1 G2 G3 ... B1 B2 B3 ...", otherwise "R1 G1 B1 R2 G2 B2 ..."

8.222.3.37 void mitkRawFilesReader::SetSpacingX (float *px*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.

8.222.3.38 void mitkRawFilesReader::SetSpacingX (float *px*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.

8.222.3.39 void mitkRawFilesReader::SetSpacingY (float *py*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:
py the spacing (mm) in two adjacent voxels in y axis.

8.222.3.40 void mitkRawFilesReader::SetSpacingY (float *py*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:
py the spacing (mm) in two adjacent voxels in y axis.

8.222.3.41 void mitkRawFilesReader::SetSpacingZ (float *pz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:
pz the spacing (mm) in two adjacent voxels in z axis.

8.222.3.42 void mitkRawFilesReader::SetSpacingZ (float *pz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:
pz the spacing (mm) in two adjacent voxels in z axis.

8.222.3.43 void mitkRawFilesReader::SetTitleBytes (int *n*) [inline]

Set number of bytes in title (header). These bytes will be skipped.

Parameters:
n number of bytes in title (header)

Note:
This title has nothing to do with the image. If the file has a title, the number of bytes must be given so that the program can skip the title properly when reading the file.

8.222.3.44 void mitkRawFilesReader::SetTitleBytes (int *n*) [inline]

Set number of bytes in title (header). These bytes will be skipped.

Parameters:
n number of bytes in title (header)

Note:
This title has nothing to do with the image. If the file has a title, the number of bytes must be given so that the program can skip the title properly when reading the file.

8.222.3.45 void mitkRawFilesReader::SetWidth (int *w*) [inline]

Set image width.

Parameters:
w image width

8.222.3.46 void mitkRawFilesReader::SetWidth (int *w*) [inline]

Set image width.

Parameters:
w image width

The documentation for this class was generated from the following files:

- mitkCommon/Include/mitkRawFilesReader.h
- mitkReconstruction/Include/mitkRawFilesReader.h

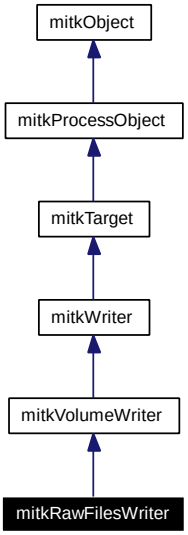
8.223 mitkRawFilesWriter Class Reference

mitkRawFilesWriter - a concrete writer for writing a volume to a a series of raw files.

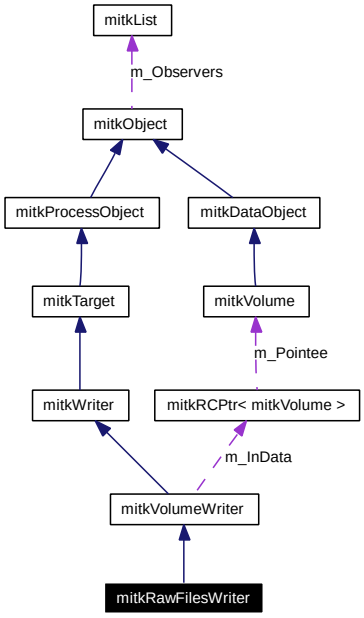
`#include <mitkRawFilesWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkRawFilesWriter:



Collaboration diagram for mitkRawFilesWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetEndian (bool isBig=false)
- void SetBigEndian (bool isBig=true)
- void SetLittleEndian (bool isBig=false)
- void SetPlanarCfg (bool isColorByPlane)
- void SetTitleBytes (int n)

8.223.1 Detailed Description

mitkRawFilesWriter - a concrete writer for writing a volume to a a series of raw files.

mitkRawFilesWriter writes a volume to a series of raw files (one slice per file with no header information). All the property of volume will lost, including the image width, height, image number, spacing information, etc. You can control the format of output file by setting the endian configuration, planar configuration and header size.

See also:
SetEndian
SetPlanarCfg
SetTitleBytes To use this writer, the code snippet is:

```
mitkRawWriter *aWriter = new mitkRawWriter;
aWriter->SetInput(aVolume);
int imageNum = aVolume->GetImageNum();
Gerenate file names into files[imageNum];
for(int i = 0; i < imageNum; i++)
    aWriter->AddFileName(files[i]);
aWriter->Run();
```

8.223.2 Member Function Documentation

8.223.2.1 virtual void mitkRawFilesWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

8.223.2.2 void mitkRawFilesWriter::SetBigEndian (bool *isBig* = true) [inline]

Provided for convenience, just the same as SetEndian().

8.223.2.3 void mitkRawFilesWriter::SetEndian (bool *isBig* = false) [inline]

Set endian if depth per pixel is bigger than 8.

Parameters:
isBig if the endian configuration is big endian (Mac)

8.223.2.4 void mitkRawFilesWriter::SetLittleEndian (bool *isBig* = `false`) `[inline]`

Provided for convenience, just the same as SetEndian().

8.223.2.5 void mitkRawFilesWriter::SetPlanarCfg (bool *isColorByPlane*) `[inline]`

Set planar configuration.

Parameters:

isColorByPlane if the planar configuration is color-by-plane

Note:

Only used when channel number is bigger than 1. For RGB images, if it is color-by-plane, it's means the pixels should be stored as "R1 R2 R3 ... G1 G2 G3 ... B1 B2 B3 ...", otherwise "R1 G1 B1 R2 G2 B2 ..."

8.223.2.6 void mitkRawFilesWriter::SetTitleBytes (int *n*) `[inline]`

Set number of bytes in title.

Parameters:

n number of bytes in title

Note:

All the bytes in the title will be set to 00H. This title has nothing to do with the image and will be ignored when read.

The documentation for this class was generated from the following file:

- mitkRawFilesWriter.h

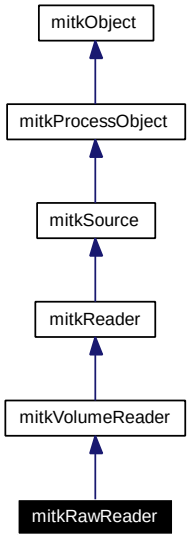
8.224 mitkRawReader Class Reference

mitkRawReader - a concrete reader for reading raw volume file(no header information).

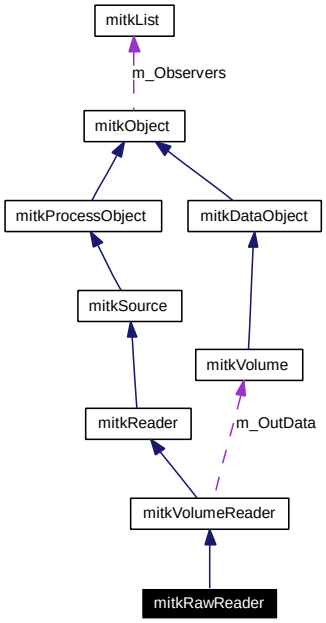
`#include <mitkRawReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkRawReader:



Collaboration diagram for mitkRawReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkRawReader ()
- void SetWidth (int w)
- void SetHeight (int h)
- void SetImageNum (int n)
- void SetSpacingX (float px)
- void SetSpacingY (float py)
- void SetSpacingZ (float pz)
- void SetChannelNum (int n)
- void SetTitleBytes (int n)
- void SetEndian (bool isBig=false)
- void SetBigEndian (bool isBig=true)
- void SetLittleEndian (bool isBig=false)
- void SetPlanarCfg (bool isColorByPlane)
- void SetDataType (int dataType)
- void SetDataTypeToFloat ()
- void SetDataTypeToDouble ()
- void SetDataTypeToInt ()
- void SetDataTypeToUnsignedInt ()
- void SetDataTypeToLong ()
- void SetDataTypeToUnsignedLong ()
- void SetDataTypeToShort ()
- void SetDataTypeToUnsignedShort ()
- void SetDataTypeToUnsignedChar ()
- void SetDataTypeToChar ()

8.224.1 Detailed Description

mitkRawReader - a concrete reader for reading raw volume file(no header information).

mitkRawReader reads a raw volume file(no any header information) to a volume. Because raw file doesn't has any header information, including width, height, image number, spacing information, etc. , you must set enough information using the series SetXXX functions. To use this reader, the code snippet is:

```
mitkRawReader *aReader = new mitkRawReader;
aReader->SetWidth(w);
aReader->SetHeight(h);
aReader->SetImageNum(d);
aReader->SetSpacingX(sx);
aReader->SetSpacingY(sy);
aReader->SetSpacingZ(sz);
aReader->SetDataType(dt);
aReader->SetChannelNum(cn);
aReader->AddFileName(filename); //Only require one file name
//the above is the required information
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

8.224.2 Constructor & Destructor Documentation

8.224.2.1 mitkRawReader::mitkRawReader ()

Default constructor.

8.224.3 Member Function Documentation

8.224.3.1 virtual void mitkRawReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

8.224.3.2 void mitkRawReader::SetBigEndian (bool *isBig* = true) [inline]

Provided for convenience, just the same as SetEndian().

8.224.3.3 void mitkRawReader::SetChannelNum (int *n*) [inline]

Set number of channel.

Parameters:
n number of channel 1—gray image 3—RGB image 4—RGBA image

8.224.3.4 void mitkRawReader::SetDataType (int *dataType*) [inline]

Set the data type of these images.

Parameters:
dataType the data type of these images, the value should be one of follows:
MITK_CHAR (char)
MITK_UNSIGNED_CHAR (unsigned char)
MITK_SHORT (short)
MITK_UNSIGNED_SHORT (unsigned short)
MITK_INT (int)
MITK_UNSIGNED_INT (unsigned int)
MITK_LONG (long)
MITK_UNSIGNED_LONG (unsigned long)
MITK_FLOAT (float)
MITK_DOUBLE (double)

8.224.3.5 void mitkRawReader::SetDataTypeToChar () [inline]

Set data type to char.

8.224.3.6 void mitkRawReader::SetDataTypeToDouble () [inline]

Set data type to double.

8.224.3.7 void mitkRawReader::SetDataTypeToFloat () [inline]

Set data type to float.

8.224.3.8 void mitkRawReader::SetDataTypeToInt () [inline]

Set data type to int.

8.224.3.9 void mitkRawReader::SetDataTypeToLong () [inline]

Set data type to long.

8.224.3.10 void mitkRawReader::SetDataTypeToShort () [inline]

Set data type to short.

8.224.3.11 void mitkRawReader::SetDataTypeToUnsignedChar () [inline]

Set data type to unsigned char.

8.224.3.12 void mitkRawReader::SetDataTypeToUnsignedInt () [inline]

Set data type to unsigned int.

8.224.3.13 void mitkRawReader::SetDataTypeToUnsignedLong () [inline]

Set data type to unsigned long.

8.224.3.14 void mitkRawReader::SetDataTypeToUnsignedShort () [inline]

Set data type to unsigned short.

8.224.3.15 void mitkRawReader::SetEndian (bool *isBig* = false) [inline]

Set endian if depth per pixel is bigger than 8.

Parameters:

isBig if the endian configuration is big endian (Mac)

8.224.3.16 void mitkRawReader::SetHeight (int *h*) [inline]

Set image height.

Parameters:
h image height

8.224.3.17 void mitkRawReader::SetImageNum (int *n*) [inline]

Set number of images.

Parameters:
n image number

8.224.3.18 void mitkRawReader::SetLittleEndian (bool *isBig* = false) [inline]

Provided for convenience, just the same as SetEndian().

8.224.3.19 void mitkRawReader::SetPlanarCfg (bool *isColorByPlane*) [inline]

Set planar configuration.

Parameters:
isColorByPlane if the planar configuration is color-by-plane

Note:
Only used when channel number is bigger than 1. For RGB images, if it is color-by-plane, it's means the pixels should be stored as "R1 R2 R3 ... G1 G2 G3 ... B1 B2 B3 ...", otherwise "R1 G1 B1 R2 G2 B2 ..."

8.224.3.20 void mitkRawReader::SetSpacingX (float *px*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.

8.224.3.21 void mitkRawReader::SetSpacingY (float *py*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:
py the spacing (mm) in two adjacent voxels in y axis.

8.224.3.22 void mitkRawReader::SetSpacingZ (float *pz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:
pz the spacing (mm) in two adjacent voxels in z axis.

8.224.3.23 void mitkRawReader::SetTitleBytes (int *n*) [inline]

Set number of bytes in title (header). These bytes will be skipped.

Parameters:
n number of bytes in title (header)

Note:
This title has nothing to do with the image. If the file has a title, the number of bytes must be given so that the program can skip the title properly when reading the file.

8.224.3.24 void mitkRawReader::SetWidth (int *w*) [inline]

Set image width.

Parameters:
w image width

The documentation for this class was generated from the following file:

- mitkRawReader.h

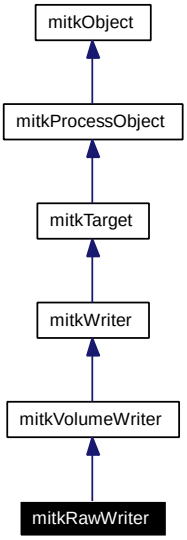
8.225 mitkRawWriter Class Reference

mitkRawWriter - a concrete writer for writing a volume to a single raw file(no any header information).

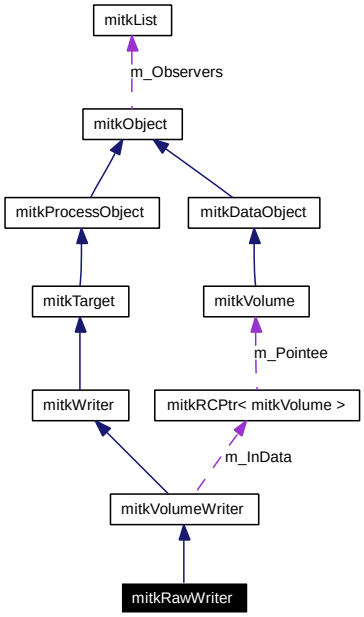
`#include <mitkRawWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkRawWriter:



Collaboration diagram for mitkRawWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetEndian (bool isBig=false)
- void SetBigEndian (bool isBig=true)
- void SetLittleEndian (bool isBig=false)
- void SetPlanarCfg (bool isColorByPlane)
- void SetTitleBytes (int n)

8.225.1 Detailed Description

mitkRawWriter - a concrete writer for writing a volume to a single raw file(no any header information).

mitkRawWriter writes a volume to a single raw file(no any header information). All the property of volume will lost, including the image width, height, image number, spacing information, etc. You can control the format of output file by setting the endian configuration, planar configuration and header size.

See also:
SetEndian
SetPlanarCfg
SetTitleBytes To use this writer, the code snippet is:

```
mitkRawWriter *aWriter = new mitkRawWriter;
aWriter->SetInput(aVolume);
aWriter->AddFileName(filename);
aWriter->Run();
```

8.225.2 Member Function Documentation

8.225.2.1 virtual void mitkRawWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

8.225.2.2 void mitkRawWriter::SetBigEndian (bool *isBig* = true) [inline]

Provided for convenience, just the same as SetEndian().

8.225.2.3 void mitkRawWriter::SetEndian (bool *isBig* = false) [inline]

Set endian if depth per pixel is bigger than 8.

Parameters:
isBig if the endian configuration is big endian (Mac)

8.225.2.4 void mitkRawWriter::SetLittleEndian (bool *isBig* = false) [inline]

Provided for convenience, just the same as SetEndian().

8.225.2.5 void mitkRawWriter::SetPlanarCfg (bool *isColorByPlane*) [inline]

Set planar configuration.

Parameters:
isColorByPlane if the planar configuration is color-by-plane

Note:
Only used when channel number is bigger than 1. For RGB images, if it is color-by-plane, it's means the pixels should be stored as "R1 R2 R3 ... G1 G2 G3 ... B1 B2 B3 ...", otherwise "R1 G1 B1 R2 G2 B2 ..."

8.225.2.6 void mitkRawWriter::SetTitleBytes (int *n*) [inline]

Set number of bytes in title.

Parameters:
n number of bytes in title

Note:
All the bytes in the title will be set to 00H. This title has nothing to do with the image and will be ignored when read.

The documentation for this class was generated from the following file:

- mitkRawWriter.h

8.226 mitkRCPtr< T > Class Template Reference

mitkRCPtr - a smart pointer class

`#include <mitkRCPtr.h>`

8.226.1 Detailed Description

template<class T> class mitkRCPtr< T >

mitkRCPtr - a smart pointer class

mitkRCPtr is a smart pointer class

The documentation for this class was generated from the following file:

- mitkRCPtr.h

8.227 mitkReader Class Reference

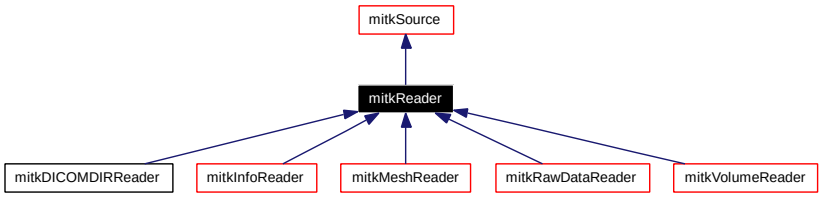
mitkReader - an abstract class represents a reader

`#include <mitkReader.h>`

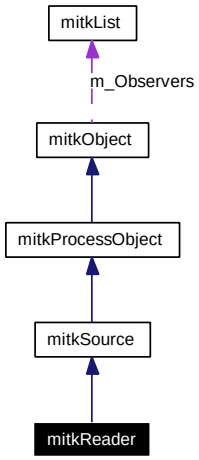
Inherits mitkSource.

Inherited by mitkDICOMDIRReader, mitkInfoReader, mitkMeshReader, mitkRawDataReader, and mitkVolumeReader.

Inheritance diagram for mitkReader:



Collaboration diagram for mitkReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddFileName (const char *inFileName)
- void SortFileNames ()

8.227.1 Detailed Description

mitkReader - an abstract class represents a reader

mitkReader defines the interface of all of the readers. To use a concrete reader, for example, mitkBMPReader, the code snippet is:

```
mitkBMPReader *aReader = new mitkBMPReader;
```

```
aReader->AddFileName(file1);
aReader->AddFileName(file2);
...  ...  ...
aReader->Run();
mitkVolume *aVolume = aReader->GetOutput();
Using aVolume
```

8.227.2 Member Function Documentation

8.227.2.1 void mitkReader::AddFileName (const char * *inFileName*)

Add a file into the internal list for reading these files.

Parameters:
inFileName C string for the file name.

8.227.2.2 virtual void mitkReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkSource.

Reimplemented in mitkAnalyzeImageReader, mitkBMPReader, mitkCacheVolumeReader, mitkDICOMInfoReader, mitkDICOMReader, mitkInfoReader, mitkJPEGReader, mitkMeshReader, mitkMetaImageReader, mitkNetCDFImageReader, mitkPLYReader, mitkRawFilesReader, mitkRawReader, mitkSTLReader, mitkTIFFReader, mitkVolumeReader, mitkRawDataReader, mitkRawFilesReader, and mitkTrackedBscanDataReader.

8.227.2.3 void mitkReader::SortFileNames ()

Sort all the file names alphabetically.

The documentation for this class was generated from the following file:

- mitkReader.h

8.228 mitkRectPlane Class Reference

mitkRectPlane - an encapsulation of a rectangle in 3D space

`#include <mitkVolumeResliceFilter.h>`

Public Member Functions

- bool CheckValidity ()

8.228.1 Detailed Description

mitkRectPlane - an encapsulation of a rectangle in 3D space

mitkRectPlane is an encapsulation of a rectangle in 3D space. The rectangle is decided by its three points: left-bottom point, right-bottom point and left-top point.

8.228.2 Member Function Documentation

8.228.2.1 bool mitkRectPlane::CheckValidity () [inline]

Check the validity of the rectangle, i.e. if $(\vec{v}_1 - \vec{v}_0) \perp (\vec{v}_2 - \vec{v}_0)$

Returns:
Return true if the rectangle is valid, otherwise return false.

The documentation for this class was generated from the following file:

- mitkVolumeResliceFilter.h

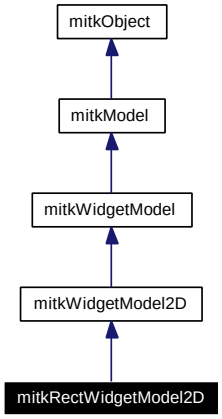
8.229 mitkRectWidgetModel2D Class Reference

mitkRectWidgetModel2D - a 2D widget for displaying a rectangle in an image scene

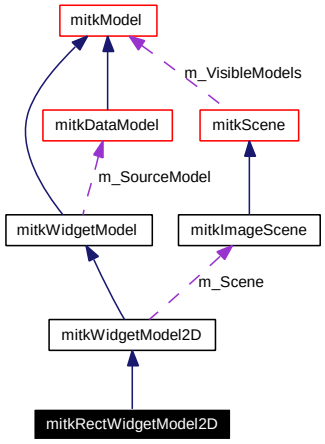
`#include <mitkRectWidgetModel2D.h>`

Inherits mitkWidgetModel2D.

Inheritance diagram for mitkRectWidgetModel2D:



Collaboration diagram for mitkRectWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- void SetStartPoint (float point[2])
- void SetStartPoint (float x, float y)
- void SetStartPoint (int sx, int sy)
- void SetMovePoint (float point[2])

- void SetMovePoint (float x, float y)
- void SetMovePoint (int sx, int sy)
- void SetEndPoint (float point[2])
- void SetEndPoint (float x, float y)
- void SetEndPoint (int sx, int sy)
- void SetUnitName (const string &name)
- const string & GetUnitName ()
- int GetWidth ()
- float GetTrueWidth ()
- int GetHeight ()
- float GetTrueHeight ()
- int GetLeft ()
- float GetTrueLeft ()
- int GetRight ()
- float GetTrueRight ()
- int GetBottom ()
- float GetTrueBottom ()
- int GetTop ()
- float GetTrueTop ()
- void GetStartPoint (int &sx, int &sy)
- void GetTrueStartPoint (float &sx, float &sy)
- void GetStartPoint (int p[2])
- void GetTrueStartPoint (float p[2])
- void GetEndPoint (int &ex, int &ey)
- void GetTrueEndPoint (float &ex, float &ey)
- void GetEndPoint (int p[2])
- void GetTrueEndPoint (float p[2])
- bool IsValid ()
- virtual mitkVolume * GetRegionMask ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.229.1 Detailed Description

mitkRectWidgetModel2D - a 2D widget for displaying a rectangle in an image scene

mitkRectWidgetModel2D is a 2D widget for displaying a rectangle in an image scene. It can respond the mouse events and return the current width and height of this rectangle. It is supposed to be attached to a 2D data model (e.g. mitkImageModel) and add to a 2D scene (e.g. mitkImageScene), and in other conditions the display could be improper.

8.229.2 Member Function Documentation

8.229.2.1 virtual void mitkRectWidgetModel2D::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.229.2.2 virtual void mitkRectWidgetModel2D::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.229.2.3 virtual void mitkRectWidgetModel2D::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.229.2.4 int mitkRectWidgetModel2D::GetBottom ()

Get the bottom of the rectangle in screen space.

Returns:

Return the bottom of the rectangle.

8.229.2.5 void mitkRectWidgetModel2D::GetEndPoint (int *p*[2]) [inline]

Get the end point of the rectangle in screen space.

Parameters:

p[0] return the x-coordinate of the start point

p[1] return the y-coordinate of the start point

8.229.2.6 void mitkRectWidgetModel2D::GetEndPoint (int & *ex*, int & *ey*) [inline]

Get the end point of the rectangle in screen space.

Parameters:

ex return the x-coordinate of the end point

ey return the y-coordinate of the end point

8.229.2.7 int mitkRectWidgetModel2D::GetHeight ()

Get the height of the rectangle in screen space.

Returns:

Return the height of the pseudocolor rectangle.

8.229.2.8 int mitkRectWidgetModel2D::GetLeft ()

Get the left of the rectangle in screen space.

Returns:

Return the left of the rectangle.

8.229.2.9 virtual mitkVolume* mitkRectWidgetModel2D::GetRegionMask () [virtual]

Get mask image where the value of widget region is 255 and the rest is 0.

Returns:

Return a pointer of mitkVolume object contains the mask image.

Note:

The returned object pointer should be deleted properly by yourself.

Reimplemented from mitkWidgetModel2D.

8.229.2.10 int mitkRectWidgetModel2D::GetRight ()

Get the right of the rectangle in screen space.

Returns:

Return the right of the rectangle.

8.229.2.11 void mitkRectWidgetModel2D::GetStartPoint (int *p*[2]) [inline]

Get the start point of the rectangle in screen space.

Parameters:
p[0] return the x-coordinate of the start point
p[1] return the y-coordinate of the start point

8.229.2.12 void mitkRectWidgetModel2D::GetStartPoint (int & *sx*, int & *sy*) [inline]

Get the start point of the rectangle in screen space.

Parameters:
sx return the x-coordinate of the start point
sy return the y-coordinate of the start point

8.229.2.13 int mitkRectWidgetModel2D::GetTop ()

Get the top of the rectangle in screen space.

Returns:
Return the left of the rectangle.

8.229.2.14 float mitkRectWidgetModel2D::GetTrueBottom ()

Get the true bottom of the rectangle in source model space.

Returns:
Return the bottom of the rectangle.

8.229.2.15 void mitkRectWidgetModel2D::GetTrueEndPoint (float *p*[2]) [inline]

Get the true end point of the rectangle in source model space.

Parameters:
p[0] return the x-coordinate of the start point
p[1] return the y-coordinate of the start point

8.229.2.16 void mitkRectWidgetModel2D::GetTrueEndPoint (float & *ex*, float & *ey*) [inline]

Get the true end point of the rectangle in source model space.

Parameters:
ex return the x-coordinate of the end point
ey return the y-coordinate of the end point

8.229.2.17 float mitkRectWidgetModel2D::GetTrueHeight ()

Get the true height of the rectangle in source model space.

Returns:
Return the height of the pseudocolor rectangle.

8.229.2.18 float mitkRectWidgetModel2D::GetTrueLeft ()

Get the true left of the rectangle in source model space.

Returns:
Return the left of the rectangle.

8.229.2.19 float mitkRectWidgetModel2D::GetTrueRight ()

Get the true right of the rectangle in source model space.

Returns:
Return the right of the rectangle.

8.229.2.20 void mitkRectWidgetModel2D::GetTrueStartPoint (float *p*[2]) [inline]

Get the true start point of the rectangle in source model space.

Parameters:
p[0] return the x-coordinate of the start point

p[1] return the y-coordinate of the start point

8.229.2.21 void mitkRectWidgetModel2D::GetTrueStartPoint (float & *sx*, float & *sy*) [inline]

Get the true start point of the rectangle in source model space.

Parameters:
sx return the x-coordinate of the start point

sy return the y-coordinate of the start point

8.229.2.22 float mitkRectWidgetModel2D::GetTrueTop ()

Get the true top of the rectangle in source model space.

Returns:
Return the left of the rectangle.

8.229.2.23 float mitkRectWidgetModel2D::GetTrueWidth ()

Get the true width of the rectangle in source model space.

Returns:
Return the width of the pseudocolor rectangle.

8.229.2.24 const string& mitkRectWidgetModel2D::GetUnitName () [inline]

Get name string of unit.

Returns:
Return a constant reference to a string contains the name of the length unit this line uses

8.229.2.25 int mitkRectWidgetModel2D::GetWidth ()

Get the width of the rectangle in screen space.

Returns:
Return the width of the pseudocolor rectangle.

8.229.2.26 bool mitkRectWidgetModel2D::IsValid () [inline]

If the widget model is valid.

Returns:
Return if the widget model is valid (i.e. can be shown in the scene).

8.229.2.27 virtual void mitkRectWidgetModel2D::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:
names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.229.2.28 virtual void mitkRectWidgetModel2D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel2D.

8.229.2.29 virtual void mitkRectWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.229.2.30 virtual int mitkRectWidgetModel2D::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:

scene the pointer of an mitkScene in which this model will be shown

Returns:

Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.229.2.31 void mitkRectWidgetModel2D::SetEndPoint (int *sx*, int *sy*) [inline]

Set position of the end point.

Parameters:

sx x-coordinate of the end point of mouse dragging on screen

sy y-coordinate of the end point of mouse dragging on screen

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.229.2.32 void mitkRectWidgetModel2D::SetEndPoint (float *x*, float *y*) [inline]

Set position of the end point in the object space.

Parameters:

x x-coordinate of the end point of mouse dragging

y y-coordinate of the end point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.229.2.33 void mitkRectWidgetModel2D::SetEndPoint (float *point*[2]) [inline]

Set position of the end point in the object space.

Parameters:

point[0] x-coordinate of the end point of mouse dragging

point[1] y-coordinate of the end point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.229.2.34 void mitkRectWidgetModel2D::SetMovePoint (int *sx*, int *sy*)

Set position of the moving end point.

Parameters:

sx x-coordinate of the moving end point of mouse dragging on screen

sy y-coordinate of the moving end point of mouse dragging on screen

Note:

The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.229.2.35 void mitkRectWidgetModel2D::SetMovePoint (float *x*, float *y*)

Set position of the moving end point in the object space.

Parameters:

x x-coordinate of the moving end point of mouse dragging

y y-coordinate of the moving end point of mouse dragging

Note:

The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.229.2.36 void mitkRectWidgetModel2D::SetMovePoint (float *point*[2]) [inline]

Set position of the moving end point in the object space.

Parameters:

point[0] x-coordinate of the moving end point of mouse dragging

point[1] y-coordinate of the moving end point of mouse dragging

Note:
The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.229.2.37 void mitkRectWidgetModel2D::SetStartPoint (int *sx*, int *sy*)

Set position of the start point.

Parameters:
sx x-coordinate of the start point of mouse dragging on screen

sy y-coordinate of the start point of mouse dragging on screen

Note:
The coordinates of the point are in the screen coordinate system. This function is useful when you can not get the original coordinates in the object space easily outside. It can do this job for you. But if this widget is attach to a data model, you must ensure the source model and the scene which contains this widget and the source model are properly set to this widget (e.g. call SetSourceModel() of this widget or call AddWidget() of the source model and SetScene() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the scene to calculate the original coordinates.

8.229.2.38 void mitkRectWidgetModel2D::SetStartPoint (float *x*, float *y*)

Set position of the start point in the object space.

Parameters:
x x-coordinate of the start point of mouse dragging

y y-coordinate of the start point of mouse dragging

Note:
The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.229.2.39 void mitkRectWidgetModel2D::SetStartPoint (float *point*[2]) `[inline]`

Set position of the start point in the object space.

Parameters:
point[0] x-coordinate of the start point of mouse dragging

point[1] y-coordinate of the start point of mouse dragging

Note:
The coordinates of the point must be the original coordinates in the object space before all geometrical transforms. (Because this widget is for 2D image, the z-coordinate will always be zero.)

8.229.2.40 void mitkRectWidgetModel2D::SetUnitName (const string & *name*) [inline]

Set name string of unit.

Parameters:

name a constant reference to a string contains the name of the length unit (e.g. mm) this line uses

The documentation for this class was generated from the following file:

- mitkRectWidgetModel2D.h

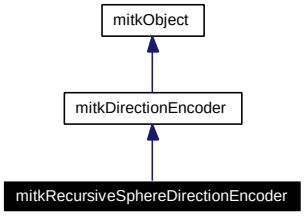
8.230 mitkRecursiveSphereDirectionEncoder Class Reference

mitkRecursiveSphereDirectionEncoder - A direction encoder based on the recursive subdivision of an octahedron.

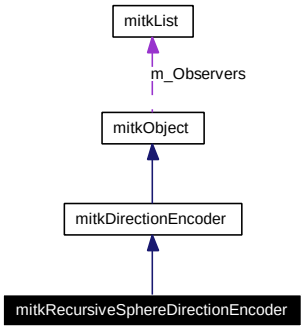
`#include <mitkRecursiveSphereDirectionEncoder.h>`

Inherits mitkDirectionEncoder.

Inheritance diagram for mitkRecursiveSphereDirectionEncoder:



Collaboration diagram for mitkRecursiveSphereDirectionEncoder:



Public Member Functions

- int GetEncodedDirection (float n[3])
- float * GetDecodedGradient (int value)
- int GetNumberOfEncodedDirections ()
- float * GetDecodedGradientTable ()
- virtual void SetRecursionDepth (int fnDepth)
- virtual int GetRecursionDepthMinValue ()
- virtual int GetRecursionDepthMaxValue ()
- virtual int GetRecursionDepth ()

8.230.1 Detailed Description

mitkRecursiveSphereDirectionEncoder - A direction encoder based on the recursive subdivision of an octahedron.

mitkRecursiveSphereDirectionEncoder is a direction encoder which uses the vertices of a recursive subdivision of an octahedron (with the vertices pushed out onto the surface of an enclosing sphere) to encode directions into a two byte value. Some codes are borrowed from VTK, and please see the copyright at end.

8.230.2 Member Function Documentation

8.230.2.1 float* mitkRecursiveSphereDirectionEncoder::GetDecodedGradient (int *value*) [virtual]

Given an encoded value, return a pointer to the normal vector

Parameters:
value Represent the encoded value

Returns:
Return a pointer to the normal vector

Implements mitkDirectionEncoder.

8.230.2.2 float* mitkRecursiveSphereDirectionEncoder::GetDecodedGradientTable () [virtual]

Get the decoded gradient table. There are this->GetNumberOfEncodedDirections() entries in the table, each containing a normal (direction) vector. This is a flat structure - 3 times the number of directions floats in an array.

Returns:
Return the decoded gradient table

Implements mitkDirectionEncoder.

8.230.2.3 int mitkRecursiveSphereDirectionEncoder::GetEncodedDirection (float *n*[3]) [virtual]

Given a normal vector n, return the encoded direction

Parameters:
n Represent the normal vector

Returns:
Return the encoded direction

Implements mitkDirectionEncoder.

8.230.2.4 int mitkRecursiveSphereDirectionEncoder::GetNumberOfEncodedDirections () [virtual]

Return the number of encoded directions

Returns:
Return the number of encoded directions

Implements mitkDirectionEncoder.

8.230.2.5 virtual int mitkRecursiveSphereDirectionEncoder::GetRecursionDepth () [inline, virtual]

Get the recursion depth for the subdivision. This indicates how many time one triangle on the initial 8-sided sphere model is replaced by four triangles formed by connecting triangle edge midpoints. A recursion level of 0 yields 8 triangles with 6 unique vertices. The normals are the vectors from the sphere center through the vertices. The number of directions will be 11 since the four normals with 0 z values will be duplicated in the table - once with +0 values and the other time with -0 values, and an addition index will be used to represent the (0,0,0) normal. If we instead choose a recursion level of 6 (the maximum that can fit within 2 bytes) the number of directions is 16643, with 16386 unique directions and a zero normal.

Returns:
Return the recursion depth value

8.230.2.6 virtual int mitkRecursiveSphereDirectionEncoder::GetRecursionDepthMaxValue () [inline, virtual]

Get the max recursion depth value

Returns:
fnDepth Return the max recursion depth value

8.230.2.7 virtual int mitkRecursiveSphereDirectionEncoder::GetRecursionDepthMinValue () [inline, virtual]

Get the min recursion depth value

Returns:
fnDepth Return the min recursion depth value

8.230.2.8 virtual void mitkRecursiveSphereDirectionEncoder::SetRecursionDepth (int *fnDepth*) [virtual]

Set the recursion depth for the subdivision. This indicates how many time one triangle on the initial 8-sided sphere model is replaced by four triangles formed by connecting triangle edge midpoints. A recursion level of 0 yields 8 triangles with 6 unique vertices. The normals are the vectors from the sphere center through the vertices. The number of directions will be 11 since the four normals with 0 z values will be duplicated in the table - once with +0 values and the other time with -0 values, and an addition index will be used to represent the (0,0,0) normal. If we instead choose a recursion level of 6 (the maximum that can fit within 2 bytes) the number of directions is 16643, with 16386 unique directions and a zero normal.

Parameters:
fnDepth Represent the recursion depth value

The documentation for this class was generated from the following file:

- mitkRecursiveSphereDirectionEncoder.h

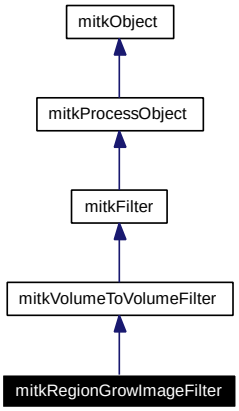
8.231 mitkRegionGrowImageFilter Class Reference

mitkRegionGrowImageFilter - A class that implement region grow segmentation arithmetic

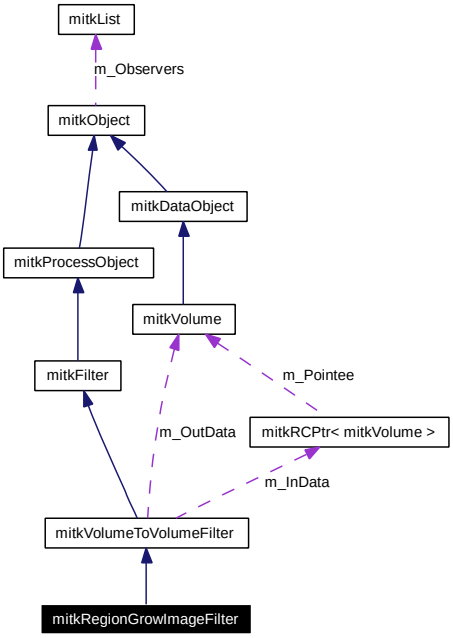
`#include <mitkRegionGrowImageFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkRegionGrowImageFilter:



Collaboration diagram for mitkRegionGrowImageFilter:



Public Member Functions

- mitkRegionGrowImageFilter ()
- virtual void PrintSelf (ostream &os)

- virtual ~mitkRegionGrowImageFilter ()
- void SetSeedPoint (int x, int y, int z)
- void SetMaxDifferentValue (double v)
- void SetMaxChangeValue (double v)

8.231.1 Detailed Description

mitkRegionGrowImageFilter - A class that implement region grow segmentation arithmetic

mitkRegionGrowImageFilter - A class that implement region grow segmentation arithmetic

8.231.2 Constructor & Destructor Documentation

8.231.2.1 mitkRegionGrowImageFilter::mitkRegionGrowImageFilter ()

Constructor of the class.

8.231.2.2 virtual mitkRegionGrowImageFilter::~mitkRegionGrowImageFilter () [virtual]

Constructor of the class.

8.231.3 Member Function Documentation

8.231.3.1 virtual void mitkRegionGrowImageFilter::PrintSelf (ostream & *os*) [inline, virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.231.3.2 void mitkRegionGrowImageFilter::SetMaxChangeValue (double *v*) [inline]

Set the max change value

Parameters:

v Represent the max change value

8.231.3.3 void mitkRegionGrowImageFilter::SetMaxDifferentValue (double *v*) [inline]

Set the max different value

Parameters:

v Represent the max different value

8.231.3.4 void mitkRegionGrowImageFilter::SetSeedPoint (int *x*, int *y*, int *z*) [inline]

Set the coordinateof the seed point

Parameters:

- ***x*** Represent the x coordinate of the point
- ***y*** Represent the y coordinate of the point
- ***z*** Represett the z coordinate of the point

The documentation for this class was generated from the following file:

- mitkRegionGrowImageFilter.h

8.232 mitkRegistrationFilter Class Reference

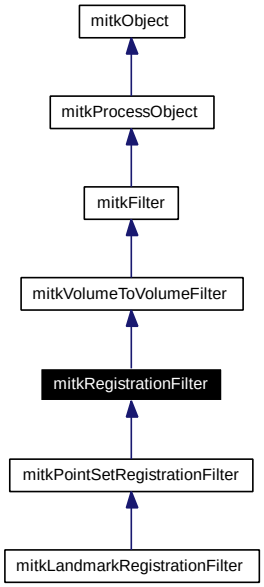
mitkRegistrationFilter - a class for registration filter

`#include <mitkRegistrationFilter.h>`

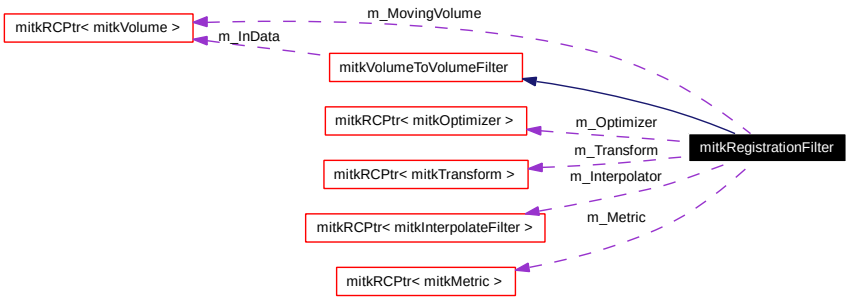
Inherits mitkVolumeToVolumeFilter.

Inherited by mitkPointSetRegistrationFilter.

Inheritance diagram for mitkRegistrationFilter:



Collaboration diagram for mitkRegistrationFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetFixedVolume (mitkVolume *fixedVolume)
- void SetMovingVolume (mitkVolume *movingVolume)
- mitkVolume * GetMovingVolume ()
- void SetOptimizer (mitkOptimizer *optimizer)

- virtual mitkOptimizer * GetOptimizer ()
- void SetMetric (mitkMetric *metric)
- virtual mitkMetric * GetMetric ()
- void SetInterpolator (mitkInterpolateFilter *interpolator)
- mitkInterpolateFilter * GetInterpolator ()
- void SetTransform (mitkTransform *transform)
- virtual mitkTransform * GetTransform ()
- void SetInitialParameters (VectorParameterType &parameters)
- void SetInitialParameters (double *parameters, unsigned int num)
- VectorParameterType & GetInitialParameters ()
- VectorParameterType & GetLastParameters ()
- void GetLastParameters (double *parameters)
- void SetRegisrationRegion (int r[6])
- bool IsOptimizationFinished ()
- mitkRegistrationFilter ()
- mitkVolume * GetOutput ()

8.232.1 Detailed Description

mitkRegistrationFilter - a class for registration filter

mitkRegistrationFilter is a class for registration filter. This type of filter has two volume as input and generates a volume as output.

A generic Transform is used by this class. That allows to select at run time the particular type of transformation that is to be applied for registering the images.

This method use a generic Metric in order to compare the two images. the final goal of the registration method is to find the set of parameters of the Transformation that optimizes the metric.

The registration method also support a generic optimizer that can be selected at run-time. The only restriction for the optimizer is that it should be able to operate in single-valued cost functions given that the metrics used to compare images provide a single value as output.

The terms : Fixed image and Moving image are used in this class to indicate what image is being mapped by the transform.

This class uses the coordinate system of the Fixed image as a reference and searchs for a Transform that will map points from the space of the Fixed image to the space of the Moving image.

For doing so, a Metric will be continously applied to compare the Fixed image with the Transformed Moving image. This process also requires to interpolate values from the Moving image.

8.232.2 Constructor & Destructor Documentation

8.232.2.1 mitkRegistrationFilter::mitkRegistrationFilter ()

Constructor.

8.232.3 Member Function Documentation

8.232.3.1 VectorParameterType& mitkRegistrationFilter::GetInitialParameters ()

Get the initial transform parameters

Returns:
Return the initial transform parameters

8.232.3.2 mitkInterpolateFilter* mitkRegistrationFilter::GetInterpolator ()

Get the interpolate method

Returns:
Return interpolate method

8.232.3.3 void mitkRegistrationFilter::GetLastParameters (double * *parameters*)

Get the last transform parameters (Array Version)

Parameters:
parameters An array pointer to store the parameter values.

8.232.3.4 VectorParameterType& mitkRegistrationFilter::GetLastParameters ()

Get the last transform parameters.

Returns:
Return the last transform parameters

8.232.3.5 virtual mitkMetric* mitkRegistrationFilter::GetMetric () [virtual]

Get the similarity metric method

Returns:
Return similarity metric method

Reimplemented in mitkPointSetRegistrationFilter.

8.232.3.6 mitkVolume* mitkRegistrationFilter::GetMovingVolume () [inline]

Get the Fixed volume

Returns:
Return the Fixed volume

8.232.3.7 virtual mitkOptimizer* mitkRegistrationFilter::GetOptimizer () [virtual]

Get the optimize method

Returns:
Return the optimize method

Reimplemented in mitkPointSetRegistrationFilter.

8.232.3.8 mitkVolume* mitkRegistrationFilter::GetOutput ()

Get the output volume

Returns:
Return the output volume

Reimplemented from mitkVolumeToVolumeFilter.

8.232.3.9 virtual mitkTransform* mitkRegistrationFilter::GetTransform () [virtual]

Get the transform method

Returns:
Return transform method

Reimplemented in mitkPointSetRegistrationFilter.

8.232.3.10 bool mitkRegistrationFilter::IsOptimizationFinished () [inline]

Get the status of registration process.

Returns:
m_FinishOptimization Return true if optimization was done.

8.232.3.11 virtual void mitkRegistrationFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

Reimplemented in mitkLandmarkRegistrationFilter, and mitkPointSetRegistrationFilter.

8.232.3.12 void mitkRegistrationFilter::SetFixedVolume (mitkVolume * *fixedVolume*)

Set the Fixed volume

Parameters:
FixedVolume Specify the Fixed volume

8.232.3.13 void mitkRegistrationFilter::SetInitialParameters (double * *parameters*, unsigned int *num*)

Set the initial transform parameters.

Parameters:
initialTransformParameters Specify the initial transform parameters
numberOfParameters Number of initialized parameters.

8.232.3.14 void mitkRegistrationFilter::SetInitialParameters (VectorParameterType & *parameters*)

Set the initial transform parameters.

Parameters:
initialTransformParameters Specify the initial transform parameters

8.232.3.15 void mitkRegistrationFilter::SetInterpolator (mitkInterpolateFilter * *interpolator*) [inline]

Set the interpolate method

Parameters:
interpolator Specify interpolate method

8.232.3.16 void mitkRegistrationFilter::SetMetric (mitkMetric * *metric*) [inline]

Set the similarity metric method

Parameters:
metric Specify similarity metric method

8.232.3.17 void mitkRegistrationFilter::SetMovingVolume (mitkVolume * *movingVolume*)

Set the Moving volume

Parameters:
movingVolume Specify the Moving volume

8.232.3.18 void mitkRegistrationFilter::SetOptimizer (mitkOptimizer * *optimizer*)

Set the optimize method

Parameters:
optimizer Specify the optimize method

8.232.3.19 void mitkRegistrationFilter::SetRegisrationRegion (int *r*[6])

Set the registration region of volumes.

Parameters:
r[0] The first (smaller) index in x axis, the unit is pixel.
r[1] The second (bigger) index in x axis, the unit is pixel.
r[2] The first (smaller) index in y axis, the unit is pixel.
r[3] The second (bigger) index in y axis, the unit is pixel.
r[4] The first (smaller) index in z axis, the unit is pixel.
r[5] The second (bigger) index in z axis, the unit is pixel.

8.232.3.20 void mitkRegistrationFilter::SetTransform (mitkTransform * *transform*)

Set the transform method

Parameters:
transform Specify transform method

The documentation for this class was generated from the following file:

- mitkRegistrationFilter.h

8.233 mitkRenderer Class Reference

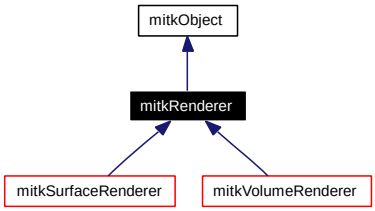
mitkRenderer - an abstract class to define the common interface of a renderer

`#include <mitkRenderer.h>`

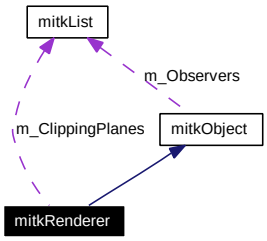
Inherits mitkObject.

Inherited by mitkSurfaceRenderer, and mitkVolumeRenderer.

Inheritance diagram for mitkRenderer:



Collaboration diagram for mitkRenderer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddClippingPlane (mitkPlane *plane)
- void RemoveClippingPlane (mitkPlane *plane)
- void RemoveAllClippingPlanes ()
- mitkList * GetClippingPlanes (void)
- mitkPlane * GetClippingPlane (int planeIndex)
- int GetClippingPlaneCount (void)
- void ClippingOn ()
- void ClippingOff ()
- void SetClipping (bool enableClipping)
- bool GetClipping ()

8.233.1 Detailed Description

mitkRenderer - an abstract class to define the common interface of a renderer

mitkRenderer defines the common interface of a renderer. A renderer render a model (either surface model or volume model) to a view. Its concrete subclasses implement the actual rendering algorithm.

8.233.2 Member Function Documentation

8.233.2.1 void mitkRenderer::AddClippingPlane (mitkPlane * *plane*)

Add a specified clipping plane to this renderer(at most 6 clipping planes can be specified).

Parameters:
plane The specified plane

8.233.2.2 void mitkRenderer::ClippingOff () [inline]

Disable the clipping

8.233.2.3 void mitkRenderer::ClippingOn () [inline]

Enable the clipping

8.233.2.4 bool mitkRenderer::GetClipping () [inline]

Get the status of clipping

Returns:
Return truethe clipping is enabled Return false, the clipping is disabled

8.233.2.5 mitkPlane* mitkRenderer::GetClippingPlane (int *planeIndex*)

Get the clipping plane in the specified index.

Parameters:
planeIndex Specify the plane index(zero based) in the plane list.

Returns:
Return the clipping plane in the specified index

8.233.2.6 int mitkRenderer::GetClippingPlaneCount (void) [inline]

Get the count of clipping plane in this renderer.

Returns:
Return the count of clipping plane in this renderer.

8.233.2.7 mitkList* mitkRenderer::GetClippingPlanes (void) [inline]

Get the plane list of this renderer.

Returns:
Return the internal plane list

8.233.2.8 virtual void mitkRenderer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkOoCSurfaceRendererStandard, mitkOoCSurfaceRendererUseVA, mitkOoCVolumeRendererRayCasting, mitkOoCVolumeRendererRayCastingEx, mitkOoCVolumeRendererShearWarp, mitkOoCVolumeRendererTexture3D, mitkOoCVolumeRendererTexture3DEx, mitkSurfaceRenderer, mitkSurfaceRendererStandard, mitkSurfaceRendererUseVA, mitkSurfaceRendererUseVBO, mitkVolumeRenderer, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, and mitkVolumeRendererTexture3D.

8.233.2.9 void mitkRenderer::RemoveAllClippingPlanes ()

Remove all clipping planes from this renderer.

8.233.2.10 void mitkRenderer::RemoveClippingPlane (mitkPlane * *plane*)

Remove a specified clipping plane from this renderer.

Parameters:
plane The specified plane. If plane is not in the plane list of this renderer, nothing happens. Otherwise, plane is removed from the plane list.

8.233.2.11 void mitkRenderer::SetClipping (bool *enableClipping*) [inline]

Set the status of clipping

Parameters:
enableClipping enableClipping = trueenable the clipping enableClipping = falsedisable the clipping

The documentation for this class was generated from the following file:

- mitkRenderer.h

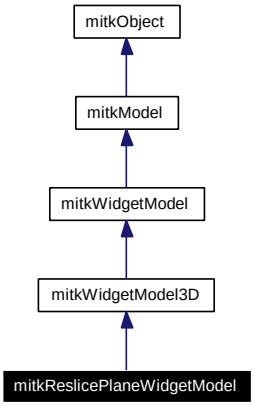
8.234 mitkReslicePlaneWidgetModel Class Reference

mitkReslicePlaneWidgetModel - a 3D widget for re-slice plane

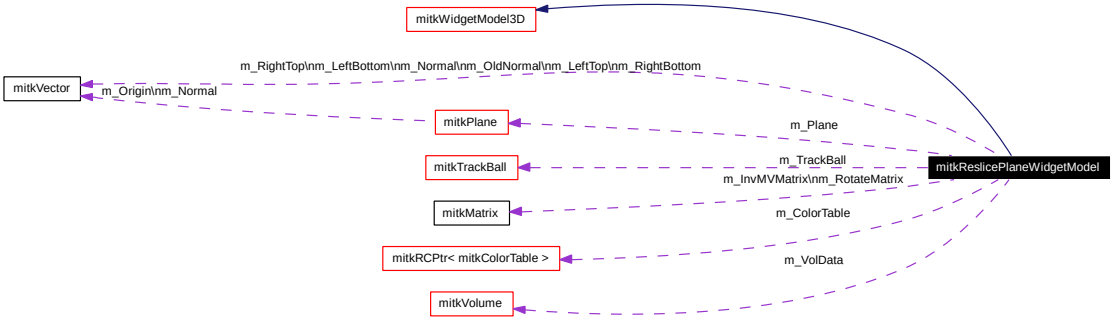
`#include <mitkReslicePlaneWidgetModel.h>`

Inherits mitkWidgetModel3D.

Inheritance diagram for mitkReslicePlaneWidgetModel:



Collaboration diagram for mitkReslicePlaneWidgetModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkReslicePlaneWidgetModel (coord_type v0x, coord_type v0y, coord_type v0z, coord_type v1x, coord_type v1y, coord_type v1z, coord_type cx, coord_type cy, coord_type cz)
- virtual int Render (mitkScene *scene)
- virtual void Pick (const WidgetNames &names)
- virtual void Release ()
- virtual void SetSourceModel (mitkDataModel *model)
- void SetPlanePosition (coord_type v0x, coord_type v0y, coord_type v0z, coord_type v1x, coord_type v1y, coord_type v1z, coord_type cx, coord_type cy, coord_type cz)
- void SetPlaneOpacity (float opacity)
- void SetSliceOpacity (float opacity)
- float GetPlaneOpacity ()

- float GetSliceOpacity ()
- void SetVolumeData (mitkVolume *vol)
- mitkVolume * GetVolumeData ()
- void SetSliceImageWidth (int width)
- void SetSliceImageHeight (int height)
- int GetSliceImageWidth ()
- int GetSliceImageHeight ()
- void GetLeftBottomPoint (coord_type &x, coord_type &y, coord_type &z)
- mitkVector const * GetLeftBottomPoint ()
- void GetRightBottomPoint (coord_type &x, coord_type &y, coord_type &z)
- mitkVector const * GetRightBottomPoint ()
- void GetRightTopPoint (coord_type &x, coord_type &y, coord_type &z)
- mitkVector const * GetRightTopPoint ()
- void GetLeftTopPoint (coord_type &x, coord_type &y, coord_type &z)
- mitkVector const * GetLeftTopPoint ()
- virtual void Update ()
- void EnableReslice (bool enable=true)
- void DisableReslice ()
- void SetColorTable (mitkColorTable *ct)
- mitkColorTable * GetColorTable ()
- void EnablePseudocolor (bool enable=true)
- void DisablePseudocolor ()
- bool IsResliceEnabled ()
- bool IsPseudocolorEnabled ()
- void StoreEnableState ()
- mitkVolume * GetReslicedImage ()
- void RotateRadAroundXAxisOfPlane (float angle)
- void RotateRadAroundYAxisOfPlane (float angle)
- void RotateRadAroundZAxisOfPlane (float angle)
- void RotateDegAroundXAxisOfPlane (float angle)
- void RotateDegAroundYAxisOfPlane (float angle)
- void RotateDegAroundZAxisOfPlane (float angle)
- void TranslatePlane (float tx, float ty, float tz)
- void SetPlaneCenter (float ox, float oy, float oz)
- void GetPlaneNormal (float &nx, float &ny, float &nz)
- void GetPlaneNormal (float n[3])
- void SetUnitName (const string &name)
- const string & GetUnitName ()
- float GetRotatedAngleRad ()
- float GetRotatedAngleDeg ()
- float GetMovedDistance ()
- float GetVertMoveFromStart ()
- float const * GetStartOrigin ()
- void GetStartOrigin (float so[3])
- bool IsBallSelected ()
- bool IsLineSelected ()

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)

8.234.1 Detailed Description

mitkReslicePlaneWidgetModel - a 3D widget for re-slice plane

mitkReslicePlaneWidgetModel is a 3D widget for re-slice plane. It can clip the source model at an arbitrary position and along an arbitrary direction, and at the same time, reconstruct the slice of the transection and display it on the clipping plane.

The clipping rectangle plane is initialized by its left-bottom point, right-bottom point and center point in the constructor of this class.

8.234.2 Constructor & Destructor Documentation

8.234.2.1 mitkReslicePlaneWidgetModel::mitkReslicePlaneWidgetModel (coord_type *v0x*, coord_type *v0y*, coord_type *v0z*, coord_type *v1x*, coord_type *v1y*, coord_type *v1z*, coord_type *cx*, coord_type *cy*, coord_type *cz*)

A constructor. The parameters specify a rectangle in the model space via its left-bottom point, right-bottom point and center point.

Parameters:

v0x the x-coordinate of the left-bottom point

v0y the y-coordinate of the left-bottom point

v0z the z-coordinate of the left-bottom point

v1x the x-coordinate of the right-bottom point

v1y the y-coordinate of the right-bottom point

v1z the z-coordinate of the right-bottom point

cx the x-coordinate of the center point

cy the y-coordinate of the center point

cz the z-coordinate of the center point

8.234.3 Member Function Documentation

8.234.3.1 virtual void mitkReslicePlaneWidgetModel::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkWidgetModel.

8.234.3.2 virtual void mitkReslicePlaneWidgetModel::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implements mitkWidgetModel.

8.234.3.3 virtual void mitkReslicePlaneWidgetModel::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkWidgetModel.

8.234.3.4 void mitkReslicePlaneWidgetModel::DisablePseudocolor () [inline]

Display pseudo-color display for re-slice.

8.234.3.5 void mitkReslicePlaneWidgetModel::DisableReslice () [inline]

Disable re-slice (do not reconstruct the slice of the transection).

8.234.3.6 void mitkReslicePlaneWidgetModel::EnablePseudocolor (bool *enable* = true)

Enable pseudo-color display for re-slice.

Parameters:

enable if enable is true, the pseudo-color function is enabled, otherwise it is disabled

8.234.3.7 void mitkReslicePlaneWidgetModel::EnableReslice (bool *enable* = `true`)

Enable re-slice (reconstruct the slice of the transection and display it on the clipping plane).

Parameters:

enable if enable is true, the re-slice function is enabled, otherwise it is disabled

8.234.3.8 mitkColorTable* mitkReslicePlaneWidgetModel::GetColorTable ()

Get color table.

Returns:

Return the pointer to the mitkColorTable object.

8.234.3.9 mitkVector const* mitkReslicePlaneWidgetModel::GetLeftBottomPoint () `[inline]`

Get the left-bottom point of the re-slice rectangle plane in the model space.

Returns:

Return a point to a const mitkVector object which contains the coordinates of the point.

8.234.3.10 void mitkReslicePlaneWidgetModel::GetLeftBottomPoint (coord_type & *x*, coord_type & *y*, coord_type & *z*) `[inline]`

Get the left-bottom point of the re-slice rectangle plane in the model space.

Parameters:

x the x-coordinate of the point

y the y-coordinate of the point

z the z-coordinate of the point

8.234.3.11 mitkVector const* mitkReslicePlaneWidgetModel::GetLeftTopPoint () `[inline]`

Get the left-top point of the re-slice rectangle plane in the model space.

Returns:

Return a point to a const mitkVector object which contains the coordinates of the point.

8.234.3.12 void mitkReslicePlaneWidgetModel::GetLeftTopPoint (coord_type & *x*, coord_type & *y*, coord_type & *z*) `[inline]`

Get the left-top point of the re-slice rectangle plane in the model space.

Parameters:

x the x-coordinate of the point

y the y-coordinate of the point

z the z-coordinate of the point

8.234.3.13 float mitkReslicePlaneWidgetModel::GetMovedDistance ()

Get moved distance since mouse button is pressed.

Returns:
Return the value of moved distance.

8.234.3.14 void mitkReslicePlaneWidgetModel::GetPlaneNormal (float *n*[3]) [inline]

Get the normal of the plane.

Parameters:
n[0] return the x weight of the normal

n[1] return the y weight of the normal

n[2] return the z weight of the normal

8.234.3.15 void mitkReslicePlaneWidgetModel::GetPlaneNormal (float & *nx*, float & *ny*, float & *nz*)

Get the normal of the plane.

Parameters:
nx return the x weight of the normal

ny return the y weight of the normal

nz return the z weight of the normal

8.234.3.16 float mitkReslicePlaneWidgetModel::GetPlaneOpacity () [inline]

Get the opacity of the plane.

Returns:
Return the opacity of the plane (the value is between 0.0f and 1.0f, and the default value is 0.0f)

8.234.3.17 mitkVolume* mitkReslicePlaneWidgetModel::GetReslicedImage ()

Get current re-sliced image.

Returns:
Return a pointer of mitkVolume object contains the current re-sliced image.

Note:
The returned object pointer should be deleted properly by yourself.

8.234.3.18 mitkVector const* mitkReslicePlaneWidgetModel::GetRightBottomPoint () [inline]

Get the right-bottom point of the re-slice rectangle plane in the model space.

Returns:
Return a point to a const mitkVector object which contains the coordinates of the point.

8.234.3.19 void mitkReslicePlaneWidgetModel::GetRightBottomPoint (coord_type & *x*, coord_type & *y*, coord_type & *z*) [inline]

Get the right-bottom point of the re-slice rectangle plane in the model space.

Parameters:
- ***x*** the x-coordinate of the point
- ***y*** the y-coordinate of the point
- ***z*** the z-coordinate of the point

8.234.3.20 mitkVector const* mitkReslicePlaneWidgetModel::GetRightTopPoint () [inline]

Get the right-top point of the re-slice rectangle plane in the model space.

Returns:
Return a point to a const mitkVector object which contains the coordinates of the point.

8.234.3.21 void mitkReslicePlaneWidgetModel::GetRightTopPoint (coord_type & *x*, coord_type & *y*, coord_type & *z*) [inline]

Get the right-top point of the re-slice rectangle plane in the model space.

Parameters:
- ***x*** the x-coordinate of the point
- ***y*** the y-coordinate of the point
- ***z*** the z-coordinate of the point

8.234.3.22 float mitkReslicePlaneWidgetModel::GetRotatedAngleDeg ()

Get the rotated angle of the plane in degree (around x/y axis) since mouse button is pressed.

Returns:
Return the value of rotated angel in degree.

8.234.3.23 float mitkReslicePlaneWidgetModel::GetRotatedAngleRad ()

Get the rotated angle of the plane in radian (around x/y axis) since mouse button is pressed.

Returns:
Return the value of rotated angel in radian.

8.234.3.24 int mitkReslicePlaneWidgetModel::GetSliceImageHeight () [inline]

Get the image height of the reconstructed slice.

Returns:
Return the image height (measured by pixels) of the reconstructed slice.

8.234.3.25 int mitkReslicePlaneWidgetModel::GetSliceImageWidth () [inline]

Get the image width of the reconstructed slice.

Returns:
Return the image width (measured by pixels) of the reconstructed slice.

8.234.3.26 float mitkReslicePlaneWidgetModel::GetSliceOpacity () [inline]

Get the opacity of the reconstructed slice.

Returns:
Return the opacity of the plane (the value is between 0.0f and 1.0f, and the default value is 1.0f)

8.234.3.27 void mitkReslicePlaneWidgetModel::GetStartOrigin (float *so*[3]) [inline]

Get the start origin (start plane center) of the plane.

Parameters:
so[0] return the x-coordinate the start origin
so[1] return the y-coordinate the start origin
so[2] return the z-coordinate the start origin

8.234.3.28 float const* mitkReslicePlaneWidgetModel::GetStartOrigin () [inline]

Get the start origin (start plane center) of the plane.

Returns:
Return the start origin (start plane center) of the plane.

8.234.3.29 const string& mitkReslicePlaneWidgetModel::GetUnitName () [inline]

Get name string of unit.

Returns:
Return a constant reference to a string contains the name

8.234.3.30 float mitkReslicePlaneWidgetModel::GetVertMoveFromStart ()

Get vertical movement from the start position (plane center).

Returns:
Return the vertical movement from the start position.

8.234.3.31 mitkVolume* mitkReslicePlaneWidgetModel::GetVolumeData () [inline]

Get the source volume data for reconstructing slice.

Returns:
Return the pointer to the source volume object.

8.234.3.32 bool mitkReslicePlaneWidgetModel::IsBallSelected ()

Whether any one of the balls is selected.

Returns:
Return true if one of the four balls is selected.

8.234.3.33 bool mitkReslicePlaneWidgetModel::IsLineSelected ()

Whether any one of the lines is selected.

Returns:
Return true if one of the four lines is selected.

8.234.3.34 bool mitkReslicePlaneWidgetModel::IsPseudocolorEnabled () [inline]

Test if the pseudo-color function is enabled.

8.234.3.35 bool mitkReslicePlaneWidgetModel::IsResliceEnabled () [inline]

Test if the re-slice function is enabled.

8.234.3.36 virtual void mitkReslicePlaneWidgetModel::Pick (const WidgetNames & *names*) [virtual]

Maintain the selection status when this widget is picked.

Parameters:
names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implements mitkWidgetModel.

8.234.3.37 virtual void mitkReslicePlaneWidgetModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel3D.

8.234.3.38 virtual void mitkReslicePlaneWidgetModel::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

8.234.3.39 virtual int mitkReslicePlaneWidgetModel::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Reimplemented from mitkModel.

8.234.3.40 void mitkReslicePlaneWidgetModel::RotateDegAroundXAxisOfPlane (float *angle*)

Rotate the plane around the x axis of the plane. This function is provided for convenience. It behaves essentially like RotateRadAroundXAxisOfPlane(), but the parameter angle is in degree.

Parameters:
angle rotation angle in degree

8.234.3.41 void mitkReslicePlaneWidgetModel::RotateDegAroundYAxisOfPlane (float *angle*)

Rotate the plane around the y axis of the plane. This function is provided for convenience. It behaves essentially like RotateRadAroundYAxisOfPlane(), but the parameter angle is in degree.

Parameters:
angle rotation angle in degree

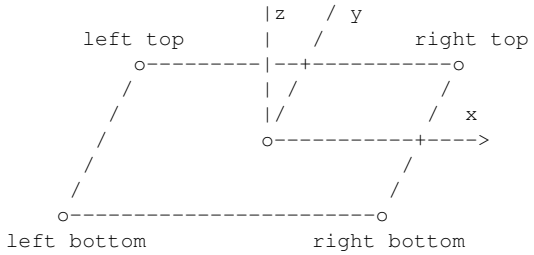
8.234.3.42 void mitkReslicePlaneWidgetModel::RotateDegAroundZAxisOfPlane (float *angle*)

Rotate the plane around the z axis of the plane. This function is provided for convenience. It behaves essentially like RotateRadAroundZAxisOfPlane(), but the parameter angle is in degree.

Parameters:
angle rotation angle in degree

8.234.3.43 void mitkReslicePlaneWidgetModel::RotateRadAroundXAxisOfPlane (float *angle*)

Rotate the plane around the x axis of the plane.

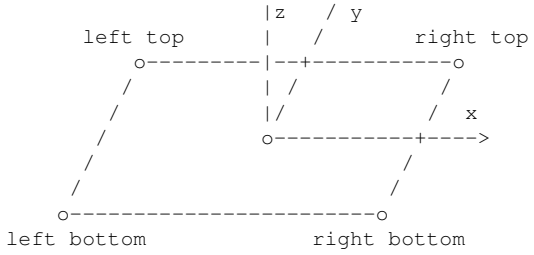


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

8.234.3.44 void mitkReslicePlaneWidgetModel::RotateRadAroundYAxisOfPlane (float *angle*)

Rotate the plane around the y axis of the plane.

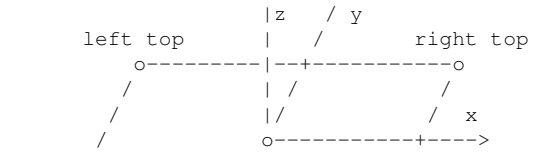


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

8.234.3.45 void mitkReslicePlaneWidgetModel::RotateRadAroundZAxisOfPlane (float *angle*)

Rotate the plane around the z axis of the plane.





Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

8.234.3.46 void mitkReslicePlaneWidgetModel::SetColorTable (mitkColorTable * *ct*) [inline]

Set color table.

Parameters:
ct the pointer to a mitkColorTable object

8.234.3.47 void mitkReslicePlaneWidgetModel::SetPlaneCenter (float *ox*, float *oy*, float *oz*)

Set the center of the plane.

Parameters:
ox the x coordinate of the center
oy the y coordinate of the center
oz the z coordinate of the center

8.234.3.48 void mitkReslicePlaneWidgetModel::SetPlaneOpacity (float *opacity*)

Set the opacity of the plane.

Parameters:
opacity the opacity of the plane (the value is between 0.0f and 1.0f, and the default value is 0.0f)

8.234.3.49 void mitkReslicePlaneWidgetModel::SetPlanePosition (coord_type *v0x*, coord_type *v0y*, coord_type *v0z*, coord_type *v1x*, coord_type *v1y*, coord_type *v1z*, coord_type *cx*, coord_type *cy*, coord_type *cz*)

Set the position of this plane. The parameters specify a rectangle in the model space via its left-bottom point, right-bottom point and center point.

Parameters:
v0x the x-coordinate of the left-bottom point
v0y the y-coordinate of the left-bottom point
v0z the z-coordinate of the left-bottom point
v1x the x-coordinate of the right-bottom point

v1y the y-coordinate of the right-bottom point

v1z the z-coordinate of the right-bottom point

cx the x-coordinate of the center point

cy the y-coordinate of the center point

cz the z-coordinate of the center point

8.234.3.50 void mitkReslicePlaneWidgetModel::SetSliceImageHeight (int *height*)

Set the image height of the reconstructed slice.

Parameters:

height the image height (measured by pixels) of the reconstructed slice

8.234.3.51 void mitkReslicePlaneWidgetModel::SetSliceImageWidth (int *width*)

Set the image width of the reconstructed slice.

Parameters:

width the image width (measured by pixels) of the reconstructed slice

8.234.3.52 void mitkReslicePlaneWidgetModel::SetSliceOpacity (float *opacity*)

Set the opacity of the reconstructed slice.

Parameters:

opacity the opacity of the plane (the value is between 0.0f and 1.0f, and the default value is 1.0f)

8.234.3.53 virtual void mitkReslicePlaneWidgetModel::SetSourceModel (mitkDataModel * *model*) [virtual]

Associate this widget with a data model.

Parameters:

model pointer to an mitkDataModel with which this widget is associated

Note:

The parameter model can be NULL. It indicates that manipulation on this widget does not have an effect on other data models.

Reimplemented from mitkWidgetModel3D.

8.234.3.54 void mitkReslicePlaneWidgetModel::SetUnitName (const string & *name*) [inline]

Set name string of unit.

Parameters:

name a constant reference to a string contains the name of the length unit (e.g. mm) this line uses

8.234.3.55 void mitkReslicePlaneWidgetModel::SetVolumeData (mitkVolume * *vol*)

Set the source volume data for reconstructing slice.

Parameters:

vol the pointer to the source volume object

8.234.3.56 void mitkReslicePlaneWidgetModel::StoreEnableState () [inline]

Store enabled states of the widget's re-slice and pseudo-color function, these states will be restored after rendering of the widget model.

8.234.3.57 void mitkReslicePlaneWidgetModel::TranslatePlane (float *tx*, float *ty*, float *tz*)

Translate the plane with translation vector (tx, ty, tz).

Parameters:

tx the x coordinate of the translation vector

ty the y coordinate of the translation vector

tz the z coordinate of the translation vector

8.234.3.58 virtual void mitkReslicePlaneWidgetModel::Update () [virtual]

Update the parameters of the widget.

Reimplemented from mitkWidgetModel3D.

The documentation for this class was generated from the following file:

- mitkReslicePlaneWidgetModel.h

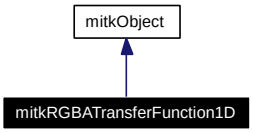
8.235 mitkRGBATransferFunction1D Class Reference

mitkRGBATransferFunction1D - 1D-Transfer-Function with RGBA control points.

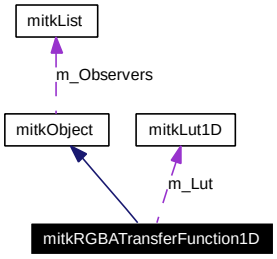
`#include <mitkRGBATransferFunction1D.h>`

Inherits mitkObject.

Inheritance diagram for mitkRGBATransferFunction1D:



Collaboration diagram for mitkRGBATransferFunction1D:



Public Member Functions

- mitkVis2Vector4 * GetData ()
- void SetSampleNumber (int Number)
- int GetSampleNumber () const
- void SetMinPos (float Min)
- float GetMinPos () const
- void SetMaxPos (float Max)
- float GetMaxPos () const
- const mitkLut1D & GetLut1D ()
- int AddPoint (float pos, mitkVis2Vector4 value)
- int GetPointCount () const
- float GetPos (int index) const
- mitkVis2Vector4 GetValue (int index) const
- void SetRGBAValue (int index, mitkVis2Vector4 value)
- void RemovePoint (int index)
- void RemoveAllPoints ()

8.235.1 Detailed Description

mitkRGBATransferFunction1D - 1D-Transfer-Function with RGBA control points.

mitkRGBATransferFunction1D defines 1D-Transfer-Functions with RGBA control points. The generated transfer function contains SampleNumber sample points mapping the scalar values ranging from MinPos to MaxPos.

8.235.2 Member Function Documentation

8.235.2.1 int mitkRGBATransferFunction1D::AddPoint (float *pos*, mitkVis2Vector4 *value*)

Add a control point

Parameters:

pos The scalar value position of the control point

value The color value of the control point

Returns:

Return the index of the added control point

8.235.2.2 mitkVis2Vector4* mitkRGBATransferFunction1D::GetData ()

Get the array of sample points

Returns:

Return the array of sample points

8.235.2.3 const mitkLut1D& mitkRGBATransferFunction1D::GetLut1D ()

Get the 1D Lut Data Object

Returns:

Return the 1D Lut Data Object

8.235.2.4 float mitkRGBATransferFunction1D::GetMaxPos () const

Get the maximum position of value mapping

Returns:

Return the maximum position of scalar value mapping

8.235.2.5 float mitkRGBATransferFunction1D::GetMinPos () const

Get the minimum position of value mapping

Returns:

Return the minimum position of scalar value mapping

8.235.2.6 int mitkRGBATransferFunction1D::GetPointCount () const

Get the number of the control points

Returns:

Return the number of the control points

8.235.2.7 float mitkRGBATransferFunction1D::GetPos (int *index*) const

Get the scalar value position of a certain control point

Parameters:
index The index of the control point

Returns:
Return the scalar value position of the given control point

8.235.2.8 int mitkRGBATransferFunction1D::GetSampleNumber () const

Get the number of sample points

Returns:
Return the number of sample points

8.235.2.9 mitkVis2Vector4 mitkRGBATransferFunction1D::GetValue (int *index*) const

Get the color value of a certain control point

Parameters:
index The index of the control point

Returns:
Return the color value of the given control point

8.235.2.10 void mitkRGBATransferFunction1D::RemoveAllPoints ()

Remove all control points

8.235.2.11 void mitkRGBATransferFunction1D::RemovePoint (int *index*)

Remove a certain control point

Parameters:
index The index of the control point

8.235.2.12 void mitkRGBATransferFunction1D::SetMaxPos (float *Max*)

Set the maximum position of value mapping

Parameters:
Max The maximum position of scalar value mapping

8.235.2.13 void mitkRGBATransferFunction1D::SetMinPos (float *Min*)

Set the minimum position of value mapping

Parameters:

Min The minimum position of scalar value mapping

8.235.2.14 void mitkRGBATransferFunction1D::SetRGBAValue (int *index*, mitkVis2Vector4 *value*)

Set the color value of a certain control point

Parameters:

index The index of the control point

value The color value of the control point

8.235.2.15 void mitkRGBATransferFunction1D::SetSampleNumber (int *Number*)

Set the number of sample points

Parameters:

Number The number of sample points

The documentation for this class was generated from the following file:

- mitkRGBATransferFunction1D.h

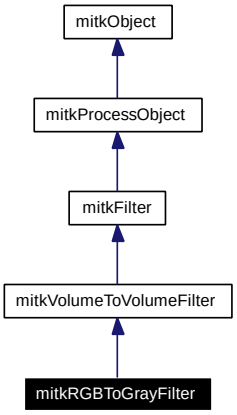
8.236 mitkRGBToGrayFilter Class Reference

mitkRGBToGrayFilter - a filter to transfer RGB volume to gray volume

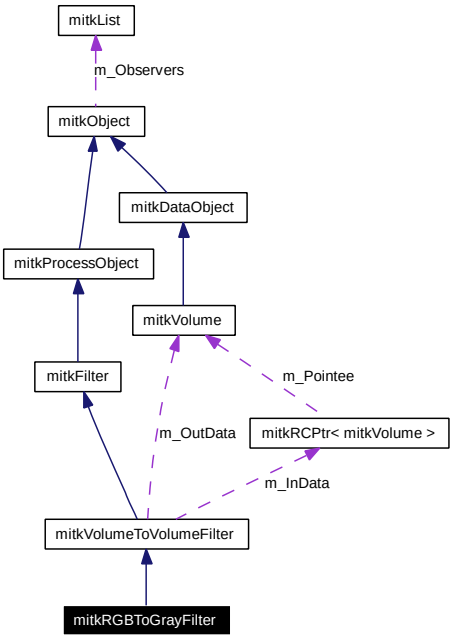
`#include <mitkRGBToGrayFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkRGBToGrayFilter:



Collaboration diagram for mitkRGBToGrayFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetOutputDataType (int dataType)
- void SetCoefficients (float coefRed, float coefGreen, float coefBlue)

8.236.1 Detailed Description

mitkRGBToGrayFilter - a filter to transfer RGB volume to gray volume

mitkRGBToGrayFilter is a filter to transfer RGB volume (with 3 channels) to gray volume (with 1 channel). The equation of the transform is:

$$\mathbf{GRAY} = c_{\mathbf{RED}} * \mathbf{RED} + c_{\mathbf{GREEN}} * \mathbf{GREEN} + c_{\mathbf{BLUE}} * \mathbf{BLUE}$$ where $c_{\mathbf{RED}} + c_{\mathbf{GREEN}} + c_{\mathbf{BLUE}} = 1.0$.

You can set $c_{\mathbf{RED}}$, $c_{\mathbf{GREEN}}$ and $c_{\mathbf{BLUE}}$ by SetCoefficients().

8.236.2 Member Function Documentation

8.236.2.1 virtual void mitkRGBToGrayFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.236.2.2 void mitkRGBToGrayFilter::SetCoefficients (float *coefRed*, float *coefGreen*, float *coefBlue*)

Set the transform coefficients. The coefficients will be normalized first so that $c_{\mathbf{RED}}+c_{\mathbf{GREEN}}+c_{\mathbf{BLUE}} = 1.0$. The final transform will be:

$$\mathbf{GRAY} = c_{\mathbf{RED}} * \mathbf{RED} + c_{\mathbf{GREEN}} * \mathbf{GREEN} + c_{\mathbf{BLUE}} * \mathbf{BLUE} .$$

Parameters:

coefRed red coefficient

coefGreen green coefficient

coefBlue blue coefficient

8.236.2.3 void mitkRGBToGrayFilter::SetOutputDataType (int *dataType*) [inline]

Set data type of the output volume. MITK supports various data type.

Parameters:

data_type Its valid value and meaning is shown as follows:

MITK_CHAR The data type is char

MITK_UNSIGNED_CHAR The data type is unsigned char

MITK_SHORT The data type is short

MITK_UNSIGNED_SHORT The data type is unsigned short

MITK_INT The data type is int

MITK_UNSIGNED_INT The data type is unsigned int

MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

The documentation for this class was generated from the following file:

- mitkRGBToGrayFilter.h

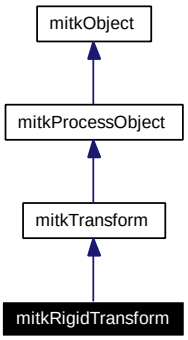
8.237 mitkRigidTransform Class Reference

mitkRigidTransform - a concrete transform to perform rigid transform

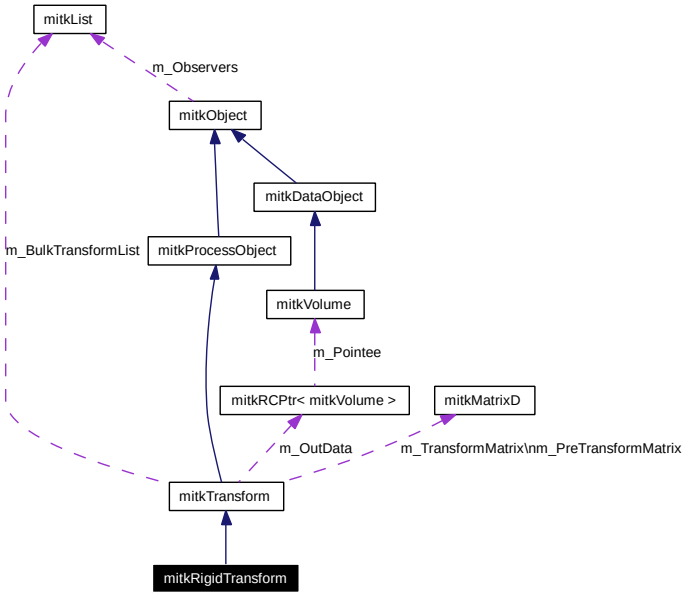
`#include <mitkRigidTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkRigidTransform:



Collaboration diagram for mitkRigidTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkRigidTransform ()
- mitkRigidTransform (unsigned int dim)
- const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- void GetAngles (ScalarParameterType &angle)

- void GetAngles (ScalarParameterType &alpha, ScalarParameterType &beta, ScalarParameterType &gamma)
- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty)
- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty, ScalarParameterType &tz)
- virtual void ConvertTransformMaxtrixToVector ()

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.237.1 Detailed Description

mitkRigidTransform - a concrete transform to perform rigid transform

This transform applies a rotation and translation to the 2D and 3D image space. User should use the constructor to specify the image dimensions, e.g. mitkRigidTransform(2) (default) for 2d image, mitkRigidTransform(3) for 3d image.

Transform Mode Number of parameters p[0] p[1] p[2] p[3] p[4] p[5] p[6] p[7] p[8]

MITK_TRANSFORM_RIGID (2D) 3 angle tx ty - - - - - -

MITK_TRANSFORM_RIGID_CENTERED (2D) 5 angle tx ty cx cy - - - -

MITK_TRANSFORM_RIGID (3D) 6 angle_x angle_y angle_z tx ty tz - - -

MITK_TRANSFORM_RIGID_CENTERED (3D) 9 angle_x angle_y angle_z tx ty tz cx cy cz

8.237.2 Constructor & Destructor Documentation

8.237.2.1 mitkRigidTransform::mitkRigidTransform ()

Default Constructor.

8.237.2.2 mitkRigidTransform::mitkRigidTransform (unsigned int *dim*)

Constructor with specific image dimensions setting.

8.237.3 Member Function Documentation

8.237.3.1 virtual void mitkRigidTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.237.3.2 virtual void mitkRigidTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.237.3.3 void mitkRigidTransform::GetAngles (ScalarParameterType & *alpha*, ScalarParameterType & *beta*, ScalarParameterType & *gamma*)

Get the rotation parameters from the transform parameter vector. (for 3d rotation)

Parameters:

alpha The returned angle value. (rotate around x-axis)

beta The returned angle value. (rotate around y-axis)

gamma The returned angle value. (rotate around z-axis)

8.237.3.4 void mitkRigidTransform::GetAngles (ScalarParameterType & *angle*)

Get the rotation parameters from the transform parameter vector. (for 2d rotation)

Parameters:

angle The returned angle value.

8.237.3.5 const MatrixType& mitkRigidTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:

inPoint The index of the point in image.

Returns:

Return the pointer to the Jacobian matrix.

Reimplemented from mitkTransform.

8.237.3.6 void mitkRigidTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*, ScalarParameterType & *tz*)

Get the translation parameters from the transform parameter vector. (for 3d translation)

Parameters:

tx The returned x-translation value.

ty The returned y-translation value.

tz The returned z-translation value.

8.237.3.7 void mitkRigidTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*)

Get the translation parameters from the transform parameter vector. (for 2d translation)

Parameters:

tx The returned x-translation value.

ty The returned y-translation value.

8.237.3.8 virtual void mitkRigidTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkRigidTransform.h

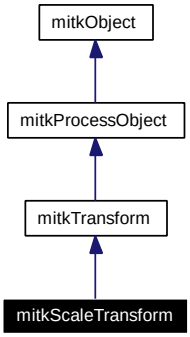
8.238 mitkScaleTransform Class Reference

mitkScaleTransform - a concrete transform to perform anisotropic scaling transform

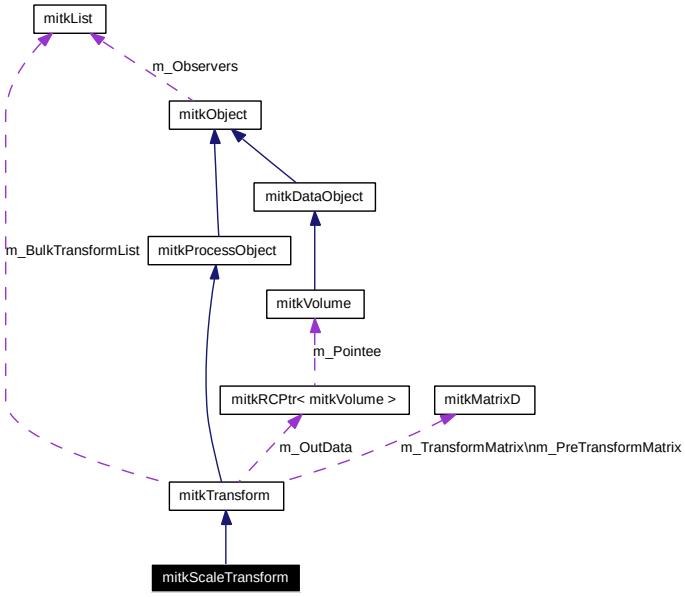
`#include <mitkScaleTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkScaleTransform:



Collaboration diagram for mitkScaleTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkScaleTransform ()
- mitkScaleTransform (unsigned int dim)
- virtual void SetTransformMode (int transformMode, unsigned int dim)
- const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- virtual void ConvertTransformMaxtrixToVector ()

- void GetScales (ScalarParameterType &sx, ScalarParameterType &sy)
- void GetScales (ScalarParameterType &sx, ScalarParameterType &sy, ScalarParameterType &sz)

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.238.1 Detailed Description

mitkScaleTransform - a concrete transform to perform anisotropic scaling transform

mitkScaleTransform is a concrete transform to perform anisotropic scaling transform to 2D and 3D image space. User should use the constructor to specify the image dimensions, e.g. mitkScaleTransform(2) (default) for 2d image, mitkScaleTransform(3) for 3d image. To improve the behavior of the scaling parameters when they are modified by optimizers, a logarithm based scaling transform was implemented.

Transform Mode Number of parameters p[0] p[1] p[2]

MITK_TRANSFORM_SCALE (2D) 2 sx sy -

MITK_TRANSFORM_SCALE (3D) 3 sx sy sz

MITK_TRANSFORM_SCALE_LOGARITHMIC (2D) 2 sx sy -

MITK_TRANSFORM_SCALE_LOGARITHMIC (3D) 3 sx sy sz

8.238.2 Constructor & Destructor Documentation

8.238.2.1 mitkScaleTransform::mitkScaleTransform ()

Constructor.

8.238.2.2 mitkScaleTransform::mitkScaleTransform (unsigned int *dim*)

Constructor with specific transform mode setting.

8.238.3 Member Function Documentation

8.238.3.1 virtual void mitkScaleTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.238.3.2 virtual void mitkScaleTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.238.3.3 const MatrixType& mitkScaleTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:
inPoint The index of the point in image.

Returns:
Return the pointer to the Jacobian matrix.

Reimplemented from mitkTransform.

8.238.3.4 void mitkScaleTransform::GetScales (ScalarParameterType & *sx*, ScalarParameterType & *sy*, ScalarParameterType & *sz*)

Get the scaling parameters from the transform parameter vector. (for 3d scaling)

Parameters:
sx The returned scale value along x-axis.
sy The returned scale value along y-axis.
sz The returned scale value along z-axis.

8.238.3.5 void mitkScaleTransform::GetScales (ScalarParameterType & *sx*, ScalarParameterType & *sy*)

Get the scaling parameters from the transform parameter vector. (for 2d scaling)

Parameters:
sx The returned scale value along x-axis.
sy The returned scale value along y-axis.

8.238.3.6 virtual void mitkScaleTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTransform.

8.238.3.7 virtual void mitkScaleTransform::SetTransformMode (int *transformMode*, unsigned int *dim*) [virtual]

Set the transform mode.Default is MITK_TRANSFORM_SCALE_LOGARITHMIC. param mode Specify the transform mode.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkScaleTransform.h

8.239 mitkScene Class Reference

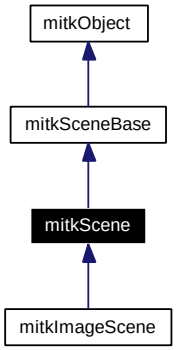
mitkScene - a scene to display 3D surface rendered or volume rendered image

`#include <mitkScene.h>`

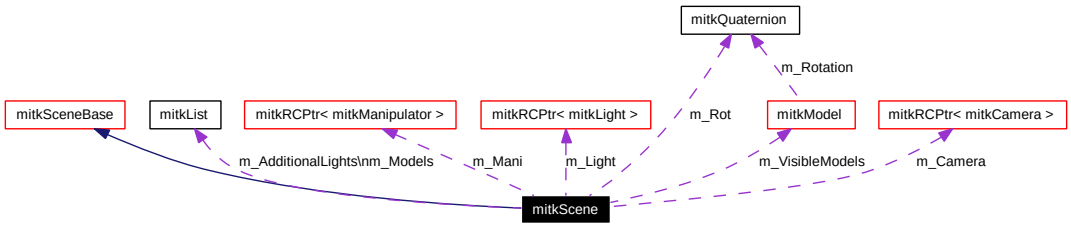
Inherits mitkSceneBase.

Inherited by mitkImageScene.

Inheritance diagram for mitkScene:



Collaboration diagram for mitkScene:



Public Member Functions

- const int * GetViewPort ()
- void GetViewPort (int vp[4])
- virtual void InitGL ()
- virtual void DestroyGL ()
- virtual void ResizeGL (int newX, int newY)
- virtual void RenderGL ()
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseWheel (bool ctrlDown, bool shiftDown, int xPos, int yPos, int delta)
- void SetBackColor (unsigned char rColor, unsigned char gColor, unsigned char bColor)
- void SetBackColor (float rColor, float gColor, float bColor)
- void SetBackColor (int rColor, int gColor, int bColor)
- void GetBackColor (float &rColor, float &gColor, float &bColor)
- void GetBackColor (unsigned char &rColor, unsigned char &gColor, unsigned char &bColor)

- void GetBackColor (int &rColor, int &gColor, int &bColor)
- virtual void ResetScene ()
- virtual void Translate (float deltX, float deltY, float deltZ)
- virtual void Rotate (float deltX, float deltY, float deltZ)
- virtual void Rotate (float ax, float ay, float az, float angle)
- virtual void Rotate (const mitkQuaternion &q)
- virtual void SetRotation (float x, float y, float z)
- virtual void SetRotation (float ax, float ay, float az, float angle)
- virtual void SetRotation (const mitkQuaternion &q)
- virtual void Scale (float delt)
- virtual mitkManipulator * CreateManipulator ()
- void SetManipulator (mitkManipulator *mani)
- void AddModel (mitkModel *model)
- void RemoveModel (mitkModel *model)
- void RemoveModel (int i)
- void RemoveAllModels ()
- void RemoveAllModel ()
- mitkModel * GetModel (int i)
- mitkList * GetModels ()
- int GetModelCount ()
- int GetNumberOfPropsRendered ()
- void SetCamera (mitkCamera *cam)
- mitkCamera * GetCamera ()
- void SetDefaultLight (mitkLight *lit)
- mitkLight * GetDefaultLight ()
- mitkLight * GetLight ()
- void AddAdditionalLight (mitkLight *lit)
- void RemoveAdditionalLight (int idx)
- void RemoveAdditionalLight (mitkLight *lit)
- void RemoveAllAdditionalLights ()
- int GetAdditionalLightCount ()
- mitkLight * GetAdditionalLight (int idx)
- void AlignScene ()
- float * GetZbufferData (int x1, int y1, int x2, int y2)
- void GetZbufferData (int x1, int y1, int x2, int y2, float *buffer)
- unsigned char * GetPixelData (int xOffset, int yOffset, int width, int height)
- void GetPixelData (int xOffset, int yOffset, int width, int height, unsigned char *buffer, int packAlignment=4, bool swapRGB=false)
- void GetPixelData24 (int xOffset, int yOffset, int width, int height, unsigned char *buffer, int packAlignment=4, bool swapRGB=false)
- void GetPixelData32 (int xOffset, int yOffset, int width, int height, unsigned char *buffer, bool swapRGB=false)
- void SetSelected (bool isSelected)
- bool GetSelected ()
- float GetTranslateSpeed ()
- float GetTranslationX ()
- float GetTranslationY ()
- float GetTranslationZ ()
- void GetTranslation (float trans[3])
- const mitkQuaternion & GetRotation ()

- float GetScale ()
- void EnableLoD ()
- void DisableLoD ()
- bool IsLoDEnabled ()
- bool IsModelsModified ()
- bool IsCameraModified ()
- bool IsAdditionalLightModified ()

8.239.1 Detailed Description

mitkScene - a scene to display 3D surface rendered or volume rendered image

mitkScene is a 3d scene to display 3D surface rendered or volume rendered image. To display 3d images, you must create one or several models (either mitkSurfaceModel or mitkVolumeModel) firstly, then add them using the function

```
AddModel()
```

. mitkScene needs a container subclasses from mitkSceneContainer. You can use mitkGLWindow for the container, or implement your own container. This class is added from Mitk 2.3, in order to allow the users have their own OpenGL context encapsulation. (For example, the using the QGLWidget class provided by QT).

8.239.2 Member Function Documentation

8.239.2.1 void mitkScene::AddAdditionalLight (mitkLight * *lit*)

Add an additional light to this scene.

Parameters:
lit the light to add

Note:
At most 5 additional light can be added to the scene.

8.239.2.2 void mitkScene::AddModel (mitkModel * *model*)

Add a model to this scene for display. Scene can contain many models, including surface model, volume model and widget model, and display them simultaneity. So scene provides a set of functions to manage the models.

See also:
RemoveModel()
RemoveAllModel()
GetModel()
GetModels()
GetModelCount()

Parameters:
model Specify the model that will be added to this scene.

8.239.2.3 void mitkScene::AlignScene ()

Warning:
Internal function. Don't call it directly.

8.239.2.4 virtual mitkManipulator* mitkScene::CreateManipulator () [virtual]

Create a default manipulator for the mouse events processing. Subclass can override this function to create a proper manipulator.

Reimplemented in mitkImageScene.

8.239.2.5 virtual void mitkScene::DestroyGL () [virtual]

Free OpenGL resources

Reimplemented from mitkSceneBase.

8.239.2.6 void mitkScene::DisableLoD ()

Disable LoD rendering mode.

Note:
It is only a hint to rendering method and depends on actual implementation. Maybe it will not take any effect.

8.239.2.7 void mitkScene::EnableLoD ()

Enable LoD rendering mode.

Note:
It is only a hint to rendering method and depends on actual implementation. Maybe it will not take any effect.

8.239.2.8 mitkLight* mitkScene::GetAdditionalLight (int *idx*)

Get the idx'th additional light.

Parameters:
idx the index of the light to get

8.239.2.9 int mitkScene::GetAdditionalLightCount ()

Get the count of additional lights.

Returns:
Return the count of additional lights.

8.239.2.10 void mitkScene::GetBackColor (int & *rColor*, int & *gColor*, int & *bColor*)

Get the background color of this scene.

Parameters:

rColor Return the red component of the background color. The value range is from 0 to 255.

gColor Return the green component of the background color. The value range is from 0 to 255.

bColor Return the blue component of the background color. The value range is from 0 to 255.

8.239.2.11 void mitkScene::GetBackColor (unsigned char & *rColor*, unsigned char & *gColor*, unsigned char & *bColor*)

Get the background color of this scene.

Parameters:

rColor Return the red component of the background color. The value range is from 0 to 255.

gColor Return the green component of the background color. The value range is from 0 to 255.

bColor Return the blue component of the background color. The value range is from 0 to 255.

8.239.2.12 void mitkScene::GetBackColor (float & *rColor*, float & *gColor*, float & *bColor*)

Get the background color of this scene.

Parameters:

rColor Return the red component of the background color. The value range is from 0.0 to 1.0.

gColor Return the green component of the background color. The value range is from 0.0 to 1.0.

bColor Return the blue component of the background color. The value range is from 0.0 to 1.0.

8.239.2.13 mitkCamera* mitkScene::GetCamera ()

Get the camera of this scene.

Returns:

Return the camera of this scene.

8.239.2.14 mitkLight* mitkScene::GetDefaultLight ()

Get the default light of this scene.

Returns:

Return the light of this scene.

Note:

The position of the default light is calculated by the scene automatically, so any changes of the position will be discarded.

8.239.2.15 mitkLight* mitkScene::GetLight () [inline]

Provided for back compatibility, just the same as SetSliceNum().

8.239.2.16 mitkModel* mitkScene::GetModel (int *i*)

Get the model in specified position.

Parameters:
i Zero based index to specify the position of the model.

Returns:
Return the ith model in this scene.

8.239.2.17 int mitkScene::GetModelCount ()

Get the model count in this scene.

Returns:
Return the model count in this scene.

8.239.2.18 mitkList* mitkScene::GetModels ()

Get the list of models in this scene.

Returns:
Return the internal list of models. User can access each model through the interface of mitkList.

8.239.2.19 int mitkScene::GetNumberOfPropsRendered () [inline]

Warning:
Internal function. Don't call it directly.

8.239.2.20 void mitkScene::GetPixelData (int *xOffset*, int *yOffset*, int *width*, int *height*, unsigned char * *buffer*, int *packAlignment* = 4, bool *swapRGB* = false) [inline]

Get frame buffer data (RGB pixel data) in the specified region. This function is provided for convenience. It does the same thing as GetPixelData24().

Parameters:
xOffset specify the x coordinate of the left bottom corner

yOffset specify the y coordinate of the left bottom corner

width specify the width of the region

height specify the height of the region

buffer return the the frame buffer data. The data is RGB color data, and is arranged in the order of R1G1B1R2G2B2...

packAlignment pack alignment bytes

swapRGB whether swap RGB channel or not (i.e. true: use GL_BGR format to get pixel data, false: use GL_RGB format to get pixel data

Warning:
The parameter "buffer" must point to a memory block which can contain width∗height RGB pixel samples. If packAlignment is 4, the total bytes of one line (contain width samples) must be devided exactly by 4, i.e. one line should be ((width∗3+3) & (∼3)) bytes. Otherwise, one line should be exactly width∗3 bytes. Furthermore, this memory block must be allocated by the user before call this function.

8.239.2.21 unsigned char∗ mitkScene::GetPixelData (int *xOffset*, int *yOffset*, int *width*, int *height*)

Get frame buffer data in the specified region.

Parameters:

xOffset Specify the x coordinate of the left bottom corner

yOffset Specify the y coordinate of the left bottom corner

width Specify the width of the region

height Specify the height of the region

Returns:
Return the address of the frame buffer data. The data is RGB color data, and is arranged in the order of R1G1B1R2G2B2...

Note:
This function is obsolete, please use another function to get frame buffer data.

8.239.2.22 void mitkScene::GetPixelData24 (int *xOffset*, int *yOffset*, int *width*, int *height*, unsigned char ∗ *buffer*, int *packAlignment* = `4`, bool *swapRGB* = `false`)

Get frame buffer data (RGB pixel data) in the specified region.

Parameters:

xOffset specify the x coordinate of the left bottom corner

yOffset specify the y coordinate of the left bottom corner

width specify the width of the region

height specify the height of the region

buffer return the the frame buffer data. The data is RGB color data, and is arranged in the order of R1G1B1R2G2B2...

packAlignment pack alignment bytes

swapRGB whether swap RGB channel or not (i.e. true: use GL_BGR format to get pixel data, false: use GL_RGB format to get pixel data

Warning:
The parameter "buffer" must point to a memory block which can contain width∗height RGB pixel samples. If packAlignment is 4, the total bytes of one line (contain width samples) must be devided exactly by 4, i.e. one line should be ((width∗3+3) & (∼3)) bytes. Otherwise, one line should be exactly width∗3 bytes. Furthermore, this memory block must be allocated by the user before call this function.

8.239.2.23 void mitkScene::GetPixelData32 (int *xOffset*, int *yOffset*, int *width*, int *height*, unsigned char * *buffer*, bool *swapRGB* = false)

Get frame buffer data (RGBA pixel data) in the specified region.

Parameters:

xOffset specify the x coordinate of the left bottom corner

yOffset specify the y coordinate of the left bottom corner

width specify the width of the region

height specify the height of the region

buffer return the the frame buffer data. The data is RGBA color data, and is arranged in the order of R1G1B1A1R2G2B2A2...

swapRGB whether swap RGB channel or not (i.e. true: use GL_BGR format to get pixel data, false: use GL_RGB format to get pixel data

Warning:

The parameter "buffer" must point to a memory block which can contain width∗height RGBA pixel samples. This memory block must be allocated by the user before call this function.

8.239.2.24 const mitkQuaternion& mitkScene::GetRotation (void) [inline]

Get the Rotation of the scene.

Returns:

Return the quaternion of the rotation.

8.239.2.25 float mitkScene::GetScale (void) [inline]

Get the scale of the scene.

Returns:

Return the scale of the scene.

8.239.2.26 bool mitkScene::GetSelected () [inline]

Set the selected status of this scene. If scene is selected, it will be added a red frame.

Parameters:

isSelected isSelected=true, set the scene is selected. isSelected=false, set the scene is unselected.

8.239.2.27 float mitkScene::GetTranslateSpeed () [inline]

Warning:

Internal function. Don't call it directly.

8.239.2.28 void mitkScene::GetTranslation (float *trans*[3]) [inline]

Get the translation of the scene.

Parameters:

trans[0] return translation in x-axis

trans[1] return translation in y-axis

trans[2] return translation in z-axis

Warning:

This function is provided for convenience. Don't call it directly.

8.239.2.29 float mitkScene::GetTranslationX () [inline]

Get the translation along x-axis of the scene.

Returns:

Return the translation along x-axis of the scene.

Warning:

This function is provided for convenience. Don't call it directly.

8.239.2.30 float mitkScene::GetTranslationY () [inline]

Get the translation along y-axis of the scene.

Returns:

Return the translation along y-axis of the scene.

Warning:

This function is provided for convenience. Don't call it directly.

8.239.2.31 float mitkScene::GetTranslationZ () [inline]

Get the translation along z-axis of the scene.

Returns:

Return the translation along z-axis of the scene.

Warning:

This function is provided for convenience. Don't call it directly.

8.239.2.32 void mitkScene::GetViewPort (int *vp*[4]) [inline]

Get the viewport of this scene. Not all of the area of a scene is used to display the image. Only the viewport is used to display the image. The position and size of viewport are calculated automatically according to the size of image.

Parameters:

vp[0] return the left position of the viewport

vp[1] return the top position of the viewport

vp[2] return the width of the viewport

vp[3] return the height of the viewport

8.239.2.33 const int* mitkScene::GetViewPort () [inline]

Get the viewport of this scene. Not all of the area of a scene is used to display the image. Only the viewport is used to display the image. The position and size of viewport are calculated automatically according to the size of image.

Returns:

Return a pointer to the viewport. The first element is the Left position. The second element is the Top position. The third element is the Width of the viewport. The fourth element is the Height of the viewport.

8.239.2.34 void mitkScene::GetZbufferData (int *x1*, int *y1*, int *x2*, int *y2*, float * *buffer*)

Get z buffer data in the specified region.

Parameters:

x1 Specify the x coordinate of the left bottom corner

y1 Specify the y coordinate of the left bottom corner

x2 Specify the x coordinate of the right top corner. x2 >= x1 must be satisfied.

y2 Specify the y coordinate of the right top corner. y2 >= y1 must be satisfied.

buffer Return the z buffer data.

Warning:

The parameter "buffer" must point to a memory block which can contain (x2-x1+1)*(y2-y1+1) floats (i.e. (x2-x1+1)*(y2-y1+1)*4 bytes). This memory block must be allocated by the user before call this function.

8.239.2.35 float* mitkScene::GetZbufferData (int *x1*, int *y1*, int *x2*, int *y2*)

Get z buffer data in the specified region.

Parameters:

x1 Specify the x coordinate of the left bottom corner

y1 Specify the y coordinate of the left bottom corner

x2 Specify the x coordinate of the right top corner. x2 >= x1 must be satisfied.

y2 Specify the y coordinate of the right top corner. y2 >= y1 must be satisfied.

Returns:

Return the address of the z buffer data.

8.239.2.36 virtual void mitkScene::InitGL () [virtual]

Initialize OpenGL variables and resources.

Reimplemented from mitkSceneBase.

Reimplemented in mitkImageScene.

8.239.2.37 bool mitkScene::IsAdditionalLightModified () [inline]

Check whether there is a modified additional light

Returns:
Return true if there is a modified additional light, otherwise Return false

8.239.2.38 bool mitkScene::IsCameraModified () [inline]

Check whether the camera has been modified

Returns:
Return true if the camera has been modified, otherwise Return false

8.239.2.39 bool mitkScene::IsLoDEnabled ()

Test if the LoD mode is enabled.

Returns:
Return true if the LoD mode is enabled, otherwise return false.

8.239.2.40 bool mitkScene::IsModelsModified () [inline]

Check whether there is a modified model

Returns:
Return true if there is a modified model, otherwise Return false

8.239.2.41 virtual void mitkScene::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

Handle the mouse pressed event pass by the container

Reimplemented from mitkSceneBase.

8.239.2.42 virtual void mitkScene::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

Handle the mouse moved event pass by the container

Reimplemented from mitkSceneBase.

8.239.2.43 virtual void mitkScene::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

Handle the mouse released event pass by the container

Reimplemented from mitkSceneBase.

8.239.2.44 virtual void mitkScene::OnMouseWheel (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *delta*) [virtual]

Handle the mouse wheel event pass by the container

Reimplemented from mitkSceneBase.

8.239.2.45 void mitkScene::RemoveAdditionalLight (mitkLight * *lit*)

Remove an additional light in this scene.

Parameters:
lit specify the light that will be removed from this scene

Note:
If lit is found in the list of models, it will be removed from the list. Otherwise nothing happens.

8.239.2.46 void mitkScene::RemoveAdditionalLight (int *idx*)

Remove the additional light in specified position.

Parameters:
idx zero based index to specify the position of the additional light

8.239.2.47 void mitkScene::RemoveAllAdditionalLights ()

Remove all additional lights in this scene.

8.239.2.48 void mitkScene::RemoveAllModel () [inline]

Provided for back compatibility, just the same as SetSliceNum().

8.239.2.49 void mitkScene::RemoveAllModels ()

Remove all models in this scene.

8.239.2.50 void mitkScene::RemoveModel (int *i*)

Remove the model in specified position.

Parameters:
i Zero based index to specify the position of the model.

8.239.2.51 void mitkScene::RemoveModel (mitkModel * *model*)

Remove a model in this scene.

Parameters:

model Specify the model that will be removed from this scene. If model is found in the list of models, it will be removed from the list. Otherwise nothing happens.

8.239.2.52 virtual void mitkScene::RenderGL () [virtual]

Handle the render event pass by the container

Reimplemented from mitkSceneBase.

Reimplemented in mitkImageScene.

8.239.2.53 virtual void mitkScene::ResetScene () [virtual]

Reset the geometric position of all of the models in this scene.

Reimplemented in mitkImageScene.

8.239.2.54 virtual void mitkScene::ResizeGL (int *newX*, int *newY*) [virtual]

Handle the resize event pass by the container

Reimplemented from mitkSceneBase.

Reimplemented in mitkImageScene.

8.239.2.55 virtual void mitkScene::Rotate (const mitkQuaternion & *q*) [virtual]

Rotate all of the models in this scene.

Parameters:

q the quaternion of the rotation

Note:

The rotation is incremental.

Reimplemented in mitkImageScene.

8.239.2.56 virtual void mitkScene::Rotate (float *ax*, float *ay*, float *az*, float *angle*) [virtual]

Rotate all of the models in this scene around axis (ax,ay,az) for angle degrees.

Parameters:

ax the x-coordinate of rotation axis

ay the y-coordinate of rotation axis

az the z-coordinate of rotation axis

angle the angle of rotation in degrees

Note:
The rotation is incremental.

Reimplemented in mitkImageScene.

8.239.2.57 virtual void mitkScene::Rotate (float *deltX*, float *deltY*, float *deltZ*) [virtual]

Rotate all of the models in this scene around x, y, z axis.

Parameters:
deltX Specify the rotation around x-axis in degrees.

deltY Specify the rotation around y-axis in degrees.

deltZ Specify the rotation around z-axis in degrees.

Note:
The rotation is incremental.

Reimplemented in mitkImageScene.

8.239.2.58 virtual void mitkScene::Scale (float *delt*) [virtual]

Scale all of the models in this scene in x, y, z directions.

Parameters:
delt Specify the scale in x, y, z directions. This function scales the models equally in each direction.

Reimplemented in mitkImageScene.

8.239.2.59 void mitkScene::SetBackColor (int *rColor*, int *gColor*, int *bColor*)

Set the background color of this scene.

Parameters:
rColor Specify the red component of the background color. The valid range is from 0 to 255.

gColor Specify the green component of the background color. The valid range is from 0 to 255.

bColor Specify the blue component of the background color. The valid range is from 0 to 255.

8.239.2.60 void mitkScene::SetBackColor (float *rColor*, float *gColor*, float *bColor*)

Set the background color of this scene.

Parameters:
rColor Specify the red component of the background color. The valid range is from 0.0 to 1.0.

gColor Specify the green component of the background color. The valid range is from 0.0 to 1.0.

bColor Specify the blue component of the background color. The valid range is from 0.0 to 1.0.

8.239.2.61 void mitkScene::SetBackColor (unsigned char *rColor*, unsigned char *gColor*, unsigned char *bColor*)

Set the background color of this scene.

Parameters:

rColor Specify the red component of the background color. The valid range is from 0 to 255.

gColor Specify the green component of the background color. The valid range is from 0 to 255.

bColor Specify the blue component of the background color. The valid range is from 0 to 255.

8.239.2.62 void mitkScene::SetCamera (mitkCamera * *cam*)

Set the camera of this scene.

Parameters:

cam the new camera of this scene

8.239.2.63 void mitkScene::SetDefaultLight (mitkLight * *lit*)

Set the default light of this scene.

Parameters:

lit the new light of this scene

Note:

The position of the default light is calculated by the scene automatically. It is unchangeable.

8.239.2.64 void mitkScene::SetManipulator (mitkManipulator * *mani*)

Set another manipulator to replace the current manipulator. This function provides a chance for the user to override the behavior of mouse events processing.

Parameters:

mani Specify the new manipulator for process the mouse events

8.239.2.65 virtual void mitkScene::SetRotation (const mitkQuaternion & *q*) [virtual]

Set rotations of all models in this scene to the quaternion "q".

Parameters:

q the quaternion of the rotation

Reimplemented in mitkImageScene.

8.239.2.66 virtual void mitkScene::SetRotation (float *ax*, float *ay*, float *az*, float *angle*) [virtual]

Set rotations of all models in this scene.

Parameters:

ax the x-coordinate of rotation axis

ay the y-coordinate of rotation axis

az the z-coordinate of rotation axis

angle the angle of rotation in degrees

Reimplemented in mitkImageScene.

8.239.2.67 virtual void mitkScene::SetRotation (float *x*, float *y*, float *z*) [virtual]

Set rotations of all models in this scene.

Parameters:

x Specify the rotation around x-axis in degrees.

y Specify the rotation around y-axis in degrees.

z Specify the rotation around z-axis in degrees.

Reimplemented in mitkImageScene.

8.239.2.68 void mitkScene::SetSelected (bool *isSelected*) [inline]

Set the selected status of this scene. If scene is selected, it will be added a red frame.

Parameters:

isSelected isSelected=true, set the scene is selected. isSelected=false, set the scene is unselected.

8.239.2.69 virtual void mitkScene::Translate (float *deltX*, float *deltY*, float *deltZ*) [virtual]

Translate all of the models in this scene in x, y, z directions.

Parameters:

deltX Specify the translation in x direction

deltY Specify the translation in y direction

deltZ Specify the translation in z direction

Reimplemented in mitkImageScene.

The documentation for this class was generated from the following file:

- mitkScene.h

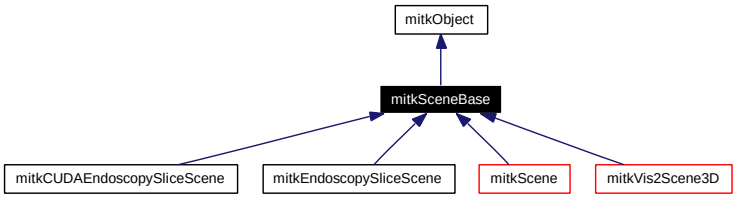
8.240 mitkSceneBase Class Reference

`#include <mitkSceneBase.h>`

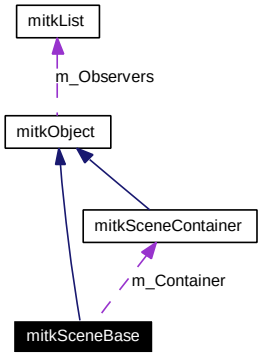
Inherits mitkObject.

Inherited by mitkCUDAEndoscopySliceScene, mitkEndoscopySliceScene, mitkScene, and mitkVis2Scene3D.

Inheritance diagram for mitkSceneBase:



Collaboration diagram for mitkSceneBase:



Public Member Functions

- void * GetWindowId ()
- const int * GetSize ()
- int GetWidth ()
- int GetHeight ()
- void Update ()
- void MakeCurrent ()
- void SwapBuffers ()
- bool HasUnprocessedMouseMessage ()
- virtual void InitGL ()
- virtual void DestroyGL ()
- virtual void ResizeGL (int newX, int newY)
- virtual void RenderGL ()
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseWheel (bool ctrlDown, bool shiftDown, int xPos, int yPos, int delta)

8.240.1 Detailed Description

mitkSceneBase - an abstract class defining the interface of a scene, which need to be referenced by the container window

mitkSceneBase is an abstract class defining the interface of a scene, which need to be referenced by the container window You can use this interface to define your own subclass of scenes. A concrete scene should reponse to the events passed by the container and do the rendering.

8.240.2 Member Function Documentation

8.240.2.1 virtual void mitkSceneBase::DestroyGL () [inline, virtual]

The Interface to free OpenGL resources

Reimplemented in mitkScene.

8.240.2.2 int mitkSceneBase::GetHeight () [inline]

Get the height of the container window. Provided for convinience.

8.240.2.3 const int* mitkSceneBase::GetSize () [inline]

Get the size of the container window. Provided for convinience.

8.240.2.4 int mitkSceneBase::GetWidth () [inline]

Get the width of the container window. Provided for convinience.

8.240.2.5 void* mitkSceneBase::GetWindowId (void) [inline]

Get the system dependent id of the container window. Provided for convinience.

8.240.2.6 bool mitkSceneBase::HasUnprocessedMouseMessage () [inline]

Test if there is any unprocessed mouse message (button down/up) in the message queue of rendering thread. Calls the container. Provided for convinience.

8.240.2.7 virtual void mitkSceneBase::InitGL () [inline, virtual]

The Interface to initialize OpenGL variables and resources.

Reimplemented in mitkImageScene, and mitkScene.

8.240.2.8 void mitkSceneBase::MakeCurrent () [inline]

Make the OpenGL context of the container window current. Provided for convinience.

8.240.2.9 virtual void mitkSceneBase::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [inline, virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented in mitkScene, mitkEndoscopySliceScene, mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkIsosurfaceVRScene, mitkVis2Scene3D, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAEndoscopySliceScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatileVRScene.

8.240.2.10 virtual void mitkSceneBase::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [inline, virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented in mitkScene, mitkEndoscopySliceScene, mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkIsosurfaceVRScene, mitkVis2Scene3D, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAEndoscopySliceScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatileVRScene.

8.240.2.11 virtual void mitkSceneBase::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [inline, virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented in mitkScene, mitkEndoscopySliceScene, mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkIsosurfaceVRScene, mitkVis2Scene3D, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAEndoscopySliceScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatileVRScene.

8.240.2.12 virtual void mitkSceneBase::OnMouseWheel (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *delta*) [inline, virtual]

The Interface for handling the mouse wheel event pass by the container

Reimplemented in mitkScene, mitkEndoscopySliceScene, and mitkCUDAEndoscopySliceScene.

8.240.2.13 virtual void mitkSceneBase::RenderGL () [inline, virtual]

The Interface for handling the render event pass by the container

Reimplemented in mitkImageScene, mitkScene, mitkEndoscopySliceScene, mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkGLSLMIPVRScene, mitkIsosurfaceVRScene, mitkMIPVRScene, mitkTargetMeasurementScene, mitkVis2Scene3D, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAEndoscopySliceScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAMIPVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitk-

CUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatile-VRScene.

8.240.2.14 virtual void mitkSceneBase::ResizeGL (int *newX*, int *newY*) [inline, virtual]

The Interface for handling the resize event pass by the container

Reimplemented in mitkImageScene, mitkScene, and mitkVis2Scene3D.

8.240.2.15 void mitkSceneBase::SwapBuffers () [inline]

Swap the buffers of the container window. Provided for convinience.

8.240.2.16 void mitkSceneBase::Update () [inline]

Update the scene. Actually calls the container->Update(). Provided for convinience.

The documentation for this class was generated from the following file:

- mitkSceneBase.h

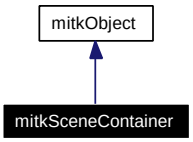
8.241 mitkSceneContainer Class Reference

`#include <mitkSceneContainer.h>`

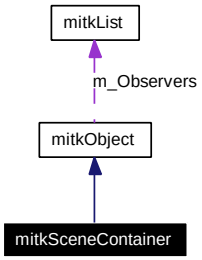
Inherits mitkObject.

Inherited by mitkGLWindow.

Inheritance diagram for mitkSceneContainer:



Collaboration diagram for mitkSceneContainer:



Public Member Functions

- virtual void * GetWindowId ()=0
- virtual const int * GetSize ()=0
- virtual int GetWidth ()=0
- virtual int GetHeight ()=0
- virtual void Update ()=0
- virtual void MakeCurrent ()=0
- virtual void SwapBuffers ()=0
- virtual bool HasUnprocessedMouseMessage ()
- virtual void MoveMouseTo (int x, int y)=0

8.241.1 Detailed Description

mitkSceneContainer - an abstract class defining the interface of a scene container window, which need to be back-referenced by the embeding scene

mitkSceneContainer is an abstract class defining the interface of a scene container window, which need to be back-referenced by the embeding scene You can use this interface to define your own conatiner of a scene. A concrete scene container should maintain an OpenGL context and pass window events to the embeding scene.

8.241.2 Member Function Documentation

8.241.2.1 virtual int mitkSceneContainer::GetHeight () [pure virtual]

The Interface to get the height of the container window

8.241.2.2 virtual const int* mitkSceneContainer::GetSize () [pure virtual]

The Interface to get the size of the container window

8.241.2.3 virtual int mitkSceneContainer::GetWidth () [pure virtual]

The Interface to get the width of the container window

8.241.2.4 virtual void* mitkSceneContainer::GetWindowId () [pure virtual]

The Interface to get the system dependent id of the container window

8.241.2.5 virtual bool mitkSceneContainer::HasUnprocessedMouseMessage () [inline, virtual]

The Interface to tests if there is any unprocessed mouse message (button down/up) in the container window

8.241.2.6 virtual void mitkSceneContainer::MakeCurrent () [pure virtual]

The Interface to make the OpenGL context of the container window current

8.241.2.7 virtual void mitkSceneContainer::MoveMouseTo (int *x*, int *y*) [pure virtual]

Warp mouse to the position (x,y)

Parameters:

x

y

8.241.2.8 virtual void mitkSceneContainer::SwapBuffers () [pure virtual]

The Interface to swap the buffers of the container window

8.241.2.9 virtual void mitkSceneContainer::Update () [pure virtual]

The Interface to update the container window

The documentation for this class was generated from the following file:

- mitkSceneContainer.h

8.242 mitkSceneParam Class Reference

mitkSceneParam - storage class for scene parameters

`#include <mitkSceneParam.h>`

Public Member Functions

- void InitSceneParam (const float *size3D)

  m_Fovy & m_AspectRatio should be set before calling!

8.242.1 Detailed Description

mitkSceneParam - storage class for scene parameters

mitkSceneParam is a storage class for scene parameters, used by mitkVis2Scene3D

The documentation for this class was generated from the following file:

- mitkSceneParam.h

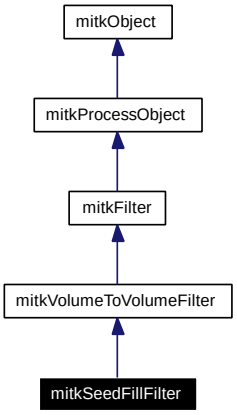
8.243 mitkSeedFillFilter Class Reference

mitkSeedFillFilter - a concrete filter for seed fill algorithm

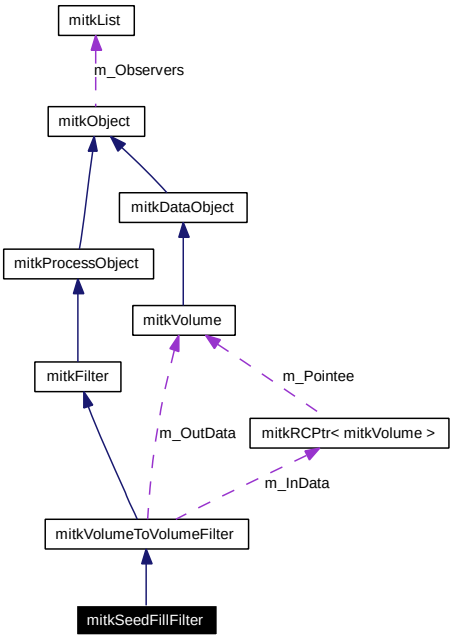
`#include <mitkSeedFillFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkSeedFillFilter:



Collaboration diagram for mitkSeedFillFilter:



Public Member Functions

- mitkSeedFillFilter ()
- virtual void PrintSelf (ostream &os)
- virtual ~mitkSeedFillFilter ()

8.243.1 Detailed Description

mitkSeedFillFilter - a concrete filter for seed fill algorithm

mitkSeedFillFilter is a concrete filter for seed fill algorithm.

Note:
Both the input and the output are bivalue images. The contour of the input image is not zero. Datatype of both the input and out are char.

8.243.2 Constructor & Destructor Documentation

8.243.2.1 mitkSeedFillFilter::mitkSeedFillFilter () [inline]

Constructor of the class.

8.243.2.2 virtual mitkSeedFillFilter::~mitkSeedFillFilter () [inline, virtual]

Constructor of the class.

8.243.3 Member Function Documentation

8.243.3.1 virtual void mitkSeedFillFilter::PrintSelf (ostream & *os*) [inline, virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkSeedFillFilter.h

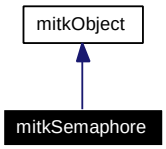
8.244 mitkSemaphore Class Reference

mitkSemaphore - a class that encapsulates a semaphore

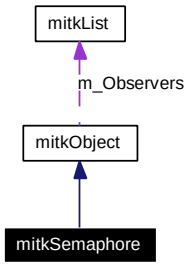
`#include <mitkSemaphore.h>`

Inherits mitkObject.

Inheritance diagram for mitkSemaphore:



Collaboration diagram for mitkSemaphore:



Public Member Functions

- void Wait ()
- void Post (int count=1)

8.244.1 Detailed Description

mitkSemaphore - a class that encapsulates a semaphore

mitkSemaphore encapsulates a semaphore.

8.244.2 Member Function Documentation

8.244.2.1 void mitkSemaphore::Post (int *count* = 1) [inline]

Add count to the semaphore object

Parameters:

count Specify the count to be added to the semaphore object

8.244.2.2 void mitkSemaphore::Wait () [inline]

Wait for the semaphore object and decrease the count

The documentation for this class was generated from the following file:

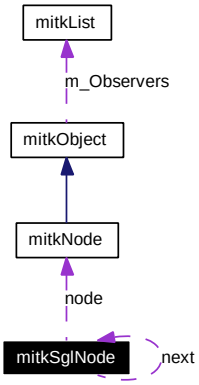
- mitkSemaphore.h

8.245 mitkSglNode Class Reference

mitkSglNode - a class that define mitkSglNode used by mitkNodeHeap

`#include <mitkNodeHeap.h>`

Collaboration diagram for mitkSglNode:



Public Member Functions

- mitkSglNode ()
- ~mitkSglNode ()

8.245.1 Detailed Description

mitkSglNode - a class that define mitkSglNode used by mitkNodeHeap

mitkSglNode define a single node of a SgnNode link

8.245.2 Constructor & Destructor Documentation

8.245.2.1 mitkSglNode::mitkSglNode () [inline]

Constructor of the class

8.245.2.2 mitkSglNode::~mitkSglNode () [inline]

Destructor of the class

The documentation for this class was generated from the following file:

- mitkNodeHeap.h

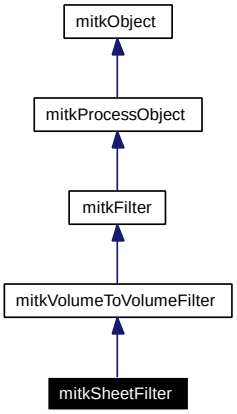
8.246 mitkSheetFilter Class Reference

mitkSheetFilter - a concrete class for implementation of 3D Sheet Structure Enhancement

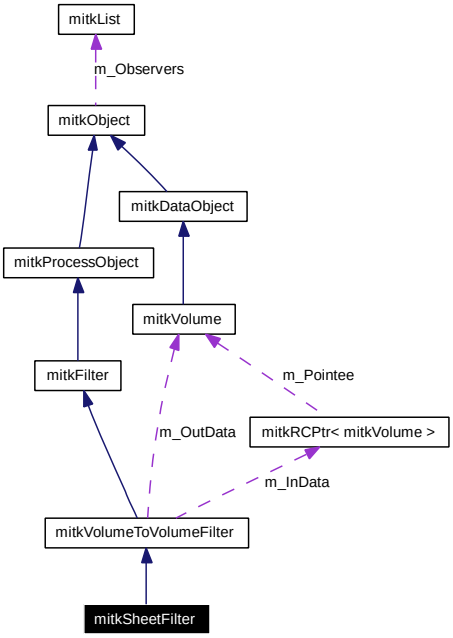
`#include <mitkSheetFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkSheetFilter:



Collaboration diagram for mitkSheetFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetGauss (bool applyGauss, float StandardDeviation)

- void SetAlpha (float alpha)
- void SetGamma (float gamma)

8.246.1 Detailed Description

mitkSheetFilter - a concrete class for implementation of 3D Sheet Structure Enhancement

mitkSheetFilter - a concrete class for implementation of 3D Sheet Structure Enhancement

8.246.2 Member Function Documentation

8.246.2.1 virtual void mitkSheetFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.246.2.2 void mitkSheetFilter::SetAlpha (float *alpha*) [inline]

Set the Alpha Parameter

Parameters:
the alpha parameter

8.246.2.3 void mitkSheetFilter::SetGamma (float *gamma*) [inline]

Set the Gamma Parameter

Parameters:
the gamma parameter

8.246.2.4 void mitkSheetFilter::SetGauss (bool *applyGauss*, float *StandardDeviation*)

Set the Gauss Filtering Parameters

Parameters:
applyGauss,: whether gauss filtering should be applied
StandardDeviation,: the standard deviation used in gauss filtering

The documentation for this class was generated from the following file:

- mitkSheetFilter.h

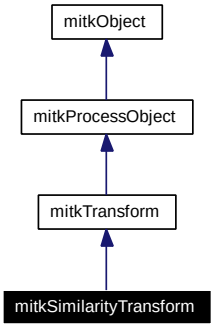
8.247 mitkSimilarityTransform Class Reference

mitkSimilarityTransform - a concrete transform to perform Similarity transformation.

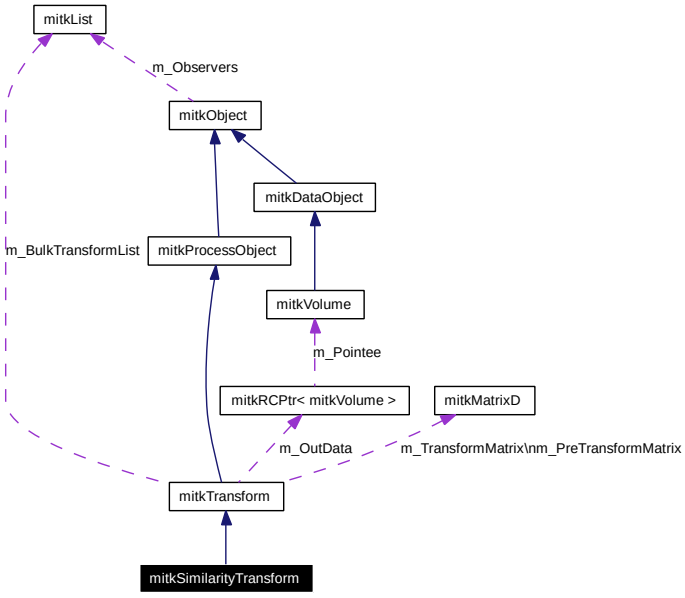
`#include <mitkSimilarityTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkSimilarityTransform:



Collaboration diagram for mitkSimilarityTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkSimilarityTransform ()
- mitkSimilarityTransform (unsigned int dim)
- const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- virtual void ConvertTransformMaxtrixToVector ()
- void GetAngles (ScalarParameterType &angle)

- void GetAngles (ScalarParameterType &alpha, ScalarParameterType &beta, ScalarParameterType &gamma)
- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty)
- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty, ScalarParameterType &tz)
- void GetScales (ScalarParameterType &scale)

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.247.1 Detailed Description

mitkSimilarityTransform - a concrete transform to perform Similarity transformation.

This transform applies a rotation, translation and scaling to the 2D and 3D image space. User should use the constructor to specify the image dimensions, e.g. mitkSimilarityTransform(2) (default) for 2d image, mitkSimilarityTransform(3) for 3d image.

Transform Mode Number of parameters p[0] p[1] p[2] p[3] p[4] p[5] p[6]

MITK_TRANSFORM_SIMILARITY (2D) 4 angle tx ty scale - - -

MITK_TRANSFORM_SIMILARITY (3D) 7 angle_x angle_y angle_z tx ty tz scale

8.247.2 Constructor & Destructor Documentation

8.247.2.1 mitkSimilarityTransform::mitkSimilarityTransform ()

Constructor.

8.247.2.2 mitkSimilarityTransform::mitkSimilarityTransform (unsigned int *dim*)

Constructor with specific transform mode setting.

8.247.3 Member Function Documentation

8.247.3.1 virtual void mitkSimilarityTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.247.3.2 virtual void mitkSimilarityTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.247.3.3 void mitkSimilarityTransform::GetAngles (ScalarParameterType & *alpha*, ScalarParameterType & *beta*, ScalarParameterType & *gamma*)

Get the rotation parameters from the transform parameter vector. (for 3d rotation)

Parameters:

alpha The returned angle value. (rotate around x-axis)

beta The returned angle value. (rotate around y-axis)

gamma The returned angle value. (rotate around z-axis)

8.247.3.4 void mitkSimilarityTransform::GetAngles (ScalarParameterType & *angle*)

Get the rotation parameters from the transform parameter vector. (for 2d rotation)

Parameters:

angle The returned angle value.

8.247.3.5 const MatrixType& mitkSimilarityTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:

inPoint The index of the point in image.

Returns:

Return the pointer to the Jacobian matrix.

Reimplemented from mitkTransform.

8.247.3.6 void mitkSimilarityTransform::GetScales (ScalarParameterType & *scale*)

Get the scaling parameters from the transform parameter vector.

Parameters:

scale The returned isotropic scale value.

8.247.3.7 void mitkSimilarityTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*, ScalarParameterType & *tz*)

Get the translation parameters from the transform parameter vector. (for 3d translation)

Parameters:

tx The returned x-translation value.

ty The returned y-translation value.

tz The returned z-translation value.

8.247.3.8 void mitkSimilarityTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*)

Get the translation parameters from the transform parameter vector. (for 2d translation)

Parameters:

tx The returned x-translation value.

ty The returned y-translation value.

8.247.3.9 virtual void mitkSimilarityTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkSimilarityTransform.h

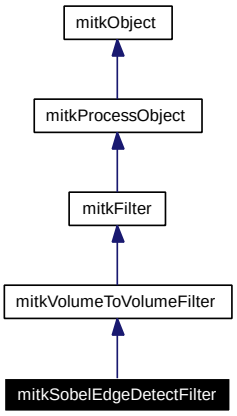
8.248 mitkSobelEdgeDetectFilter Class Reference

mitkSobelEdgeDetectFilter - a class used to detect the edge in an image

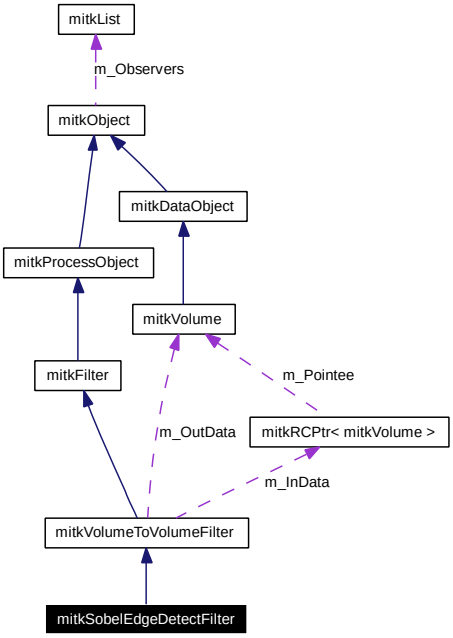
`#include <mitkSobelEdgeDetectFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkSobelEdgeDetectFilter:



Collaboration diagram for mitkSobelEdgeDetectFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetThreshold (float Threshold)

8.248.1 Detailed Description

mitkSobelEdgeDetectFilter - a class used to detect the edge in an image

mitkSobelEdgeDetectFilter is a filter based on the sobel edge detect method. this filter comprise the considerations of both the vertical trend and the horizontal trend.the result is also quite good contrast to its simplicity.But, sometimes the processed image may present a little discontinuity.

8.248.2 Member Function Documentation

8.248.2.1 virtual void mitkSobelEdgeDetectFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.248.2.2 void mitkSobelEdgeDetectFilter::SetThreshold (float *Threshold*) [inline]

Set the member variable m_Threshold.

Parameters:
Threshold represent the cut value,the value below Threshold is set to 0,and the value above Threshold is set to 255.

Warning:
you need to choose an appropriate value(<50) to get a satisfying result.

The documentation for this class was generated from the following file:

- mitkSobelEdgeDetectFilter.h

8.249 mitkSource Class Reference

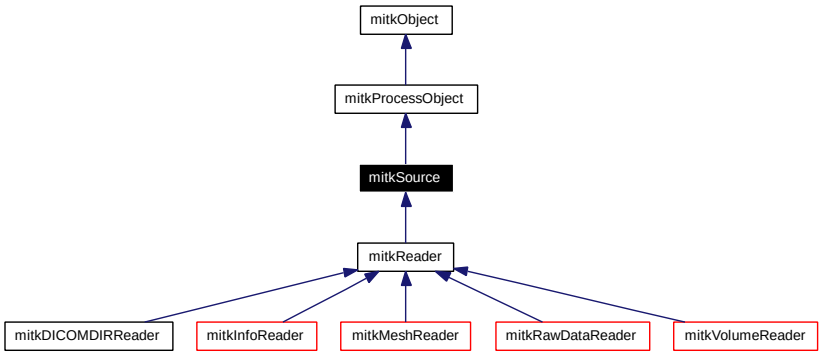
mitkSource - abstract class specifies interface for source object

`#include <mitkSource.h>`

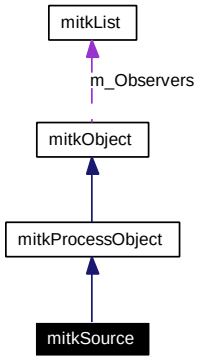
Inherits mitkProcessObject.

Inherited by mitkDeformFieldGenerator, and mitkReader.

Inheritance diagram for mitkSource:



Collaboration diagram for mitkSource:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.249.1 Detailed Description

mitkSource - abstract class specifies interface for source object

mitkSource is an abstract object that specifies behavior and interface of source objects. Source objects are creators of data objects, including reader and procedual source.

8.249.2 Member Function Documentation

8.249.2.1 virtual void mitkSource::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProcessObject.

Reimplemented in mitkAnalyzeImageReader, mitkBMPReader, mitkCacheVolumeReader, mitkDICOMInfoReader, mitkDICOMReader, mitkInfoReader, mitkJPEGReader, mitkMeshReader, mitkMetaImageReader, mitkNetCDFImageReader, mitkPLYReader, mitkRawFilesReader, mitkRawReader, mitkReader, mitkSTLReader, mitkTIFFReader, mitkVolumeReader, mitkRawDataReader, mitkRawFilesReader, and mitkTrackedBscanDataReader.

The documentation for this class was generated from the following file:

- mitkSource.h

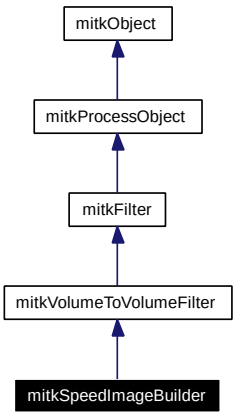
8.250 mitkSpeedImageBuilder Class Reference

mitkSpeedImageBuilder - a class that build SpeedImage for mitkFastMarchingImageFilter

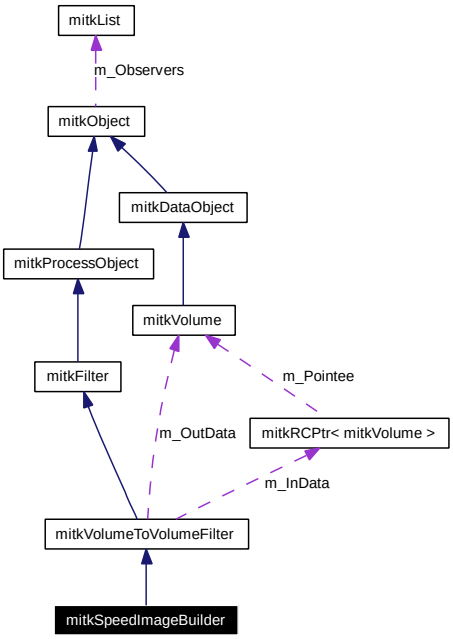
`#include <mitkSpeedImageBuilder.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkSpeedImageBuilder:



Collaboration diagram for mitkSpeedImageBuilder:



8.250.1 Detailed Description

mitkSpeedImageBuilder - a class that build SpeedImage for mitkFastMarchingImageFilter

mitkSpeedImageBuilder build speed image for mitkFastMarchingImageFilter. Datatype of the output of this filter is double no matter the datatype of the input data.

Some codes are borrowed from ITK, and please see the copyright at end.

The documentation for this class was generated from the following file:

- mitkSpeedImageBuilder.h

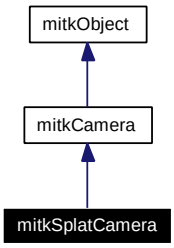
8.251 mitkSplatCamera Class Reference

mitkSplatCamera - a camera in 3D scene

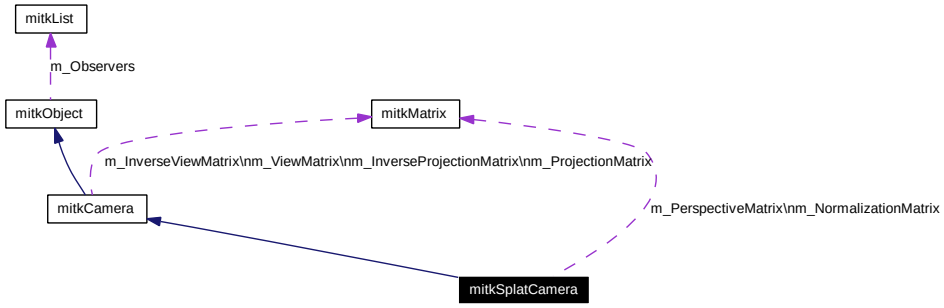
`#include <mitkSplatCamera.h>`

Inherits mitkCamera.

Inheritance diagram for mitkSplatCamera:



Collaboration diagram for mitkSplatCamera:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Frustum (float left, float right, float bottom, float top, float zNear, float zFar)
- virtual void Perspective (float fovy, float aspect, float zNear, float zFar)
- void GetPerspectiveMatrix (mitkMatrix *m)
- void GetPerspectiveMatrix (float m[16])
- const mitkMatrix * GetPerspectiveMatrix (void)
- void GetNormalizationMatrix (mitkMatrix *m)
- void GetNormalizationMatrix (float m[16])
- const mitkMatrix * GetNormalizationMatrix (void)

8.251.1 Detailed Description

mitkSplatCamera - a camera in 3D scene

mitkSplatCamera is a camera in 3D scene. The only way you can access it is to get a pointer to it by using the member function GetCamera() of mitkScene. Then you can control it through its interface, and the change will affect the image rendered in the scene. This class derives from mitkCamera, and separates projection matrix into perspective matrix and normalization matrix.

Note:
If a mitkVolumeRendererSplatting object is used to render the volume, you should set the scene's camera to be mitkSplatCamera object.

See also:
Please see more information from mitkCamera class.

8.251.2 Member Function Documentation

8.251.2.1 virtual void mitkSplatCamera::Frustum (float *left*, float *right*, float *bottom*, float *top*, float *zNear*, float *zFar*) [virtual]

Perspective Projection

Reimplemented from mitkCamera.

8.251.2.2 const mitkMatrix* mitkSplatCamera::GetNormalizationMatrix (void) [inline]

Get the Normalization transform matrix.

8.251.2.3 void mitkSplatCamera::GetNormalizationMatrix (float *m*[16])

Get the Normalization transform matrix.

8.251.2.4 void mitkSplatCamera::GetNormalizationMatrix (mitkMatrix * *m*)

Get the Normalization transform matrix.

8.251.2.5 const mitkMatrix* mitkSplatCamera::GetPerspectiveMatrix (void) [inline]

Get the perspective transform matrix.

8.251.2.6 void mitkSplatCamera::GetPerspectiveMatrix (float *m*[16])

Get the perspective transform matrix.

8.251.2.7 void mitkSplatCamera::GetPerspectiveMatrix (mitkMatrix * *m*)

Get the perspective transform matrix.

8.251.2.8 virtual void mitkSplatCamera::Perspective (float *fovy*, float *aspect*, float *zNear*, float *zFar*) [virtual]

Perspective Projection

Reimplemented from mitkCamera.

8.251.2.9 virtual void mitkSplatCamera::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkCamera.

The documentation for this class was generated from the following file:

- mitkSplatCamera.h

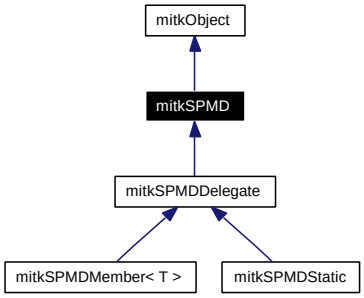
8.252 mitkSPMD Class Reference

`#include <mitkSPMD.h>`

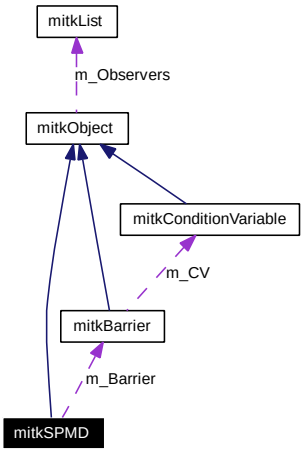
Inherits mitkObject.

Inherited by mitkSPMDDelegate.

Inheritance diagram for mitkSPMD:



Collaboration diagram for mitkSPMD:



Public Member Functions

- void SetNumberOfThreads (int NumberOfThreads)
- int GetNumberOfThreads () const
- void RunSPMD ()

Static Public Member Functions

- static int GetGlobalDefaultNumberOfThreads ()

8.252.1 Detailed Description

mitkSPMD - A threading tool that implements a SPMD (Single Program Multiple Data) style multi-thread execution.

mitkSPMD is a threading tool that implements a SPMD (Single Program Multiple Data) style multi-thread execution. mitkSPMD will create a number of threads, and each thread will end up calling the virtual protected function ThreadExecute with their unique identifier, which can be used to decide the part of data to be processed, since the memory is shared among threads.

8.252.2 Member Function Documentation

8.252.2.1 static int mitkSPMD::GetGlobalDefaultNumberOfThreads () [static]

Acquire the default number of threads according to the number of hardware processors(cores)

Returns:
Return the global default number of threads

8.252.2.2 int mitkSPMD::GetNumberOfThreads () const [inline]

Get the number of threads to be created

Returns:
return the number of threads to be created the default value is the GlobalDefaultNumberOfThreads

8.252.2.3 void mitkSPMD::RunSPMD ()

Perform the SPMD execution

8.252.2.4 void mitkSPMD::SetNumberOfThreads (int *NumberOfThreads*)

Set the number of threads to be created

Parameters:
NumberOfThreads Specify the number of threads to be created must be greater or equal than 1, and the default value is the GlobalDefaultNumberOfThreads

The documentation for this class was generated from the following file:

- mitkSPMD.h

8.253 mitkSPMDDelegate Class Reference

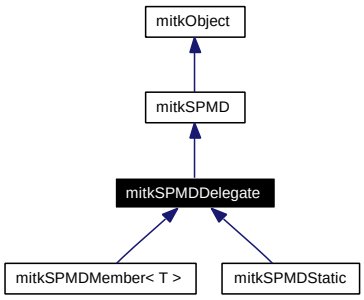
mitkSPMDDelegate - The base class of mitkSPMDStatic and mitkSPMDMember,

`#include <mitkSPMD.h>`

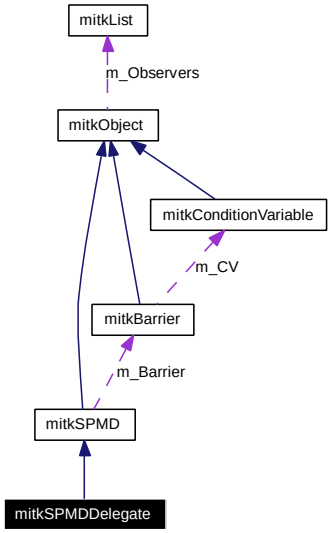
Inherits mitkSPMD.

Inherited by mitkSPMDMember< T >, and mitkSPMDStatic.

Inheritance diagram for mitkSPMDDelegate:



Collaboration diagram for mitkSPMDDelegate:



8.253.1 Detailed Description

mitkSPMDDelegate - The base class of mitkSPMDStatic and mitkSPMDMember,

mitkSPMDDelegate - The base class of mitkSPMDStatic and mitkSPMDMember, which redirect SPMD functions to some user functions. This class allows user to set a parameter to be passed to the user entrance.

The documentation for this class was generated from the following file:

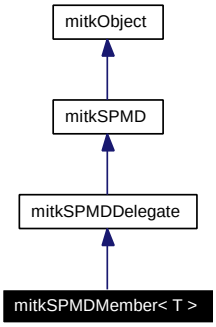
- mitkSPMD.h

8.254 mitkSPMDMember< T > Class Template Reference

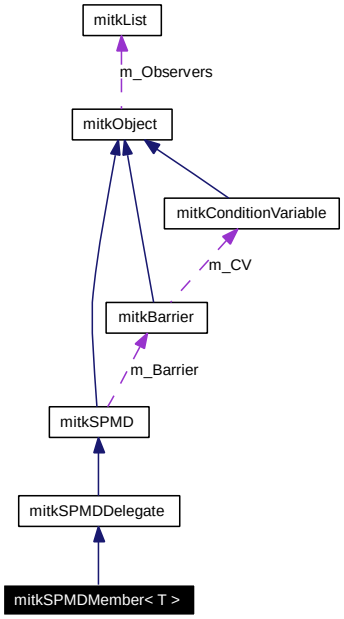
`#include <mitkSPMD.h>`

Inherits mitkSPMDDelegate.

Inheritance diagram for mitkSPMDMember< T >:



Collaboration diagram for mitkSPMDMember< T >:



8.254.1 Detailed Description

template<class T> class mitkSPMDMember< T >

mitkSPMDMember - A utility allowing user to conveniently create a delegate SPMD object that redirects SPMD functions to specified member functions.

mitkSPMDMember is a utility allowing user to conveniently create a delegate SPMD object that redirects SPMD functions to specified member functions.

The documentation for this class was generated from the following file:

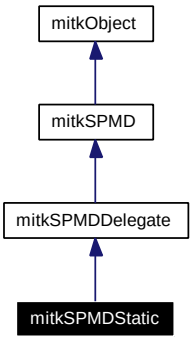
- mitkSPMD.h

8.255 mitkSPMDStatic Class Reference

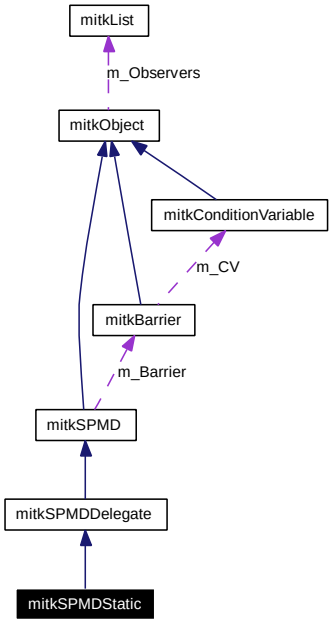
`#include <mitkSPMD.h>`

Inherits mitkSPMDDelegate.

Inheritance diagram for mitkSPMDStatic:



Collaboration diagram for mitkSPMDStatic:



8.255.1 Detailed Description

mitkSPMDStatic - A utility allowing user to conveniently create a delegate SPMD object that redirects SPMD functions to specified static functions or global functions.

mitkSPMDStatic is a utility allowing user to conveniently create a delegate SPMD object that redirects SPMD functions to specified static functions or global functions.

The documentation for this class was generated from the following file:

- mitkSPMD.h

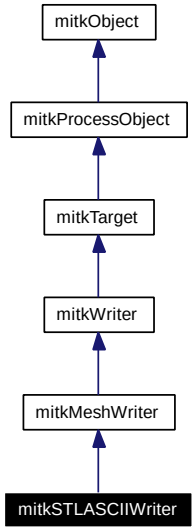
8.256 mitkSTLASCIIWriter Class Reference

mitkSTLASCIIWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using ASCII format

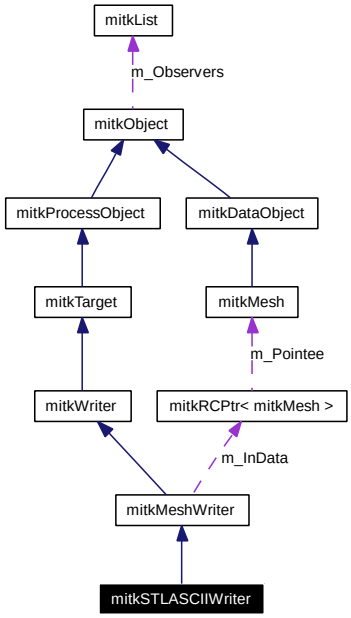
`#include <mitkSTLASCIIWriter.h>`

Inherits mitkMeshWriter.

Inheritance diagram for mitkSTLASCIIWriter:



Collaboration diagram for mitkSTLASCIIWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.256.1 Detailed Description

mitkSTLASCIIWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using ASCII format

mitkSTLASCIIWriter writes a mesh to a STL file using ASCII format. To use this writer, the code snippet is:

```
mitkIM0Writer *aWriter = new mitkSTLASCIIWriter;
aWriter->SetInput(aMesh);
aWriter->AddFileName(filename);
aWriter->Run();
```

8.256.2 Member Function Documentation

8.256.2.1 virtual void mitkSTLASCIIWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMeshWriter.

The documentation for this class was generated from the following file:

- mitkSTLASCIIWriter.h

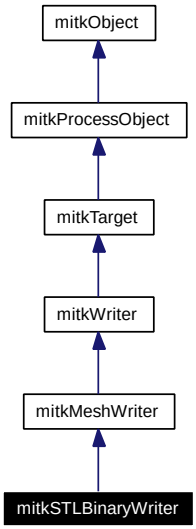
8.257 mitkSTLBinaryWriter Class Reference

mitkSTLBinaryWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using binary format

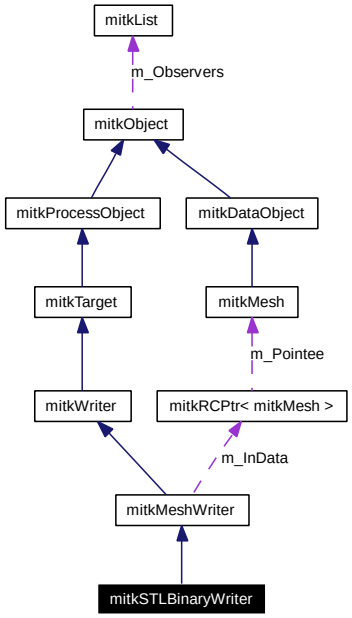
`#include <mitkSTLBinaryWriter.h>`

Inherits mitkMeshWriter.

Inheritance diagram for mitkSTLBinaryWriter:



Collaboration diagram for mitkSTLBinaryWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.257.1 Detailed Description

mitkSTLBinaryWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using binary format

mitkSTLBinaryWriter writes a mesh to a STL file using binary format. To use this writer, the code snippet is:

```
mitkIMOWriter *aWriter = new mitkSTLBinaryWriter;
aWriter->SetInput(aMesh);
aWriter->AddFileName(filename);
aWriter->Run();
```

8.257.2 Member Function Documentation

8.257.2.1 virtual void mitkSTLBinaryWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMeshWriter.

The documentation for this class was generated from the following file:

- mitkSTLBinaryWriter.h

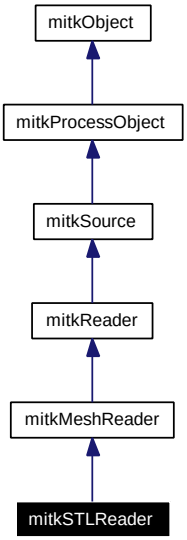
8.258 mitkSTLReader Class Reference

mitkSTLReader - a concrete reader for reading a STL file

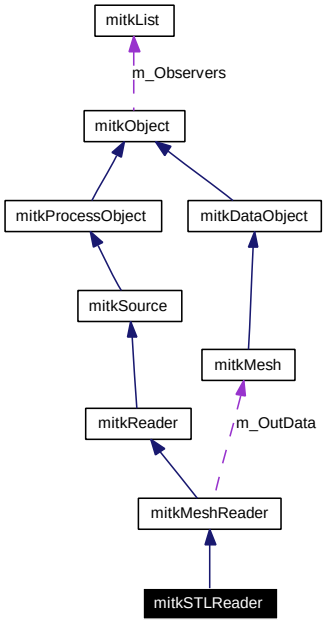
`#include <mitkSTLReader.h>`

Inherits mitkMeshReader.

Inheritance diagram for mitkSTLReader:



Collaboration diagram for mitkSTLReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.258.1 Detailed Description

mitkSTLReader - a concrete reader for reading a STL file

mitkSTLReader is a concrete reader for reading a STL file. the code snippet is:

```
mitkSTLReader *aReader = new mitkSTLReader;
aReader->AddFileName(filename);
if (aReader->Run())
{
    mitkMesh *aMesh = aReader->GetOutput();
    Using aMesh
}
```

8.258.2 Member Function Documentation

8.258.2.1 virtual void mitkSTLReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMeshReader.

The documentation for this class was generated from the following file:

- mitkSTLReader.h

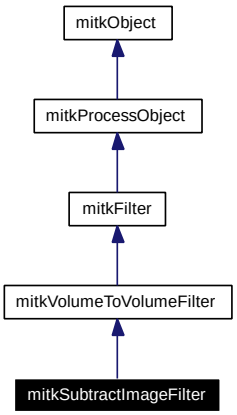
8.259 mitkSubtractImageFilter Class Reference

mitkSubtractImageFilter - a concrete class for subtraction of two volumes

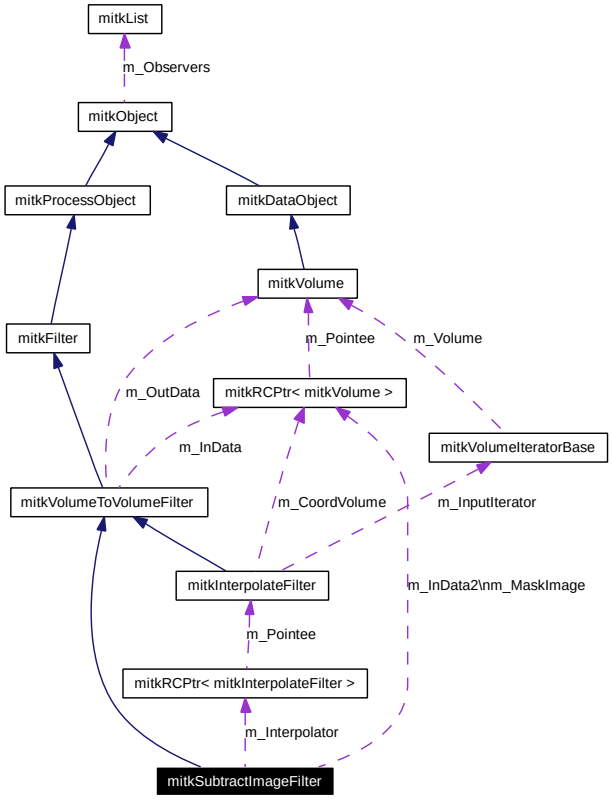
`#include <mitkSubtractImageFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkSubtractImageFilter:



Collaboration diagram for mitkSubtractImageFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkSubtractImageFilter (int mode)
- mitkSubtractImageFilter ()
- void SetInput1 (mitkVolume *inData)
- void SetInput2 (mitkVolume *inData)
- void SetSubtractMode (int mode)
- void SetOutputDatatype (int dataType)

8.259.1 Detailed Description

mitkSubtractImageFilter - a concrete class for subtraction of two volumes

mitkSubtractImageFilter is a concrete class for subtraction of two volumes. mitkSubtractImageFilter can only handle gray level images in same size. This class provide several subtract modes,including SubGray,AddGray,SubColor AddColor, GridGray. The particular functions of these modes is list below:

Output Channels Remarks

SubGray : Output = Input1 - Input2 1

AddGray : Output = Input1 + Input2 1

Output(R) = Input1 SubColor: Output(G) = Input1 - Input2 3 Output(B) = 255 - Input2

Output(R) = Input1 AddColor: Output(G) = 0 3 Output(B) = Input2

GridGray: 3 Input1 and Input2 are alternately presented on 4x4 grid. Usage: mitkSubtractImageFilter* subtract = new mitkSubtractImageFilter(MITK_SUBSTRACT_MODE_SUBCOLOR); subtract->SetInput1(volume1); subtract->SetInput2(volume2); subtract->Run(); mitkVolume* outVolume = subtract->GetOutput();

8.259.2 Constructor & Destructor Documentation

8.259.2.1 mitkSubtractImageFilter::mitkSubtractImageFilter (int *mode*)

Constructor with Specific subtract mode. Default is MITK_SUBSTRACT_MODE_SUB.

Parameters:

mode Specify the image subtract mode.

8.259.2.2 mitkSubtractImageFilter::mitkSubtractImageFilter ()

Constructor.

8.259.3 Member Function Documentation

8.259.3.1 virtual void mitkSubtractImageFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.259.3.2 void mitkSubtractImageFilter::SetInput1 (mitkVolume * *inData*) [inline]

Set the first input volume. (Do NOT call SetInput() directly.) param inData the pointer to the input volume.

8.259.3.3 void mitkSubtractImageFilter::SetInput2 (mitkVolume * *inData*) [inline]

Set the second input volume. param inData the pointer to the input volume.

8.259.3.4 void mitkSubtractImageFilter::SetOutputDatatype (int *dataType*) [inline]

Set the data type for output volume.Default is MITK_FLOAT.

Parameters:

dataType Specify the output volume's data type.

8.259.3.5 void mitkSubtractImageFilter::SetSubtractMode (int *mode*) [inline]

Set the image subtract mode.Default is MITK_SUBSTRACT_MODE_SUBGRAY. param mode Specify the image subtract mode.

The documentation for this class was generated from the following file:

- mitkSubtractImageFilter.h

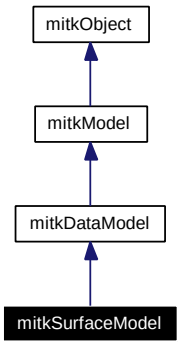
8.260 mitkSurfaceModel Class Reference

mitkSurfaceModel - an 3D entity in a rendering scene represented in surface

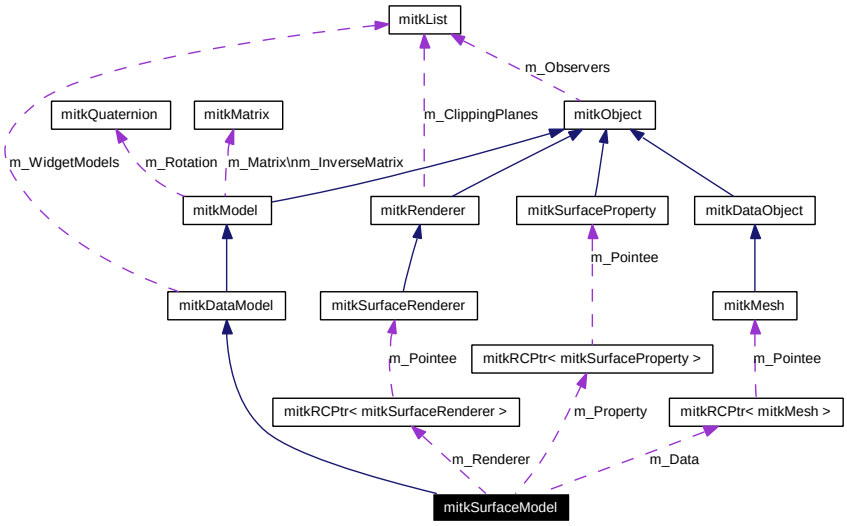
`#include <mitkSurfaceModel.h>`

Inherits mitkDataModel.

Inheritance diagram for mitkSurfaceModel:



Collaboration diagram for mitkSurfaceModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkSurfaceModel ()
- virtual int Render (mitkScene *scene)
- virtual mitkRenderer * GetBasicRenderer ()
- void SetRenderer (mitkSurfaceRenderer *renderer)
- mitkSurfaceRenderer * GetRenderer ()
- void SetProperty (mitkSurfaceProperty *prop)
- mitkSurfaceProperty * GetProperty ()

- void SetData (mitkMesh *data)
- mitkMesh * GetData ()
- virtual bool IsOpaque ()

8.260.1 Detailed Description

mitkSurfaceModel - an 3D entity in a rendering scene represented in surface

mitkSurfaceModel is an 3D entity in a rendering scene represented in surface. It contains an mitkMesh object which should be shown in the rendering scene. Not like widget model, it is a kind of mitkDataModel and can not be manipulated by itself.

8.260.2 Constructor & Destructor Documentation

8.260.2.1 mitkSurfaceModel::mitkSurfaceModel ()

Default constructor.

8.260.3 Member Function Documentation

8.260.3.1 virtual mitkRenderer* mitkSurfaceModel::GetBasicRenderer () [inline, virtual]

Get the renderer for widget.

Returns:
Return pointer to an mitkRenderer which renders this model actually.

Note:
This function is written for the widget who needs to know the information about the surface model's renderer.

Reimplemented from mitkDataModel.

8.260.3.2 mitkMesh* mitkSurfaceModel::GetData (void) [inline]

Get the surface data.

Returns:
Return pointer to the surface model's mesh data.

8.260.3.3 mitkSurfaceProperty* mitkSurfaceModel::GetProperty ()

Get the surface property.

Returns:
Return pointer to this surface model's property.

8.260.3.4 mitkSurfaceRenderer* mitkSurfaceModel::GetRenderer ()

Get the surface renderer.

Returns:
Return a pointer to this surface model's render.

8.260.3.5 virtual bool mitkSurfaceModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Implements mitkModel.

8.260.3.6 virtual void mitkSurfaceModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkDataModel.

8.260.3.7 virtual int mitkSurfaceModel::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Warning:
Don't call this function directly.

Reimplemented from mitkModel.

8.260.3.8 void mitkSurfaceModel::SetData (mitkMesh * *data*)

Set the surface data.

Parameters:
data pointer to an mitkMesh

8.260.3.9 void mitkSurfaceModel::SetProperty (mitkSurfaceProperty * *prop*) [inline]

Set the surface property.

Parameters:
prop pointer to an mitkSurfaceProperty

8.260.3.10 void mitkSurfaceModel::SetRenderer (mitkSurfaceRenderer * *renderer*) [inline]

Set the surface renderer.

Parameters:
renderer pointer to an mitkSurfaceRenderer

The documentation for this class was generated from the following file:

- mitkSurfaceModel.h

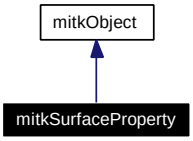
8.261 mitkSurfaceProperty Class Reference

mitkSurfaceProperty - properties of an mitkSurfaceModel

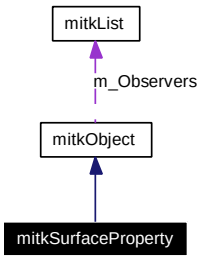
`#include <mitkSurfaceProperty.h>`

Inherits mitkObject.

Inheritance diagram for mitkSurfaceProperty:



Collaboration diagram for mitkSurfaceProperty:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkSurfaceProperty ()
- bool IsModified () const
- void SetUnmodified ()
- void SetInterpolationType (int intpType)
- int GetInterpolationType () const
- void SetInterpolationTypeToFlat ()
- void SetInterpolationTypeToGouraud ()
- void SetInterpolationTypeToPhong ()
- const char * GetInterpolationTypeAsString () const
- void SetRepresentationType (int repType)
- int GetRepresentationType () const
- void SetRepresentationTypeToPoints ()
- void SetRepresentationTypeToWireframe ()
- void SetRepresentationTypeToSurface ()
- const char * GetRepresentationTypeAsString () const
- void SetAmbient (float value)
- float GetAmbient () const
- void SetDiffuse (float value)
- float GetDiffuse () const
- void SetSpecular (float value)

- float GetSpecular () const
- void SetSpecularPower (float value)
- float GetSpecularPower () const
- void SetOpacity (float value)
- float GetOpacity () const
- void SetLineWidth (float value)
- float GetLineWidth () const
- void SetPointSize (float value)
- float GetPointSize () const
- void SetLineStipplePattern (unsigned short value)
- unsigned short GetLineStipplePattern () const
- void SetLineStippleRepeatFactor (int value)
- int GetLineStippleRepeatFactor () const
- void SetColor (float red, float green, float blue, float alpha=1.0f)
- void SetColor (float color[4])
- float * GetColor ()
- void GetColor (float color[4])
- mitkSetColorMacro (AmbientColor, float)
- mitkGetColorMacro (AmbientColor, float)
- mitkSetColorMacro (DiffuseColor, float)
- mitkGetColorMacro (DiffuseColor, float)
- mitkSetColorMacro (SpecularColor, float)
- mitkGetColorMacro (SpecularColor, float)
- mitkSetColorMacro (EmissionColor, float)
- mitkGetColorMacro (EmissionColor, float)
- mitkSetColorMacro (EdgeColor, float)
- mitkGetColorMacro (EdgeColor, float)

8.261.1 Detailed Description

mitkSurfaceProperty - properties of an mitkSurfaceModel

mitkSurfaceProperty is a class contains properties of an mitkSurfaceModel including geometry representation parameters and material parameters.

8.261.2 Constructor & Destructor Documentation

8.261.2.1 mitkSurfaceProperty::mitkSurfaceProperty ()

Default constructor.

8.261.3 Member Function Documentation

8.261.3.1 float mitkSurfaceProperty::GetAmbient () const [inline]

Get the ambient coefficient.

Returns:
Return the ambient coefficient.

8.261.3.2 void mitkSurfaceProperty::GetColor (float *color*[4])

Get the color of the surface decided by the value and coefficient of ambient, diffuse, specular and emission color(material) of the surface.

Returns:
color[4] return a float array contains red, green, blue and alpha value in turn.

8.261.3.3 float* mitkSurfaceProperty::GetColor ()

Get the color of the surface decided by the value and coefficient of ambient, diffuse, specular and emission color(material) of the surface.

Returns:
Return a float array contains red, green, blue and alpha value in turn.

8.261.3.4 float mitkSurfaceProperty::GetDiffuse () const [inline]

Get the diffuse coefficient.

Returns:
Return the diffuse coefficient.

8.261.3.5 int mitkSurfaceProperty::GetInterpolationType () const [inline]

Get the interpolation type for sampling a surface.

Returns:
Return interpolation type for sampling a surface, the value could be one of the follows:
MITK_SURFACE_FLAT (flat interpolation type)
MITK_SURFACE_GOURAUD (smooth interpolation type)
MITK_SURFACE_PHONG (smooth interpolation type).

8.261.3.6 const char * mitkSurfaceProperty::GetInterpolationTypeAsString () const [inline]

Get the interpolation type for sampling a surface as a string.

Returns:
Return the string of interpolation type.

8.261.3.7 unsigned short mitkSurfaceProperty::GetLineStipplePattern () const [inline]

Get the stippling pattern of a Line, as a 16-bit binary pattern (1 = pixel on, 0 = pixel off). This is only implemented for OpenGL. The default is 0xFFFF.

Returns:
Return a 16-bit binary pattern represent the stippling pattern of a Line (1 = pixel on, 0 = pixel off).

8.261.3.8 int mitkSurfaceProperty::GetLineStippleRepeatFactor () const [inline]

Get the stippling repeat factor of a Line, which specifies how many times each bit in the pattern is to be repeated.

Returns:
Return the stippling repeat factor of a Line, which specifies how many times each bit in the pattern is to be repeated

8.261.3.9 float mitkSurfaceProperty::GetLineWidth () const [inline]

Get the line width. The width is expressed in screen units.

Returns:
Return the line width.

8.261.3.10 float mitkSurfaceProperty::GetOpacity () const [inline]

Get the object's opacity. 1.0 is totally opaque and 0.0 is completely transparent.

Returns:
Return the object's opacity.

8.261.3.11 float mitkSurfaceProperty::GetPointSize () const [inline]

Get the point size. The size is expressed in screen units.

Returns:
Return the point size.

8.261.3.12 int mitkSurfaceProperty::GetRepresentationType () const [inline]

Get the geometry representation of the surface.

Returns:
Return geometry representation of a surface, the value should be one of the follows:
MITK_MESH_POINTS (points representation)
MITK_MESH_WIREFRAME (wireframe representation)
MITK_MESH_SURFACE (surface representation).

8.261.3.13 const char * mitkSurfaceProperty::GetRepresentationTypeAsString () const [inline]

Get the geometry representation of the surface as a string.

Returns:
Return the string of geometry representation.

8.261.3.14 float mitkSurfaceProperty::GetSpecular () const [inline]

Get the specular coefficient.

Returns:
Return the specular coefficient.

8.261.3.15 float mitkSurfaceProperty::GetSpecularPower () const [inline]

Get the specular power.

Returns:
Return the specular power (between 0.0 and 128.0).

8.261.3.16 bool mitkSurfaceProperty::IsModified () const [inline]

Test if some of the properties are modified.

Returns:
Return true if some of the properties are modified. Otherwise return false.

8.261.3.17 mitkSurfaceProperty::mitkGetColorMacro (EdgeColor, float)

A group of functions to get the color of edges.

This macro will generate three functions:

float∗ GetEdgeColor();

void GetEdgeColor(float &red, float &green, float &blue, float &alpha = 1.0f);

void GetEdgeColor(float color[4]);

8.261.3.18 mitkSurfaceProperty::mitkGetColorMacro (EmissionColor, float)

A group of functions to get the emission color of the surface.

This macro will generate three functions:

float∗ GetEmissionColor();

void GetEmissionColor(float &red, float &green, float &blue, float &alpha = 1.0f);

void GetEmissionColor(float color[4]);

8.261.3.19 mitkSurfaceProperty::mitkGetColorMacro (SpecularColor, float)

A group of functions to get the specular color of the surface.

This macro will generate three functions:

float∗ GetSpecularColor();

void GetSpecularColor(float &red, float &green, float &blue, float &alpha = 1.0f);

void GetSpecularColor(float color[4]);

8.261.3.20 mitkSurfaceProperty::mitkGetColorMacro (DiffuseColor, float)

A group of functions to get the diffuse color of the surface.
This macro will generate three functions:
float∗ GetDiffuseColor();
void GetDiffuseColor(float &red, float &green, float &blue, float &alpha = 1.0f);
void GetDiffuseColor(float color[4]);

8.261.3.21 mitkSurfaceProperty::mitkGetColorMacro (AmbientColor, float)

A group of functions to get the ambient color of the surface.
This macro will generate three functions:
float∗ GetAmbientColor();
void GetAmbientColor(float &red, float &green, float &blue, float &alpha = 1.0f);
void GetAmbientColor(float color[4]);

8.261.3.22 mitkSurfaceProperty::mitkSetColorMacro (EdgeColor, float)

A group of functions to set color of edges.
This macro will generate two functions:
void SetEdgeColor(float red, float green, float blue, float alpha = 1.0f);
void SetEdgeColor(float color[4]);

8.261.3.23 mitkSurfaceProperty::mitkSetColorMacro (EmissionColor, float)

A group of functions to set the emission color of the surface.
This macro will generate two functions:
void SetEmissionColor(float red, float green, float blue, float alpha = 1.0f);
void SetEmissionColor(float color[4]);

8.261.3.24 mitkSurfaceProperty::mitkSetColorMacro (SpecularColor, float)

A group of functions to set the specular color of the surface.
This macro will generate two functions:
void SetSpecularColor(float red, float green, float blue, float alpha = 1.0f);
void SetSpecularColor(float color[4]);

8.261.3.25 mitkSurfaceProperty::mitkSetColorMacro (DiffuseColor, float)

A group of functions to set the diffuse color of the surface.
This macro will generate two functions:

void SetDiffuseColor(float red, float green, float blue, float alpha = 1.0f);

void SetDiffuseColor(float color[4]);

8.261.3.26 mitkSurfaceProperty::mitkSetColorMacro (AmbientColor, float)

A group of functions to set the ambient color of the surface.

This macro will generate two functions:

void SetAmbientColor(float red, float green, float blue, float alpha = 1.0f);

void SetAmbientColor(float color[4]);

8.261.3.27 virtual void mitkSurfaceProperty::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

8.261.3.28 void mitkSurfaceProperty::SetAmbient (float *value*) [inline]

Set the ambient coefficient.

Parameters:

value the value of ambient coefficient

8.261.3.29 void mitkSurfaceProperty::SetColor (float *color*[4]) [inline]

Set the color of the surface. Has the side effect of setting the ambient, diffuse and specular colors as well. This is basically a quick overall color setting method.

Parameters:

color[0] the red value of the color

color[1] the green value of the color

color[2] the blue value of the color

color[3] the alpha value of the color (default is 1.0)

8.261.3.30 void mitkSurfaceProperty::SetColor (float *red*, float *green*, float *blue*, float *alpha* = 1.0f)

Set the color of the surface. Has the side effect of setting the ambient, diffuse and specular colors as well. This is basically a quick overall color setting method.

Parameters:

red the red value of the color

green the green value of the color

blue the blue value of the color

alpha the alpha value of the color (default is 1.0)

8.261.3.31 void mitkSurfaceProperty::SetDiffuse (float *value*) [inline]

Set the diffuse coefficient.

Parameters:
value the diffuse coefficient

8.261.3.32 void mitkSurfaceProperty::SetInterpolationType (int *intpType*) [inline]

Set the interpolation type for sampling a surface.

Parameters:
intpType interpolation type for sampling a surface, the value should be one of the follows:
MITK_SURFACE_FLAT (flat interpolation type)
MITK_SURFACE_GOURAUD (smooth interpolation type)
MITK_SURFACE_PHONG (smooth interpolation type)

8.261.3.33 void mitkSurfaceProperty::SetInterpolationTypeToFlat () [inline]

Set the interpolation type for sampling a surface to MITK_SURFACE_FLAT.

8.261.3.34 void mitkSurfaceProperty::SetInterpolationTypeToGouraud () [inline]

Set the interpolation type for sampling a surface to MITK_SURFACE_GOURAUD.

8.261.3.35 void mitkSurfaceProperty::SetInterpolationTypeToPhong () [inline]

Set the interpolation type for sampling a surface to MITK_SURFACE_PHONG.

8.261.3.36 void mitkSurfaceProperty::SetLineStipplePattern (unsigned short *value*) [inline]

Set the stippling pattern of a Line, as a 16-bit binary pattern (1 = pixel on, 0 = pixel off). This is only implemented for OpenGL. The default is 0xFFFF.

Parameters:
value a 16-bit binary pattern represent the stippling pattern of a Line (1 = pixel on, 0 = pixel off).

8.261.3.37 void mitkSurfaceProperty::SetLineStippleRepeatFactor (int *value*) [inline]

Set the stippling repeat factor of a Line, which specifies how many times each bit in the pattern is to be repeated. This is only implemented for OpenGL. The default is 1.

Parameters:
value the stippling repeat factor of a Line pecifies how many times each bit in the pattern is to be repeated

8.261.3.38 void mitkSurfaceProperty::SetLineWidth (float *value*) [inline]

Set the line width. The width is expressed in screen units. This is only implemented for OpenGL. The default is 1.0.

Parameters:
value the line width (default value is 1.0)

8.261.3.39 void mitkSurfaceProperty::SetOpacity (float *value*) [inline]

Set the object's opacity. 1.0 is totally opaque and 0.0 is completely transparent.

Parameters:
value the object's opacity

8.261.3.40 void mitkSurfaceProperty::SetPointSize (float *value*) [inline]

Set the point size. The size is expressed in screen units. This is only implemented for OpenGL. The default is 1.0.

Parameters:
value the point size (default value is 1.0)

8.261.3.41 void mitkSurfaceProperty::SetRepresentationType (int *repType*) [inline]

Set the geometry representation of the surface.

Parameters:
repType geometry representation of a surface, the value should be one of the follows:
MITK_MESH_POINTS (points representation)
MITK_MESH_WIREFRAME (wireframe representation)
MITK_MESH_SURFACE (surface representation)

8.261.3.42 void mitkSurfaceProperty::SetRepresentationTypeToPoints () [inline]

Set the geometry representation of the surface to MITK_MESH_POINTS.

8.261.3.43 void mitkSurfaceProperty::SetRepresentationTypeToSurface () [inline]

Set the geometry representation of the surface to MITK_MESH_SURFACE.

8.261.3.44 void mitkSurfaceProperty::SetRepresentationTypeToWireframe () [inline]

Set the geometry representation of the surface to MITK_MESH_WIREFRAME.

8.261.3.45 void mitkSurfaceProperty::SetSpecular (float *value*) [inline]

Set the specular coefficient.

Parameters:
value the specular coefficient

8.261.3.46 void mitkSurfaceProperty::SetSpecularPower (float *value*) [inline]

Set the specular power.

Parameters:
value the specular power (between 0.0 and 128.0)

8.261.3.47 void mitkSurfaceProperty::SetUnmodified () [inline]

Reset to unmodified after changes have been done according to the new properties.

The documentation for this class was generated from the following file:

- mitkSurfaceProperty.h

8.262 mitkSurfaceRenderer Class Reference

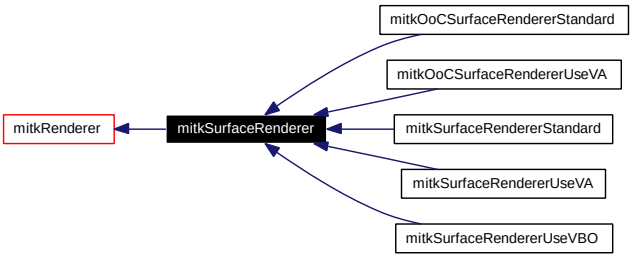
mitkSurfaceRenderer - an abstract class for surface renderer object

`#include <mitkSurfaceRenderer.h>`

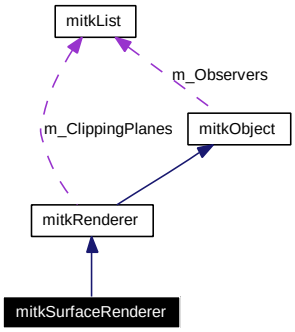
Inherits mitkRenderer.

Inherited by mitkOoCSurfaceRendererStandard, mitkOoCSurfaceRendererUseVA, mitkSurfaceRenderer-Standard, mitkSurfaceRendererUseVA, and mitkSurfaceRendererUseVBO.

Inheritance diagram for mitkSurfaceRenderer:



Collaboration diagram for mitkSurfaceRenderer:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.262.1 Detailed Description

mitkSurfaceRenderer - an abstract class for surface renderer object

mitkSurfaceRenderer is an abstract class for surface renderer object.

8.262.2 Member Function Documentation

8.262.2.1 virtual void mitkSurfaceRenderer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkRenderer.

Reimplemented in mitkOoCSurfaceRendererStandard, mitkOoCSurfaceRendererUseVA, mitkSurfaceRendererStandard, mitkSurfaceRendererUseVA, and mitkSurfaceRendererUseVBO.

The documentation for this class was generated from the following file:

- mitkSurfaceRenderer.h

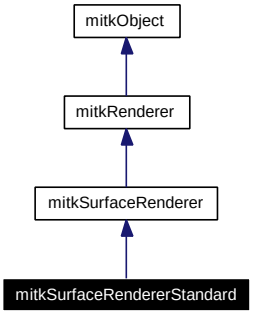
8.263 mitkSurfaceRendererStandard Class Reference

mitkSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

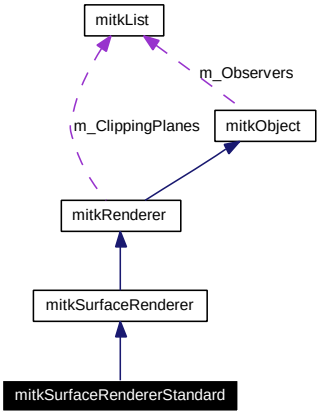
`#include <mitkSurfaceRendererStandard.h>`

Inherits mitkSurfaceRenderer.

Inheritance diagram for mitkSurfaceRendererStandard:



Collaboration diagram for mitkSurfaceRendererStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.263.1 Detailed Description

mitkSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

mitkSurfaceRendererStandard is a standard renderer object for mitkSurfaceModel using general OpenGL.

8.263.2 Member Function Documentation

8.263.2.1 virtual void mitkSurfaceRendererStandard::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkSurfaceRenderer.

The documentation for this class was generated from the following file:

- mitkSurfaceRendererStandard.h

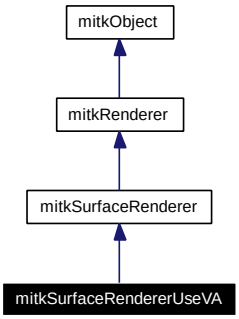
8.264 mitkSurfaceRendererUseVA Class Reference

mitkSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

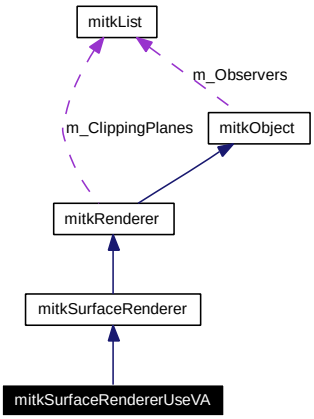
`#include <mitkSurfaceRendererUseVA.h>`

Inherits mitkSurfaceRenderer.

Inheritance diagram for mitkSurfaceRendererUseVA:



Collaboration diagram for mitkSurfaceRendererUseVA:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.264.1 Detailed Description

mitkSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

mitkSurfaceRendererUseVA is a renderer for mitkSurfaceModel using vertex array

8.264.2 Member Function Documentation

8.264.2.1 virtual void mitkSurfaceRendererUseVA::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkSurfaceRenderer.

The documentation for this class was generated from the following file:

- mitkSurfaceRendererUseVA.h

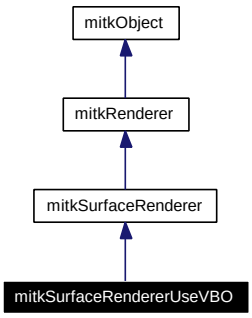
8.265 mitkSurfaceRendererUseVBO Class Reference

mitkSurfaceRendererUseVBO - a renderer for mitkSurfaceModel using vertex buffer object

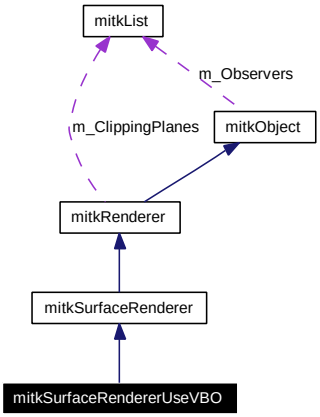
`#include <mitkSurfaceRendererUseVBO.h>`

Inherits mitkSurfaceRenderer.

Inheritance diagram for mitkSurfaceRendererUseVBO:



Collaboration diagram for mitkSurfaceRendererUseVBO:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.265.1 Detailed Description

mitkSurfaceRendererUseVBO - a renderer for mitkSurfaceModel using vertex buffer object

mitkSurfaceRendererUseVBO is a renderer for mitkSurfaceModel using OpenGL extension of vertex buffer object (VBO).

8.265.2 Member Function Documentation

8.265.2.1 virtual void mitkSurfaceRendererUseVBO::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkSurfaceRenderer.

The documentation for this class was generated from the following file:

- mitkSurfaceRendererUseVBO.h

8.266 mitkTarget Class Reference

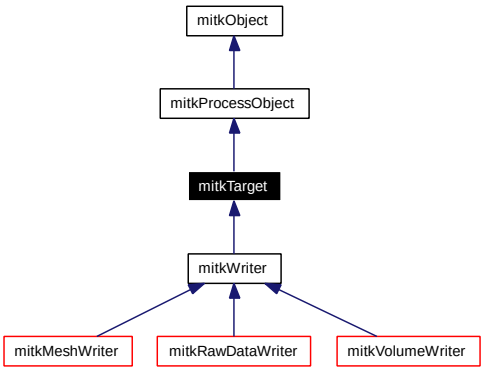
mitkTarget - abstract class specifies interface for Target object

`#include <mitkTarget.h>`

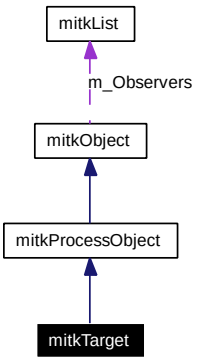
Inherits mitkProcessObject.

Inherited by mitkWriter.

Inheritance diagram for mitkTarget:



Collaboration diagram for mitkTarget:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.266.1 Detailed Description

mitkTarget - abstract class specifies interface for Target object

mitkTarget is an abstract object that specifies behavior and interface of mapper objects. Mapper object representas end point for a data processing pipeline, e.g. view or writer. It only accepts input data and either displays the input data to a view or writes the input data to a disk file.

8.266.2 Member Function Documentation

8.266.2.1 virtual void mitkTarget::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProcessObject.

Reimplemented in mitkAnalyzeImageWriter, mitkBMPWriter, mitkCacheVolumeWriter, mitkDICOMWriter, mitkJPEGWriter, mitkMeshWriter, mitkMetaImageWriter, mitkNetCDFImageWriter, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkRawFilesWriter, mitkRawWriter, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkTIFFWriter, mitkVolumeWriter, mitkWriter, mitkRawDataWriter, and mitkTrackedBscanDataWriter.

The documentation for this class was generated from the following file:

- mitkTarget.h

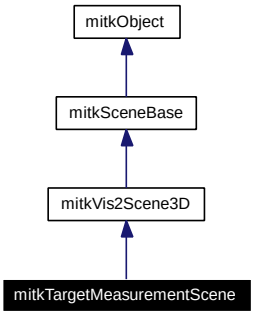
8.267 mitkTargetMeasurementScene Class Reference

mitkTargetMeasurementScene - a 3D scene to display a target region from a volume data, long/short axis are detected.

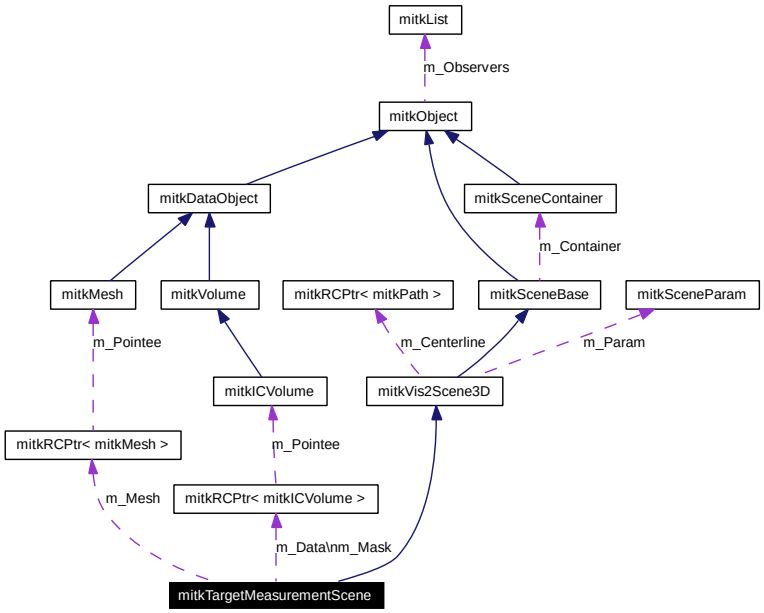
`#include <mitkTargetMeasurementScene.h>`

Inherits mitkVis2Scene3D.

Inheritance diagram for mitkTargetMeasurementScene:



Collaboration diagram for mitkTargetMeasurementScene:



Public Member Functions

- virtual void RenderGL ()
- void SetData (mitkICVolume *data, mitkICVolume *mask)
- void SetDataModified ()
- void SetBackGroundColor (float r, float g, float b)
- void GetBackGroundColor (float &r, float &g, float &b)

- void SetSurfaceColor (float r, float g, float b)
- void GetSurfaceColor (float &r, float &g, float &b)
- void SetLightIntensity (float intensity)
- void SetLightColor (float r, float g, float b)
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()

8.267.1 Detailed Description

mitkTargetMeasurementScene - a 3D scene to display a target region from a volume data, long/short axis are detected.

mitkTargetMeasurementScene is a 3D scene to display a target region from a volume data, long/short axis are detected.

8.267.2 Member Function Documentation

8.267.2.1 float mitkTargetMeasurementScene::GetAmbient ()

Get the ambient factor of the material of the isosurface

Returns:
Return The ambient factor of the material of the isosurface

8.267.2.2 void mitkTargetMeasurementScene::GetBackGroundColor (float & *r*, float & *g*, float & *b*)

Return the background color of the scene

Parameters:
r Return the red component of the background color.

g Return the green component of the background color.

b Return the blue component of the background color.

8.267.2.3 float mitkTargetMeasurementScene::GetDiffuse ()

Get the diffuse factor of the material of the isosurface

Returns:
Return The diffuse factor of the material of the isosurface

8.267.2.4 float mitkTargetMeasurementScene::GetSpecular ()

Get the specular factor of the material of the isosurface

Returns:
Return The specular factor of the material of the isosurface

8.267.2.5 float mitkTargetMeasurementScene::GetSpecularPower ()

Get the specular power of the material of the isosurface

Returns:
Return The specular power of the material of the isosurface

8.267.2.6 void mitkTargetMeasurementScene::GetSurfaceColor (float & *r*, float & *g*, float & *b*)

Get the color of the isosurface to be displayed

Parameters:
r Return the red component of the surface color.
g Return the green component of the surface color.
b Return the blue component of the surface color.

8.267.2.7 virtual void mitkTargetMeasurementScene::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkVis2Scene3D.

8.267.2.8 void mitkTargetMeasurementScene::SetAmbient (float *value*)

Set the ambient factor of the material of the isosurface

Parameters:
value The ambient factor of the material of the isosurface

8.267.2.9 void mitkTargetMeasurementScene::SetBackGroundColor (float *r*, float *g*, float *b*)

Set the background color of the scene

Parameters:
r Specify the red component of the background color.
g Specify the green component of the background color.
b Specify the blue component of the background color.

8.267.2.10 void mitkTargetMeasurementScene::SetData (mitkICVolume * *data*, mitkICVolume * *mask*)

Set the data object(ICVolume) to be rendered.

Parameters:
data The data object to be rendered.

8.267.2.11 void mitkTargetMeasurementScene::SetDataModified ()

Tell the scene the data is modified when any modification is done to the data object

8.267.2.12 void mitkTargetMeasurementScene::SetDiffuse (float *value*)

Set the diffuse factor of the material of the isosurface

Parameters:
value The diffuse factor of the material of the isosurface

8.267.2.13 void mitkTargetMeasurementScene::SetLightColor (float *r*, float *g*, float *b*)

Set the color of the light source

Parameters:
r Specify the red component of the light source color.
g Specify the green component of the light source color.
b Specify the blue component of the light source color.

8.267.2.14 void mitkTargetMeasurementScene::SetLightIntensity (float *intensity*)

Set the light's intensity

Parameters:
intensity Light's intensity

8.267.2.15 void mitkTargetMeasurementScene::SetSpecular (float *value*)

Set the specular factor of the material of the isosurface

Parameters:
value The specular factor of the material of the isosurface

8.267.2.16 void mitkTargetMeasurementScene::SetSpecularPower (float *value*)

Set the specular power of the material of the isosurface

Parameters:
value The specular power of the material of the isosurface

8.267.2.17 void mitkTargetMeasurementScene::SetSurfaceColor (float *r*, float *g*, float *b*)

Set the color of the isosurface to be displayed

Parameters:

- ***r*** Specify the red component of the surface color.
- ***g*** Specify the green component of the surface color.
- ***b*** Specify the blue component of the surface color.

The documentation for this class was generated from the following file:

- mitkTargetMeasurementScene.h

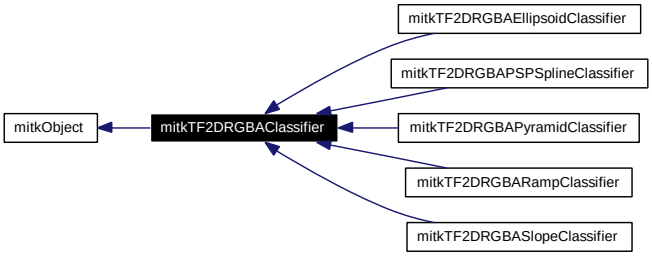
8.268 mitkTF2DRGBAClassifier Class Reference

`#include <mitkTF2DRGBAClassifier.h>`

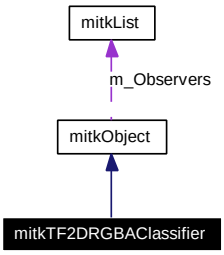
Inherits mitkObject.

Inherited by mitkTF2DRGBAEllipsoidClassifier, mitkTF2DRGBAPSPSplineClassifier, mitkTF2DRGBAPyramidClassifier, mitkTF2DRGBARampClassifier, and mitkTF2DRGBASlopeClassifier.

Inheritance diagram for mitkTF2DRGBAClassifier:



Collaboration diagram for mitkTF2DRGBAClassifier:



Public Member Functions

- void SetLeft (int left)
- void SetRight (int right)
- void SetBottom (int bottom)
- void SetTop (int top)
- void SetCenter (int centerX, int centerY)
- void Move (int deltaX, int deltaY)
- void SetMaxOpacity (float MaxOpacity)
- void SetMinOpacity (float MinOpacity)
- void SetColor (float r, float g, float b)
- int GetLeft () const
- int GetRight () const
- int GetBottom () const
- int GetTop () const
- void GetCenter (int &centerX, int &centerY) const
- float GetMaxOpacity () const
- float GetMinOpacity () const
- const float * GetColor () const

- void GetColor (float &r, float &g, float &b) const
- bool IsModified ()
- void SetUnmodified ()
- float * GetCurve ()
- void Update ()

8.268.1 Detailed Description

mitkTF2DRGBAClassifier - An Abstract class defining the interface of a classifier for 2D-RGBA-Transfer-Function.

mitkTF2DRGBAClassifier - is an abstract class defining the interface of a classifier for 2D-RGBA-TransferFunction. A 2D-RGBA-TransferFunction maps scalar values along with gredient values to RGBA values, which can be used to render a volume. The shape of a classifier is determined by its 4 boundaries and a center control point. The actual behaviour is defined in subclasses.

8.268.2 Member Function Documentation

8.268.2.1 int mitkTF2DRGBAClassifier::GetBottom () const [inline]

Get the bottom boundary (minimum gradient value) of the classifier

Returns:
Return the bottom boundary (minimum gradient value) of the classifier

8.268.2.2 void mitkTF2DRGBAClassifier::GetCenter (int & *centerX*, int & *centerY*) const [inline]

Set the coordinate of the center control point

Parameters:
centerX Recieve X coordinate(scalar value) of the center control point
centerY Recieve Y coordinate(scalar value) of the center control point

8.268.2.3 void mitkTF2DRGBAClassifier::GetColor (float & *r*, float & *g*, float & *b*) const [inline]

Get the color of the classifier

Parameters:
r Receive the red value of the classifier
g Receive the green value of the classifier
b Receive the blue value of the classifier

8.268.2.4 const float* mitkTF2DRGBAClassifier::GetColor () const [inline]

Get the color of the classifier

Returns:
Return the color of the classifier

8.268.2.5 float* mitkTF2DRGBAClassifier::GetCurve ()

Get curve values of the classifier, it is calculated by subclasses and is the range 0∼1 Call Update to apply the changes to parameters.

Returns:
Return opacity values of the classifier

8.268.2.6 int mitkTF2DRGBAClassifier::GetLeft () const [inline]

Get the left boundary (minimum scalar value) of the classifier

Returns:
Return the left boundary (minimum scalar value) of the classifier

8.268.2.7 float mitkTF2DRGBAClassifier::GetMaxOpacity () const [inline]

Get the maximum opacity of the classifier

Returns:
Return the maximum opacity of the classifier

8.268.2.8 float mitkTF2DRGBAClassifier::GetMinOpacity () const [inline]

Get the minimum opacity of the classifier

Returns:
Return the minimum opacity of the classifier

8.268.2.9 int mitkTF2DRGBAClassifier::GetRight () const [inline]

Get the right boundary (maximum scalar value) of the classifier

Returns:
Return the right boundary (maximum scalar value) of the classifier

8.268.2.10 int mitkTF2DRGBAClassifier::GetTop () const [inline]

Get the top boundary (maximum gradient value) of the classifier

Returns:
Return the top boundary (maximum gradient value) of the classifier

8.268.2.11 bool mitkTF2DRGBAClassifier::IsModified () [inline]

Get the modified status of the classifier, after last call to SetUnmodified, Call Update to apply the changes to parameters.

Returns:
Return the modified status of the classifier

8.268.2.12 void mitkTF2DRGBAClassifier::Move (int *deltaX*, int *deltaY*)

Quickly move the classifier without changing its shape

Parameters:

deltaX Specify shift along X

deltaY Specify shift along Y

8.268.2.13 void mitkTF2DRGBAClassifier::SetBottom (int *bottom*)

Set the bottom boundary (minimum gradient value) of the classifier

Parameters:

bottom Specify the bottom boundary (minimum gradient value) of the classifier

8.268.2.14 void mitkTF2DRGBAClassifier::SetCenter (int *centerX*, int *centerY*)

Set the coordinate of the center control point

Parameters:

centerX Specify X coordinate(scalar value) of the center control point

centerY Specify Y coordinate(scalar value) of the center control point

8.268.2.15 void mitkTF2DRGBAClassifier::SetColor (float *r*, float *g*, float *b*)

Set the color of the classifier, the default is 1,1,1

Parameters:

r Specify the red value of the classifier

g Specify the green value of the classifier

b Specify the blue value of the classifier

8.268.2.16 void mitkTF2DRGBAClassifier::SetLeft (int *left*)

Set the left boundary (minimum scalar value) of the classifier

Parameters:

left Specify the left boundary (minimum scalar value) of the classifier

8.268.2.17 void mitkTF2DRGBAClassifier::SetMaxOpacity (float *MaxOpacity*)

Set the maximum opacity of the classifier, the default is 1

Parameters:

MaxOpacity Specify the maximum opacity of the classifier.

8.268.2.18 void mitkTF2DRGBAClassifier::SetMinOpacity (float *MinOpacity*)

Set the minimum opacity of the classifier, the default is 0

Parameters:
MinOpacity Specify the minimum opacity of the classifier.

8.268.2.19 void mitkTF2DRGBAClassifier::SetRight (int *right*)

Set the right boundary (maximum scalar value) of the classifier

Parameters:
right Specify the right boundary (maximum scalar value) of the classifier

8.268.2.20 void mitkTF2DRGBAClassifier::SetTop (int *top*)

Set the top boundary (maximum gradient value) of the classifier

Parameters:
top Specify the top boundary (maximum gradient value) of the classifier

8.268.2.21 void mitkTF2DRGBAClassifier::SetUnmodified () `[inline]`

Set the modified status of the classifier to false.

8.268.2.22 void mitkTF2DRGBAClassifier::Update ()

Apply the changes to parameters. If any change is made, the modified status will be set to true

The documentation for this class was generated from the following file:

- mitkTF2DRGBAClassifier.h

8.269 mitkTF2DRGBAEllipsoidClassifier Class Reference

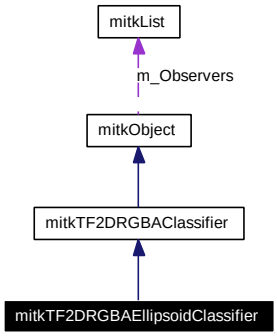
`#include <mitkTF2DRGBAEllipsoidClassifier.h>`

Inherits mitkTF2DRGBAClassifier.

Inheritance diagram for mitkTF2DRGBAEllipsoidClassifier:



Collaboration diagram for mitkTF2DRGBAEllipsoidClassifier:



8.269.1 Detailed Description

mitkTF2DRGBAEllipsoidClassifier - A concrete class implementing a half ellipsoid shaped classifier.

mitkTF2DRGBAEllipsoidClassifier is a concrete class implementing a half ellipsoid shaped classifier. The max-opacity point is located at the center point, and the ellipsoid is extended to the nearest boundaries of the classifier

The documentation for this class was generated from the following file:

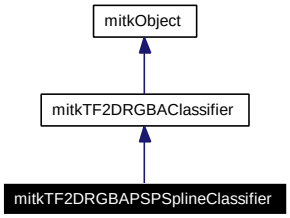
- mitkTF2DRGBAEllipsoidClassifier.h

8.270 mitkTF2DRGBAPSPSplineClassifier Class Reference

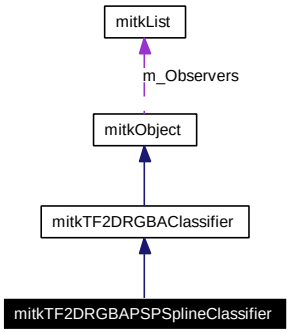
`#include <mitkTF2DRGBAPSPSplineClassifier.h>`

Inherits mitkTF2DRGBAClassifier.

Inheritance diagram for mitkTF2DRGBAPSPSplineClassifier:



Collaboration diagram for mitkTF2DRGBAPSPSplineClassifier:



Public Member Functions

- void SetSmoothnessDegree (int smoothnessdegree)
- void SetDelta (float delta)
- int GetSmoothnessDegree () const
- float GetDelta () const

8.270.1 Detailed Description

mitkTF2DRGBAPSPSplineClassifier - A concrete class implementing a half ellipsoid shaped classifier.

mitkTF2DRGBAPSPSplineClassifier is a concrete class implementing a partial shape preserving spline classifier. The max-opacity point is located at the center point.

8.270.2 Member Function Documentation

8.270.2.1 float mitkTF2DRGBAPSPSplineClassifier::GetDelta () const [inline]

Get the softness of the classifier

Returns:
Return the softness of the classifier

8.270.2.2 int mitkTF2DRGBAPSPSplineClassifier::GetSmoothnessDegree () const [inline]

Get the smoothness degree of the classifier

Returns:
Return the smoothness degree of the classifier

8.270.2.3 void mitkTF2DRGBAPSPSplineClassifier::SetDelta (float *delta*)

Set the softness of the classifier

Parameters:
delta a positive number used to specify the softness of the designed implicit polygon

8.270.2.4 void mitkTF2DRGBAPSPSplineClassifier::SetSmoothnessDegree (int *smoothnessdegree*)

Set the smoothness degree of the classifier

Parameters:
smoothnessdegree a positive integer used to specify the degree of smoothness

The documentation for this class was generated from the following file:

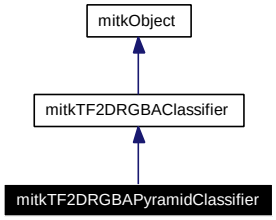
- mitkTF2DRGBAPSPSplineClassifier.h

8.271 mitkTF2DRGBAPyramidClassifier Class Reference

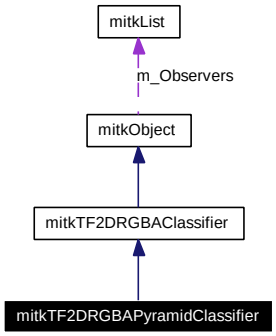
`#include <mitkTF2DRGBAPyramidClassifier.h>`

Inherits mitkTF2DRGBAClassifier.

Inheritance diagram for mitkTF2DRGBAPyramidClassifier:



Collaboration diagram for mitkTF2DRGBAPyramidClassifier:



8.271.1 Detailed Description

mitkTF2DRGBAPyramidClassifier - A concrete class implementing a pyramid shaped classifier.

mitkTF2DRGBAPyramidClassifier is a concrete class implementing a pyramid shaped classifier. The max-opacity point is located at the center point, and the pyramid is extended to the boundaries of the classifier

The documentation for this class was generated from the following file:

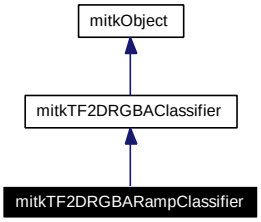
- mitkTF2DRGBAPyramidClassifier.h

8.272 mitkTF2DRGBARampClassifier Class Reference

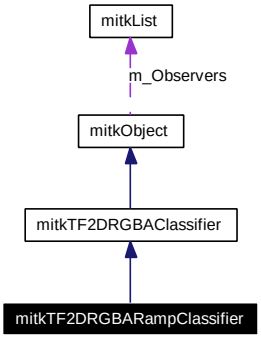
`#include <mitkTF2DRGBARampClassifier.h>`

Inherits mitkTF2DRGBAClassifier.

Inheritance diagram for mitkTF2DRGBARampClassifier:



Collaboration diagram for mitkTF2DRGBARampClassifier:



8.272.1 Detailed Description

mitkTF2DRGBARampClassifier - A concrete class implementing a ramp shaped classifier.

mitkTF2DRGBARampClassifier is a concrete class implementing a ramp shaped classifier. There is a vertical max-opacity line passing the center point and the ramp is extended to the left and right boundaries of the classifier

The documentation for this class was generated from the following file:

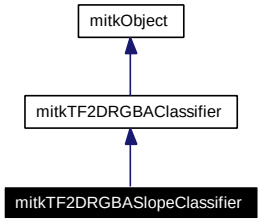
- mitkTF2DRGBARampClassifier.h

8.273 mitkTF2DRGBASlopeClassifier Class Reference

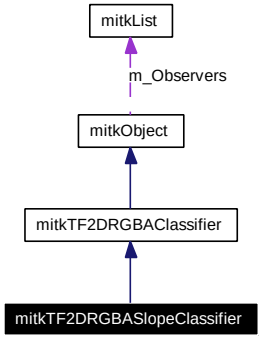
`#include <mitkTF2DRGBASlopeClassifier.h>`

Inherits mitkTF2DRGBAClassifier.

Inheritance diagram for mitkTF2DRGBASlopeClassifier:



Collaboration diagram for mitkTF2DRGBASlopeClassifier:



8.273.1 Detailed Description

mitkTF2DRGBASlopeClassifier - A concrete class implementing a slope shaped classifier.

mitkTF2DRGBASlopeClassifier is a concrete class implementing a slope shaped classifier. The top boundary is the max-opacity line,and the center point is passed by the mid-opacity line. The slope is extended to the bottom boundary of the classifier

The documentation for this class was generated from the following file:

- mitkTF2DRGBASlopeClassifier.h

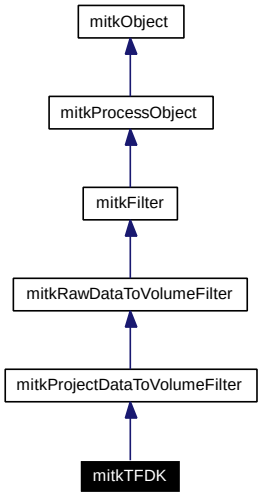
8.274 mitkTFDK Class Reference

mitkTFDK - concret class implement cone-beam reconstruction using TFDK Algorithm.

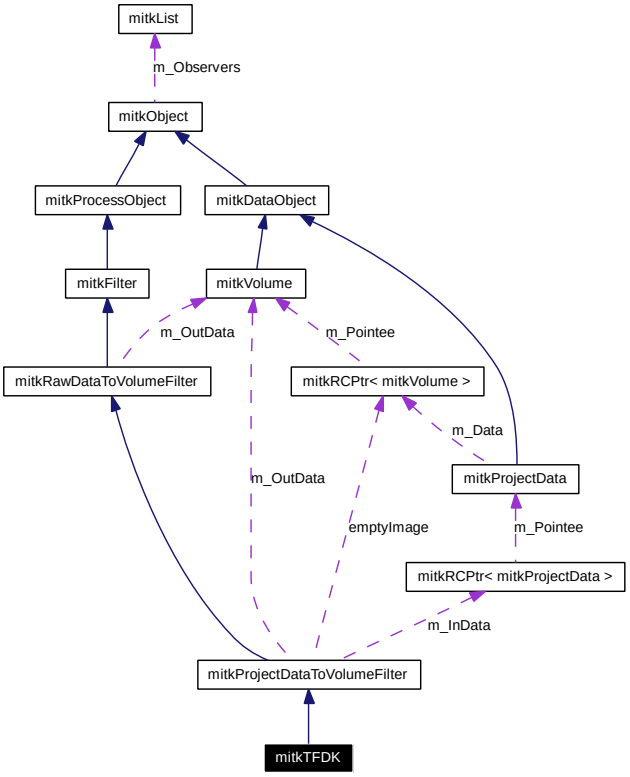
`#include <mitkTFDK.h>`

Inherits mitkProjectDataToVolumeFilter.

Inheritance diagram for mitkTFDK:



Collaboration diagram for mitkTFDK:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.274.1 Detailed Description

mitkTFDK - concret class implement cone-beam reconstruction using TFDK Algorithm.

mitkTFDK - concret class implement cone-beam reconstruction TFDK

8.274.2 Member Function Documentation

8.274.2.1 virtual void mitkTFDK::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkProjectDataToVolumeFilter.

The documentation for this class was generated from the following file:

- mitkTFDK.h

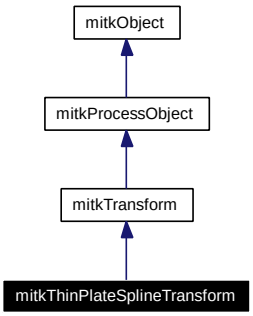
8.275 mitkThinPlateSplineTransform Class Reference

mitkThinPlateSplineTransform - a concrete transform to perform thin-plate spline transform

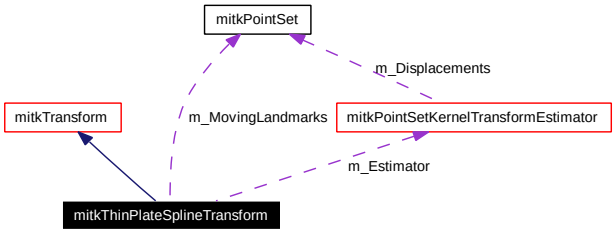
`#include <mitkThinPlateSplineTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkThinPlateSplineTransform:



Collaboration diagram for mitkThinPlateSplineTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkThinPlateSplineTransform ()
- mitkThinPlateSplineTransform (unsigned int imageDimension, int transformMode=MITK_-TRANSFORM_THINPLATESPLINE_R)
- const MatrixType & GetJacobian (VectorParameterType &inPoint)
- virtual void ConvertTransformMaxtrixToVector ()

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.275.1 Detailed Description

mitkThinPlateSplineTransform - a concrete transform to perform thin-plate spline transform

8.275.2 Constructor & Destructor Documentation

8.275.2.1 mitkThinPlateSplineTransform::mitkThinPlateSplineTransform ()

Constructor.

8.275.2.2 mitkThinPlateSplineTransform::mitkThinPlateSplineTransform (unsigned int *imageDimension*, int *transformMode* = MITK_TRANSFORM_THINPLATESPLINE_R)

Constructor with specific image dimensions setting.

8.275.3 Member Function Documentation

8.275.3.1 virtual void mitkThinPlateSplineTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.275.3.2 virtual void mitkThinPlateSplineTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.275.3.3 const MatrixType& mitkThinPlateSplineTransform::GetJacobian (VectorParameterType & *inPoint*)

Get the Jacobian matrix.

Parameters:
inPoint The index of the point in image.

Returns:
Return the pointer to the Jacobian matrix.

8.275.3.4 virtual void mitkThinPlateSplineTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkThinPlateSplineTransform.h

8.276 mitkThread Class Reference

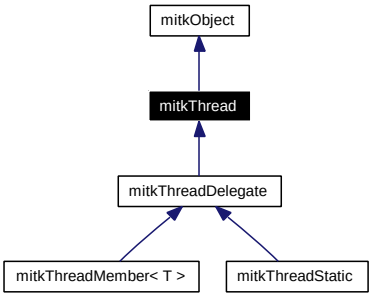
mitkThread - a class that encapsulates a thread

`#include <mitkThread.h>`

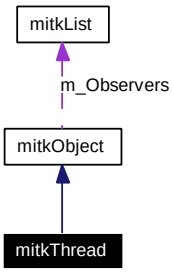
Inherits mitkObject.

Inherited by mitkThreadDelegate.

Inheritance diagram for mitkThread:



Collaboration diagram for mitkThread:



Public Member Functions

- void Spawn ()

8.276.1 Detailed Description

mitkThread - a class that encapsulates a thread

mitkThread encapsulates a thread.

8.276.2 Member Function Documentation

8.276.2.1 void mitkThread::Spawn () `[inline]`

Create a thread an start to execute

The documentation for this class was generated from the following file:

- mitkThread.h

8.277 mitkThreadDelegate Class Reference

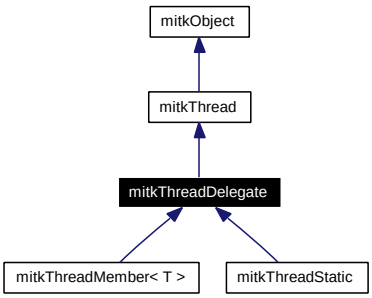
mitkThreadDelegate - The base class of mitkThreadStatic and mitkThreadMember,

`#include <mitkThread.h>`

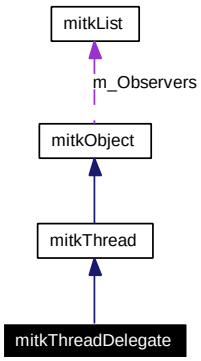
Inherits mitkThread.

Inherited by mitkThreadMember< T >, and mitkThreadStatic.

Inheritance diagram for mitkThreadDelegate:



Collaboration diagram for mitkThreadDelegate:



8.277.1 Detailed Description

mitkThreadDelegate - The base class of mitkThreadStatic and mitkThreadMember,

mitkThreadDelegate - The base class of mitkThreadStatic and mitkThreadMember, which redirect thread entrance to some user function. This class allows user to set a parameter to be passed to the user entrance.

The documentation for this class was generated from the following file:

- mitkThread.h

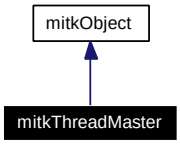
8.278 mitkThreadMaster Class Reference

mitkThreadMaster - encapsulates a master of the master/worker model

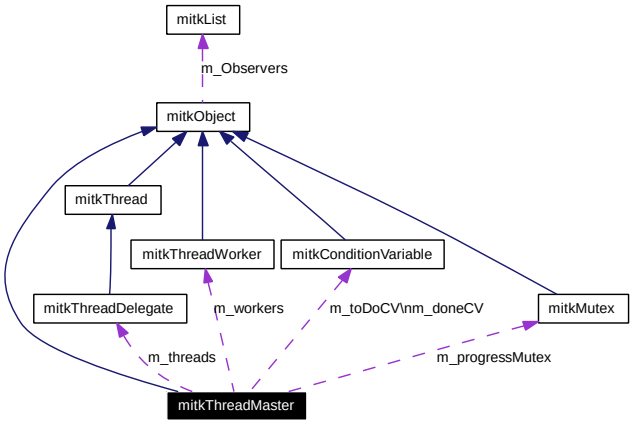
`#include <mitkThreadMasterWorker.h>`

Inherits mitkObject.

Inheritance diagram for mitkThreadMaster:



Collaboration diagram for mitkThreadMaster:



Public Member Functions

- int GetThreadNumber () const
- void DoTasks (unsigned long taskNumber, unsigned long startID=0, void *taskBag=NULL)

8.278.1 Detailed Description

mitkThreadMaster - encapsulates a master of the master/worker model

mitkThreadMaster encapsulates a master of the master/worker model It creates workers when it is initialized either by a "worker creator" or by duplicating an existing "sample worker". If workers are created by duplication, the "sample worker" itself will be referenced as one of the workers. And then, the master creates corresponding threads to drive the workers.

8.278.2 Member Function Documentation

8.278.2.1 void mitkThreadMaster::DoTasks (unsigned long *taskNumber*, unsigned long *startID* = 0, void * *taskBag* = NULL)

When this function is called, threads are waken up to drive the workers. Tasks are dynamically scheduled, which means a worker will receive a new task immediately after it finishes its current one. The function won't return until all scheduled tasks are done.

Parameters:

taskNumber Specify how many tasks to be scheduled

startID Specify where the ID of tasks starts from

taskBag Sent to workers to describe current group of tasks

8.278.2.2 int mitkThreadMaster::GetThreadNumber () const [inline]

Get the number of threads spawned.

Returns:

Return the number of threads spawned.

The documentation for this class was generated from the following file:

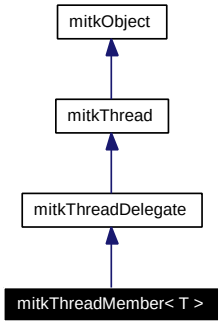
- mitkThreadMasterWorker.h

8.279 mitkThreadMember< T > Class Template Reference

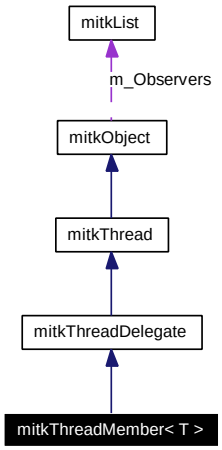
`#include <mitkThread.h>`

Inherits mitkThreadDelegate.

Inheritance diagram for mitkThreadMember< T >:



Collaboration diagram for mitkThreadMember< T >:



8.279.1 Detailed Description

template<class T> class mitkThreadMember< T >

mitkThreadMember - A utility allowing user to conveniently create a delegate thread object that redirects Run to a specified member function.

mitkThreadMember is a utility allowing user to conveniently create a delegate thread object that redirects Run to a specified member function.

The documentation for this class was generated from the following file:

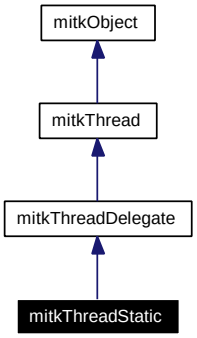
- mitkThread.h

8.280 mitkThreadStatic Class Reference

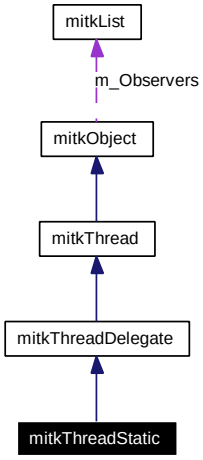
`#include <mitkThread.h>`

Inherits mitkThreadDelegate.

Inheritance diagram for mitkThreadStatic:



Collaboration diagram for mitkThreadStatic:



8.280.1 Detailed Description

mitkThreadStatic - A utility allowing user to conveniently create a delegate thread object that redirects Run to a specified static function or global function.

mitkThreadStatic is a utility allowing user to conveniently create a delegate thread object that redirects Run to a specified static function or global function.

The documentation for this class was generated from the following file:

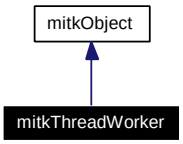
- mitkThread.h

8.281 mitkThreadWorker Class Reference

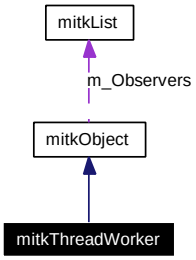
`#include <mitkThreadMasterWorker.h>`

Inherits mitkObject.

Inheritance diagram for mitkThreadWorker:



Collaboration diagram for mitkThreadWorker:



Protected Member Functions

- virtual int TasksPerTime ()
- virtual void SendTaskBag (void *taskBag)
- virtual void DoTask (int taskID, int taskNumber)=0
- virtual mitkThreadWorker * Duplicate ()=0

Friends

- class mitkThreadMaster

8.281.1 Detailed Description

mitkThreadWorker - encapsulates a worker of the Master/Worker model. A worker can be driven by a thread of the master to fulfil the scheduled tasks.

mitkThreadWorker encapsules a worker of the Master/Worker model. A worker can be driven by a thread of the master to fulfil the scheduled tasks. The class can be subclassed as a specific implementation of some parallel part of an algorithm.

8.281.2 Member Function Documentation

8.281.2.1 virtual void mitkThreadWorker::DoTask (int *taskID*, int *taskNumber*) [protected, pure virtual]

Do the scheduled task, called automatically by master

Parameters:

taskID Sent by master indicating the task (or the first of the tasks) that should be done.

taskNumber Sent by master indicating how many tasks actually should be done. It is no less than 1 and no more than the return value of TasksPerTime

8.281.2.2 virtual mitkThreadWorker* mitkThreadWorker::Duplicate () [protected, pure virtual]

Should be implemented. Used by master to duplicate workers from a single "sample worker".

8.281.2.3 virtual void mitkThreadWorker::SendTaskBag (void * *taskBag*) [inline, protected, virtual]

Called by the master when a group of tasks are sent to be done

Parameters:

taskBag Information of the group of tasks

8.281.2.4 virtual int mitkThreadWorker::TasksPerTime () [inline, protected, virtual]

Overwrite this function when more than one task will be done in do task function. The master use this function to decide the progress.

Returns:

Return the number of tasks done in each call to DoTask

The documentation for this class was generated from the following file:

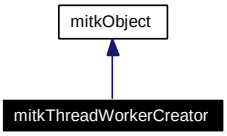
- mitkThreadMasterWorker.h

8.282 mitkThreadWorkerCreator Class Reference

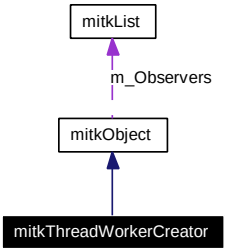
`#include <mitkThreadMasterWorker.h>`

Inherits mitkObject.

Inheritance diagram for mitkThreadWorkerCreator:



Collaboration diagram for mitkThreadWorkerCreator:



Public Member Functions

- virtual int GetWorkerNum ()
- virtual mitkThreadWorker * CreateThreadWorker (int WorkerID)=0

8.282.1 Detailed Description

mitkThreadWorkerCreator - An abstract class whose descendant can be used by master to create workers. Subclass it when different kind of workers should be used is calculation.

mitkThreadWorkerCreator is an abstract class whose descendant can be used by master to create workers. Subclass it when different kind of workers should be used is calculation.

8.282.2 Member Function Documentation

8.282.2.1 virtual mitkThreadWorker* mitkThreadWorkerCreator::CreateThreadWorker (int *WorkerID*) [pure virtual]

Called by master to create workers.

Parameters:

WorkerID id of the worker currently being created

Returns:

Return a pointer to the created worker

8.282.2.2 virtual int mitkThreadWorkerCreator::GetWorkerNum () [virtual]

Called by master to decided how many workers(threads) should be created

Returns:
Return the desired number of workers

The documentation for this class was generated from the following file:

- mitkThreadMasterWorker.h

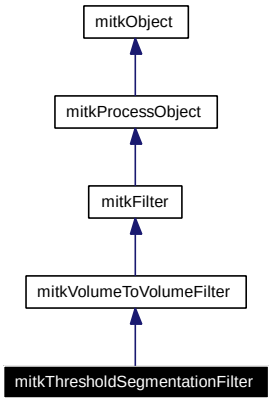
8.283 mitkThresholdSegmentationFilter Class Reference

mitkThresholdSegmentationFilter - a class implement threshold segmentation arithmetic

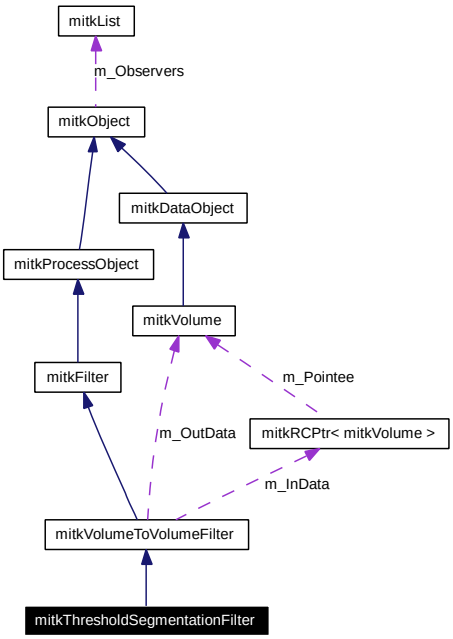
`#include <mitkThresholdSegmentationFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkThresholdSegmentationFilter:



Collaboration diagram for mitkThresholdSegmentationFilter:



Public Member Functions

- mitkThresholdSegmentationFilter ()
- virtual void PrintSelf (ostream &os)
- void SetLowThreshValue (double d)

- int GetLowThreshValue ()
- void SetHighThreshValue (double d)
- int GetHighThreshValue ()

8.283.1 Detailed Description

mitkThresholdSegmentationFilter - a class implement threshold segmentation arithmetic

mitkThresholdSegmentationFilter implement threshold segmentation arithmetic. The output is of the same datatype of the input.

8.283.2 Constructor & Destructor Documentation

8.283.2.1 mitkThresholdSegmentationFilter::mitkThresholdSegmentationFilter ()

Constructor of the class

8.283.3 Member Function Documentation

8.283.3.1 int mitkThresholdSegmentationFilter::GetHighThreshValue () [inline]

Get the high thresh value used by this class

Returns:
Return the high thresh value

8.283.3.2 int mitkThresholdSegmentationFilter::GetLowThreshValue () [inline]

Get the low thresh value used by this class

Returns:
Return the low thresh value

8.283.3.3 virtual void mitkThresholdSegmentationFilter::PrintSelf (ostream & *os*) [inline, virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.283.3.4 void mitkThresholdSegmentationFilter::SetHighThreshValue (double *d*) [inline]

Set the high thresh value. Pixels whose values are higher than it are set to zero.

Parameters:
d Represent the high thresh value

8.283.3.5 void mitkThresholdSegmentationFilter::SetLowThreshValue (double *d*) [inline]

Set the low thresh value. Pixels whose values are smaller than it are set to zero.

Parameters:

d Represent the low thresh value

The documentation for this class was generated from the following file:

- mitkThresholdSegmentationFilter.h

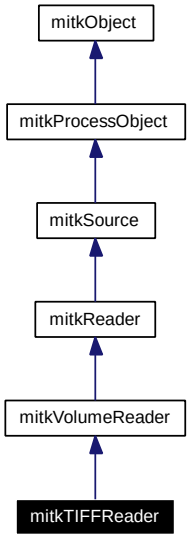
8.284 mitkTIFFReader Class Reference

mitkTIFFReader - a concrete reader for reading TIFF image files

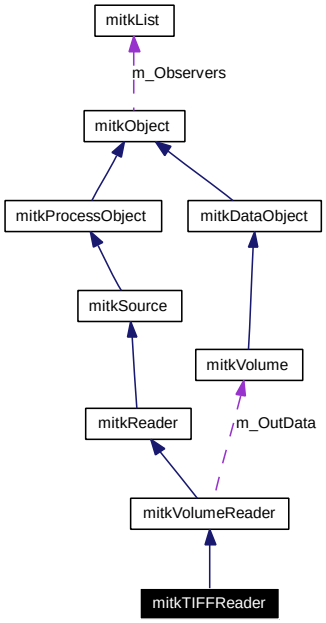
`#include <mitkTIFFReader.h>`

Inherits mitkVolumeReader.

Inheritance diagram for mitkTIFFReader:



Collaboration diagram for mitkTIFFReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetSpacingX (float px)
- void SetSpacingY (float py)
- void SetSpacingZ (float pz)
- void SetSpacings (float s[3])

8.284.1 Detailed Description

mitkTIFFReader - a concrete reader for reading TIFF image files

mitkTIFFReader reads a set of TIFF image files to a volume. Because TIFF file doesn't have the spacing information, you must set them using the SetSpacingX, SetSpacingY, SetSpacingZ functions. To use this reader, the code snippet is:

```
mitkTIFFReader *aReader = new mitkTIFFReader;
aReader->SetSpacingX(sx);
aReader->SetSpacingY(sy);
aReader->SetSpacingZ(sz);
aReader->AddFileName(file1);
aReader->AddFileName(file2);
...  ...  ...
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

Warning:
All of the images must have equal width and height. Otherwise they can't form a volume. Now MITK only supports single frame TIFF file.

8.284.2 Member Function Documentation

8.284.2.1 virtual void mitkTIFFReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeReader.

8.284.2.2 void mitkTIFFReader::SetSpacings (float *s*[3]) [inline]

Set spacing information in x, y, z axis, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.
py the spacing (mm) in two adjacent voxels in y axis.
pz the spacing (mm) in two adjacent voxels in z axis.

8.284.2.3 void mitkTIFFReader::SetSpacingX (float *px*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:
px the spacing (mm) in two adjacent voxels in x axis.

8.284.2.4 void mitkTIFFReader::SetSpacingY (float *py*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:
py the spacing (mm) in two adjacent voxels in y axis.

8.284.2.5 void mitkTIFFReader::SetSpacingZ (float *pz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:
pz the spacing (mm) in two adjacent voxels in z axis.

The documentation for this class was generated from the following file:

- mitkTIFFReader.h

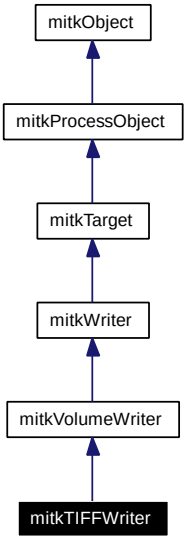
8.285 mitkTIFFWriter Class Reference

mitkTIFFWriter - a concrete writer for writing a volume to TIFF image files

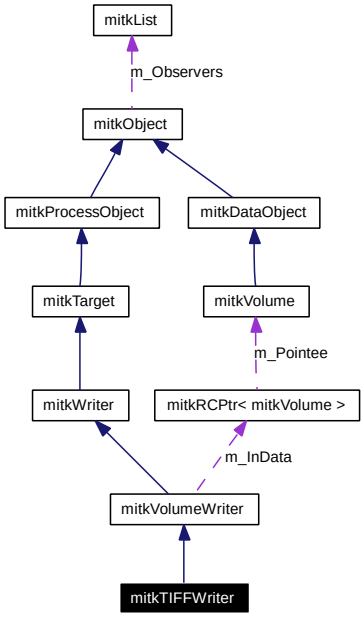
`#include <mitkTIFFWriter.h>`

Inherits mitkVolumeWriter.

Inheritance diagram for mitkTIFFWriter:



Collaboration diagram for mitkTIFFWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.285.1 Detailed Description

mitkTIFFWriter - a concrete writer for writing a volume to TIFF image files

mitkTIFFWriter writes a volume to a set of TIFF image files. Because the volume is a 3D dataset, it may contain many slices. So the file names must be generated and passed to writer properly. To use this writer, the code snippet is:

```
mitkTIFFWriter *aWriter = new mitkTIFFWriter;
aWriter->SetInput(aVolume);
int imageNum = aVolume->GetImageNum();
Gerenate file names into files[imageNum];
for(int i = 0; i < imageNum; i++)
    aWriter->AddFileName(files[i]);
aWriter->Run();
```

8.285.2 Member Function Documentation

8.285.2.1 virtual void mitkTIFFWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeWriter.

The documentation for this class was generated from the following file:

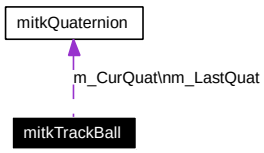
- mitkTIFFWriter.h

8.286 mitkTrackBall Class Reference

mitkTrackBall - a virtual trackball for tracking the movement of mouse

`#include <mitkTrackBall.h>`

Collaboration diagram for mitkTrackBall:



8.286.1 Detailed Description

mitkTrackBall - a virtual trackball for tracking the movement of mouse

mitkTrackBall is a virtual trackball for tracking the movement of mouse. Hacked into C++ from SGI's trackball.h & trackball.cpp —see copyright at end.

The documentation for this class was generated from the following file:

- mitkTrackBall.h

8.287 mitkTrackedBscanData Class Reference

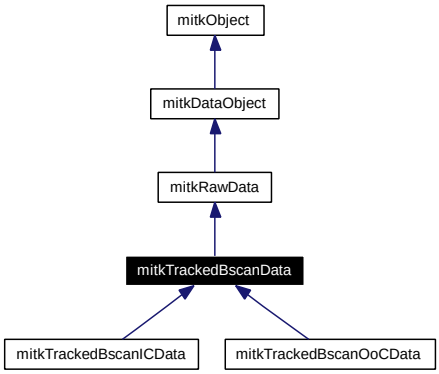
mitkTrackedBscanData - a concrete class encapsulating Raw Data.

`#include <mitkTrackedBscanData.h>`

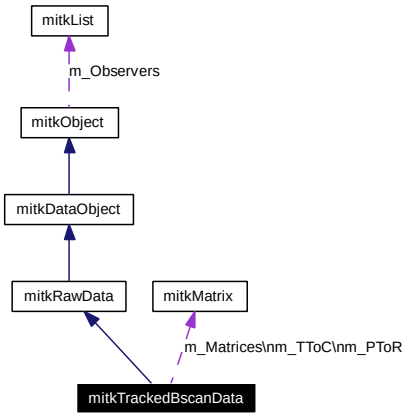
Inherits mitkRawData.

Inherited by mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

Inheritance diagram for mitkTrackedBscanData:



Collaboration diagram for mitkTrackedBscanData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()
- virtual int GetDataObjectType () const
- void SetImageWidth (int w)
- int GetImageWidth ()
- void SetImageHeight (int h)
- int GetImageHeight ()
- void SetNumberOfSlices (int s)
- int GetNumberOfSlices ()

- void GetVolumeDimensions (int dims[3])
- void SetVolumeDimensions (int dims[3])
- void GetPixelSpacings (float s[3])
- void SetPixelSpacings (float s[3])
- void GetVoxelSpacings (float s[3])
- void SetVoxelSpacings (float s[3])
- void SetNumberOfChannel (int n)
- int GetNumberOfChannel ()
- virtual void * GetData ()=0
- virtual void * GetSliceData (int sliceIdx)=0
- virtual bool WriteSliceData (int sliceIdx, void const *src)=0
- bool SetRToTMatrix (int sliceIdx, mitkMatrix const *ptor)
- mitkMatrix * GetRToTMatrix (int sliceIdx)
- mitkMatrix * GetRToTMatrix ()
- virtual bool Allocate ()=0
- virtual unsigned long long GetActualMemorySize () const =0
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0
- void SetPToRMatrix (mitkMatrix const *ptor)
- mitkMatrix GetPToRMatrix ()
- void SetTToCMatrix (mitkMatrix const *ttoc)
- mitkMatrix GetTToCMatrix ()
- void SetPredetermined (bool predet)
- bool IsPredetermined ()

8.287.1 Detailed Description

mitkTrackedBscanData - a concrete class encapsulating Raw Data.

mitkTrackedBscanData is a concrete class encapsulating Raw Data for 3D ultrasound Reconstruction. A mitkTrackedBscanData object include a group of B-Scan Images and the relative Matrices attached to the B-Scan Images. Concrete mitk3DUSReconstructionFilter classes reconstruct mitkVolume from mitkTrackedBscanData.

Note: If the mitkTrackedBscanData object was generated by a mitkTrackedBscanDataReader object, the parameters about the reconstruction volume (TTOC matrix, volume size and voxel size) were given. Then you need not set anything to the object, just use it. BUT, If you create and set the mitkTrackedBscanData object by yourself, please remember to compute the TTOC matrix, volume size and voxel size and set them to the mitkTrackedBscanData object, and use SetPredetermined(true) to notify the mitkTrackedBscanData object you have set these parameters. If you use SetPredetermined(false) after you have set these parameters, the mitk3DUSPixelBased object will automatically compute these parameters and use the automatically computed ones for volume reconstruction. If you really want the mitk3DUSPixelBased object to compute these parameters, just use SetPredetermined(false). If you forget to use SetPredetermined(), this filter will use the default status (false), that is, the filter will compute the parameters automatically.

8.287.2 Member Function Documentation

8.287.2.1 virtual bool mitkTrackedBscanData::Allocate () [pure virtual]

Allocate necessary space for holding the B-Scan data. It calculate the memory size using current Dimensions, DataType and NumberOfChannel settings. The equation is shown as follow:

Width ∗ Height ∗ SliceNum ∗ sizeof(DataType) ∗ NumberOfChannel

In addition, it calculates memory size for holding Matrices attached to B-Scans. The previous allocated data will be deleted if exists.

Returns:
If success, return the allocated memory address. Otherwise return NULL.

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the memory return by this function.

Implemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.2 virtual void mitkTrackedBscanData::DeepCopy (mitkDataObject ∗ *src*) [pure virtual]

Warning:
Haven't been implemented, don't call this function directly.

Implements mitkDataObject.

Implemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.3 virtual unsigned long long mitkTrackedBscanData::GetActualMemorySize () const [pure virtual]

Return the actual memory size occupied by the B-Scan data. The unit is BYTE.

Returns:
Return the actual memory size occupied by the B-Scan data. The unit is BYTE.

Implements mitkDataObject.

Implemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.4 virtual void∗ mitkTrackedBscanData::GetData () [pure virtual]

Get data pointer of the B-Scan image data (changeable).

Returns:
Return a void pointer to data.

Note:
The returned type is void ∗, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.5 virtual int mitkTrackedBscanData::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_VOLUME

Reimplemented from mitkRawData.

Reimplemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.6 int mitkTrackedBscanData::GetImageHeight () [inline]

Get the height of the Image (dimension in y). Every image have the same size.

Returns:
Return the height of this volume, the unit is pixel.

8.287.2.7 int mitkTrackedBscanData::GetImageWidth () [inline]

Get the width of the Image (dimension in x). Every image have the same size.

Returns:
Return the width of this volume, the unit is pixel.

8.287.2.8 int mitkTrackedBscanData::GetNumberOfChannel () [inline]

Get the channel number of the B-Scan image.

Returns:
Return the channel number of the B-Scan image.
return 1 — gray image
return 3 — RGB image
return 4 — RGBA image

Note:
Currently only support gray image.

8.287.2.9 int mitkTrackedBscanData::GetNumberOfSlices () [inline]

Get the number of the slices/images.

Returns:
Return the number of images/slices.

8.287.2.10 void mitkTrackedBscanData::GetPixelSpacings (float *s*[3]) [inline]

Get spacing information in x, y and z axis of the B-Scan image, the unit is mm.

Parameters:

s[0] Return the spacing (mm) in two adjacent pixels in x axis.

s[1] Return the spacing (mm) in two adjacent pixels in y axis.

s[2] Return the spacing (mm) in two adjacent pixels in z axis (thick of the B-Scan).

8.287.2.11 mitkMatrix mitkTrackedBscanData::GetPToRMatrix () [inline]

Get the calibrated transformation matrix from the B-Scan Plane coordinate system to the Receiver coordinate system.

Returns:

Return the value of the B-Scan Plane to the Receiver transformation matrix.

8.287.2.12 mitkMatrix* mitkTrackedBscanData::GetRToTMatrix ()

Get the pointer of the transformation matrix (from the Receiver coordinate system to the Transmitter coordinate system) from the TrackedBscanData.

Returns:

Return the pointer (the first) of the Receiver to the Transmitter transformation matrix.

8.287.2.13 mitkMatrix* mitkTrackedBscanData::GetRToTMatrix (int *sliceIdx*)

Get the pointer of the transformation matrix (from the Receiver coordinate system to the Transmitter coordinate system) from the TrackedBscanData.

Parameters:

sliceIdx the index of the slice to write (in [0, GetNumberOfSlices()-1])

Returns:

Return the pointer (the sliceIdx-th) of the Receiver to the Transmitter transformation matrix.

8.287.2.14 virtual void* mitkTrackedBscanData::GetSliceData (int *sliceIdx*) [pure virtual]

Copy slice data from the B-Scan image data to a specified memory buffer.

Parameters:

sliceIdx the index of the slice to get (in [0, GetNumberOfSlices()-1])

Returns:

Return a void pointer to data.

Note:

The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.15 mitkMatrix mitkTrackedBscanData::GetTToCMatrix () [inline]

Get the predetermined transformation matrix from the Transmitter coordinate system to the Reconstruction Volume coordinate system.

Returns:
Return the value of the Transmitter to the Reconstruction Volume transformation matrix.

8.287.2.16 void mitkTrackedBscanData::GetVolumeDimensions (int *dims*[3]) [inline]

Get the predetermined width, height and depth of the reconstructed volume.

Parameters:
dims[0] Return the predicted width of the reconstructed volume.
dims[1] Return the predicted height of the reconstructed volume.
dims[2] Return the predicted depth of the reconstructed volume.

8.287.2.17 void mitkTrackedBscanData::GetVoxelSpacings (float *s*[3]) [inline]

Get predetermined spacings in x, y and z axis of the reconstructed volume, the unit is mm.

Parameters:
s[0] Return the spacing (mm) in two adjacent voxels in x axis.
s[1] Return the spacing (mm) in two adjacent voxels in y axis.
s[2] Return the spacing (mm) in two adjacent voxels in z axis.

8.287.2.18 virtual void mitkTrackedBscanData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Implements mitkDataObject.

Reimplemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.19 bool mitkTrackedBscanData::IsPredetermined () [inline]

Get predetermination status of the reconstruction volume.

Returns:
Return the predetermination status.

8.287.2.20 virtual void mitkTrackedBscanData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRawData.

Reimplemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.21 void mitkTrackedBscanData::SetImageHeight (int *h*) [inline]

Set the height of the Image (dimension in y). Every image have the same size.

Parameters:
h The height of this volume, the unit is pixel.

8.287.2.22 void mitkTrackedBscanData::SetImageWidth (int *w*) [inline]

Set the width of the Image (dimension in x). Every image have the same size.

Parameters:
w The width of this volume, the unit is pixel.

8.287.2.23 void mitkTrackedBscanData::SetNumberOfChannel (int *n*) [inline]

Set the channel number of the B-Scan image.

Parameters:
n The channel number of the B-Scan image.
n = 1 — gray image
n = 3 — RGB image
n = 4 — RGBA image

Note:
Currently only support gray image.

8.287.2.24 void mitkTrackedBscanData::SetNumberOfSlices (int *s*)

Set the number of the slices/images.

Parameters:
s The number of images/slices.

8.287.2.25 void mitkTrackedBscanData::SetPixelSpacings (float *s*[3]) [inline]

Set spacing information in x, y and z axis of the B-Scan image, the unit is mm.

Parameters:

s[0] The spacing (mm) in two adjacent pixels in x axis.

s[1] The spacing (mm) in two adjacent pixels in y axis.

s[2] The spacing (mm) in two adjacent pixels in z axis (thick of the B-Scan).

8.287.2.26 void mitkTrackedBscanData::SetPredetermined (bool *predet*) [inline]

Set predetermination status of the reconstruction volume (default is false). The predetermination status.

8.287.2.27 void mitkTrackedBscanData::SetPToRMatrix (mitkMatrix const * *ptor*) [inline]

Set the calibrated transformation matrix from the B-Scan Plane coordinate system to the Receiver coordinate system. The pointer of the B-Scan Plane to the Receiver transformation matrix.

8.287.2.28 bool mitkTrackedBscanData::SetRToTMatrix (int *sliceIdx*, mitkMatrix const * *ptor*)

Copy the transformation matrix (from the Receiver coordinate system to the Transmitter coordinate system) from a specified mitkMatrix to the TrackedBscanData.

Parameters:

sliceIdx the index of the slice to write (in [0, GetNumberOfSlices()-1]) The pointer of the Receiver to the Transmitter transformation matrix.

8.287.2.29 void mitkTrackedBscanData::SetTToCMatrix (mitkMatrix const * *ttoc*) [inline]

Set the predetermined transformation matrix from the Transmitter coordinate system to the Reconstruction Volume coordinate system. The pointer of the Transmitter to the Reconstruction Volume transformation matrix.

8.287.2.30 void mitkTrackedBscanData::SetVolumeDimensions (int *dims*[3]) [inline]

Set the predetermined width, height and depth of the reconstructed volume.

Parameters:

dims[0] The predicted width of the reconstructed volume.

dims[1] The predicted height of the reconstructed volume.

dims[2] The predicted depth of the reconstructed volume.

8.287.2.31 void mitkTrackedBscanData::SetVoxelSpacings (float *s*[3]) [inline]

Set predetermined spacings in x, y and z axis of the reconstructed volume, the unit is mm.

Parameters:

s[0] The spacing (mm) in two adjacent voxels in x axis.

s[1] The spacing (mm) in two adjacent voxels in y axis.

s[2] The spacing (mm) in two adjacent voxels in z axis.

8.287.2.32 virtual void mitkTrackedBscanData::ShallowCopy (mitkDataObject * *src*) [pure virtual]

Warning:

Haven't been implemented, don't call this function directly.

Implements mitkDataObject.

Implemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

8.287.2.33 virtual bool mitkTrackedBscanData::WriteSliceData (int *sliceIdx*, void const * *src*) [pure virtual]

Copy slice data from a specified memory buffer to the B-Scan image data.

Parameters:

sliceIdx the index of the slice to write (in [0, GetNumberOfSlices()-1])

src the pointer to the source memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of source memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implemented in mitkTrackedBscanICData, and mitkTrackedBscanOoCData.

The documentation for this class was generated from the following file:

- mitkTrackedBscanData.h

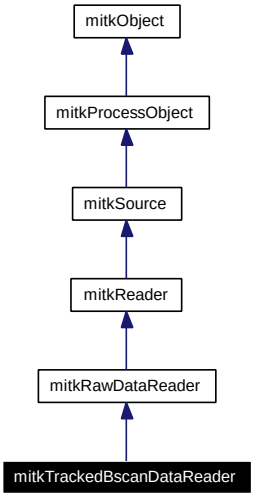
8.288 mitkTrackedBscanDataReader Class Reference

mitkTrackedBscanDataReader - A concrete class to read PAI files.

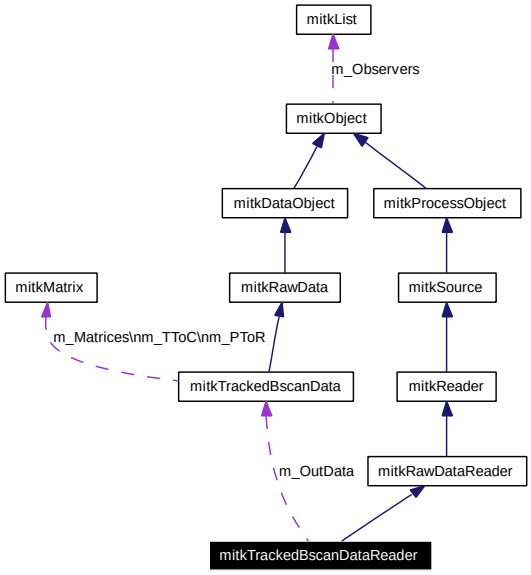
`#include <mitkTrackedBscanDataReader.h>`

Inherits mitkRawDataReader.

Inheritance diagram for mitkTrackedBscanDataReader:



Collaboration diagram for mitkTrackedBscanDataReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkTrackedBscanData * GetOutput ()

8.288.1 Detailed Description

mitkTrackedBscanDataReader - A concrete class to read PAI files.

mitkTrackedBscanDataReader is a concrete class to read PAI files to generate mitkTrackedBscanData object for 3D Ultrasound Reconstruction with concrete classes inheriting mitk3DUSReconstructionFilter. Raw Data acquired for 3D Ultrasound Reconstruction are stored as PAI (Positions And Images) files. A HPAI (Head for PAIs) file is stored together with the PAI files. The HPAI contains number of PAI files, B-Scan image size and spacings, calibrated PToR matrix, predetermined TToC matrix, predetermined volume size and volume spacings. Each PAI file contains the data of a B-scan image and the relative RToT matrix.

To use this reader, the code snippet is:

```
mitkTrackedBscanDataReader *aReader = new mitkTrackedBscanDataReader;
aReader->AddFileName(filename); //Only require the head file name "*.hpai"
if (aReader->Run())
{
    mitkTrackedBscanData *aTrackedBscanData = aReader->GetOutput();
    Using aTrackedBscanData...
}
```

Note: The HPAI and PAI are customized file formats defined by the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences. Please read the Manual or Examples to find the file formats.

8.288.2 Member Function Documentation

8.288.2.1 mitkTrackedBscanData* mitkTrackedBscanDataReader::GetOutput ()

Get the output the Tracked B-Scan Data the reader has read.

Returns:
the output Tracked B-Scan Data.

8.288.2.2 virtual void mitkTrackedBscanDataReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRawDataReader.

The documentation for this class was generated from the following file:

- mitkTrackedBscanDataReader.h

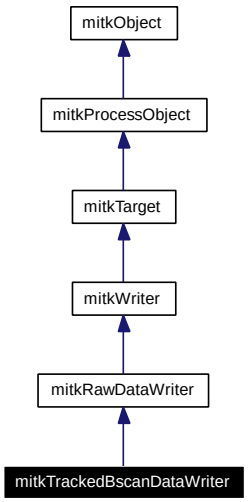
8.289 mitkTrackedBscanDataWriter Class Reference

mitkTrackedBscanDataWriter - A concrete class to write TrackedBscanData to PAI files.

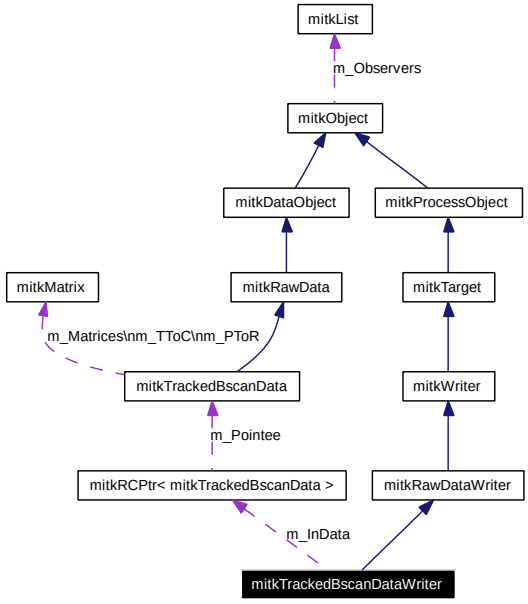
`#include <mitkTrackedBscanDataWriter.h>`

Inherits mitkRawDataWriter.

Inheritance diagram for mitkTrackedBscanDataWriter:



Collaboration diagram for mitkTrackedBscanDataWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- void SetInput (mitkTrackedBscanData *inData)
- mitkTrackedBscanData * GetInput ()

8.289.1 Detailed Description

mitkTrackedBscanDataWriter - A concrete class to write TrackedBscanData to PAI files.

mitkTrackedBscanDataWriter is a concrete class to write mitkTrackedBscanData object to PAI files. Raw Data acquired for 3D Ultrasound Reconstruction are stored as PAI (Positions And Images) files on the hard disk. A HPAI (Head for PAIs) file is stored together with the PAI files. The HPAI contains number of PAI files, B-Scan image size and spacings, calibrated PToR matrix, predetermined TToC matrix, predetermined volume size and volume spacings. Each PAI file contains the data of a B-scan image and the relative RToT matrix.

To use this writer, the code snippet is:

```
mitkTrackedBscanDataWriter *aWriter = new mitkTrackedBscanDataWriter;
aWriter->SetInput(TrackedBscanData);
aWriter->AddFileName(filename); //Only require the head file name "*.hpai"
aWriter->Run();
```

Note: The HPAI and PAI are customized file formats defined by the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences. Please read the Manual or Examples to find the file formats.

8.289.2 Member Function Documentation

8.289.2.1 mitkTrackedBscanData* mitkTrackedBscanDataWriter::GetInput () [inline]

Get input volume.

Returns:
Return the input volume

8.289.2.2 virtual void mitkTrackedBscanDataWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRawDataWriter.

8.289.2.3 void mitkTrackedBscanDataWriter::SetInput (mitkTrackedBscanData * *inData*) [inline]

Set input volume to write to disk file.

Parameters:
inData Input volume

The documentation for this class was generated from the following file:

- mitkTrackedBscanDataWriter.h

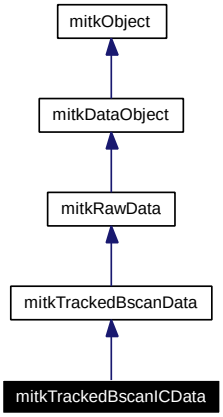
8.290 mitkTrackedBscanICData Class Reference

mitkTrackedBscanICData - a concrete class encapsulating Raw Data.

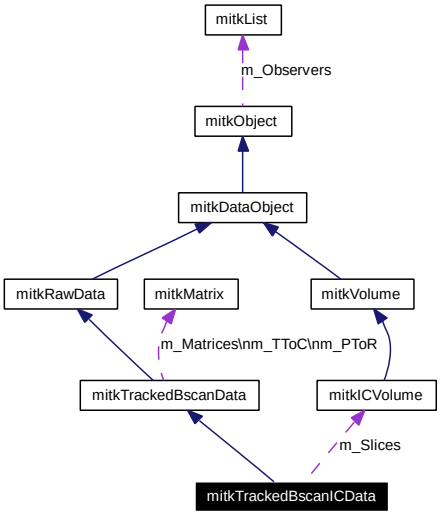
`#include <mitkTrackedBscanICData.h>`

Inherits mitkTrackedBscanData.

Inheritance diagram for mitkTrackedBscanICData:



Collaboration diagram for mitkTrackedBscanICData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()
- virtual int GetDataObjectType () const
- void * GetData ()
- void * GetSliceData (int sliceIdx)
- bool WriteSliceData (int sliceIdx, void const *src)

- bool Allocate ()
- virtual unsigned long long GetActualMemorySize () const
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)

8.290.1 Detailed Description

mitkTrackedBscanICData - a concrete class encapsulating Raw Data.

mitkTrackedBscanICData is a concrete class encapsulating Raw Data for 3D ultrasound Reconstruction. A mitkTrackedBscanICData object include a group of B-Scan Images and the relative Matrices attached to the B-Scan Images. Concrete mitk3DUSReconstructionFilter classes reconstruct mitkVolume from mitkTrackedBscanICData.

Note: If the mitkTrackedBscanICData object was generated by a mitkTrackedBscanICDataReader object, the parameters about the reconstruction volume (TTOC matrix, volume size and voxel size) were given. Then you need not set anything to the object, just use it. BUT, If you create and set the mitkTrackedBscanICData object by yourself, please remember to compute the TTOC matrix, volume size and voxel size and set them to the mitkTrackedBscanICData object, and use SetPredetermined(true) to notify the mitkTrackedBscanICData object you have set these parameters. If you use SetPredetermined(false) after you have set these parameters, the mitk3DUSPixelBased object will automatically compute these parameters and use the automatically computed ones for volume reconstruction. If you really want the mitk3DUSPixelBased object to compute these parameters, just use SetPredetermined(false). If you forget to use SetPredetermined(), this filter will use the default status (false), that is, the filter will compute the parameters automatically.

8.290.2 Member Function Documentation

8.290.2.1 bool mitkTrackedBscanICData::Allocate () [virtual]

Allocate necessary space for holding the B-Scan data. It calculate the memory size using current Dimensions, DataType and NumberOfChannel settings. The equation is shown as follow:

Width * Height * SliceNum * sizeof(DataType) * NumberOfChannel

In addition, it calculates memory size for holding Matrices attached to B-Scans. The previous allocated data will be deleted if exists.

Returns:
If success, return the allocated memory address. Otherwise return NULL.

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the memory return by this function.

Implements mitkTrackedBscanData.

8.290.2.2 virtual void mitkTrackedBscanICData::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Haven't been implemented, don't call this function directly.

Implements mitkTrackedBscanData.

8.290.2.3 virtual unsigned long long mitkTrackedBscanICData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by the B-Scan data. The unit is BYTE.

Returns:
Return the actual memory size occupied by the B-Scan data. The unit is BYTE.

Implements mitkTrackedBscanData.

8.290.2.4 void* mitkTrackedBscanICData::GetData () [virtual]

Get data pointer of the B-Scan image data (changeable).

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkTrackedBscanData.

8.290.2.5 virtual int mitkTrackedBscanICData::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_VOLUME

Reimplemented from mitkTrackedBscanData.

8.290.2.6 void* mitkTrackedBscanICData::GetSliceData (int *sliceIdx*) [virtual]

Copy slice data from the B-Scan image data to a specified memory buffer.

Parameters:
sliceIdx the index of the slice to get (in [0, GetNumberOfSlices()-1])

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkTrackedBscanData.

8.290.2.7 virtual void mitkTrackedBscanICData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Reimplemented from mitkTrackedBscanData.

8.290.2.8 virtual void mitkTrackedBscanICData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTrackedBscanData.

8.290.2.9 virtual void mitkTrackedBscanICData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Haven't been implemented, don't call this function directly.

Implements mitkTrackedBscanData.

8.290.2.10 bool mitkTrackedBscanICData::WriteSliceData (int *sliceIdx*, void const * *src*) [virtual]

Copy slice data from a specified memory buffer to the B-Scan image data.

Parameters:
sliceIdx the index of the slice to write (in [0, GetNumberOfSlices()-1])

src the pointer to the source memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of source memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implements mitkTrackedBscanData.

The documentation for this class was generated from the following file:

- mitkTrackedBscanICData.h

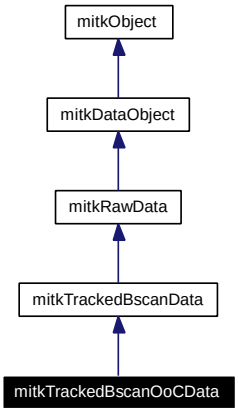
8.291 mitkTrackedBscanOoCData Class Reference

mitkTrackedBscanOoCData - a concrete class encapsulating Raw Data.

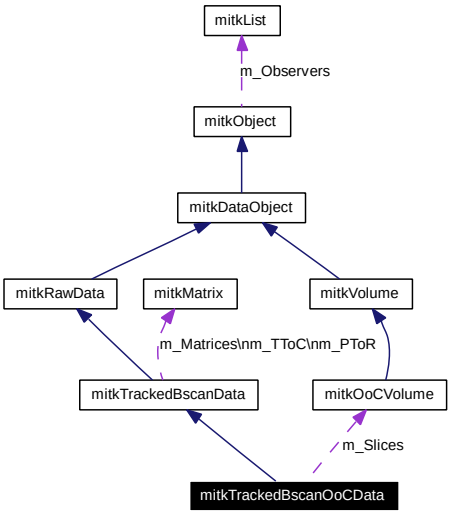
`#include <mitkTrackedBscanOoCData.h>`

Inherits mitkTrackedBscanData.

Inheritance diagram for mitkTrackedBscanOoCData:



Collaboration diagram for mitkTrackedBscanOoCData:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()
- virtual int GetDataObjectType () const
- void * GetData ()
- void * GetSliceData (int sliceIdx)
- bool WriteSliceData (int sliceIdx, void const *src)

- bool Allocate ()
- virtual unsigned long long GetActualMemorySize () const
- void SetBufferedSliceNum (unsigned int n)
- unsigned int GetBufferedSliceNum ()
- void SetPathOfDiskBuffer (char const *path)
- char const * GetPathOfDiskBuffer ()
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)

8.291.1 Detailed Description

mitkTrackedBscanOoCData - a concrete class encapsulating Raw Data.

mitkTrackedBscanOoCData is a concrete class encapsulating Raw Data for 3D ultrasound Reconstruction. A mitkTrackedBscanOoCData object include a group of B-Scan Images and the relative Matrices attached to the B-Scan Images. Concrete mitk3DUSReconstructionFilter classes reconstruct mitkVolume from mitkTrackedBscanOoCData.

Note: If the mitkTrackedBscanOoCData object was generated by a mitkTrackedBscanOoCDataReader object, the parameters about the reconstruction volume (TTOC matrix, volume size and voxel size) were given. Then you need not set anything to the object, just use it. BUT, If you create and set the mitkTrackedBscanOoCData object by yourself, please remember to compute the TTOC matrix, volume size and voxel size and set them to the mitkTrackedBscanOoCData object, and use SetPredetermined(true) to notify the mitkTrackedBscanOoCData object you have set these parameters. If you use SetPredetermined(false) after you have set these parameters, the mitk3DUSPixelBased object will automatically compute these parameters and use the automatically computed ones for volume reconstruction. If you really want the mitk3DUSPixelBased object to compute these parameters, just use SetPredetermined(false). If you forget to use SetPredetermined(), this filter will use the default status (false), that is, the filter will compute the parameters automatically.

8.291.2 Member Function Documentation

8.291.2.1 bool mitkTrackedBscanOoCData::Allocate () [virtual]

Allocate necessary space for holding the B-Scan data. It calculate the memory size using current Dimensions, DataType and NumberOfChannel settings. The equation is shown as follow:

Width * Height * SliceNum * sizeof(DataType) * NumberOfChannel

In addition, it calculates memory size for holding Matrices attached to B-Scans. The previous allocated data will be deleted if exists.

Returns:
If success, return the allocated memory address. Otherwise return NULL.

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the memory return by this function.

Implements mitkTrackedBscanData.

8.291.2.2 virtual void mitkTrackedBscanOoCData::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Haven't been implemented, don't call this function directly.

Implements mitkTrackedBscanData.

8.291.2.3 virtual unsigned long long mitkTrackedBscanOoCData::GetActualMemorySize () const [virtual]

Return the actual memory size occupied by the B-Scan data. The unit is BYTE.

Returns:
Return the actual memory size occupied by the B-Scan data. The unit is BYTE.

Implements mitkTrackedBscanData.

8.291.2.4 unsigned int mitkTrackedBscanOoCData::GetBufferedSliceNum () [inline]

Get the number of buffered slice in memory.

Returns:
Return the number of buffered slice.

8.291.2.5 void* mitkTrackedBscanOoCData::GetData () [virtual]

Get data pointer of the B-Scan image data (changeable).

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkTrackedBscanData.

8.291.2.6 virtual int mitkTrackedBscanOoCData::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_VOLUME

Reimplemented from mitkTrackedBscanData.

8.291.2.7 char const* mitkTrackedBscanOoCData::GetPathOfDiskBuffer ()

Get the full path of the disk buffer which contains all the slices of the volume.

Returns:
Return the full path of the disk buffer.

8.291.2.8 void* mitkTrackedBscanOoCData::GetSliceData (int *sliceIdx*) [virtual]

Copy slice data from the B-Scan image data to a specified memory buffer.

Parameters:
sliceIdx the index of the slice to get (in [0, GetNumberOfSlices()-1])

Returns:
Return a void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implements mitkTrackedBscanData.

8.291.2.9 virtual void mitkTrackedBscanOoCData::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Reimplemented from mitkTrackedBscanData.

8.291.2.10 virtual void mitkTrackedBscanOoCData::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTrackedBscanData.

8.291.2.11 void mitkTrackedBscanOoCData::SetBufferedSliceNum (unsigned int *n*) [inline]

Set the number of buffered slice in memory.

Parameters:
n the number of buffered slice

Note:
When calling this function, the buffer size is also calculated by (slice size) * (slice number), so it is unnecessary to call SetMemoryBufferSize() again.

8.291.2.12 void mitkTrackedBscanOoCData::SetPathOfDiskBuffer (char const * *path*)

Set the full path of the disk buffer to contain all the slices of the volume.

Parameters:
path the full path of the disk buffer

8.291.2.13 virtual void mitkTrackedBscanOoCData::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Haven't been implemented, don't call this function directly.

Implements mitkTrackedBscanData.

8.291.2.14 bool mitkTrackedBscanOoCData::WriteSliceData (int *sliceIdx*, void const * *src*) [virtual]

Copy slice data from a specified memory buffer to the B-Scan image data.

Parameters:
sliceIdx the index of the slice to write (in [0, GetNumberOfSlices()-1])

src the pointer to the source memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of source memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implements mitkTrackedBscanData.

The documentation for this class was generated from the following file:

- mitkTrackedBscanOoCData.h

8.292 mitkTransferFunction Class Reference

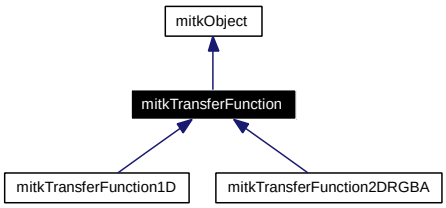
mitkTransferFunction - an abstract class to map the data property to opacity

`#include <mitkTransferFunction.h>`

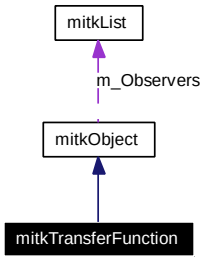
Inherits mitkObject.

Inherited by mitkTransferFunction1D, and mitkTransferFunction2DRGBA.

Inheritance diagram for mitkTransferFunction:



Collaboration diagram for mitkTransferFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- float * GetData ()
- virtual int GetDimension ()=0
- bool IsModified () const
- void SetUnmodified ()

8.292.1 Detailed Description

mitkTransferFunction - an abstract class to map the data property to opacity

mitkTransferFunction is an abstract transfer function to map the data property, including scalar value, gradient value, etc., to opacity. MITK supports multi-dimensional transfer function by subclassing it.

See also:
mitkTransferFunction1D

8.292.2 Member Function Documentation

8.292.2.1 float* mitkTransferFunction::GetData (void) [inline]

Get the address of the opacity in the transfer function.

Returns:
Return a pointer to the address of the opacity in the transfer function. It may be a multi-dimensiona array, which depends on the dimensionality of this transfer function.

8.292.2.2 virtual int mitkTransferFunction::GetDimension () [pure virtual]

Get the dimensionality of this transfer function.

Note:
Pure virtual function. Its concrete subclass must implement this function

Implemented in mitkTransferFunction1D, and mitkTransferFunction2DRGBA.

8.292.2.3 bool mitkTransferFunction::IsModified () const [inline]

Test if some of the transfer function is modified.

Returns:
Return true if some of the properties are modified. Otherwise return false.

8.292.2.4 virtual void mitkTransferFunction::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkTransferFunction1D.

8.292.2.5 void mitkTransferFunction::SetUnmodified () [inline]

Reset to unmodified after changes have been done according to the new transfer function.

Reimplemented in mitkTransferFunction2DRGBA.

The documentation for this class was generated from the following file:

- mitkTransferFunction.h

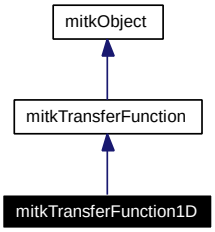
8.293 mitkTransferFunction1D Class Reference

mitkTransferFunction1D - a concrete 1D transfer function to map the data property to opacity

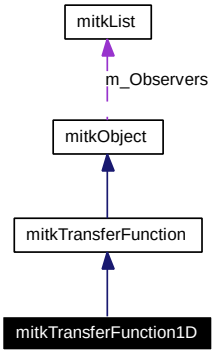
`#include <mitkTransferFunction1D.h>`

Inherits mitkTransferFunction.

Inheritance diagram for mitkTransferFunction1D:



Collaboration diagram for mitkTransferFunction1D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddPoint (int x, float val)
- void RemovePoint (int x)
- void RemoveAllPoints ()
- int GetPointsCount ()
- int GetDataValueAtPoint (int pointIndex)
- float GetOpacityAtPoint (int pointIndex)
- void SetMax (int xMax, float valMax)
- int GetMaxX ()
- float GetValue (int x)
- virtual int GetDimension ()

8.293.1 Detailed Description

mitkTransferFunction1D - a concrete 1D transfer function to map the data property to opacity

mitkTransferFunction1D is a 1D transfer function to map the data property, including scalar value, gradient value, etc., to opacity. The data property value has a range. The minimum value is always 0, the maximum value can be set by calling the function SetMax(). In general, you firstly get the maximum value from a volume, and then pass it to mitkTransferFunction1D.

8.293.2 Member Function Documentation

8.293.2.1 void mitkTransferFunction1D::AddPoint (int *x*, float *val*)

Add a point to this transfer function

Parameters:

x Specify the data value, such as scalar value, gradient value

val Specify the corresponding opacity

8.293.2.2 int mitkTransferFunction1D::GetDataValueAtPoint (int *pointIndex*)

Get the data value at pointIndex point

Parameters:

pointIndex Specify zero-based point index

Returns:

Return the data value at the specified point

8.293.2.3 virtual int mitkTransferFunction1D::GetDimension () [inline, virtual]

Get the dimensionality of this transfer function.

Returns:

Always return 1

Implements mitkTransferFunction.

8.293.2.4 int mitkTransferFunction1D::GetMaxX () [inline]

Get the maximum data value of this transfer function.

Returns:

Return the maximum data value of this transfer function.

8.293.2.5 float mitkTransferFunction1D::GetOpacityAtPoint (int *pointIndex*)

Get the opacity at pointIndex point

Parameters:

pointIndex Specify zero-based point index

Returns:

Return the opacity at the specified point

8.293.2.6 int mitkTransferFunction1D::GetPointsCount ()

Get the number of points in this transfer function

Returns:
Return the points number

8.293.2.7 float mitkTransferFunction1D::GetValue (int *x*)

Get the opacity corresponding to the specified data value.

Parameters:
x The specified data value.

Returns:
Return the opacity corresponding to the data value x.

8.293.2.8 virtual void mitkTransferFunction1D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTransferFunction.

8.293.2.9 void mitkTransferFunction1D::RemoveAllPoints ()

Remove all points from this transfer function

8.293.2.10 void mitkTransferFunction1D::RemovePoint (int *x*)

Remove a point from this transfer function

Parameters:
x Specify the data value, such as scalar value, gradient value. If the transfer function has a point which data value is equal to x, then this point will be removed. Otherwise nothing happens.

8.293.2.11 void mitkTransferFunction1D::SetMax (int *xMax*, float *valMax*)

Set the maximum data value and its corresponding opacity.

Parameters:
xMax Specify the maximum data value. The data value range of this transfer function is from 0 to xMax.

valMax Specify the corresponding opacity

Note:

This function will call RemoveAllPoints(), so it will clear the previous transfer function. After the calling of this function, the transfer function has two points. One is (0, 0.0), and the other is (xMax, valMax)

The documentation for this class was generated from the following file:

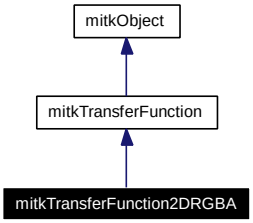
- mitkTransferFunction1D.h

8.294 mitkTransferFunction2DRGBA Class Reference

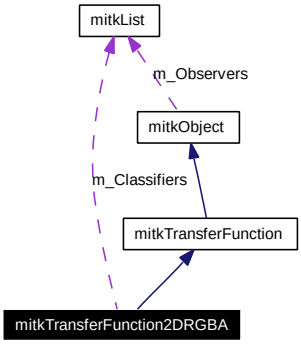
`#include <mitkTransferFunction2DRGBA.h>`

Inherits mitkTransferFunction.

Inheritance diagram for mitkTransferFunction2DRGBA:



Collaboration diagram for mitkTransferFunction2DRGBA:



Public Member Functions

- virtual int GetDimension ()
- float * GetRed ()
- float * GetGreen ()
- float * GetBlue ()
- float * GetAlpha ()
- void SetMax (int maxX, int maxY)
- int GetMaxX ()
- int GetMaxY ()
- bool AddClassifier (mitkTF2DRGBAClassifier *Classifier)
- bool RemoveClassifier (mitkTF2DRGBAClassifier *Classifier)
- bool RemoveAllClassifiers ()
- bool IsModified ()
- void SetUnmodified ()
- void Update ()

8.294.1 Detailed Description

mitkTransferFunction2DRGBA - a concrete 2D transfer function to map the data property (value and gredient) to opacity and color

mitkTransferFunction2DRGBA is a 2D transfer function to map the data property, including scalar value, gradient value, etc., to opacity and color. The transfer function is generated by combining classifiers that added to it.

8.294.2 Member Function Documentation

8.294.2.1 bool mitkTransferFunction2DRGBA::AddClassifier (mitkTF2DRGBAClassifier * *Classifier*)

Add a classifier to the transfer function

Parameters:
Classifier Specify the classifier to be added to the transfer function

Returns:
Return true if the addition is successful

8.294.2.2 float* mitkTransferFunction2DRGBA::GetAlpha ()

Get alpha channel values of the transfer function\

Returns:
Return alpha channel values of the transfer function

8.294.2.3 float* mitkTransferFunction2DRGBA::GetBlue ()

Get blue channel values of the transfer function

Returns:
Return blue channel values of the transfer function

8.294.2.4 virtual int mitkTransferFunction2DRGBA::GetDimension () [inline, virtual]

Get the dimensionality of this transfer function.

Returns:
always return 2

Implements mitkTransferFunction.

8.294.2.5 float* mitkTransferFunction2DRGBA::GetGreen ()

Get green channel values of the transfer function

Returns:
Return green channel values of the transfer function

8.294.2.6 int mitkTransferFunction2DRGBA::GetMaxX () [inline]

Get Max value

Returns:
Return max value

8.294.2.7 int mitkTransferFunction2DRGBA::GetMaxY () [inline]

Get Max gradient

Returns:
Return Max gradient

8.294.2.8 float* mitkTransferFunction2DRGBA::GetRed ()

Get red channel values of the transfer function

Returns:
Return red channel values of the transfer function

8.294.2.9 bool mitkTransferFunction2DRGBA::IsModified () [inline]

Return the modified status of the transfer function Be sure to call Update to apply changes to classifiers

Returns:
true is the transfer function is modified

8.294.2.10 bool mitkTransferFunction2DRGBA::RemoveAllClassifiers ()

Remove all classifiers from the transfer function

Returns:
Return true if all removal is successful

8.294.2.11 bool mitkTransferFunction2DRGBA::RemoveClassifier (mitkTF2DRGBAClassifier * *Classifier*)

Remove a classifier from the transfer function

Parameters:
Classifier Specify the classifier to be removed from the transfer function

Returns:
Return true if the removal is successful

8.294.2.12 void mitkTransferFunction2DRGBA::SetMax (int *maxX*, int *maxY*)

Set Max value and gradient

Parameters:

maxX Specify max value

maxY Specify max gradient

8.294.2.13 void mitkTransferFunction2DRGBA::SetUnmodified () [inline]

Set the the modified status of the transfer function to false

Reimplemented from mitkTransferFunction.

8.294.2.14 void mitkTransferFunction2DRGBA::Update ()

Apply all changes to classifiers as well as changes to the classifier membership. If any change is made, the modified status of the transfer function is set to true.

The documentation for this class was generated from the following file:

- mitkTransferFunction2DRGBA.h

8.295 mitkTransform Class Reference

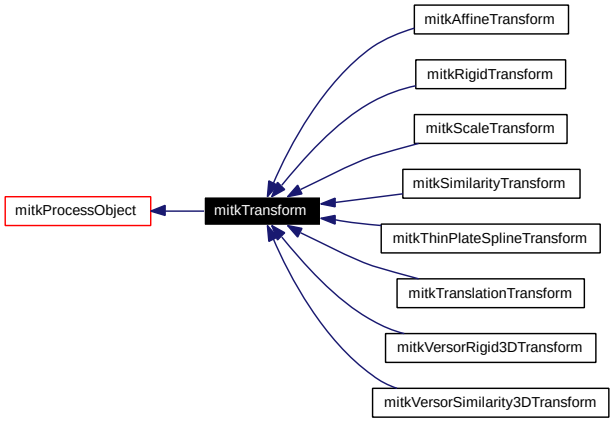
mitkTransform - an abstract class to perform coordinates transformation

`#include <mitkTransform.h>`

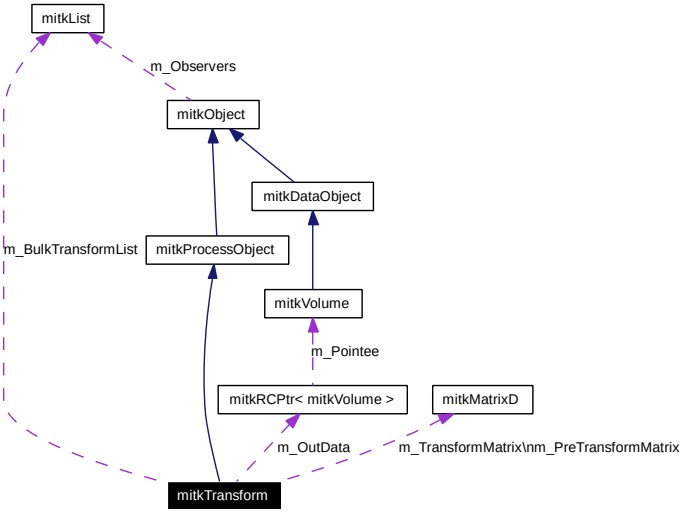
Inherits mitkProcessObject.

Inherited by mitkAffineTransform, mitkBSplineFreeFormTransform, mitkRigidTransform, mitkScaleTransform, mitkSimilarityTransform, mitkThinPlateSplineTransform, mitkTranslationTransform, mitkVersorRigid3DTransform, and mitkVersorSimilarity3DTransform.

Inheritance diagram for mitkTransform:



Collaboration diagram for mitkTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void SetParameters (const VectorParameterType &parameters)
- virtual VectorParameterType & GetParameters ()

- void SetDimensions (int d[3])
- const VectorIndexType & GetDimensions () const
- unsigned int GetSpaceDimension ()
- void SetSpacings (float s[3])
- const VectorParameterType & GetSpacings () const
- mitkVolume * GetOutput ()
- unsigned int GetNumberOfParameters ()
- void SetCentreTransformFlag (bool flag)
- void SetRegion (const VectorIndexType &r)
- virtual bool Update ()
- virtual const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- virtual void TransformPoint (const ScalarIndexType &x, const ScalarIndexType &y, const ScalarIndexType &z, float *outPoint)
- virtual void TransformPoint (const ScalarParameterType &x, const ScalarParameterType &y, const ScalarParameterType &z, ScalarParameterType *outPoint)
- void SetIdentityMatrix ()
- virtual void ComputeTransformMatrix ()
- void SetComputeMatrixFlag (bool flag)
- void SetComputeJacobianFlag (bool flag)
- void SetComputeTransformParametersFlag (bool flag)
- void SetTransformMatrix (mitkMatrixScalarType *matrix, bool ToCentered=true)
- void SetTransformMatrix (mitkMatrixType *matrix, bool ToCentered=true)
- mitkMatrixType * GetTransformMatrix ()
- void ConcatenationTransform (mitkMatrixType *matrix, bool ToCentered=true)
- bool GetConcatenationTransformFlag ()
- virtual void SetTransformMode (int transformMode, unsigned int dim)
- int GetTransformMode ()
- virtual void ConvertTransformMaxtrixToVector ()
- virtual void ConvertTransformVectorToMatrix ()
- bool TransformPointSet (mitkPointSet *p)
- bool TransformPointSet (mitkPointSet *p1, mitkPointSet *p2)
- void SetOutputDatatype (int dataType)

8.295.1 Detailed Description

mitkTransform - an abstract class to perform coordinates transformation

mitkTransform is an abstract class to perform coordinates transformation, which will convert one point from the Reference object space to the Target object space.

8.295.2 Member Function Documentation

8.295.2.1 virtual void mitkTransform::ComputeTransformMatrix () [virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented in mitkAffineTransform, mitkRigidTransform, mitkScaleTransform, mitkSimilarityTransform, mitkThinPlateSplineTransform, mitkTranslationTransform, mitkVersorRigid3DTransform, and mitkVersorSimilarity3DTransform.

8.295.2.2 void mitkTransform::ConcatenationTransform (mitkMatrixType * *matrix*, bool *ToCentered* = true)

Compute PreTransformMatrix in order to perform concatenate transform.

Parameters:

matrix The pointer to the matrix which multiplies with PreTransformMatrix.

8.295.2.3 virtual void mitkTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector.

Reimplemented in mitkAffineTransform, mitkRigidTransform, mitkScaleTransform, mitkSimilarityTransform, mitkThinPlateSplineTransform, mitkTranslationTransform, mitkVersorRigid3DTransform, and mitkVersorSimilarity3DTransform.

8.295.2.4 virtual void mitkTransform::ConvertTransformVectorToMatrix () [virtual]

Convert transform parameter vector to transform matrix.

8.295.2.5 bool mitkTransform::GetConcatenationTransformFlag () [inline]

Get the Concatenation Transform Flag.

Returns:

Return the flag value.

8.295.2.6 const VectorIndexType& mitkTransform::GetDimensions () const [inline]

Get dimension in x, y, z direction of fixed volume.

Parameters:

d[0] Get the dimension in x direction.

d[1] Get the dimension in y direction.

d[2] Get the dimension in z direction.

8.295.2.7 virtual const MatrixType& mitkTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:

x The x index of the point in image.

y The y index of the point in image.

z The z index of the point in image.

Returns:

Return the pointer to the Jacobian matrix.

Reimplemented in mitkAffineTransform, mitkRigidTransform, mitkScaleTransform, mitkSimilarityTransform, mitkTranslationTransform, mitkVersorRigid3DTransform, and mitkVersorSimilarity3DTransform.

8.295.2.8 unsigned int mitkTransform::GetNumberOfParameters () [inline]

Get number of transform parameters.

Returns:
Return the number of transform parameters.

8.295.2.9 mitkVolume* mitkTransform::GetOutput ()

Get the output volume.

Returns:
Return the output volume.

8.295.2.10 virtual VectorParameterType& mitkTransform::GetParameters () [virtual]

Get Transform Parameters.

Returns:
Return the transform parameters.

8.295.2.11 unsigned int mitkTransform::GetSpaceDimension () [inline]

Get transform space dimension.

Returns:
Return the space dimension.

8.295.2.12 const VectorParameterType& mitkTransform::GetSpacings () const [inline]

Get spacing information in x, y and z axis of the fixed volume, the unit is mm.

Parameters:

s[0] Return the spacing (mm) in two adjacent voxels in x axis. It is equal to the return value of GetSpacingX()

s[1] Return the spacing (mm) in two adjacent voxels in y axis. It is equal to the return value of GetSpacingY()

s[2] Return the spacing (mm) in two adjacent voxels in z axis. It is equal to the return value of GetSpacingZ()

8.295.2.13 mitkMatrixType* mitkTransform::GetTransformMatrix ()

Get the transform matrix.

Returns:
Return the pointer to the transform matrix.

8.295.2.14 int mitkTransform::GetTransformMode () [inline]

Get the transform mode.

Returns:
Return the mode of transform.

8.295.2.15 virtual void mitkTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkProcessObject.

Reimplemented in mitkAffineTransform, mitkRigidTransform, mitkScaleTransform, mitkSimilarityTransform, mitkThinPlateSplineTransform, mitkTranslationTransform, mitkVersorRigid3DTransform, and mitkVersorSimilarity3DTransform.

8.295.2.16 void mitkTransform::SetCentreTransformFlag (bool *flag*) [inline]

Set the flag of center transform.

Parameters:
flag The flag of center transform.

8.295.2.17 void mitkTransform::SetComputeJacobianFlag (bool *flag*) [inline]

Set the compute Jacobian parameters flag.

Parameters:
flag Compute Jacobian parameters if the flag is true.

8.295.2.18 void mitkTransform::SetComputeMatrixFlag (bool *flag*) [inline]

Set the compute matrix flag.

Parameters:
flag Compute transform matrix if the flag is true.

8.295.2.19 void mitkTransform::SetComputeTransformParametersFlag (bool *flag*) [inline]

Set the compute transform parameter vector flag.

Parameters:
flag Compute transform parameter vector if the flag is true.

8.295.2.20 void mitkTransform::SetDimensions (int *d*[3])

Set dimension in x, y, z direction of fixed volume.

Parameters:
dims[0] The dimension in x direction.
dims[1] The dimension in y direction.
dims[2] The dimension in z direction.

8.295.2.21 void mitkTransform::SetIdentityMatrix ()

Set the transform matrix to identity.

8.295.2.22 void mitkTransform::SetOutputDatatype (int *dataType*) [inline]

Set the data type for output volume.

Parameters:
dataType Specify the output volume's data type.

8.295.2.23 virtual void mitkTransform::SetParameters (const VectorParameterType & *parameters*) [virtual]

Set the transform parameters.

Parameters:
parameters The transform parameters.

8.295.2.24 void mitkTransform::SetRegion (const VectorIndexType & *r*)

Set the region of transform

Parameters:
r[0] The first (smaller) index in x axis, the unit is pixel.
r[1] The second (bigger) index in x axis, the unit is pixel.
r[2] The first (smaller) index in y axis, the unit is pixel.
r[3] The second (bigger) index in y axis, the unit is pixel.
r[4] The first (smaller) index in z axis, the unit is pixel.
r[5] The second (bigger) index in z axis, the unit is pixel.

8.295.2.25 void mitkTransform::SetSpacings (float *s*[3])

Set spacing information in x, y and z axis of the fixed volume, the unit is mm.

Parameters:

s[0] Set the spacing (mm) in two adjacent voxels in x axis.

s[1] Set the spacing (mm) in two adjacent voxels in y axis.

s[2] Set the spacing (mm) in two adjacent voxels in z axis.

8.295.2.26 void mitkTransform::SetTransformMatrix (mitkMatrixType * *matrix*, bool *ToCentered* = `true`)

Initialize transform matrix using a matrix.

Parameters:

matrix The pointer to the transform matrix user defined.

8.295.2.27 void mitkTransform::SetTransformMatrix (mitkMatrixScalarType * *matrix*, bool *ToCentered* = `true`)

Initialize transform matrix using an array.

Parameters:

matrix The pointer to the transform matrix user defined.

8.295.2.28 virtual void mitkTransform::SetTransformMode (int *transformMode*, unsigned int *dim*) `[virtual]`

Set the transform mode. param mode Specify the transform mode.

Reimplemented in mitkScaleTransform.

8.295.2.29 virtual void mitkTransform::TransformPoint (const ScalarParameterType & *x*, const ScalarParameterType & *y*, const ScalarParameterType & *z*, ScalarParameterType * *outPoint*) `[virtual]`

Perform a point transform. (point transform) /param point The transformed point need to be compute /param x The x index of the point in image. /param y The y index of the point in image. /param z The z index of the point in image. /return Return true if the transformation is performed without errors.

8.295.2.30 virtual void mitkTransform::TransformPoint (const ScalarIndexType & *x*, const ScalarIndexType & *y*, const ScalarIndexType & *z*, float * *outPoint*) `[virtual]`

Perform a point transform. (pixel transform) /param point The transformed point need to be compute /param x The x index of the point in image. /param y The y index of the point in image. /param z The z index of the point in image. /return Return true if the transformation is performed without errors.

8.295.2.31 bool mitkTransform::TransformPointSet (mitkPointSet * *p1*, mitkPointSet * *p2*)

Perform point set transform. /param p1 The pointer to a input point set. /param p2 The pointer to a output point set. /return Return true if the transformation is performed without errors.

8.295.2.32 bool mitkTransform::TransformPointSet (mitkPointSet * *p*)

Perform a point set transform. /param p The pointer to a input point set. /return Return true if the transformation is performed without errors.

8.295.2.33 virtual bool mitkTransform::Update () [virtual]

Initialization, perform before Run() function

The documentation for this class was generated from the following file:

- mitkTransform.h

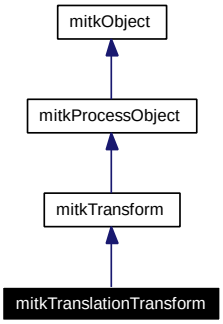
8.296 mitkTranslationTransform Class Reference

mitkTranslationTransform - a concrete transform to perform translation transform

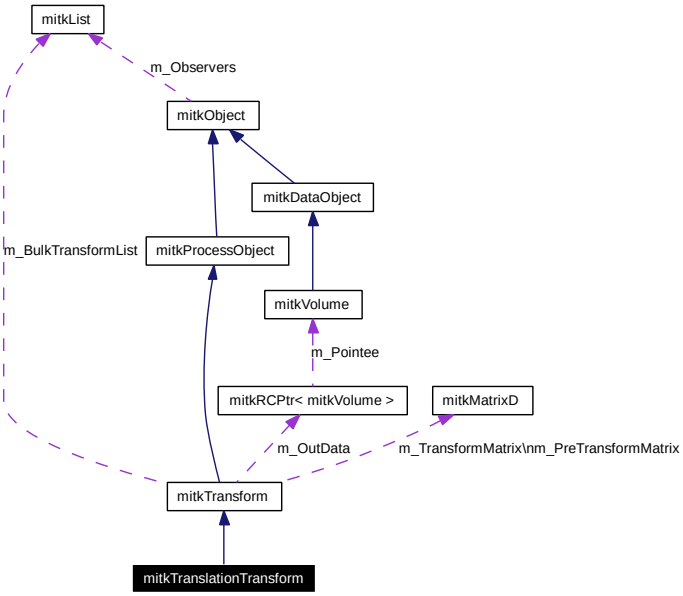
`#include <mitkTranslationTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkTranslationTransform:



Collaboration diagram for mitkTranslationTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkTranslationTransform ()
- mitkTranslationTransform (unsigned int dim)
- const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- virtual void ConvertTransformMaxtrixToVector ()
- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty)

- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty, ScalarParameterType &tz)

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.296.1 Detailed Description

mitkTranslationTransform - a concrete transform to perform translation transform

mitkTranslationTransform is a concrete transform to perform translation transformation. This transform applies a translation to the 2d and 3d space.User should use the constructor to specify the image dimensions, e.g. mitkTranslationTransform(2) (default) for 2d image, mitkTranslationTransform(3) for 3d image.

Transform Mode Number of parameters p[0] p[1] p[2]

MITK_TRANSFORM_TRANSLATION (2D) 2 tx ty -

MITK_TRANSFORM_TRANSLATION (3D) 3 tx ty tz

8.296.2 Constructor & Destructor Documentation

8.296.2.1 mitkTranslationTransform::mitkTranslationTransform ()

Constructor.

8.296.2.2 mitkTranslationTransform::mitkTranslationTransform (unsigned int *dim*)

Constructor with specific transform mode setting.

8.296.3 Member Function Documentation

8.296.3.1 virtual void mitkTranslationTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.296.3.2 virtual void mitkTranslationTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.296.3.3 const MatrixType& mitkTranslationTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:
inPoint The index of the point in image.

Returns:
Return the pointer to the Jacobian matrix.

Reimplemented from mitkTransform.

8.296.3.4 void mitkTranslationTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*, ScalarParameterType & *tz*)

Get the translation parameters from the transform parameter vector. (for 3d translation)

Parameters:
tx The returned x-translation value.
ty The returned y-translation value.
tz The returned z-translation value.

8.296.3.5 void mitkTranslationTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*)

Get the translation parameters from the transform parameter vector. (for 2d translation)

Parameters:
tx The returned x-translation value.
ty The returned y-translation value.

8.296.3.6 virtual void mitkTranslationTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkTranslationTransform.h

8.297 mitkTriangleMesh Class Reference

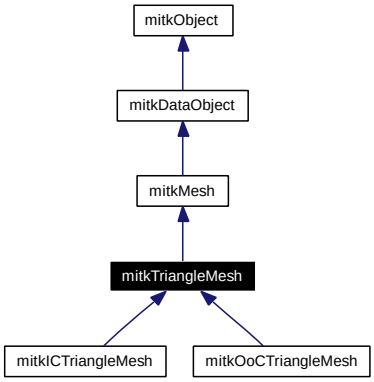
mitkTriangleMesh - an abstract class for triangle meshes

`#include <mitkTriangleMesh.h>`

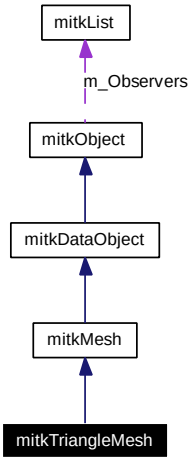
Inherits mitkMesh.

Inherited by mitkICTriangleMesh, and mitkOoCTriangleMesh.

Inheritance diagram for mitkTriangleMesh:



Collaboration diagram for mitkTriangleMesh:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkTriangleMesh ()
- bool CreateFrom (mitkHETriangleMesh *mesh)
- virtual int GetDataObjectType () const
- virtual void Initialize ()
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)

- virtual size_type GetVertexNumber () const
- virtual size_type GetFaceNumber () const
- index_type AddFace (TriangleFace &face)

8.297.1 Detailed Description

mitkTriangleMesh - an abstract class for triangle meshes

mitkTriangleMesh is an abstract class for triangle meshes.

See also:
mitkICTriangleMesh and mitkOoCTriangleMesh

8.297.2 Constructor & Destructor Documentation

8.297.2.1 mitkTriangleMesh::mitkTriangleMesh ()

Default constructor of this class.

8.297.3 Member Function Documentation

8.297.3.1 index_type mitkTriangleMesh::AddFace (TriangleFace & *face*) [inline]

Add a triangle face.

Parameters:
face the face to add

Returns:
Return the index of the face added.

Note:
The struct TriangleFace is equal to follows:

```
struct TriangleFace
{
    enum { vertNum = 3 };
    index_type verts[vertNum];
};
```

Warning:
Each index in the "verts" array of the face must be valid, i.e. represents a existent vertex in the mesh (has been added before).

See also:
mitkGeometryTypes.h

Reimplemented from mitkMesh.

8.297.3.2 bool mitkTriangleMesh::CreateFrom (mitkHETriangleMesh * *mesh*)

Create mitkTriangleMesh object from a mitkHETriangleMesh object which is based on half edge structure.

Parameters:
mesh the pointer to a mitkHETriangleMesh object this mitkTriangleMesh object is created from

Returns:
Return true if this operation succeed, otherwise return false.

8.297.3.3 virtual void mitkTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkMesh.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.297.3.4 virtual int mitkTriangleMesh::GetDataObjectType () const [inline, virtual]

Return what type of data object this is.

Returns:
Return the type of this data object.

Reimplemented from mitkMesh.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.297.3.5 virtual size_type mitkTriangleMesh::GetFaceNumber () const [inline, virtual]

Get the mesh's face number.

Returns:
Return the number of faces.

Implements mitkMesh.

8.297.3.6 virtual size_type mitkTriangleMesh::GetVertexNumber () const [inline, virtual]

Get the mesh's vertex number.

Returns:
Return the number of vertices.

Implements mitkMesh.

8.297.3.7 virtual void mitkTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkMesh.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.297.3.8 virtual void mitkTriangleMesh::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkMesh.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

8.297.3.9 virtual void mitkTriangleMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkMesh.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

The documentation for this class was generated from the following file:

- mitkTriangleMesh.h

8.298 mitkTriangleMeshSimplification Class Reference

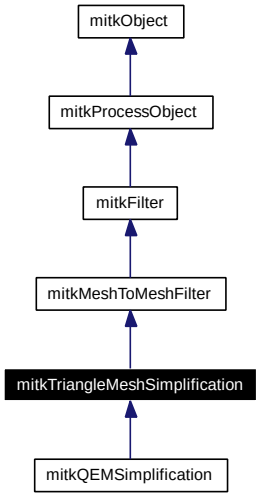
mitkTriangleMeshSimplification - abstract class for triangle mesh simplification algorithms

`#include <mitkTriangleMeshSimplification.h>`

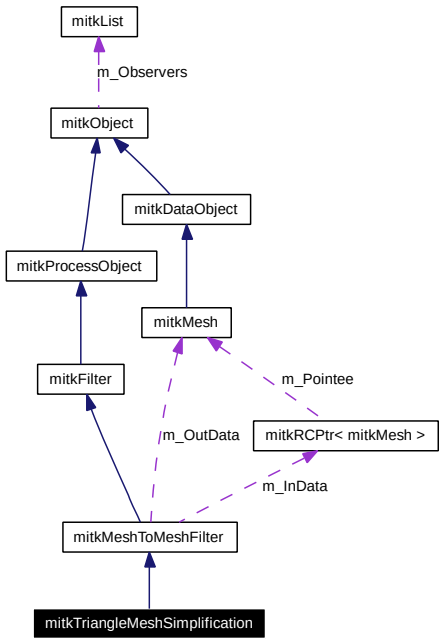
Inherits mitkMeshToMeshFilter.

Inherited by mitkQEMSimplification.

Inheritance diagram for mitkTriangleMeshSimplification:



Collaboration diagram for mitkTriangleMeshSimplification:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetTargetFaceNumber (size_type num)

8.298.1 Detailed Description

mitkTriangleMeshSimplification - abstract class for triangle mesh simplification algorithms

mitkTriangleMeshSimplification is an abstract class for triangle mesh simplification algorithms. The subclasses which implement concrete triangle mesh simplification algorithms should override the pure virual function _simplificationProcess() and put the implementation in it. Use the parameter "mesh" which is a pointer to a mitkHETriangleMesh object prepared and transfered by this class to do simplification, because mitkHETriangleMesh (designed based on half edge structure) is more suitable for mesh processing algorithms. The Execute() function of this class will generate the right mitkHETriangleMesh object for you, and call _simplificationProcess() with the object as parameter to perform real simplification, so do not override Execute() in your subclasses.

8.298.2 Member Function Documentation

8.298.2.1 virtual void mitkTriangleMeshSimplification::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkMeshToMeshFilter.

Reimplemented in mitkQEMSimplification.

8.298.2.2 void mitkTriangleMeshSimplification::SetTargetFaceNumber (size_type *num*) [inline]

Set the target face number of simplification.

Parameters:

num the target face number

The documentation for this class was generated from the following file:

- mitkTriangleMeshSimplification.h

8.299 mitkVector Class Reference

mitkVector - an encapsulation of a 4-element vector

`#include <mitkMatrix.h>`

8.299.1 Detailed Description

mitkVector - an encapsulation of a 4-element vector

mitkVector provides an encapsulation of vector operations.

The documentation for this class was generated from the following file:

- mitkMatrix.h

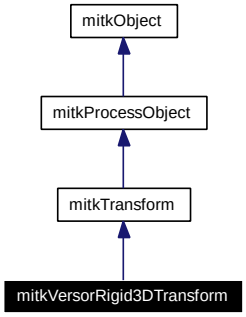
8.300 mitkVersorRigid3DTransform Class Reference

mitkVersorRigid3DTransform - a concrete transform to perform 3d rigid transformation

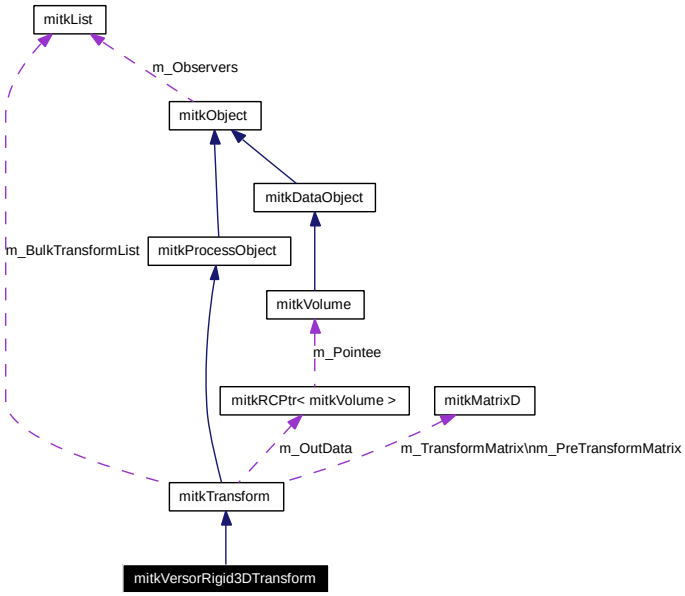
`#include <mitkVersorRigid3DTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkVersorRigid3DTransform:



Collaboration diagram for mitkVersorRigid3DTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkVersorRigid3DTransform ()
- const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- virtual void ConvertTransformMaxtrixToVector ()
- void GetVersor (ScalarParameterType &x, ScalarParameterType &y, ScalarParameterType &z, ScalarParameterType &w)

- void GetAngles (ScalarParameterType &alpha, ScalarParameterType &beta, ScalarParameterType &gamma)
- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty, ScalarParameterType &tz)

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.300.1 Detailed Description

mitkVersorRigid3DTransform - a concrete transform to perform 3d rigid transformation

This transform applies rotation and translation to the 3D image space. There are 6 elements in transform parameter vector, the first 3 elements are the components of the versor representation of 3D rotation, the last 3 parameters defines the translation in each dimension.

Transform Mode Number of parameters p[0] p[1] p[2] p[3] p[4] p[5]

MITK_TRANSFORM_VERSOR_RIGID_3D 6 vx vy vz tx ty tz

8.300.2 Constructor & Destructor Documentation

8.300.2.1 mitkVersorRigid3DTransform::mitkVersorRigid3DTransform ()

Constructor.

8.300.3 Member Function Documentation

8.300.3.1 virtual void mitkVersorRigid3DTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.300.3.2 virtual void mitkVersorRigid3DTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.300.3.3 void mitkVersorRigid3DTransform::GetAngles (ScalarParameterType & *alpha*, ScalarParameterType & *beta*, ScalarParameterType & *gamma*)

Get the rotation parameters from the transform parameter vector.

Parameters:

alpha The returned angle value. (rotate around x-axis)

beta The returned angle value. (rotate around y-axis)

gamma The returned angle value. (rotate around z-axis)

8.300.3.4 const MatrixType& mitkVersorRigid3DTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:

inPoint The index of the point in image.

Returns:

Return the pointer to the Jacobian matrix.

Reimplemented from mitkTransform.

8.300.3.5 void mitkVersorRigid3DTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*, ScalarParameterType & *tz*)

Get the translation parameters from the transform parameter vector.

Parameters:

tx The returned x-translation value.

ty The returned y-translation value.

tz The returned z-translation value.

8.300.3.6 void mitkVersorRigid3DTransform::GetVersor (ScalarParameterType & *x*, ScalarParameterType & *y*, ScalarParameterType & *z*, ScalarParameterType & *w*)

Get the rotation parameters from the transform parameter vector.

Parameters:

alpha The returned angle value. (rotate around x-axis)

beta The returned angle value. (rotate around y-axis)

gamma The returned angle value. (rotate around z-axis)

8.300.3.7 virtual void mitkVersorRigid3DTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkVersorRigid3DTransform.h

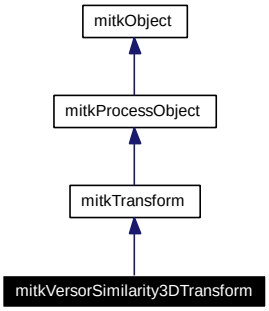
8.301 mitkVersorSimilarity3DTransform Class Reference

mitkVersorSimilarity3DTransform - a concrete transform to perform 3d similarity transform

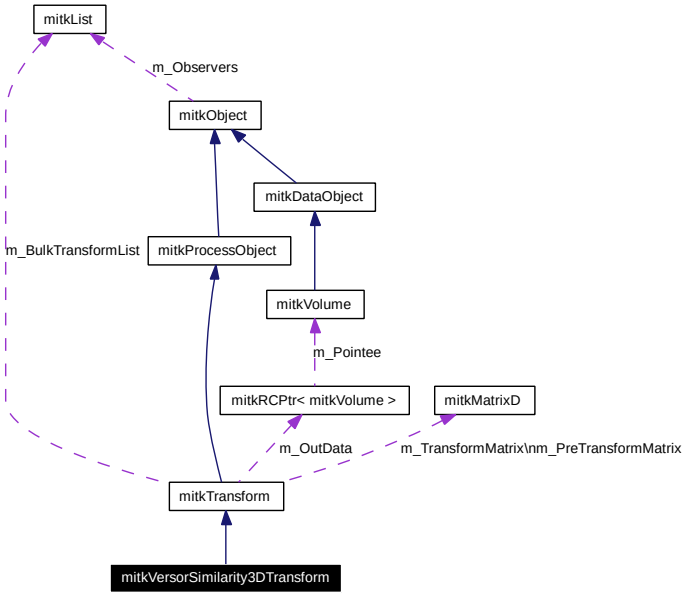
`#include <mitkVersorSimilarity3DTransform.h>`

Inherits mitkTransform.

Inheritance diagram for mitkVersorSimilarity3DTransform:



Collaboration diagram for mitkVersorSimilarity3DTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkVersorSimilarity3DTransform ()
- const MatrixType & GetJacobian (const VectorParameterType &inPoint)
- virtual void ConvertTransformMaxtrixToVector ()
- void GetVersor (ScalarParameterType &x, ScalarParameterType &y, ScalarParameterType &z, ScalarParameterType &w)

- void GetAngles (ScalarParameterType &alpha, ScalarParameterType &beta, ScalarParameterType &gamma)
- void GetTranslations (ScalarParameterType &tx, ScalarParameterType &ty, ScalarParameterType &tz)
- void GetScales (ScalarParameterType &scale)

Protected Member Functions

- virtual void ComputeTransformMatrix ()

8.301.1 Detailed Description

mitkVersorSimilarity3DTransform - a concrete transform to perform 3d similarity transform

This transform applies a similarity transform to the 3D image space. There are 7 elements in transform parameter vector, the first 3 elements are the components of the versor representation of 3D rotation, the next 3 parameters defines the translation in each dimension, the last 1 parameter defines the isotropic scaling.

Transform Mode Number of parameters p[0] p[1] p[2] p[3] p[4] p[5] p[6]

MITK_TRANSFORM_VERSOR_SIMILARITY_3D 7 angle_x angle_y angle_z tx ty tz scale

8.301.2 Constructor & Destructor Documentation

8.301.2.1 mitkVersorSimilarity3DTransform::mitkVersorSimilarity3DTransform ()

Constructor.

8.301.3 Member Function Documentation

8.301.3.1 virtual void mitkVersorSimilarity3DTransform::ComputeTransformMatrix () [protected, virtual]

Calculate the transform matrix.

Returns:
Return true if the computation is performed without error.

Reimplemented from mitkTransform.

8.301.3.2 virtual void mitkVersorSimilarity3DTransform::ConvertTransformMaxtrixToVector () [virtual]

Convert transform matrix to transform parameter vector. This function operates on build-in data.

Reimplemented from mitkTransform.

8.301.3.3 void mitkVersorSimilarity3DTransform::GetAngles (ScalarParameterType & *alpha*, ScalarParameterType & *beta*, ScalarParameterType & *gamma*)

Get the rotation parameters from the transform parameter vector.

Parameters:

alpha The returned angle value. (rotate around x-axis)

beta The returned angle value. (rotate around y-axis)

gamma The returned angle value. (rotate around z-axis)

8.301.3.4 const MatrixType& mitkVersorSimilarity3DTransform::GetJacobian (const VectorParameterType & *inPoint*) [virtual]

Get the Jacobian matrix.

Parameters:

inPoint The index of the point in image.

Returns:

Return the pointer to the Jacobian matrix.

Reimplemented from mitkTransform.

8.301.3.5 void mitkVersorSimilarity3DTransform::GetScales (ScalarParameterType & *scale*)

Get the scaling parameters from the transform parameter vector.

Parameters:

scale The returned isotropic scale value.

8.301.3.6 void mitkVersorSimilarity3DTransform::GetTranslations (ScalarParameterType & *tx*, ScalarParameterType & *ty*, ScalarParameterType & *tz*)

Get the translation parameters from the transform parameter vector.

Parameters:

tx The returned x-translation value.

ty The returned y-translation value.

tz The returned z-translation value.

8.301.3.7 void mitkVersorSimilarity3DTransform::GetVersor (ScalarParameterType & *x*, ScalarParameterType & *y*, ScalarParameterType & *z*, ScalarParameterType & *w*)

Get the rotation parameters from the transform parameter vector.

Parameters:

alpha The returned angle value. (rotate around x-axis)

beta The returned angle value. (rotate around y-axis)

gamma The returned angle value. (rotate around z-axis)

8.301.3.8 virtual void mitkVersorSimilarity3DTransform::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkTransform.

The documentation for this class was generated from the following file:

- mitkVersorSimilarity3DTransform.h

8.302 mitkView Class Reference

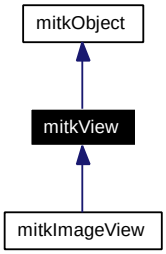
mitkView - a view to display 3D surface rendered or volume rendered image

`#include <mitkView.h>`

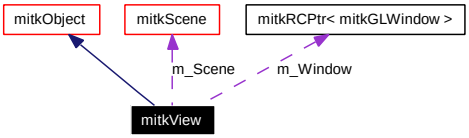
Inherits mitkObject.

Inherited by mitkImageView.

Inheritance diagram for mitkView:



Collaboration diagram for mitkView:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void * GetWindowId ()
- void * GetParent ()
- void * GetApplicationId ()
- void * GetDeviceContext ()
- void * GetRenderContext ()
- void SetParent (void *parentId)
- const int * GetPosition ()
- int GetLeft ()
- int GetTop ()
- void SetPosition (int leftPos, int topPos)
- void SetPosition (int a[2])
- void SetLeft (int leftPos)
- void SetTop (int topPos)
- int GetTitleBar ()
- void SetTitleBar (int titleBar)
- void SetTitleBarOn ()
- void SetTitleBarOff ()
- const int * GetSize ()
- int GetWidth ()

- int GetHeight ()
- void SetSize (int widthSize, int heightSize)
- void SetSize (int a[2])
- void SetWidth (int widthSize)
- void SetHeight (int heightSize)
- const int * GetViewPort ()
- void GetViewPort (int vp[4])
- const char * GetWindowName ()
- void SetWindowName (const char *winName)
- void Show ()
- void Hide ()
- void Update ()
- void MakeCurrent ()
- void SwapBuffers ()
- void SetBackColor (unsigned char rColor, unsigned char gColor, unsigned char bColor)
- void SetBackColor (float rColor, float gColor, float bColor)
- void SetBackColor (int rColor, int gColor, int bColor)
- void GetBackColor (float &rColor, float &gColor, float &bColor)
- void GetBackColor (unsigned char &rColor, unsigned char &gColor, unsigned char &bColor)
- void GetBackColor (int &rColor, int &gColor, int &bColor)
- void ResetScene ()
- void Translate (float deltX, float deltY, float deltZ)
- void Rotate (float deltX, float deltY, float deltZ)
- void Rotate (float ax, float ay, float az, float angle)
- void Rotate (const mitkQuaternion &q)
- void SetRotation (float x, float y, float z)
- void SetRotation (float ax, float ay, float az, float angle)
- void SetRotation (const mitkQuaternion &q)
- void Scale (float delt)
- mitkManipulator * CreateManipulator ()
- void SetManipulator (mitkManipulator *mani)
- void AddModel (mitkModel *model)
- void RemoveModel (mitkModel *model)
- void RemoveModel (int i)
- void RemoveAllModels ()
- void RemoveAllModel ()
- mitkModel * GetModel (int i)
- mitkList * GetModels ()
- int GetModelCount ()
- int GetNumberOfPropsRendered ()
- void SetCamera (mitkCamera *cam)
- mitkCamera * GetCamera ()
- void SetDefaultLight (mitkLight *lit)
- mitkLight * GetDefaultLight ()
- mitkLight * GetLight ()
- void AddAdditionalLight (mitkLight *lit)
- void RemoveAdditionalLight (int idx)
- void RemoveAdditionalLight (mitkLight *lit)
- void RemoveAllAdditionalLights ()
- int GetAdditionalLightCount ()

- mitkLight * GetAdditionalLight (int idx)
- void AlignScene ()
- float * GetZbufferData (int x1, int y1, int x2, int y2)
- void GetZbufferData (int x1, int y1, int x2, int y2, float *buffer)
- unsigned char * GetPixelData (int xOffset, int yOffset, int width, int height)
- void GetPixelData (int xOffset, int yOffset, int width, int height, unsigned char *buffer, int packAlignment=4, bool swapRGB=false)
- void GetPixelData24 (int xOffset, int yOffset, int width, int height, unsigned char *buffer, int packAlignment=4, bool swapRGB=false)
- void GetPixelData32 (int xOffset, int yOffset, int width, int height, unsigned char *buffer, bool swapRGB=false)
- void SetSelected (bool isSelected)
- bool GetSelected ()
- float GetTranslateSpeed ()
- float GetTranslationX ()
- float GetTranslationY ()
- float GetTranslationZ ()
- void GetTranslation (float trans[3])
- const mitkQuaternion & GetRotation ()
- float GetScale ()
- bool HasUnprocessedMouseMessage ()
- void EnableLoD ()
- void DisableLoD ()
- bool IsLoDEnabled ()
- double GetFrameSpeed ()
- double GetRenderTime ()
- void EnableCalculateFrameSpeed ()
- void DisableCalculateFrameSpeed ()
- bool IsModelsModified ()
- bool IsCameraModified ()
- bool IsAdditionalLightModified ()
- mitkScene * GetScene ()

8.302.1 Detailed Description

mitkView - a view to display 3D surface rendered or volume rendered image

mitkView is a 3d view to display 3D surface rendered or volume rendered image. To display 3d images, you must create one or several models (either mitkSurfaceModel or mitkVolumeModel) firstly, then add them using the function

```
AddModel()
```

. mitkView can be easily integrated into any user interface toolkit, such as MFC in Microsoft Visual C++, VCL in Borland C++ Builder, Trolltech Qt, etc. The following code snippet demonstrate this point:

```
mitkView *aView = new mitkView;
//Set the parent window, and fill out the whole parent window
aView->SetParent(parentWindowId);
aView->SetLeft(0);
aView->SetTop(0);
aView->SetWidth(parentWindowWidth);
aView->SetHeight(parentWindowHeight);
aView->Show();
```

So mitkView only requires a parent window id and then can integrate into that window seamless. From Mitk 2.3, mitkView is decomposed into mitkGLWindow and mitkScene, it is recommanded to use those two classes instead of this one. This class serves as a wrapper that provides the compatibility to the tranditional routines.

8.302.2 Member Function Documentation

8.302.2.1 void mitkView::AddAdditionalLight (mitkLight * *lit*) [inline]

Add an additional light to this view.

Parameters:
lit the light to add

Note:
At most 5 additional light can be added to the view.

8.302.2.2 void mitkView::AddModel (mitkModel * *model*) [inline]

Add a model to this view for display. View can contain many models, including surface model, volume model and widget model, and display them simultaneity. So view provides a set of functions to manage the models.

See also:
RemoveModel()
RemoveAllModel()
GetModel()
GetModels()
GetModelCount()

Parameters:
model Specify the model that will be added to this view.

8.302.2.3 void mitkView::AlignScene () [inline]

Warning:
Internal function. Don't call it directly.

8.302.2.4 mitkManipulator* mitkView::CreateManipulator ()

Create a default manipulator for the mouse events processing. Subclass can override this function to create a proper manipulator.

8.302.2.5 void mitkView::DisableCalculateFrameSpeed () [inline]

Disable frame speed calculation.

Note:
It is only a hint to actual implementor and depends on actual implementation. Maybe it will not take any effect.

8.302.2.6 void mitkView::DisableLoD () [inline]

Disable LoD rendering mode.

Note:
It is only a hint to rendering method and depends on actual implementation. Maybe it will not take any effect.

8.302.2.7 void mitkView::EnableCalculateFrameSpeed () [inline]

Enable frame speed calculation.

Note:
It is only a hint to actual implementor and depends on actual implementation. Maybe it will not take any effect.

8.302.2.8 void mitkView::EnableLoD () [inline]

Enable LoD rendering mode.

Note:
It is only a hint to rendering method and depends on actual implementation. Maybe it will not take any effect.

8.302.2.9 mitkLight* mitkView::GetAdditionalLight (int *idx*) [inline]

Get the idx'th additional light.

Parameters:
idx the index of the light to get

8.302.2.10 int mitkView::GetAdditionalLightCount () [inline]

Get the count of additional lights.

Returns:
Return the count of additional lights.

8.302.2.11 void* mitkView::GetApplicationId (void) [inline]

Get the system dependent application id

Returns:
Return the system dependent parent window id

8.302.2.12 void mitkView::GetBackColor (int & *rColor*, int & *gColor*, int & *bColor*) [inline]

Get the background color of this view.

Parameters:

rColor Return the red component of the background color. The value range is from 0 to 255.

gColor Return the green component of the background color. The value range is from 0 to 255.

bColor Return the blue component of the background color. The value range is from 0 to 255.

8.302.2.13 void mitkView::GetBackColor (unsigned char & *rColor*, unsigned char & *gColor*, unsigned char & *bColor*) [inline]

Get the background color of this view.

Parameters:

rColor Return the red component of the background color. The value range is from 0 to 255.

gColor Return the green component of the background color. The value range is from 0 to 255.

bColor Return the blue component of the background color. The value range is from 0 to 255.

8.302.2.14 void mitkView::GetBackColor (float & *rColor*, float & *gColor*, float & *bColor*) [inline]

Get the background color of this view.

Parameters:

rColor Return the red component of the background color. The value range is from 0.0 to 1.0.

gColor Return the green component of the background color. The value range is from 0.0 to 1.0.

bColor Return the blue component of the background color. The value range is from 0.0 to 1.0.

8.302.2.15 mitkCamera* mitkView::GetCamera () [inline]

Get the camera of this view.

Returns:

Return the camera of this view.

8.302.2.16 mitkLight* mitkView::GetDefaultLight () [inline]

Get the default light of this view.

Returns:

Return the light of this view.

Note:

The position of the default light is calculated by the view automatically, so any changes of the position will be discarded.

8.302.2.17 void* mitkView::GetDeviceContext (void) [inline]

Get the system dependent device context id

Returns:
Return the system dependent device context id

8.302.2.18 double mitkView::GetFrameSpeed () [inline]

Get frame speed.

Returns:
Return the frame speed.

8.302.2.19 int mitkView::GetHeight () [inline]

Get the Height of this view

Returns:
Return the height of this view

8.302.2.20 int mitkView::GetLeft () [inline]

Get the Left position in local coordinate of its parent window

Returns:
Return the Left position in local coordinate of its parent window

8.302.2.21 mitkLight* mitkView::GetLight () [inline]

Provided for back compatibility, just the same as SetSliceNum().

8.302.2.22 mitkModel* mitkView::GetModel (int *i*) [inline]

Get the model in specified position.

Parameters:
i Zero based index to specify the position of the model.

Returns:
Return the ith model in this view.

8.302.2.23 int mitkView::GetModelCount () [inline]

Get the model count in this view.

Returns:
Return the model count in this view.

8.302.2.24 mitkList* mitkView::GetModels () [inline]

Get the list of models in this view.

Returns:
Return the internal list of models. User can access each model through the interface of mitkList.

8.302.2.25 int mitkView::GetNumberOfPropsRendered () [inline]

Warning:
Internal function. Don't call it directly.

8.302.2.26 void* mitkView::GetParent (void) [inline]

Get the system dependent parent window id

Returns:
Return the system dependent parent window id

8.302.2.27 void mitkView::GetPixelData (int *xOffset*, int *yOffset*, int *width*, int *height*, unsigned char * *buffer*, int *packAlignment* = 4, bool *swapRGB* = false) [inline]

Get frame buffer data (RGB pixel data) in the specified region. This function is provided for convenience. It does the same thing as GetPixelData24().

Parameters:
xOffset specify the x coordinate of the left bottom corner

yOffset specify the y coordinate of the left bottom corner

width specify the width of the region

height specify the height of the region

buffer return the the frame buffer data. The data is RGB color data, and is arranged in the order of R1G1B1R2G2B2...

packAlignment pack alignment bytes

swapRGB whether swap RGB channel or not (i.e. true: use GL_BGR format to get pixel data, false: use GL_RGB format to get pixel data

Warning:
The parameter "buffer" must point to a memory block which can contain width*height RGB pixel samples. If packAlignment is 4, the total bytes of one line (contain width samples) must be devided exactly by 4, i.e. one line should be ((width*3+3) & (~3)) bytes. Otherwise, one line should be exactly width*3 bytes. Furthermore, this memory block must be allocated by the user before call this function.

8.302.2.28 unsigned char* mitkView::GetPixelData (int *xOffset*, int *yOffset*, int *width*, int *height*) [inline]

Get frame buffer data in the specified region.

Parameters:

xOffset Specify the x coordinate of the left bottom corner

yOffset Specify the y coordinate of the left bottom corner

width Specify the width of the region

height Specify the height of the region

Returns:

Return the address of the frame buffer data. The data is RGB color data, and is arranged in the order of R1G1B1R2G2B2...

Note:

This function is obsolete, please use another function to get frame buffer data.

8.302.2.29 void mitkView::GetPixelData24 (int *xOffset*, int *yOffset*, int *width*, int *height*, unsigned char * *buffer*, int *packAlignment* = `4`, bool *swapRGB* = `false`) `[inline]`

Get frame buffer data (RGB pixel data) in the specified region.

Parameters:

xOffset specify the x coordinate of the left bottom corner

yOffset specify the y coordinate of the left bottom corner

width specify the width of the region

height specify the height of the region

buffer return the the frame buffer data. The data is RGB color data, and is arranged in the order of R1G1B1R2G2B2...

packAlignment pack alignment bytes

swapRGB whether swap RGB channel or not (i.e. true: use GL_BGR format to get pixel data, false: use GL_RGB format to get pixel data

Warning:

The parameter "buffer" must point to a memory block which can contain width∗height RGB pixel samples. If packAlignment is 4, the total bytes of one line (contain width samples) must be devided exactly by 4, i.e. one line should be ((width∗3+3) & (∼3)) bytes. Otherwise, one line should be exactly width∗3 bytes. Furthermore, this memory block must be allocated by the user before call this function.

8.302.2.30 void mitkView::GetPixelData32 (int *xOffset*, int *yOffset*, int *width*, int *height*, unsigned char * *buffer*, bool *swapRGB* = `false`) `[inline]`

Get frame buffer data (RGBA pixel data) in the specified region.

Parameters:

xOffset specify the x coordinate of the left bottom corner

yOffset specify the y coordinate of the left bottom corner

width specify the width of the region

height specify the height of the region

buffer return the the frame buffer data. The data is RGBA color data, and is arranged in the order of R1G1B1A1R2G2B2A2...

swapRGB whether swap RGB channel or not (i.e. true: use GL_BGR format to get pixel data, false: use GL_RGB format to get pixel data

Warning:
The parameter "buffer" must point to a memory block which can contain width∗height RGBA pixel samples. This memory block must be allocated by the user before call this function.

8.302.2.31 const int∗ mitkView::GetPosition () [inline]

Get the position (Left and Top) in local coordinate of its parent window

Returns:
Return a pointer to the position. The first element is Left, and the second element is Top.

8.302.2.32 void∗ mitkView::GetRenderContext (void) [inline]

Get the system dependent render context id

Returns:
Return the system dependent render context id

8.302.2.33 double mitkView::GetRenderTime () [inline]

Get the elapsed time of render process in seconds.

Returns:
the elapsed time in seconds

8.302.2.34 const mitkQuaternion& mitkView::GetRotation (void) [inline]

Get the Rotation of the view.

Returns:
Return the quaternion of the rotation.

8.302.2.35 float mitkView::GetScale (void) [inline]

Get the scale of the view.

Returns:
Return the scale of the view.

8.302.2.36 mitkScene∗ mitkView::GetScene () [inline]

Get the scene embeded in the container window

Returns:
Return the scene embeded in the container window

8.302.2.37 bool mitkView::GetSelected () [inline]

Set the selected status of this view. If view is selected, it will be added a red frame.

Parameters:
isSelected isSelected=true, set the view is selected. isSelected=false, set the view is unselected.

8.302.2.38 const int* mitkView::GetSize () [inline]

Get the size (Width and Height) of this view

Returns:
Return a pointer to the size. The first element is Width, and the second element is Height.

8.302.2.39 int mitkView::GetTitleBar () [inline]

Check if the window has a title bar

8.302.2.40 int mitkView::GetTop () [inline]

Get the Top position in local coordinate of its parent window

Returns:
Return the Top position in local coordinate of its parent window

8.302.2.41 float mitkView::GetTranslateSpeed () [inline]

Warning:
Internal function. Don't call it directly.

8.302.2.42 void mitkView::GetTranslation (float *trans*[3]) [inline]

Get the translation of the view.

Parameters:
trans[0] return translation in x-axis

trans[1] return translation in y-axis

trans[2] return translation in z-axis

Warning:
This function is provided for convenience. Don't call it directly.

8.302.2.43 float mitkView::GetTranslationX () [inline]

Get the translation along x-axis of the view.

Returns:
Return the translation along x-axis of the view.

Warning:
This function is provided for convenience. Don't call it directly.

8.302.2.44 float mitkView::GetTranslationY () [inline]

Get the translation along y-axis of the view.

Returns:
Return the translation along y-axis of the view.

Warning:
This function is provided for convenience. Don't call it directly.

8.302.2.45 float mitkView::GetTranslationZ () [inline]

Get the translation along z-axis of the view.

Returns:
Return the translation along z-axis of the view.

Warning:
This function is provided for convenience. Don't call it directly.

8.302.2.46 void mitkView::GetViewPort (int *vp*[4]) [inline]

Get the viewport of this view. Not all of the area of a view is used to display the image. Only the viewport is used to display the image. The position and size of viewport are calculated automatically according to the size of image.

Parameters:
vp[0] return the left position of the viewport
vp[1] return the top position of the viewport
vp[2] return the width of the viewport
vp[3] return the height of the viewport

8.302.2.47 const int* mitkView::GetViewPort () [inline]

Get the viewport of this view. Not all of the area of a view is used to display the image. Only the viewport is used to display the image. The position and size of viewport are calculated automatically according to the size of image.

Returns:
Return a pointer to the viewport. The first element is the Left position. The second element is the Top position. The third element is the Width of the viewport. The fourth element is the Height of the viewport.

8.302.2.48 int mitkView::GetWidth () [inline]

Get the Width of this view

Returns:
Return the width of this view

8.302.2.49 void* mitkView::GetWindowId (void) [inline]

Get the system dependent window id

Returns:
Return the system dependent window id

8.302.2.50 const char* mitkView::GetWindowName () [inline]

Get the name of the window

8.302.2.51 void mitkView::GetZbufferData (int *x1*, int *y1*, int *x2*, int *y2*, float * *buffer*) [inline]

Get z buffer data in the specified region.

Parameters:
x1 Specify the x coordinate of the left bottom corner
y1 Specify the y coordinate of the left bottom corner
x2 Specify the x coordinate of the right top corner. x2 >= x1 must be satisfied.
y2 Specify the y coordinate of the right top corner. y2 >= y1 must be satisfied.
buffer Return the z buffer data.

Warning:
The parameter "buffer" must point to a memory block which can contain (x2-x1+1)*(y2-y1+1) floats (i.e. (x2-x1+1)*(y2-y1+1)*4 bytes). This memory block must be allocated by the user before call this function.

8.302.2.52 float* mitkView::GetZbufferData (int *x1*, int *y1*, int *x2*, int *y2*) [inline]

Get z buffer data in the specified region.

Parameters:
x1 Specify the x coordinate of the left bottom corner
y1 Specify the y coordinate of the left bottom corner

x2 Specify the x coordinate of the right top corner. x2 >= x1 must be satisfied.

y2 Specify the y coordinate of the right top corner. y2 >= y1 must be satisfied.

Returns:
Return the address of the z buffer data.

8.302.2.53 bool mitkView::HasUnprocessedMouseMessage () [inline]

Test if there is any unprocessed mouse message (button down/up) in the message queue of rendering thread. Use it in your rendering algorithm to achieve a fine mouse response.

Returns:
Return ture if there are some unprocessed mouse messages, otherwise return false.

8.302.2.54 void mitkView::Hide () [inline]

Hide this view.

8.302.2.55 bool mitkView::IsAdditionalLightModified () [inline]

Check whether there is a modified additional light

Returns:
Return true if there is a modified additional light, otherwise Return false

8.302.2.56 bool mitkView::IsCameraModified () [inline]

Check whether the camera has been modified

Returns:
Return true if the camera has been modified, otherwise Return false

8.302.2.57 bool mitkView::IsLoDEnabled () [inline]

Test if the LoD mode is enabled.

Returns:
Return true if the LoD mode is enabled, otherwise return false.

8.302.2.58 bool mitkView::IsModelsModified () [inline]

Check whether there is a modified model

Returns:
Return true if there is a modified model, otherwise Return false

8.302.2.59 void mitkView::MakeCurrent () [inline]

Make the OpenGL context of the container window current

8.302.2.60 virtual void mitkView::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkImageView.

8.302.2.61 void mitkView::RemoveAdditionalLight (mitkLight * *lit*) [inline]

Remove an additional light in this view.

Parameters:
lit specify the light that will be removed from this view

Note:
If lit is found in the list of models, it will be removed from the list. Otherwise nothing happens.

8.302.2.62 void mitkView::RemoveAdditionalLight (int *idx*) [inline]

Remove the additional light in specified position.

Parameters:
idx zero based index to specify the position of the additional light

8.302.2.63 void mitkView::RemoveAllAdditionalLights () [inline]

Remove all additional lights in this view.

8.302.2.64 void mitkView::RemoveAllModel () [inline]

Provided for back compatibility, just the same as SetSliceNum().

8.302.2.65 void mitkView::RemoveAllModels () [inline]

Remove all models in this view.

8.302.2.66 void mitkView::RemoveModel (int *i*) [inline]

Remove the model in specified position.

Parameters:
i Zero based index to specify the position of the model.

8.302.2.67 void mitkView::RemoveModel (mitkModel * *model*) [inline]

Remove a model in this view.

Parameters:

model Specify the model that will be removed from this view. If model is found in the list of models, it will be removed from the list. Otherwise nothing happens.

8.302.2.68 void mitkView::ResetScene () [inline]

Reset the geometric position of all of the models in this view.

8.302.2.69 void mitkView::Rotate (const mitkQuaternion & *q*) [inline]

Rotate all of the models in this view.

Parameters:

q the quaternion of the rotation

Note:

The rotation is incremental.

8.302.2.70 void mitkView::Rotate (float *ax*, float *ay*, float *az*, float *angle*) [inline]

Rotate all of the models in this view around axis (ax,ay,az) for angle degrees.

Parameters:

ax the x-coordinate of rotation axis

ay the y-coordinate of rotation axis

az the z-coordinate of rotation axis

angle the angle of rotation in degrees

Note:

The rotation is incremental.

8.302.2.71 void mitkView::Rotate (float *deltX*, float *deltY*, float *deltZ*) [inline]

Rotate all of the models in this view around x, y, z axis.

Parameters:

deltX Specify the rotation around x-axis in degrees.

deltY Specify the rotation around y-axis in degrees.

deltZ Specify the rotation around z-axis in degrees.

Note:

The rotation is incremental.

8.302.2.72 void mitkView::Scale (float *delt*) [inline]

Scale all of the models in this view in x, y, z directions.

Parameters:

delt Specify the scale in x, y, z directions. This function scales the models equally in each direction.

8.302.2.73 void mitkView::SetBackColor (int *rColor*, int *gColor*, int *bColor*) [inline]

Set the background color of this view.

Parameters:

rColor Specify the red component of the background color. The valid range is from 0 to 255.

gColor Specify the green component of the background color. The valid range is from 0 to 255.

bColor Specify the blue component of the background color. The valid range is from 0 to 255.

8.302.2.74 void mitkView::SetBackColor (float *rColor*, float *gColor*, float *bColor*) [inline]

Set the background color of this view.

Parameters:

rColor Specify the red component of the background color. The valid range is from 0.0 to 1.0.

gColor Specify the green component of the background color. The valid range is from 0.0 to 1.0.

bColor Specify the blue component of the background color. The valid range is from 0.0 to 1.0.

8.302.2.75 void mitkView::SetBackColor (unsigned char *rColor*, unsigned char *gColor*, unsigned char *bColor*) [inline]

Set the background color of this view.

Parameters:

rColor Specify the red component of the background color. The valid range is from 0 to 255.

gColor Specify the green component of the background color. The valid range is from 0 to 255.

bColor Specify the blue component of the background color. The valid range is from 0 to 255.

8.302.2.76 void mitkView::SetCamera (mitkCamera * *cam*) [inline]

Set the camera of this view.

Parameters:

cam the new camera of this view

8.302.2.77 void mitkView::SetDefaultLight (mitkLight * *lit*) [inline]

Set the default light of this view.

Parameters:
lit the new light of this view

Note:
The position of the default light is calculated by the view automatically. It is unchangeable.

8.302.2.78 void mitkView::SetHeight (int *heightSize*) [inline]

Set the Height of this view

Parameters:
heightSize Specify the height of this view.

8.302.2.79 void mitkView::SetLeft (int *leftPos*) [inline]

Set the Left position in local coordinate of its parent window

Parameters:
leftPos Specify the Left coordinate.

8.302.2.80 void mitkView::SetManipulator (mitkManipulator * *mani*)

Set another manipulator to replace the current manipulator. This function provides a chance for the user to override the behavior of mouse events processing.

Parameters:
mani Specify the new manipulator for process the mouse events

8.302.2.81 void mitkView::SetParent (void * *parentId*) [inline]

Set the parent window id.

Parameters:
parentId Parent window Id. This value is OS dependent, and the type in different OS is different. For example, in Windows OS, the type is HWND. You can convert the OS dependent type to void* directly.

8.302.2.82 void mitkView::SetPosition (int *a*[2]) [inline]

Set the position (Left and Top) in local coordinate of its parent window

Parameters:
a[0] Specify the Left coordinate.
a[1] Specify the Top coordinate.

8.302.2.83 void mitkView::SetPosition (int *leftPos*, int *topPos*) [inline]

Set the position (Left and Top) in local coordinate of its parent window

Parameters:

leftPos Specify the Left coordinate.

topPos Specify the Top coordinate.

8.302.2.84 void mitkView::SetRotation (const mitkQuaternion & *q*) [inline]

Set rotations of all models in this view to the quaternion "q".

Parameters:

q the quaternion of the rotation

8.302.2.85 void mitkView::SetRotation (float *ax*, float *ay*, float *az*, float *angle*) [inline]

Set rotations of all models in this view.

Parameters:

ax the x-coordinate of rotation axis

ay the y-coordinate of rotation axis

az the z-coordinate of rotation axis

angle the angle of rotation in degrees

8.302.2.86 void mitkView::SetRotation (float *x*, float *y*, float *z*) [inline]

Set rotations of all models in this view.

Parameters:

x Specify the rotation around x-axis in degrees.

y Specify the rotation around y-axis in degrees.

z Specify the rotation around z-axis in degrees.

8.302.2.87 void mitkView::SetSelected (bool *isSelected*) [inline]

Set the selected status of this view. If view is selected, it will be added a red frame.

Parameters:

isSelected isSelected=true, set the view is selected. isSelected=false, set the view is unselected.

8.302.2.88 void mitkView::SetSize (int *a*[2]) [inline]

Set the size (Width and Height) of this view

Parameters:

a[0] Specify the width of this view.

a[1] Specify the height of this view.

8.302.2.89 void mitkView::SetSize (int *widthSize*, int *heightSize*) [inline]

Set the size (Width and Height) of this view

Parameters:
widthSize Specify the width of this view.
heightSize Specify the height of this view.

8.302.2.90 void mitkView::SetTitleBar (int *titleBar*) [inline]

Set if the window has a title bar

8.302.2.91 void mitkView::SetTitleBarOff () [inline]

Turn off the title bar

8.302.2.92 void mitkView::SetTitleBarOn () [inline]

Turn on the title bar

8.302.2.93 void mitkView::SetTop (int *topPos*) [inline]

Get the Top position in local coordinate of its parent window

Parameters:
topPos Specify the Top coordinate.

8.302.2.94 void mitkView::SetWidth (int *widthSize*) [inline]

Set the Width of this view

Parameters:
widthSize Specify the width of this view.

8.302.2.95 void mitkView::SetWindowName (const char * *winName*) [inline]

Set the name of the window

8.302.2.96 void mitkView::Show () [inline]

Show this view. Usually call it when a view is created. And the later updating of this view should call Update().

8.302.2.97 void mitkView::SwapBuffers () [inline]

Swap the buffers of the container window

8.302.2.98 void mitkView::Translate (float *deltX*, float *deltY*, float *deltZ*) [inline]

Translate all of the models in this view in x, y, z directions.

Parameters:

deltX Specify the translation in x direction

deltY Specify the translation in y direction

deltZ Specify the translation in z direction

8.302.2.99 void mitkView::Update () [inline]

Update the content of this view.

The documentation for this class was generated from the following file:

- mitkView.h

8.303 mitkVis2Scene3D Class Reference

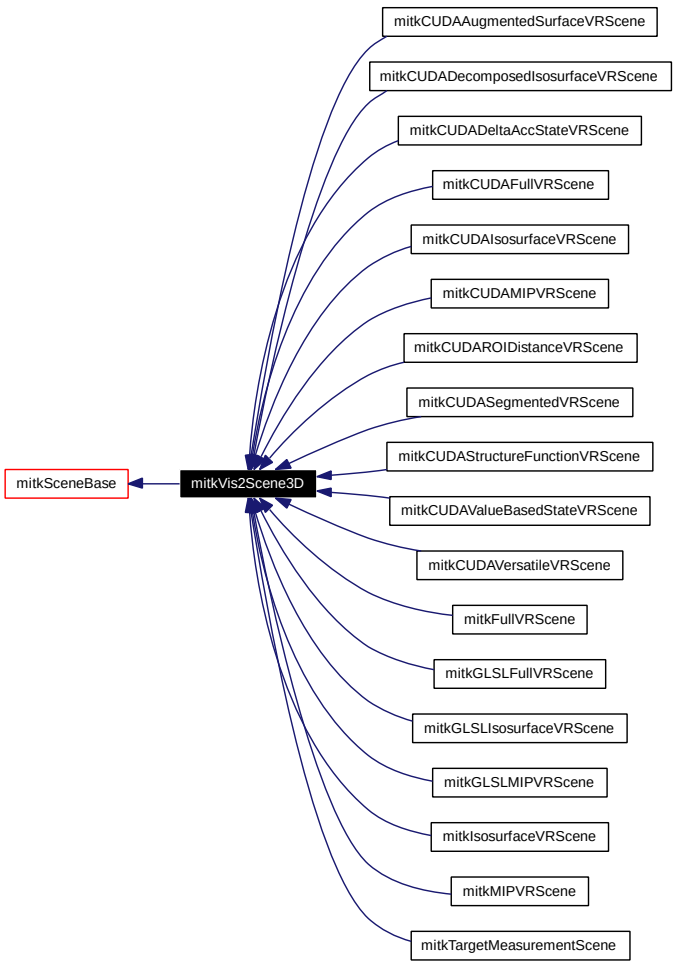
mitkVis2Scene3D - a simple base class for 3D scenes.

`#include <mitkVis2Scene3D.h>`

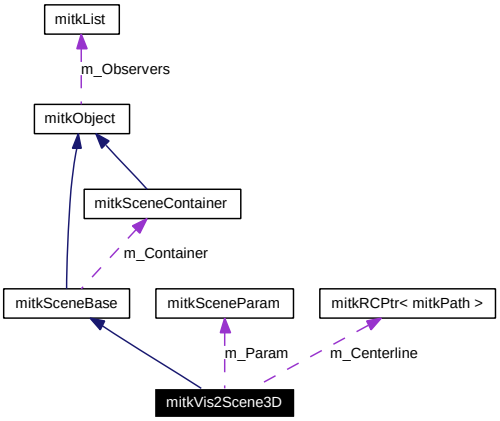
Inherits mitkSceneBase.

Inherited by mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAMIPVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, mitkCUDAVersatileVRScene, mitkCUDAWallCarvingVRScene, mitkCUDAWallPenetrateVRScene, mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkGLSLMIPVRScene, mitkIsosurfaceVRScene, mitkMIPVRScene, mitkPathNetworkScene, mitkTargetMeasurementScene, and mitkVis2SurfaceScene.

Inheritance diagram for mitkVis2Scene3D:



Collaboration diagram for mitkVis2Scene3D:



Public Member Functions

- void SetFovy (float fovy)
- void SetRotateRadPerPixel (float radPerPixel)
- float GetRotateRadPerPixel () const
- void SetScaleRatePerPixel (float scaleRatePerPixel)
- float GetScaleRatePerPixel () const
- virtual void ResizeGL (int newX, int newY)
- virtual void RenderGL ()
- virtual void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void GetTranslate (float translate[3]) const
- float GetScale () const
- const mitkVis2Quaternion & GetRotation () const
- void SetTranslate (const float translate[3])
- void SetScale (float scale)
- void SetRotation (const mitkVis2Quaternion &rotation)
- const mitkVis2Matrix4x4 & GetViewMatrix () const
- const mitkVis2Matrix4x4 & GetProjectionMatrix () const
- float GetZTranslateBase () const
- void GetViewPort (int ∗viewport) const
- float GetFovy () const
- void SetEndoscopy (bool turnOn)
- bool GetEndoscopy ()
- void MoveTo (float x, float y, float z)
- void Move (float dx, float dy, float dz)
- void Turn (float dx, float dy)
- void Roll (float rad)

8.303.1 Detailed Description

mitkVis2Scene3D - a simple base class for 3D scenes.

mitkVis2Scene3D is a simple base class for 3D scenes.

8.303.2 Member Function Documentation

8.303.2.1 bool mitkVis2Scene3D::GetEndoscopy () [inline]

Get the endoscopy status

Returns:
: the endoscopy status

8.303.2.2 float mitkVis2Scene3D::GetFovy () const [inline]

Get the fovy angle.

Returns:
: The fovy angle.

8.303.2.3 const mitkVis2Matrix4x4& mitkVis2Scene3D::GetProjectionMatrix () const

Get the projection matrix of the scene

Returns:
Return the projection matrix of the scene

8.303.2.4 float mitkVis2Scene3D::GetRotateRadPerPixel () const [inline]

Get the interaction sensitivity of rotation.

Returns:
Return the rad the object is rotated when the mouse is moved 1 pixel

8.303.2.5 const mitkVis2Quaternion& mitkVis2Scene3D::GetRotation (void) const [inline]

Get the rotation status of the scene.

Returns:
Return the rotation quaternion of the scene.

8.303.2.6 float mitkVis2Scene3D::GetScale (void) const [inline]

Get the scale status of the scene.

Returns:
Return the scaling rate of the scene.

8.303.2.7 float mitkVis2Scene3D::GetScaleRatePerPixel () const [inline]

Get the interaction sensitivity of scaling

Returns:
Return the rate the object is scaled when the mouse is moved 1 pixel

8.303.2.8 void mitkVis2Scene3D::GetTranslate (float *translate*[3]) const

Get the translate status of the scene.

Parameters:
translate[0] Return the translate of the scene in x-direction.
translate[1] Return the translate of the scene in y-direction.
translate[2] Return the translate of the scene in z-direction.

8.303.2.9 const mitkVis2Matrix4x4& mitkVis2Scene3D::GetViewMatrix () const

Get the view matrix of the scene

Returns:
Return the view matrix of the scene

8.303.2.10 void mitkVis2Scene3D::GetViewPort (int * *viewport*) const [inline]

Get the viewport of this view. Not all of the area of a view is used to display the image. Only the viewport is used to display the image. The position and size of viewport are calculated automatically according to the size of image.

Parameters:
viewport,: the viewport size

8.303.2.11 float mitkVis2Scene3D::GetZTranslateBase () const [inline]

Get the Z direction base translate after scene initialization

Returns:
Return the Z direction base translate

8.303.2.12 void mitkVis2Scene3D::Move (float *dx*, float *dy*, float *dz*)

Endoscopy control: move the view position by (dx,dy,dz) in physics unit

Parameters:
dx
dy
dz

8.303.2.13 void mitkVis2Scene3D::MoveTo (float *x*, float *y*, float *z*)

Endoscopy control: move the view position to (x,y,z) in physics unit

Parameters:

x

y

z

8.303.2.14 virtual void mitkVis2Scene3D::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse pressed event pass by the container

Reimplemented from mitkSceneBase.

Reimplemented in mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkIsosurfaceVRScene, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatileVRScene.

8.303.2.15 virtual void mitkVis2Scene3D::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse moved event pass by the container

Reimplemented from mitkSceneBase.

Reimplemented in mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkIsosurfaceVRScene, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatileVRScene.

8.303.2.16 virtual void mitkVis2Scene3D::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [virtual]

The Interface for handling the mouse released event pass by the container

Reimplemented from mitkSceneBase.

Reimplemented in mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkIsosurfaceVRScene, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatileVRScene.

8.303.2.17 virtual void mitkVis2Scene3D::RenderGL () [virtual]

The Interface for handling the render event pass by the container

Reimplemented from mitkSceneBase.

Reimplemented in mitkFullVRScene, mitkGLSLFullVRScene, mitkGLSLIsosurfaceVRScene, mitkGLSLMIPVRScene, mitkIsosurfaceVRScene, mitkMIPVRScene, mitkTargetMeasurementScene, mitkCUDAAugmentedSurfaceVRScene, mitkCUDADecomposedIsosurfaceVRScene, mitkCUDADeltaAccStateVRScene, mitkCUDAFullVRScene, mitkCUDAIsosurfaceVRScene, mitkCUDAMIPVRScene, mitkCUDAROIDistanceVRScene, mitkCUDASegmentedVRScene, mitkCUDAStructureFunctionVRScene, mitkCUDAValueBasedStateVRScene, and mitkCUDAVersatileVRScene.

8.303.2.18 virtual void mitkVis2Scene3D::ResizeGL (int *newX*, int *newY*) [virtual]

The Interface for handling the resize event pass by the container

Reimplemented from mitkSceneBase.

8.303.2.19 void mitkVis2Scene3D::Roll (float *rad*)

Endoscopy control: roll the screen

Parameters:
rad

8.303.2.20 void mitkVis2Scene3D::SetEndoscopy (bool *turnOn*)

Set the endoscopy status

Parameters:
turnOn,: the endoscopy status

8.303.2.21 void mitkVis2Scene3D::SetFovy (float *fovy*)

Set the fovy angle.

Parameters:
fovy. The fovy angle.

8.303.2.22 void mitkVis2Scene3D::SetRotateRadPerPixel (float *radPerPixel*) [inline]

Set the interaction sensitivity of rotation.

Parameters:
radPerPixel. The rad the object is rotated when the mouse is moved 1 pixel

8.303.2.23 void mitkVis2Scene3D::SetRotation (const mitkVis2Quaternion & *rotation*)

Set the rotation of the scene.

Parameters:
rotation The rotation quaternion of the scene.

8.303.2.24 void mitkVis2Scene3D::SetScale (float *scale*)

Set the scale status of the scene.

Parameters:
scale The scaling rate of the scene.

8.303.2.25 void mitkVis2Scene3D::SetScaleRatePerPixel (float *scaleRatePerPixel*) [inline]

Set the interaction sensitivity of scaling

Parameters:
scaleRatePerPixel. The rate the object is scaled when the mouse is moved 1 pixel

8.303.2.26 void mitkVis2Scene3D::SetTranslate (const float *translate*[3])

Set the translate of the scene.

Parameters:
translate[0] The translate of the scene in x-direction.
translate[1] The translate of the scene in y-direction.
translate[2] The translate of the scene in z-direction.

8.303.2.27 void mitkVis2Scene3D::Turn (float *dx*, float *dy*)

Endoscopy control: turn the view direction

Parameters:
dx
dy

The documentation for this class was generated from the following file:

- mitkVis2Scene3D.h

8.304 mitkVolume Class Reference

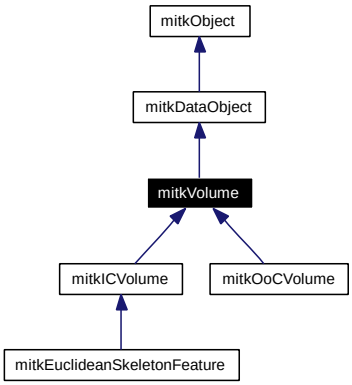
mitkVolume - an abstract class to represent a multi-dimensional medical image dataset

`#include <mitkVolume.h>`

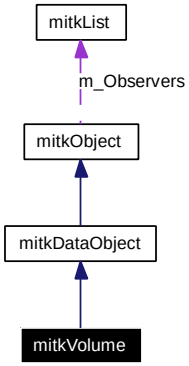
Inherits mitkDataObject.

Inherited by mitkICVolume, and mitkOoCVolume.

Inheritance diagram for mitkVolume:



Collaboration diagram for mitkVolume:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Initialize ()
- virtual int GetDataObjectType () const
- void SetDimensions (int dims[3])
- void GetDimensions (int dims[3]) const
- void SetWidth (int w)
- int GetWidth () const
- void SetHeight (int h)
- int GetHeight () const
- void SetSliceNum (int s)

- void SetImageNum (int s)
- int GetSliceNum () const
- int GetImageNum () const
- int GetDimensionality () const
- void SetSpacings (float s[3])
- void GetSpacings (float s[3]) const
- void SetSpacingX (float px)
- float GetSpacingX () const
- void SetSpacingY (float py)
- float GetSpacingY () const
- void SetSpacingZ (float pz)
- float GetSpacingZ () const
- void SetChannelNum (int n)
- void SetNumberOfChannel (int n)
- int GetChannelNum () const
- int GetNumberOfChannel () const
- void GetIncrements (int incs[3]) const
- int GetIncrementX () const
- int GetIncrementY () const
- int GetIncrementZ () const
- double GetData (int x, int y, int z, int c=0)
- void GetPointGradient (int i, int j, int k, float g[3])
- double GetDataTypeMin () const
- double GetDataTypeMax () const
- void SetWindowWidth (float wWidth)
- void SetWindowCenter (float wCenter)
- float GetWindowWidth ()
- float GetWindowCenter ()
- void GetWidthCenter (float &wWidth, float &wCenter)
- bool HasWidthCenter () const
- int GetDataTypeSize () const
- void SetDataType (int data_type)
- int GetDataType () const
- void SetDataTypeToFloat ()
- void SetDataTypeToDouble ()
- void SetDataTypeToInt ()
- void SetDataTypeToUnsignedInt ()
- void SetDataTypeToLong ()
- void SetDataTypeToUnsignedLong ()
- void SetDataTypeToShort ()
- void SetDataTypeToUnsignedShort ()
- void SetDataTypeToUnsignedChar ()
- void SetDataTypeToChar ()
- void GetMinMaxValue (int sliceIdx, double &minValue, double &maxValue, bool needRecalculate=false)
- void GetMinMaxValue (double &minValue, double &maxValue, bool needRecalculate=false)
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)
- virtual void FreezeSlice (int sliceIdx)
- virtual void UnFreezeSlice (int sliceIdx)

- virtual void const * GetData () const =0
- virtual void * GetData ()=0
- void * GetSliceData (int sliceIdx)
- virtual void const * GetSliceForRead (int sliceIdx)=0
- virtual void * GetSliceForWrite (int sliceIdx)=0
- virtual void * GetSliceForReadWrite (int sliceIdx)=0
- virtual bool ReadSliceData (int sliceIdx, void *dst)=0
- virtual bool ReadYZSliceData (int xIdx, void *dst)=0
- virtual bool ReadXZSliceData (int yIdx, void *dst)=0
- virtual bool GetArbitrarySlice (int w, int h, double o[3], double ux[3], double uy[3], void *dst)=0
- virtual bool WriteSliceData (int sliceIdx, void const *src)=0
- virtual bool ReadSubVolume (int x, int y, int z, int w, int h, int d, int &tw, int &th, int &td, void *dst)=0
- virtual bool WriteSubVolume (int x, int y, int z, int w, int h, int d, int &tw, int &th, int &td, void const *src)=0
- virtual bool Allocate ()=0

8.304.1 Detailed Description

mitkVolume - an abstract class to represent a multi-dimensional medical image dataset

mitkVolume is a very important class in MITK. And most algorithms may use it as input or output. In concept, it is a multi-dimensional dataset. You can access the property and actual data of this dataset through the interface provided by mitkVolume. mitkVolume supports 1D, 2D and 3D dataset, various data type, multi-channel image, and out-of-core data loading. Two examples using mitkVolume are given below.

Example 1: If you want to create a new volume and fill its data, for example, read a file into volume, the code snippet is:

```
// mitkVolume is an abstract class, so you should new mitkICVolume
// (for in-core data) or mitkOoCVolume (for out-of-core data) for actual
// use.
mitkVolume *aVolume = new mitkICVolume; // or mitkOoCVolume
aVolume->SetWidth(sliceWidth);
aVolume->SetHeight(sliceHeight);
aVolume->SetSliceNum(sliceNum);
aVolume->SetSpacingX(sliceSpacingX);
aVolume->SetSpacingY(sliceSpacingY);
aVolume->SetSpacingZ(sliceSpacingZ);
aVolume->SetChannelNum(sliceChannelNum);
aVolume->SetDataType(sliceDataType);
// volume has got enough information, now allocate memory
if (aVolume->Allocate())
{
    Read the content of disk file to aVolume via WriteSliceData();
    Use aVolume ...;
}
```

Example 2: If you want to read/access the content of a volume, the code snippet is:

```
sliceWidth = aVolume->GetWidth();
sliceHeight = aVolume->GetHeight();
sliceNum = aVolume->GetSliceNum();
sliceSpacingX = aVolume->GetSpacingX();
sliceSpacingY = aVolume->GetSpacingY();
sliceSpacingZ = aVolume->GetSpacingZ();
sliceChannelNum = aVolume->GetChannelNum();
```

```
sliceDataType = aVolume->GetDataType();
for(int i = 0; i < imageNum; i++)  //Access every slice data
{
    sliceData = aVolume->GetSliceData(i);
    // or using ReadSliceData() to copy data to a user specified buffer
    Process sliceData according to sliceDataType;
}
```

8.304.2 Member Function Documentation

8.304.2.1 virtual bool mitkVolume::Allocate () [pure virtual]

Allocate necessary space for holding the image data. It calculate the memory size using current Dimensions, DataType and NumberOfChannel settings. The equation is shown as follow:

Width * Height * SliceNum * sizeof(DataType) * NumberOfChannel

The previous allocated data will be deleted if exists.

Returns:
Return true if successful, otherwise return false.

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.2 virtual void mitkVolume::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Implements mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

8.304.2.3 virtual void mitkVolume::FreezeSlice (int *sliceIdx*) [inline, virtual]

Lock the physical memory containing sliceIdx'th slice data. (only useful for out-of-core volume data (mitkOoCVolume) to avoid swapping some slice data from memory buffer to disk buffer, i.e. to keep the returned pointer of some 'Get' functions, such as GetSliceData(), GetSliceForRead() and so on, always be valid until call UnFreezeSlice().

Parameters:
sliceIdx the index of the slice to freeze.

Reimplemented in mitkOoCVolume.

8.304.2.4 virtual bool mitkVolume::GetArbitrarySlice (int *w*, int *h*, double *o*[3], double *ux*[3], double *uy*[3], void * *dst*) [pure virtual]

Get arbitrary directional slice from the volume to a specified memory buffer (i.e. re-slicing).

Parameters:
w width of the new slice (in pixels)

h height of the new slice (in pixels)

o start position (left-bottom) of the new slice in the volume space

ux the step (in pixels) vector of moving to the next pixel along the x-direction of the new slice in the volume space

uy the step (in pixels) vector of moving to the next pixel along the y-direction of the new slice in the volume space

dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be w ∗ h ∗ GetChannelNum() ∗ GetDataTypeSize().

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.5 int mitkVolume::GetChannelNum () const [inline]

Get the channel number of this volume.

Returns:
Return the channel number of this volume.
return 1 — gray image
return 3 — RGB image
return 4 — RGBA image

8.304.2.6 virtual void∗ mitkVolume::GetData () [pure virtual]

Get data pointer of the volume data (changeable).

Returns:
Return a void pointer to data.

Note:
The returned type is void ∗, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.7 virtual void const∗ mitkVolume::GetData () const [pure virtual]

Get data pointer of the volume data (unchangeable).

Returns:
Return a void pointer to const data.

Note:
The returned type is void const ∗, it must be converted to some useful data type according to the return value of GetDataType().

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.8 double mitkVolume::GetData (int *x*, int *y*, int *z*, int *c* = 0)

Get value of a specified voxel's channel.

Parameters:

- ***x*** the x-coordinate of the voxel in the volume space (in [0, GetWidth()-1])
- ***y*** the y-coordinate of the voxel in the volume space (in [0, GetHeight()-1])
- ***z*** the y-coordinate of the voxel in the volume space (in [0, GetImageNum()-1])
- ***c*** the channel of the specified voxel (in [0, GetChannelNum()-1])

Returns:

Return the value of the specified voxel's channel in double data type.

Note:

It is a safe but slow function. Use it carefully if you are concerned about the efficiency.

8.304.2.9 virtual int mitkVolume::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:

Always return MITK_VOLUME

Reimplemented from mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

8.304.2.10 int mitkVolume::GetDataType () const [inline]

Get data type of this volume. MITK supports various data type.

Returns:

All of the return values and their meaning are shown as follows:
MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

8.304.2.11 double mitkVolume::GetDataTypeMax () const [inline]

Get the maximum value of the data type of this volume.

Returns:

Return the maximum value of the data type of this volume.

8.304.2.12 double mitkVolume::GetDataTypeMin () const [inline]

Get the minimum value of the data type of this volume.

Returns:
Return the minimum value of the data type of this volume.

8.304.2.13 int mitkVolume::GetDataTypeSize () const [inline]

Get the size of the data type in bytes.

Returns:
Return the size of the data type in bytes.

8.304.2.14 int mitkVolume::GetDimensionality () const [inline]

Get the dimension of this volume. return 2 — 2D image

return 3 — 3D image

8.304.2.15 void mitkVolume::GetDimensions (int *dims*[3]) const [inline]

Get dimension in x, y, z direction of this volume.

Parameters:
dims[0] Return the dimension in x direction. It is equal to the return value of GetWidth()
dims[1] Return the dimension in y direction. It is equal to the return value of GetHeight()
dims[2] Return the dimension in z direction. It is equal to the return value of GetSliceNum()

8.304.2.16 int mitkVolume::GetHeight () const [inline]

Get the height of this volume (dimension in y).

Returns:
Return the height of this volume, the unit is pixel.

8.304.2.17 int mitkVolume::GetImageNum () const [inline]

Provided for convenience, just the same as GetSliceNum().

8.304.2.18 void mitkVolume::GetIncrements (int *incs*[3]) const [inline]

Get the increments in x, y and z directions.

Parameters:
incs[0] the increment in x directions. It is equal to the return value of GetIncrementX()
incs[1] the increment in y directions. It is equal to the return value of GetIncrementY()
incs[2] the increment in z directions. It is equal to the return value of GetIncrementZ()

8.304.2.19 int mitkVolume::GetIncrementX () const [inline]

Get the increment in x direction.

Returns:
Return the increment in x direction. If the channel number is channelNum, then the return value is channelNum.

Note:
This function is provided for the convenient traversal of the volume. It doesn't take data type of the volume into account. So if you get the data of a volume in void* form, you must convert it to unsigned char*, and the increment must multiply Sizeof(data type of volume).

8.304.2.20 int mitkVolume::GetIncrementY () const [inline]

Get the increment in y direction.

Returns:
Return the increment in y direction. If the channel number is channelNum, then the return value is channelNum * widthOfVolume.

Note:
This function is provided for the convenient traversal of the volume. It doesn't take data type of the volume into account. So if you get the data of a volume in void* form, you must convert it to unsigned char*, and the increment must multiply Sizeof(data type of volume).

8.304.2.21 int mitkVolume::GetIncrementZ () const [inline]

Get the increment in z direction.

Returns:
Return the increment in z direction. If the channel number is channelNum, then the return value is channelNum * widthOfVolume heightOfVolume.

Note:
This function is provided for the convenient traversal of the volume. It doesn't take data type of the volume into account. So if you get the data of a volume in void* form, you must convert it to unsigned char*, and the increment must multiply Sizeof(data type of volume).

8.304.2.22 void mitkVolume::GetMinMaxValue (double & *minValue*, double & *maxValue*, bool *needRecalculate* = false)

Get the minimum and maximum data value of all the volume data.

Parameters:
minValue return the minimum data value

maxValue return the maximum data value

needRecalculate If needRecalculate=true, the minValue and maxValue need be recalculated. Otherwise, the cached value is returned directly.

8.304.2.23 void mitkVolume::GetMinMaxValue (int *sliceIdx*, double & *minValue*, double & *maxValue*, bool *needRecalculate* = `false`)

Get the minimum and maximum data value in a specified slice.

Parameters:

sliceNum The specified slice number.

minValue return the minimum data value in the specified slice.

maxValue return the maximum data value in the specified slice.

needRecalculate If needRecalculate=true, the minValue and maxValue need be recalculated. Otherwise, the cached value is returned directly.

8.304.2.24 int mitkVolume::GetNumberOfChannel () const `[inline]`

Just the same as GetChannelNum().

8.304.2.25 void mitkVolume::GetPointGradient (int *i*, int *j*, int *k*, float *g*[3])

Useless. Obsolete.

8.304.2.26 void* mitkVolume::GetSliceData (int *sliceIdx*) `[inline]`

Provided for back compatibility, just the same as GetSliceForReadWrite().

8.304.2.27 virtual void const* mitkVolume::GetSliceForRead (int *sliceIdx*) `[pure virtual]`

Get data pointer of a specified slice in the volume (for read only).

Parameters:

sliceIdx the index of the slice to get (in [0, GetSliceNum()-1])

Returns:

Return a const void pointer to data.

Note:

The returned type is void const *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:

The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.28 virtual void* mitkVolume::GetSliceForReadWrite (int *sliceIdx*) `[pure virtual]`

Get data pointer of a specified slice in the volume (for read & write).

Parameters:
sliceIdx the index of the slice to get (in [0, GetSliceNum()-1])

Returns:
Return a const void pointer to data.

Note:
The returned type is void *, it must be converted to some useful data type according to the return value of GetDataType().

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.29 virtual void* mitkVolume::GetSliceForWrite (int *sliceIdx*) [pure virtual]

Get data pointer of a specified slice in the volume (for write only).

Parameters:
sliceIdx the index of the slice to get (in [0, GetSliceNum()-1])

Returns:
Return a const void pointer to data.

Note:
The returned type is void *, it should be converted to some useful data type according to the return value of GetDataType(). This function returns an empty buffer for writing the specified slice the first time.

Warning:
The memory is deleted by destructor automatically, so clients shouldn't delete the pointer returned by this function.

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.30 int mitkVolume::GetSliceNum () const [inline]

Get the slice/image number of this volume (dimension in z).

Returns:
Return the image/slice number of this volume.

8.304.2.31 void mitkVolume::GetSpacings (float *s*[3]) const [inline]

Get spacing information in x, y and z axis, the unit is mm.

Parameters:
s[0] Return the spacing (mm) in two adjacent voxels in x axis. It is equal to the return value of GetSpacingX()

s[1] Return the spacing (mm) in two adjacent voxels in y axis. It is equal to the return value of GetSpacingY()

s[2] Return the spacing (mm) in two adjacent voxels in z axis. It is equal to the return value of GetSpacingZ()

8.304.2.32 float mitkVolume::GetSpacingX () const [inline]

Get spacing information in x axis, the unit is mm.

Returns:
Return the spacing (mm) in two adjacent voxels in x axis.

8.304.2.33 float mitkVolume::GetSpacingY () const [inline]

Get spacing information in y axis, the unit is mm.

Returns:
Return the spacing (mm) in two adjacent voxels in y axis.

8.304.2.34 float mitkVolume::GetSpacingZ () const [inline]

Get spacing information in z axis, the unit is mm.

Returns:
Return the spacing (mm) in two adjacent voxels in z axis.

8.304.2.35 int mitkVolume::GetWidth () const [inline]

Get the width of this volume (dimension in x).

Returns:
Return the width of this volume, the unit is pixel.

8.304.2.36 void mitkVolume::GetWidthCenter (float & *wWidth*, float & *wCenter*)

Get the window width and center for display in mitkImageScene.

Parameters:
wWidth Return the window width of this volume.
wCenter Return the window center of this volume.

8.304.2.37 float mitkVolume::GetWindowCenter ()

Get the window center for display in mitkImageScene.

Returns:
Return the window center of this volume.

8.304.2.38 float mitkVolume::GetWindowWidth ()

Get the window width for display in mitkImageScene.

Returns:
Return the window width of this volume.

8.304.2.39 bool mitkVolume::HasWidthCenter () const [inline]

Get the flag which indicates whether this volume has window width and center

Returns:
Return true — This volume has window width/center Return false — This volume has not window width/center

8.304.2.40 virtual void mitkVolume::Initialize () [virtual]

Delete the allocated memory (if any) and initialize to default status.

Implements mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

8.304.2.41 virtual void mitkVolume::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

8.304.2.42 virtual bool mitkVolume::ReadSliceData (int *sliceIdx*, void * *dst*) [pure virtual]

Copy slice data from the volume to a specified memory buffer.

Parameters:
sliceIdx the index of the slice to read (in [0, GetSliceNum()-1])

dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.43 virtual bool mitkVolume::ReadSubVolume (int *x*, int *y*, int *z*, int *w*, int *h*, int *d*, int & *tw*, int & *th*, int & *td*, void * *dst*) [pure virtual]

Copy a sub-volume (start position = (x, y, z), size = w * h * d) from the volume to a specified memory buffer.

Parameters:

x the x-coordinate of the start point

y the y-coordinate of the start point

z the z-coordinate of the start point

w the width of the sub-volume

h the height of the sub-volume

d the depth of the sub-volume

tw return the true width of the sub-volume copied (in case that x+w > GetWidth())

th return the true height of the sub-volume copied (in case that y+h > GetHeight())

td return the true depth of the sub-volume copied (in case that z+d > GetSliceNum())

dst the pointer to the destination memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of destination memory buffer should be w * h * d * GetChannelNum() * GetDataTypeSize().

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.44 virtual bool mitkVolume::ReadXZSliceData (int *yIdx*, void * *dst*) [pure virtual]

Copy y-directional slice from the volume to a specified memory buffer.

Parameters:

yIdx the position of the y-directional slice to read (in [0, GetHeight()-1])

dst the pointer to the destination memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of destination memory buffer should be GetWidth() * GetImageNum() * GetChannelNum() * GetDataTypeSize().

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.45 virtual bool mitkVolume::ReadYZSliceData (int *xIdx*, void * *dst*) [pure virtual]

Copy x-directional slice from the volume to a specified memory buffer.

Parameters:

xIdx the position of the x-directional slice to read (in [0, GetWidth()-1])

dst the pointer to the destination memory buffer

Returns:
Return true if successful, otherwise return false.

Note:
The size of destination memory buffer should be GetHeight() ∗ GetImageNum() ∗ GetChannelNum() ∗ GetDataTypeSize().

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.46 void mitkVolume::SetChannelNum (int *n*) [inline]

Set the channel number of this volume.

Parameters:
n The channel number of this volume.
n = 1 — gray image
n = 3 — RGB image
n = 4 — RGBA image

8.304.2.47 void mitkVolume::SetDataType (int *data_type*)

Set data type of this volume. MITK supports various data type.

Parameters:
data_type Its valid value and meaning is shown as follows:
MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

8.304.2.48 void mitkVolume::SetDataTypeToChar () [inline]

Set data type of this volume to char.

8.304.2.49 void mitkVolume::SetDataTypeToDouble () [inline]

Set data type of this volume to double.

8.304.2.50 void mitkVolume::SetDataTypeToFloat () [inline]

Set data type of this volume to float.

8.304.2.51 void mitkVolume::SetDataTypeToInt () [inline]

Set data type of this volume to int.

8.304.2.52 void mitkVolume::SetDataTypeToLong () [inline]

Set data type of this volume to long.

8.304.2.53 void mitkVolume::SetDataTypeToShort () [inline]

Set data type of this volume to short.

8.304.2.54 void mitkVolume::SetDataTypeToUnsignedChar () [inline]

Set data type of this volume to unsigned char.

8.304.2.55 void mitkVolume::SetDataTypeToUnsignedInt () [inline]

Set data type of this volume to unsigned int.

8.304.2.56 void mitkVolume::SetDataTypeToUnsignedLong () [inline]

Set data type of this volume to unsigned long.

8.304.2.57 void mitkVolume::SetDataTypeToUnsignedShort () [inline]

Set data type of this volume to unsigned short.

8.304.2.58 void mitkVolume::SetDimensions (int *dims*[3]) [inline]

Set dimension in x, y, z direction of this volume.

Parameters:

dims[0] the dimension in x direction

dims[1] the dimension in y direction

dims[2] the dimension in z direction

8.304.2.59 void mitkVolume::SetHeight (int *h*) [inline]

Set the height of this volume (dimension in y).

Parameters:

h The height of this volume, the unit is pixel.

8.304.2.60 void mitkVolume::SetImageNum (int *s*) [inline]

Provided for convenience, just the same as SetSliceNum().

8.304.2.61 void mitkVolume::SetNumberOfChannel (int *n*) [inline]

Just the same as SetChannelNum().

8.304.2.62 void mitkVolume::SetSliceNum (int *s*) [inline]

Set the slice/image number of this volume (dimension in z).

Parameters:

s The image/slice number of this volume.

8.304.2.63 void mitkVolume::SetSpacings (float *s*[3]) [inline]

Set spacing information in x, y and z axis, the unit is mm.

Parameters:

s[0] the spacing (mm) in two adjacent voxels in x axis.

s[1] the spacing (mm) in two adjacent voxels in y axis.

s[2] the spacing (mm) in two adjacent voxels in z axis.

8.304.2.64 void mitkVolume::SetSpacingX (float *px*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:

px the spacing (mm) in two adjacent voxels in x axis.

8.304.2.65 void mitkVolume::SetSpacingY (float *py*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:

py the spacing (mm) in two adjacent voxels in y axis.

8.304.2.66 void mitkVolume::SetSpacingZ (float *pz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:

pz the spacing (mm) in two adjacent voxels in z axis.

8.304.2.67 void mitkVolume::SetWidth (int *w*) [inline]

Set the width of this volume (dimension in x).

Parameters:
w The width of this volume, the unit is pixel.

8.304.2.68 void mitkVolume::SetWindowCenter (float *wCenter*) [inline]

Set the window center for display in mitkImageScene.

Parameters:
wCenter The window center. For some image files, they contain the window width/center information in file header, for example, DICOM files. In this situation, the window width/center is got directly from the file header. In other situations, the window width/center is calculate from the data of the volume.

8.304.2.69 void mitkVolume::SetWindowWidth (float *wWidth*) [inline]

Set the window width for display in mitkImageScene.

Parameters:
wWidth The window width. For some image files, they contain the window width/center information in file header, for example, DICOM files. In this situation, the window width/center is got directly from the file header. In other situations, the window width/center is calculate from the data of the volume.

8.304.2.70 virtual void mitkVolume::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Implements mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

8.304.2.71 virtual void mitkVolume::UnFreezeSlice (int *sliceIdx*) [inline, virtual]

Unlock the physical memory containing sliceIdx'th slice data.

Parameters:
sliceIdx the index of the slice to un-freeze.

See also:
FreezeSlice()

Reimplemented in mitkOoCVolume.

8.304.2.72 virtual bool mitkVolume::WriteSliceData (int *sliceIdx*, void const * *src*) [pure virtual]

Copy slice data from a specified memory buffer to the volume.

Parameters:

sliceIdx the index of the slice to write (in [0, GetImageNum()-1])

src the pointer to the source memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of source memory buffer should be GetWidth() * GetHeight() * GetChannelNum() * GetDataTypeSize().

Implemented in mitkICVolume, and mitkOoCVolume.

8.304.2.73 virtual bool mitkVolume::WriteSubVolume (int *x*, int *y*, int *z*, int *w*, int *h*, int *d*, int & *tw*, int & *th*, int & *td*, void const * *src*) [pure virtual]

Copy a sub-volume (start position = (x, y, z), size = w * h * d) from a specified memory buffer to the volume.

Parameters:

x the x-coordinate of the start point

y the y-coordinate of the start point

z the z-coordinate of the start point

w the width of the sub-volume

h the height of the sub-volume

d the depth of the sub-volume

tw return the true width of the sub-volume copied (in case that x+w > GetWidth())

th return the true height of the sub-volume copied (in case that y+h > GetHeight())

td return the true depth of the sub-volume copied (in case that z+d > GetSliceNum())

src the pointer to the source memory buffer

Returns:

Return true if successful, otherwise return false.

Note:

The size of source memory buffer should be w * h * d * GetChannelNum() * GetDataTypeSize().

Implemented in mitkICVolume, and mitkOoCVolume.

The documentation for this class was generated from the following file:

- mitkVolume.h

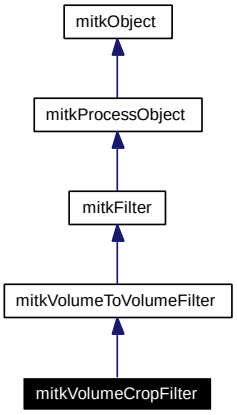
8.305 mitkVolumeCropFilter Class Reference

mitkVolumeCropFilter - A concrete Filter class to crop a volume

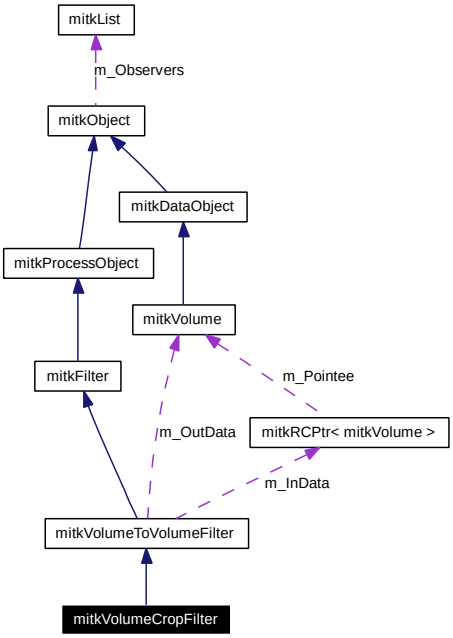
`#include <mitkVolumeCropFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkVolumeCropFilter:



Collaboration diagram for mitkVolumeCropFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetCropPosition (int xbeginp, int ybeginp, int zbeginp, int xendp, int yendp, int zendp)
- void SetCropRegion (int xbeginp, int ybeginp, int zbeginp, int xlength, int ylength, int zlength)

8.305.1 Detailed Description

mitkVolumeCropFilter - A concrete Filter class to crop a volume

mitkVolumeCropFilter is a concrete filter class which inherits from mitkVolumeToVolumeFilter to cut out a given volume. This filter needs a volume input and generates a volume output. So you should first input a volume using public member functin SetInput(), then you should set crop parameters using SetCropPosition(int xbeginp, int ybeginp, int zbeginp, int xendp, int yendp, int zendp), (xbeginp, ybeginp, zbeginp) represents one vertex of the needed crop volume, (xendp, yendp, zendp) represents the diagonal vertexes of the needed crop volume. So you only need to set any pair coordinates of the four diagonal vertexs of the needed crop volume. You can also use SetCropRegion(int xbeginp, int ybeginp, int zbeginp, int xlength, int ylength, int zlength), (xbeginp, ybeginp, zbeginp) represents one vertex of the needed crop volume, xlength, ylength, zlength represents 3 corresponding lengths from this vertex to the diagonal one. The length can be positive or negative but no zero. Two examples using mitkResizeFilter are given below.

Example 1: If you want to crop volume using SetCropPosition() the code snippet is:

```
mitkVolumeCropFilter *filter = new mitkVolumeCropFilter;
filter->SetInput(inVolume);
filter->SetCropPosition(xbeginp,ybeginp,zbeginp,xendp,yendp,zendp);
if (filter->Run())
{
      mitkVolume *outVolume = filter->GetOutput();
      Using  outVolume...
}
```

Example 2: If you want to crop volume using SetCropRegion() the code snippet is:

```
mitkResizeFilter *filter = new mitkResizeFilter;
filter->SetInput(InVolume);
filter->SetCropRegion(xbeginp,ybeginp,zbeginp,xlength,ylength,zlength);
if (filter->Run())
{
    mitkVolume * outVolume = filter->GetOutput();
     Using  outVolume...
}
```

Note:

The cropping volume must be valid ,that is, the specified cropping volume must overlap with the original volume.

8.305.2 Member Function Documentation

8.305.2.1 virtual void mitkVolumeCropFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.305.2.2 void mitkVolumeCropFilter::SetCropPosition (int *xbeginp*, int *ybeginp*, int *zbeginp*, int *xendp*, int *yendp*, int *zendp*)

Set crop parameters using a pair of diagonal vertexes. The vertex coordinate can be out of the definition area(0 $\sim$ width-1,0 $\sim$ height-1,0 $\sim$ imagenember-1).

Parameters:

xbeginp The x coordinate of one vertex(corresponding to the (x+1) column pixel in width dimension)

ybeginp The y coordinate of one vertex(corresponding to the (y+1) row pixel in height dimension)

zbeginp The z coordinate of one vertex(corresponding to the (z+1) vertical pixel in image number dimension)

xendp The x coordinate of the other vertex(corresponding to the (x+1) column pixel in width dimension)

yendp The y coordinate of the other vertex(corresponding to the (y+1) row pixel in height dimension)

zendp The z coordinate of the other vertex(corresponding to the (z+1) vertical pixel in image number dimension)

8.305.2.3 void mitkVolumeCropFilter::SetCropRegion (int *xbeginp*, int *ybeginp*, int *zbeginp*, int *xlength*, int *ylength*, int *zlength*)

Set crop parameters using one vertex and the length from this vertex to the diagonal one . The vertex coordinate can be out of the definition area(0 $\sim$ width-1,0 $\sim$ height-1,0 $\sim$ imagenember-1). And the length can be positive or negative.

Parameters:

xbeginp The x coordinate of one vertex(corresponding to the (x+1) column pixel in width dimension)

ybeginp The y coordinate of one vertex(corresponding to the (y+1) row pixel in height dimension)

zbeginp The z coordinate of one vertex(corresponding to the (z+1) vertical pixel in image number dimension)

xlength The distance from this vertex to the other in x axis(has xlength pixel units)

ylength The distance from this vertex to the other in y axis(has ylength pixel units)

zlength The distance from this vertex to the other in z axis(has zlength pixel units)

The documentation for this class was generated from the following file:

- mitkVolumeCropFilter.h

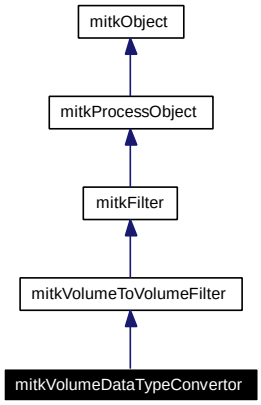
8.306 mitkVolumeDataTypeConvertor Class Reference

mitkVolumeDataTypeConvertor - a filter to convert the type of volume data

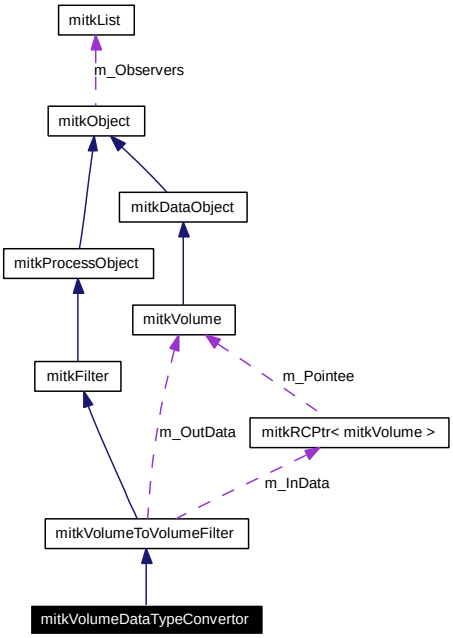
`#include <mitkVolumeDataTypeConvertor.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkVolumeDataTypeConvertor:



Collaboration diagram for mitkVolumeDataTypeConvertor:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetOutputDataType (int dataType)

8.306.1 Detailed Description

mitkVolumeDataTypeConvertor - a filter to convert the type of volume data

mitkVolumeDataTypeConvertor is a filter to convert the volume data from one type to another.

8.306.2 Member Function Documentation

8.306.2.1 virtual void mitkVolumeDataTypeConvertor::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.306.2.2 void mitkVolumeDataTypeConvertor::SetOutputDataType (int *dataType*) [inline]

Set data type of the output volume. MITK supports various data types.

Parameters:
data_type Its valid value and meaning is shown as follows:
MITK_CHAR The data type is char
MITK_UNSIGNED_CHAR The data type is unsigned char
MITK_SHORT The data type is short
MITK_UNSIGNED_SHORT The data type is unsigned short
MITK_INT The data type is int
MITK_UNSIGNED_INT The data type is unsigned int
MITK_LONG The data type is long
MITK_UNSIGNED_LONG The data type is unsigned long
MITK_FLOAT The data type is float
MITK_DOUBLE The data type is double

The documentation for this class was generated from the following file:

- mitkVolumeDataTypeConvertor.h

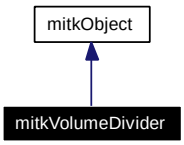
8.307 mitkVolumeDivider Class Reference

mitkVolumeDivider -

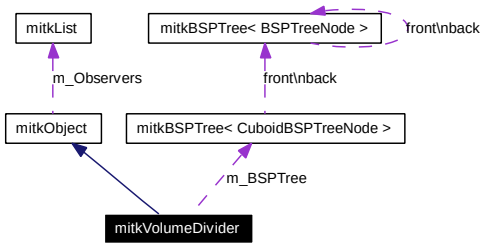
`#include <mitkVolumeDivider.h>`

Inherits mitkObject.

Inheritance diagram for mitkVolumeDivider:



Collaboration diagram for mitkVolumeDivider:



Public Member Functions

- virtual void PrintSelf (ostream &os)

8.307.1 Detailed Description

mitkVolumeDivider -

mitkVolumeDivider

8.307.2 Member Function Documentation

8.307.2.1 virtual void mitkVolumeDivider::PrintSelf (ostream & *os*) `[virtual]`

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkObject.

The documentation for this class was generated from the following file:

- mitkVolumeDivider.h

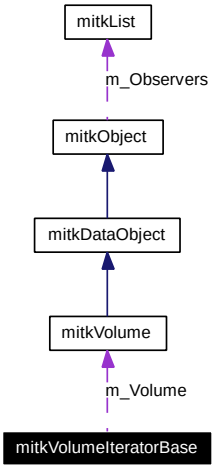
8.308 mitkVolumeIteratorBase Class Reference

mitkVolumeIterator

`#include <mitkVolumeIterator.h>`

Inherited by mitkVolumeIterator< T >.

Collaboration diagram for mitkVolumeIteratorBase:



8.308.1 Detailed Description

mitkVolumeIterator

The documentation for this class was generated from the following file:

- mitkVolumeIterator.h

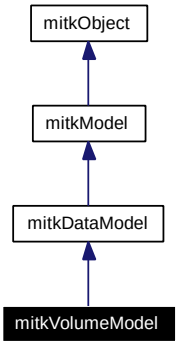
8.309 mitkVolumeModel Class Reference

mitkVolumeModel - an 3D entity in a rendering scene represented in volume

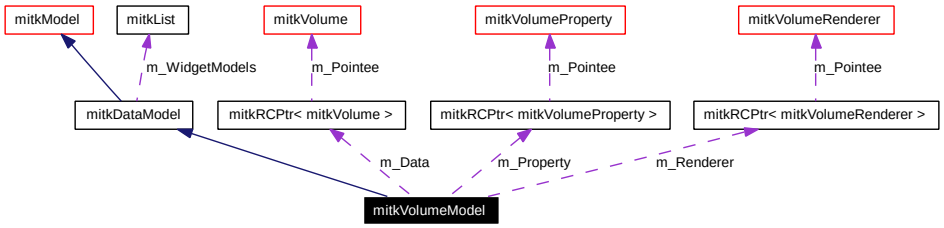
`#include <mitkVolumeModel.h>`

Inherits mitkDataModel.

Inheritance diagram for mitkVolumeModel:



Collaboration diagram for mitkVolumeModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual mitkRenderer * GetBasicRenderer ()
- void SetRenderer (mitkVolumeRenderer *renderer)
- mitkVolumeRenderer * GetRenderer (void)
- void SetProperty (mitkVolumeProperty *prop)
- mitkVolumeProperty * GetProperty (void)
- void SetData (mitkVolume *data)
- mitkVolume * GetData (void)
- virtual int Render (mitkScene *scene)
- virtual bool IsOpaque ()

8.309.1 Detailed Description

mitkVolumeModel - an 3D entity in a rendering scene represented in volume

mitkVolumeModel is an 3D entity in a rendering scene represented in volume. It contains an mitkVolume object which should be shown in the rendering scene. Not like widget model, it is a kind of mitkDataModel and can not be manipulated by itself.

8.309.2 Member Function Documentation

8.309.2.1 virtual mitkRenderer* mitkVolumeModel::GetBasicRenderer () [inline, virtual]

Get the renderer for widget.

Returns:
Return pointer to an mitkRenderer which renders this model actually.

Note:
This function is written for the widget who needs to know the information about the volume model's renderer.

Reimplemented from mitkDataModel.

8.309.2.2 mitkVolume* mitkVolumeModel::GetData (void)

Get the volume data.

Returns:
Return pointer to the volume data.

8.309.2.3 mitkVolumeProperty* mitkVolumeModel::GetProperty (void)

Get the volume property.

Returns:
Return pointer to this volume model's property.

8.309.2.4 mitkVolumeRenderer* mitkVolumeModel::GetRenderer (void)

Get the volume renderer.

Returns:
Return a pointer to this volume model's renderer.

8.309.2.5 virtual bool mitkVolumeModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Implements mitkModel.

8.309.2.6 virtual void mitkVolumeModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkDataModel.

8.309.2.7 virtual int mitkVolumeModel::Render (mitkScene * *scene*) [virtual]

Render this model.

Parameters:
scene the pointer of an mitkScene in which this model will be shown

Returns:
Return 1 if this model is rendered successfully. Otherwise return 0.

Warning:
Don't call this function directly.

Reimplemented from mitkModel.

8.309.2.8 void mitkVolumeModel::SetData (mitkVolume * *data*)

Set the volume data.

Parameters:
data pointer to an mitkVolume

8.309.2.9 void mitkVolumeModel::SetProperty (mitkVolumeProperty * *prop*)

Set the volume property.

Parameters:
prop pointer to an mitkVolumeProperty

8.309.2.10 void mitkVolumeModel::SetRenderer (mitkVolumeRenderer * *renderer*)

Set the volume renderer.

Parameters:
renderer pointer to an mitkVolumeRenderer

The documentation for this class was generated from the following file:

- mitkVolumeModel.h

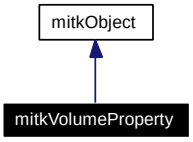
8.310 mitkVolumeProperty Class Reference

mitkVolumeProperty - properties of an mitkVolumeModel

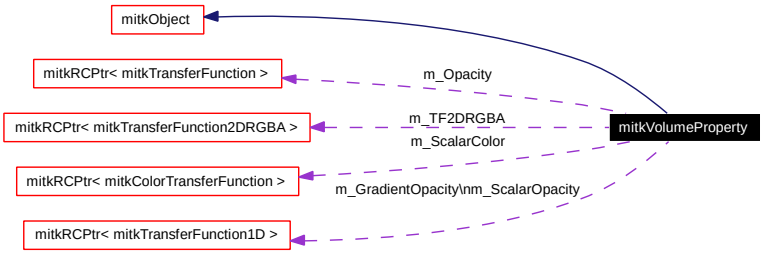
`#include <mitkVolumeProperty.h>`

Inherits mitkObject.

Inheritance diagram for mitkVolumeProperty:



Collaboration diagram for mitkVolumeProperty:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInterpolationType (int intpType)
- int GetInterpolationType ()
- void SetInterpolationTypeToNearest ()
- void SetInterpolationTypeToLinear ()
- const char * GetInterpolationTypeAsString ()
- void SetScalarColor (mitkColorTransferFunction *scFunction)
- void SetScalarOpacity (mitkTransferFunction1D *soFunction)
- void SetGradientOpacity (mitkTransferFunction1D *goFunction)
- void SetTF2DRGBA (mitkTransferFunction2DRGBA *tf2drgba)
- mitkColorTransferFunction * GetScalarColor ()
- mitkTransferFunction1D * GetScalarOpacity ()
- mitkTransferFunction1D * GetGradientOpacity ()
- mitkTransferFunction2DRGBA * GetTF2DRGBA ()
- void CorrectScalarOpacity (float sampleDistance)
- void SetOpacityTransferFunction (mitkTransferFunction *moFunction)
- mitkTransferFunction * GetOpacityTransferFunction ()
- void SetTransferFunctionStyle (TransferFunctionStyle tfStyle)
- TransferFunctionStyle GetTransferFunctionStyle ()
- void SetShade (bool shade)
- int GetShade ()

- void ShadeOn ()
- void ShadeOff ()
- void SetGradientOpacityCalculation (bool goEnable)
- int GetGradientOpacityCalculation ()
- void GradientOpacityCalculationOn ()
- void GradientOpacityCalculationOff ()
- void SetTF2DRGBACalculation (bool TF2DRGBAEnable)
- int GetTF2DRGBACalculation ()
- void TF2DRGBACalculationOn ()
- void TF2DRGBACalculationOff ()
- void SetAmbient (float value)
- float GetAmbient ()
- void SetDiffuse (float value)
- float GetDiffuse ()
- void SetSpecular (float value)
- float GetSpecular ()
- void SetSpecularPower (float value)
- float GetSpecularPower ()
- bool IsModified () const
- void SetUnmodified ()

8.310.1 Detailed Description

mitkVolumeProperty - properties of an mitkVolumeModel

mitkVolumeProperty is a class contains properties of an mitkVolumeModel including shading parameters and transfer functions.

8.310.2 Member Function Documentation

8.310.2.1 void mitkVolumeProperty::CorrectScalarOpacity (float *sampleDistance*)

Correct the scalar-opacity transfer function when the sample distance changed.

Parameters:
sampleDistance Specify the new sample distance

8.310.2.2 float mitkVolumeProperty::GetAmbient () [inline]

Get the ambient lighting coefficient.

Returns:
Return the ambient lighting coefficient.

8.310.2.3 float mitkVolumeProperty::GetDiffuse () [inline]

Get the diffuse lighting coefficient.

Returns:
Return the diffuse lighting coefficient.

8.310.2.4 mitkTransferFunction1D* mitkVolumeProperty::GetGradientOpacity ()

Get the gradient value to opacity transfer function.

Returns:
Return the gradient-opacity transfer function

8.310.2.5 int mitkVolumeProperty::GetGradientOpacityCalculation () [inline]

Get the enableness of Gradient-Opacity transfer function.

Returns:
Return non-zero value, the calculation of Gradient-Opacity transfer function is enabled Return zero, the calculation of Gradient-Opacity transfer function is disabled

8.310.2.6 int mitkVolumeProperty::GetInterpolationType () [inline]

Get the interpolation type for sampling a volume.

Returns:
Return the interpolation type. Return MITK_NEAREST_INTERPOLATION, the interpolation type is nearest interpolation Return MITK_LINEAR_INTERPOLATION, the interpolation type is linear interpolation

8.310.2.7 const char * mitkVolumeProperty::GetInterpolationTypeAsString () [inline]

Get the interpolation type as a C string.

Returns:
Return "Nearest Neighbor" if the interpolation type is nearest interpolation Return "Linear" if the interpolation type is linear interpolation Otherwise return "Unknown"

8.310.2.8 mitkTransferFunction* mitkVolumeProperty::GetOpacityTransferFunction ()

Get the general transfer function (may be multi-dimensional).

Returns:
Return the general transfer function

8.310.2.9 mitkColorTransferFunction* mitkVolumeProperty::GetScalarColor ()

Get the scalar value to color transfer function.

Returns:
Return the scalar-color transfer function

8.310.2.10 mitkTransferFunction1D* mitkVolumeProperty::GetScalarOpacity ()

Get the scalar value to opacity transfer function.

Returns:
Return the scalar-opacity transfer function

8.310.2.11 int mitkVolumeProperty::GetShade () [inline]

Get the status of shading.

Returns:
Return true, the shading is on Return false, the shading is off

8.310.2.12 float mitkVolumeProperty::GetSpecular () [inline]

Get the specular lighting coefficient.

Returns:
Return the specular lighting coefficient.

8.310.2.13 float mitkVolumeProperty::GetSpecularPower () [inline]

Get the specular power.

Returns:
Return the specular power.

8.310.2.14 mitkTransferFunction2DRGBA* mitkVolumeProperty::GetTF2DRGBA ()

Get the scalar value and gradient Magnitudes value to opacity transfer function.

Returns:
Return the Scalar-GradientMagnitudes-opacity transfer function

8.310.2.15 int mitkVolumeProperty::GetTF2DRGBACalculation () [inline]

Get the enableness of 2-D RGB and Opacity transfer function.

Returns:
Return non-zero value, the calculation of 2-D RGB and Opacity transfer function is enabled Return zero, the calculation of 2-D RGB and Opacity transfer function is disabled

8.310.2.16 TransferFunctionStyle mitkVolumeProperty::GetTransferFunctionStyle () [inline]

Get the style of transfer function.

Returns:
Return the style of transfer function.

See also:
TransferFunctionStyle

8.310.2.17 void mitkVolumeProperty::GradientOpacityCalculationOff () [inline]

Disable the calculation of Gradient-Opacity transfer function.

8.310.2.18 void mitkVolumeProperty::GradientOpacityCalculationOn () [inline]

Enable the calculation of Gradient-Opacity transfer function.

8.310.2.19 bool mitkVolumeProperty::IsModified () const

Test if some of the properties are modified.

Returns:
Return true if some of the properties are modified. Otherwise return false.

8.310.2.20 virtual void mitkVolumeProperty::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

8.310.2.21 void mitkVolumeProperty::SetAmbient (float *value*) [inline]

Set the ambient lighting coefficient.

Parameters:
value Specify the ambient lighting coefficient.

8.310.2.22 void mitkVolumeProperty::SetDiffuse (float *value*) [inline]

Set the diffuse lighting coefficient.

Parameters:
value Specify the diffuse lighting coefficient.

8.310.2.23 void mitkVolumeProperty::SetGradientOpacity (mitkTransferFunction1D * *goFunction*)

Set the gradient value to opacity transfer function.

Parameters:
goFunction Specify the gradient-opacity transfer function

8.310.2.24 void mitkVolumeProperty::SetGradientOpacityCalculation (bool *goEnable*) [inline]

Set the enableness of Gradient-Opacity transfer function.

Parameters:
goEnable goEnable = true, enable the calculation of Gradient-Opacity transfer function goEnable = false, disable the calculation of Gradient-Opacity transfer function

8.310.2.25 void mitkVolumeProperty::SetInterpolationType (int *intpType*) [inline]

Set the interpolation type for sampling a volume.

Parameters:
intpType Specify the interpolation type. intpType = MITK_NEAREST_INTERPOLATION, set the interpolation type to nearest interpolation intpType = MITK_LINEAR_INTERPOLATION, set the interpolation type to linear interpolation

8.310.2.26 void mitkVolumeProperty::SetInterpolationTypeToLinear () [inline]

Set the interpolation type to linear interpolation.

8.310.2.27 void mitkVolumeProperty::SetInterpolationTypeToNearest () [inline]

Set the interpolation type to nearest interpolation.

8.310.2.28 void mitkVolumeProperty::SetOpacityTransferFunction (mitkTransferFunction * *moFunction*)

Set the general transfer function (may be multi-dimensional).

Parameters:
moFunction Specify the general transfer function

8.310.2.29 void mitkVolumeProperty::SetScalarColor (mitkColorTransferFunction * *scFunction*)

Set the scalar value to color transfer function.

Parameters:
scFunction Specify the scalar-color transfer function

8.310.2.30 void mitkVolumeProperty::SetScalarOpacity (mitkTransferFunction1D * *soFunction*)

Set the scalar value to opacity transfer function.

Parameters:
soFunction Specify the scalar-opacity transfer function

8.310.2.31 void mitkVolumeProperty::SetShade (bool *shade*) [inline]

Set the status of shading.

Parameters:
shade shade = true, turn the shading on shade = false, turn the shading off

8.310.2.32 void mitkVolumeProperty::SetSpecular (float *value*) [inline]

Set the specular lighting coefficient.

Parameters:
value Specify the specular lighting coefficient.

8.310.2.33 void mitkVolumeProperty::SetSpecularPower (float *value*) [inline]

Set the specular power.

Parameters:
value Specify the specular power.

8.310.2.34 void mitkVolumeProperty::SetTF2DRGBA (mitkTransferFunction2DRGBA * *tf2drgba*)

Set the scalar value and gradient Magnitudes value to 2D R, G, B and opacity transfer function.

Parameters:
sgmoFunction Specify the 2D RGBOpacity transfer function

8.310.2.35 void mitkVolumeProperty::SetTF2DRGBACalculation (bool *TF2DRGBAEnable*) [inline]

Set the enableness of 2 dimensional RGB and Opacity transfer function.

Parameters:
goEnable TF2DRGBAEnable = true, enable the calculation of 2-D RGB and Opacity transfer function
TF2DRGBAEnable = false, disable the calculation of 2-D RGB and Opacity transfer function

8.310.2.36 void mitkVolumeProperty::SetTransferFunctionStyle (TransferFunctionStyle *tfStyle*) [inline]

Set the style of transfer function.

Parameters:
tfStyle Specify the style of transfer function.

See also:
TransferFunctionStyle

8.310.2.37 void mitkVolumeProperty::SetUnmodified ()

Reset to unmodified after changes have been done according to the new properties.

8.310.2.38 void mitkVolumeProperty::ShadeOff () [inline]

Turn the shading off.

8.310.2.39 void mitkVolumeProperty::ShadeOn () [inline]

Turn the shading on.

8.310.2.40 void mitkVolumeProperty::TF2DRGBACalculationOff () [inline]

Disable the calculation of 2-D RGB and Opacity transfer function.

8.310.2.41 void mitkVolumeProperty::TF2DRGBACalculationOn () [inline]

Enable the calculation of 2-D RGB and Opacity transfer function.

The documentation for this class was generated from the following file:

- mitkVolumeProperty.h

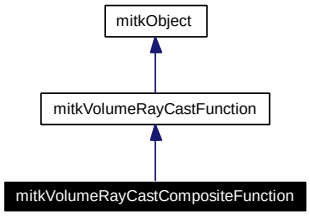
8.311 mitkVolumeRayCastCompositeFunction Class Reference

mitkVolumeRayCastCompositeFunction - a concrete ray cast function for compositing

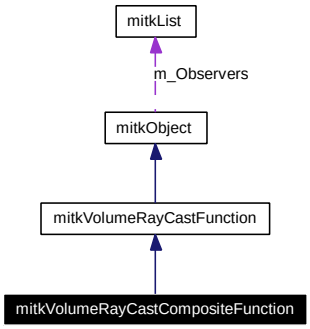
`#include <mitkVolumeRayCastCompositeFunction.h>`

Inherits mitkVolumeRayCastFunction.

Inheritance diagram for mitkVolumeRayCastCompositeFunction:



Collaboration diagram for mitkVolumeRayCastCompositeFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void CastRay (mitkRay *rayInfo)

8.311.1 Detailed Description

mitkVolumeRayCastCompositeFunction - a concrete ray cast function for compositing

mitkVolumeRayCastCompositeFunction is a concrete ray cast function by using the composition. It implements the CastRay() function defined in base class.

8.311.2 Member Function Documentation

8.311.2.1 virtual void mitkVolumeRayCastCompositeFunction::CastRay (mitkRay * *rayInfo*) [virtual]

Calculate the ray cast function using composition.

Parameters:
rayInfo Specify the ray information required by the calcualtion.

Implements mitkVolumeRayCastFunction.

8.311.2.2 virtual void mitkVolumeRayCastCompositeFunction::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRayCastFunction.

The documentation for this class was generated from the following file:

- mitkVolumeRayCastCompositeFunction.h

8.312 mitkVolumeRayCastFunction Class Reference

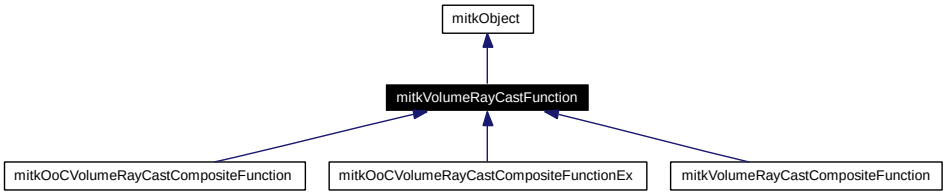
mitkVolumeRayCastFunction - an abstract class for calculating the ray casting function.

`#include <mitkVolumeRayCastFunction.h>`

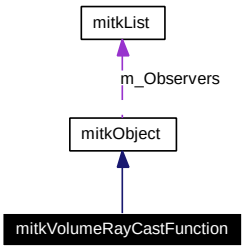
Inherits mitkObject.

Inherited by mitkOoCVolumeRayCastCompositeFunction, mitkOoCVolumeRayCastCompositeFunction-Ex, and mitkVolumeRayCastCompositeFunction.

Inheritance diagram for mitkVolumeRayCastFunction:



Collaboration diagram for mitkVolumeRayCastFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void CastRay (mitkRay *rayInfo)=0

8.312.1 Detailed Description

mitkVolumeRayCastFunction - an abstract class for calculating the ray casting function.

mitkVolumeRayCastFunction is an abstract class for calculating the ray casting function. Its subclasses must implement CastRay() to calculate the ray cast function.

8.312.2 Member Function Documentation

8.312.2.1 virtual void mitkVolumeRayCastFunction::CastRay (mitkRay * *rayInfo*) [pure virtual]

Calculate the ray cast function.

Parameters:
rayInfo Specify the ray information required by the calcualtion.

Note:
All of its subclasses must implement this pure virtual to calculate the ray cast function.

Implemented in mitkOoCVolumeRayCastCompositeFunction, mitkOoCVolumeRayCastCompositeFunctionEx, and mitkVolumeRayCastCompositeFunction.

8.312.2.2 virtual void mitkVolumeRayCastFunction::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkOoCVolumeRayCastCompositeFunction, mitkOoCVolumeRayCastCompositeFunctionEx, and mitkVolumeRayCastCompositeFunction.

The documentation for this class was generated from the following file:

- mitkVolumeRayCastFunction.h

8.313 mitkVolumeReader Class Reference

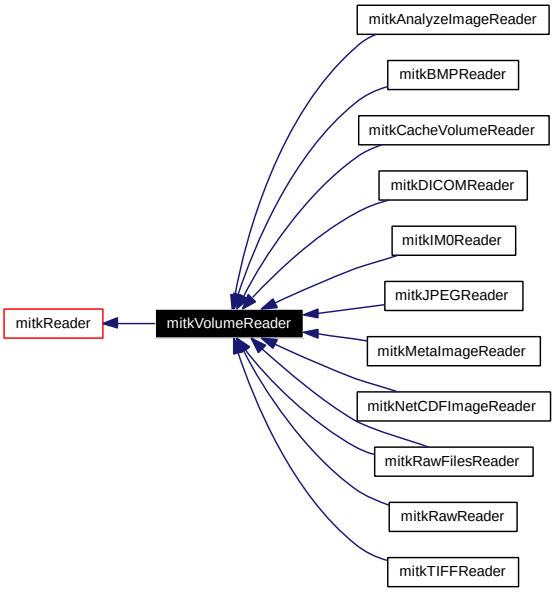
mitkVolumeReader - an abstract class represents a volume reader to read image/volume files

`#include <mitkVolumeReader.h>`

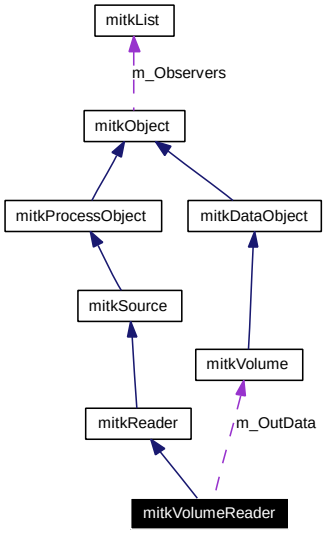
Inherits mitkReader.

Inherited by mitkAnalyzeImageReader, mitkBMPReader, mitkCacheVolumeReader, mitkDICOMReader, mitkIM0Reader, mitkJPEGReader, mitkMetaImageReader, mitkNetCDFImageReader, mitkRawFiles-Reader, mitkRawFilesReader, mitkRawReader, and mitkTIFFReader.

Inheritance diagram for mitkVolumeReader:



Collaboration diagram for mitkVolumeReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetOoCSupport (char const *diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufSliceNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)
- mitkVolume * GetOutput ()

8.313.1 Detailed Description

mitkVolumeReader - an abstract class represents a volume reader to read image/volume files

mitkVolumeReader defines the interface of all of the volume readers. To use a concrete volume reader, for example, mitkBMPReader, the code snippet is:

```
mitkBMPReader *aReader = new mitkBMPReader;
aReader->AddFileName(file1);
aReader->AddFileName(file2);
...  ...  ...
if (aReader->Run())
{
    mitkVolume *aVolume = aReader->GetOutput();
    Using aVolume
}
```

8.313.2 Member Function Documentation

8.313.2.1 mitkVolume* mitkVolumeReader::GetOutput ()

Get the output volume the reader has read.

Returns:
the output volume.

8.313.2.2 virtual void mitkVolumeReader::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkReader.

Reimplemented in mitkAnalyzeImageReader, mitkBMPReader, mitkCacheVolumeReader, mitkDICOMReader, mitkJPEGReader, mitkMetaImageReader, mitkNetCDFImageReader, mitkRawFilesReader, mitkRawReader, mitkTIFFReader, and mitkRawFilesReader.

8.313.2.3 void mitkVolumeReader::SetOoCSupport (char const * *diskPath* = mitkOoCGlobalSettings::DiskCachePath, unsigned int *bufSliceNum* = mitkOoCGlobalSettings::BufferedSliceNumber, bool *supportOoC* = true)

Let the reader support out-of-core volume data.

Parameters:

diskPath the path in the disk to cache the volume data

bufSliceNum the number of slices to cache in the main memory

supportOoC whether to turn on out-of-core support

Note:

The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

The documentation for this class was generated from the following file:

- mitkVolumeReader.h

8.314 mitkVolumeRenderer Class Reference

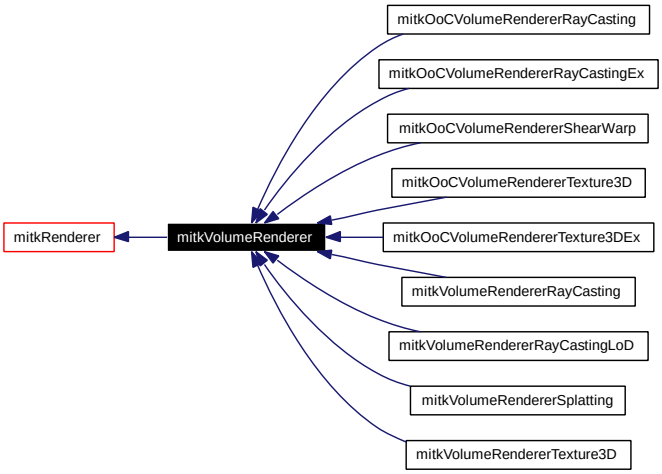
mitkVolumeRenderer - an abstract class for volume rendering

`#include <mitkVolumeRenderer.h>`

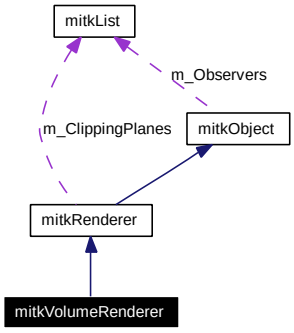
Inherits mitkRenderer.

Inherited by mitkOoCVolumeRendererRayCasting, mitkOoCVolumeRendererRayCastingEx, mitkOoCVolumeRendererShearWarp, mitkOoCVolumeRendererTexture3D, mitkOoCVolumeRendererTexture3DEx, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, and mitkVolumeRendererTexture3D.

Inheritance diagram for mitkVolumeRenderer:



Collaboration diagram for mitkVolumeRenderer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetCropping (bool isCropping)
- void CroppingOn ()
- void CroppingOff ()
- int GetCropping ()

- void SetCroppingRegionPlanes (float region[6])
- void SetCroppingRegionPlanes (float xmin, float xmax, float ymin, float ymax, float zmin, float zmax)
- float * GetCroppingRegionPlanes ()
- void GetCroppingRegionPlanes (float &xmin, float &xmax, float &ymin, float &ymax, float &zmin, float &zmax)
- void SetCroppingRegionFlags (int flags)
- int GetCroppingRegionFlags ()
- void SetCroppingRegionFlagsToSubVolume ()
- void SetCroppingRegionFlagsToFence ()
- void SetCroppingRegionFlagsToInvertedFence ()
- void SetCroppingRegionFlagsToCross ()
- void SetCroppingRegionFlagsToInvertedCross ()
- void SetClassifyMethod (int classifyMethod)
- int GetClassifyMethod ()
- void SetClassifyMethodToPreClassification ()
- void SetClassifyMethodToPostClassification ()
- void SetClassifyMethodToPreIntegration ()
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)=0

8.314.1 Detailed Description

mitkVolumeRenderer - an abstract class for volume rendering

mitkVolumeRenderer is an abstract class for volume rendering. Its concrete subclasses implement the actual volume rendering algorithm. Some codes are borrowed from VTK, and please see the copyright at end.

8.314.2 Member Function Documentation

8.314.2.1 void mitkVolumeRenderer::CroppingOff () [inline]

Turn Off cropping

8.314.2.2 void mitkVolumeRenderer::CroppingOn () [inline]

Turn On cropping

8.314.2.3 int mitkVolumeRenderer::GetClassifyMethod () [inline]

Get the classification method used by volume rendering algorithm.

Returns:

Return the classification method. Return MITK_PRE_CLASSIFICATION, the classfication method is PreClassification (classify first). Return MITK_POST_CLASSIFICATION, the classfication method is PostClassification (interpolation first). Return MITK_PRE_INTEGRATION, the classfication method is Pre-integration.

8.314.2.4 int mitkVolumeRenderer::GetCropping () [inline]

Get the status (On or Off) of cropping.

Returns:
Return non-zero value, cropping is On. Return zero, cropping is Off.

8.314.2.5 int mitkVolumeRenderer::GetCroppingRegionFlags () [inline]

Get the flags for the cropping regions.

Returns:
Return the cropping flag

8.314.2.6 void mitkVolumeRenderer::GetCroppingRegionPlanes (float & *xmin*, float & *xmax*, float & *ymin*, float & *ymax*, float & *zmin*, float & *zmax*) [inline]

Get the Cropping Region Planes (xmin, xmax, ymin, ymax, zmin, zmax)

Parameters:
xmin Return the xmin of cropping region
xmax Return the xmax of cropping region
ymin Return the ymin of cropping region
ymax Return the ymax of cropping region
zmin Return the zmin of cropping region
zmax Return the zmax of cropping region

8.314.2.7 float* mitkVolumeRenderer::GetCroppingRegionPlanes () [inline]

Get the Cropping Region Planes (xmin, xmax, ymin, ymax, zmin, zmax)

Returns:
Return a float array, the elements is xmin, xmax, ymin, ymax, zmin, zmax in turn.

8.314.2.8 virtual void mitkVolumeRenderer::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkRenderer.

Reimplemented in mitkOoCVolumeRendererRayCasting, mitkOoCVolumeRendererRayCastingEx, mitkOoCVolumeRendererShearWarp, mitkOoCVolumeRendererTexture3D, mitkOoCVolumeRenderer-Texture3DEx, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolume-RendererSplatting, and mitkVolumeRendererTexture3D.

8.314.2.9 virtual int mitkVolumeRenderer::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [pure virtual]

Internal function. Don't call it directly.

Implemented in mitkOoCVolumeRendererRayCasting, mitkOoCVolumeRendererRayCastingEx, mitkOoCVolumeRendererShearWarp, mitkOoCVolumeRendererTexture3D, mitkOoCVolumeRendererTexture3DEx, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, and mitkVolumeRendererTexture3D.

8.314.2.10 void mitkVolumeRenderer::SetClassifyMethod (int *classifyMethod*) [inline]

Set the classification method used by volume rendering algorithm.

Parameters:

classifyMethod Specify the classification method. classifyMethod = MITK_PRE_CLASSIFICATION, set the classfication method to PreClassification (classify first). classifyMethod = MITK_POST_CLASSIFICATION, set the classfication method to PostClassification (interpolation first). classifyMethod = MITK_PRE_INTEGRATION, set the classfication method to Pre-integration.

8.314.2.11 void mitkVolumeRenderer::SetClassifyMethodToPostClassification () [inline]

Set the classification method to PostClassification (interpolation first).

8.314.2.12 void mitkVolumeRenderer::SetClassifyMethodToPreClassification () [inline]

Set the classification method to PreClassification (classify first).

8.314.2.13 void mitkVolumeRenderer::SetClassifyMethodToPreIntegration () [inline]

Set the classification method to Pre-integration.

8.314.2.14 void mitkVolumeRenderer::SetCropping (bool *isCropping*) [inline]

Turn On / Off cropping

Parameters:

isCropping isCropping = true, turn on cropping isCropping = false, turn off cropping

8.314.2.15 void mitkVolumeRenderer::SetCroppingRegionFlags (int *flags*) [inline]

Set the flags for the cropping regions.

Parameters:

flags Specify the cropping flag

8.314.2.16 void mitkVolumeRenderer::SetCroppingRegionFlagsToCross () [inline]

Set the flags for the cropping regions to Cross

8.314.2.17 void mitkVolumeRenderer::SetCroppingRegionFlagsToFence () [inline]

Set the flags for the cropping regions to Fence

8.314.2.18 void mitkVolumeRenderer::SetCroppingRegionFlagsToInvertedCross () [inline]

Set the flags for the cropping regions to InvertedCross

8.314.2.19 void mitkVolumeRenderer::SetCroppingRegionFlagsToInvertedFence () [inline]

Set the flags for the cropping regions to InvertedFence

8.314.2.20 void mitkVolumeRenderer::SetCroppingRegionFlagsToSubVolume () [inline]

Set the flags for the cropping regions to SubVolume

8.314.2.21 void mitkVolumeRenderer::SetCroppingRegionPlanes (float *xmin*, float *xmax*, float *ymin*, float *ymax*, float *zmin*, float *zmax*) [inline]

Set the Cropping Region Planes (xmin, xmax, ymin, ymax, zmin, zmax)

Parameters:

xmin Specify the xmin of cropping region
xmax Specify the xmax of cropping region
ymin Specify the ymin of cropping region
ymax Specify the ymax of cropping region
zmin Specify the zmin of cropping region
zmax Specify the zmax of cropping region

8.314.2.22 void mitkVolumeRenderer::SetCroppingRegionPlanes (float *region*[6]) [inline]

Set the Cropping Region Planes (xmin, xmax, ymin, ymax, zmin, zmax)

Parameters:

region[0] Specify the xmin of cropping region
region[1] Specify the xmax of cropping region
region[2] Specify the ymin of cropping region
region[3] Specify the ymax of cropping region
region[4] Specify the zmin of cropping region
region[5] Specify the zmax of cropping region

The documentation for this class was generated from the following file:

- mitkVolumeRenderer.h

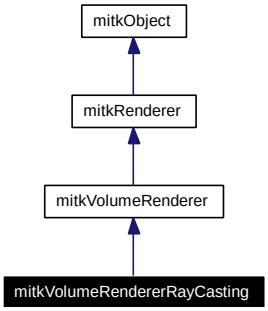
8.315 mitkVolumeRendererRayCasting Class Reference

mitkVolumeRendererRayCasting - a concrete volume renderer for rendering a volume

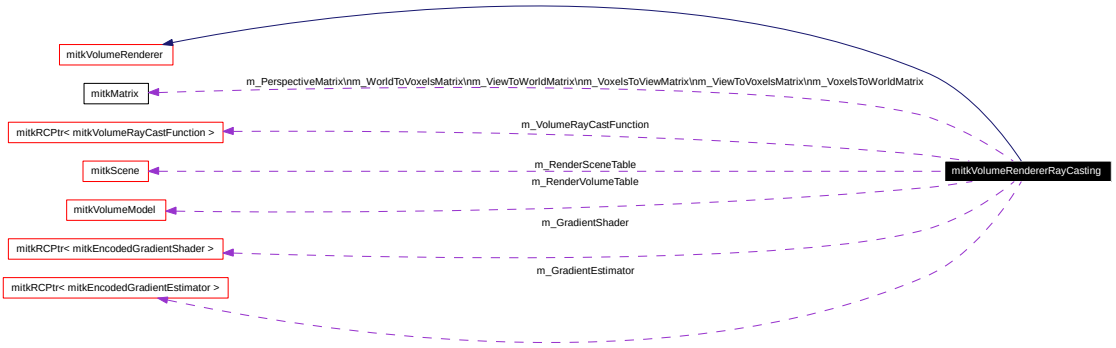
`#include <mitkVolumeRendererRayCasting.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkVolumeRendererRayCasting:



Collaboration diagram for mitkVolumeRendererRayCasting:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetSampleDistance (float fVal)
- float GetSampleDistance ()
- void SetVolumeRayCastFunction (mitkVolumeRayCastFunction *oVal)
- mitkVolumeRayCastFunction * GetVolumeRayCastFunction ()
- void SetGradientEstimator (mitkEncodedGradientEstimator *gradest)
- mitkEncodedGradientEstimator * GetGradientEstimator ()
- mitkEncodedGradientShader * GetEncodedGradientShader ()
- float GetImageSampleDistanceMinValue ()
- float GetImageSampleDistanceMaxValue ()

- void SetImageSampleDistance (float fVal)
- float GetImageSampleDistance ()
- float GetMinimumImageSampleDistanceMinValue ()
- float GetMinimumImageSampleDistanceMaxValue ()
- float GetMaximumImageSampleDistanceMinValue ()
- float GetMaximumImageSampleDistanceMaxValue ()
- void SetMinimumImageSampleDistance (float fVal)
- void SetMaximumImageSampleDistance (float fVal)
- float GetMinimumImageSampleDistance ()
- float GetMaximumImageSampleDistance ()
- void SetAutoAdjustSampleDistances (int val)
- int GetAutoAdjustSampleDistances ()
- void AutoAdjustSampleDistancesOn ()
- void AutoAdjustSampleDistancesOff ()
- void SetIntermixIntersectingGeometry (int fVal)
- int GetIntermixIntersectingGeometry ()
- void IntermixIntersectingGeometryOn ()
- void IntermixIntersectingGeometryOff ()
- virtual float GetGradientMagnitudeScale ()
- virtual float GetGradientMagnitudeBias ()

8.315.1 Detailed Description

mitkVolumeRendererRayCasting - a concrete volume renderer for rendering a volume

mitkVolumeRendererRayCasting is a concrete volume renderer for rendering a volume. Some codes are borrowed from VTK, and please see the copyright at end.

8.315.2 Member Function Documentation

8.315.2.1 void mitkVolumeRendererRayCasting::AutoAdjustSampleDistancesOff () [inline]

Set sample distances auto-adjusting off.

8.315.2.2 void mitkVolumeRendererRayCasting::AutoAdjustSampleDistancesOn () [inline]

Set sample distances auto-adjusting on.

8.315.2.3 int mitkVolumeRendererRayCasting::GetAutoAdjustSampleDistances () [inline]

Get whether to auto-adjust the sample distances.

Returns:

Return a value which indicates whether to auto-adjust the sample distances (1 means true, 0 means false).

8.315.2.4 mitkEncodedGradientShader* mitkVolumeRendererRayCasting::GetEncodedGradientShader () [inline]

Get the gradient shader

Returns:
Return the gradient shader

8.315.2.5 mitkEncodedGradientEstimator* mitkVolumeRendererRayCasting::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals

Returns:
Return the gradient estimator used to estimate normals.

8.315.2.6 virtual float mitkVolumeRendererRayCasting::GetGradientMagnitudeBias () [virtual]

Get gradient magnitude bias.

Returns:
Return the gradient magnitude bias.

Reimplemented from mitkVolumeRenderer.

8.315.2.7 virtual float mitkVolumeRendererRayCasting::GetGradientMagnitudeScale () [virtual]

Get gradient magnitude scale.

Returns:
Return the gradient magnitude scale.

Reimplemented from mitkVolumeRenderer.

8.315.2.8 float mitkVolumeRendererRayCasting::GetImageSampleDistance () [inline]

Get the sample distance in the image plane.

Returns:
Return the sample distance in the image plane.

8.315.2.9 float mitkVolumeRendererRayCasting::GetImageSampleDistanceMaxValue () [inline]

Get the maximum value of the sample distance in the image plane.

Returns:
Return the maximum value of the sample distance in the image plane.

8.315.2.10 float mitkVolumeRendererRayCasting::GetImageSampleDistanceMinValue () [inline]

Get the minimum value of the sample distance in the image plane.

Returns:
Return the minimum value of the sample distance in the image plane.

8.315.2.11 int mitkVolumeRendererRayCasting::GetIntermixIntersectingGeometry () [inline]

Get the intermix intersecting geometry value

Returns:
Return the intermix intersecting geometry value

8.315.2.12 float mitkVolumeRendererRayCasting::GetMaximumImageSampleDistance () [inline]

Get the maximum sample distance in the image plane.

Returns:
Return the maximum sample distance in the image plane.

8.315.2.13 float mitkVolumeRendererRayCasting::GetMaximumImageSampleDistanceMaxValue () [inline]

Get the maximum value of the maximum sample distance in the image plane.

Returns:
Return the maximum value of the maximum sample distance in the image plane.

8.315.2.14 float mitkVolumeRendererRayCasting::GetMaximumImageSampleDistanceMinValue () [inline]

Get the minimum value of the maximum sample distance in the image plane.

Returns:
Return the minimum value of the maximum sample distance in the image plane.

8.315.2.15 float mitkVolumeRendererRayCasting::GetMinimumImageSampleDistance () [inline]

Get the minimum sample distance in the image plane.

Returns:
Return the minimum sample distance in the image plane.

8.315.2.16 float mitkVolumeRendererRayCasting::GetMinimumImageSampleDistanceMaxValue () [inline]

Get the maximum value of the minimum sample distance in the image plane.

Returns:
Return the maximum value of the minimum sample distance in the image plane.

8.315.2.17 float mitkVolumeRendererRayCasting::GetMinimumImageSampleDistanceMinValue () [inline]

Get the minimum value of the minimum sample distance in the image plane.

Returns:
Return the minimum value of the minimum sample distance in the image plane.

8.315.2.18 float mitkVolumeRendererRayCasting::GetSampleDistance () [inline]

Get the sample distance in the ray direction.

Returns:
Return the sample distance.

8.315.2.19 mitkVolumeRayCastFunction* mitkVolumeRendererRayCasting::GetVolumeRayCastFunction () [inline]

Get the volume ray cast function.

Returns:
Return the volume ray cast function.

8.315.2.20 void mitkVolumeRendererRayCasting::IntermixIntersectingGeometryOff () [inline]

Set IntermixIntersectingGeometry to off

8.315.2.21 void mitkVolumeRendererRayCasting::IntermixIntersectingGeometryOn () [inline]

Set IntermixIntersectingGeometry to on

8.315.2.22 virtual void mitkVolumeRendererRayCasting::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.315.2.23 virtual int mitkVolumeRendererRayCasting::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.315.2.24 void mitkVolumeRendererRayCasting::SetAutoAdjustSampleDistances (int *val*) [inline]

Set whether to auto-adjust the sample distances.

Parameters:

val indicate whether to auto-adjust the sample distances (>=1 means true, <=0 means false)

8.315.2.25 void mitkVolumeRendererRayCasting::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals

Parameters:

gradest Represent the gradient estimator used to estimate normals

8.315.2.26 void mitkVolumeRendererRayCasting::SetImageSampleDistance (float *fVal*) [inline]

Set the sample distance in the image plane.

Parameters:

fVal Specify the sample distance in the image plane.

8.315.2.27 void mitkVolumeRendererRayCasting::SetIntermixIntersectingGeometry (int *fVal*) [inline]

If IntermixIntersectingGeometry is turned on, the zbuffer will be captured and used to limit the traversal of the rays.

Parameters:

fVal Represent the intermix intersecting geometry value

8.315.2.28 void mitkVolumeRendererRayCasting::SetMaximumImageSampleDistance (float *fVal*) [inline]

Set the maximum value of the sample distance in the image plane.

Parameters:

fVal the maximum value of the sample distance in the image plane.

8.315.2.29 void mitkVolumeRendererRayCasting::SetMinimumImageSampleDistance (float *fVal*) [inline]

Set the minimum value of the sample distance in the image plane.

Parameters:
fVal the minimum value of the sample distance in the image plane

8.315.2.30 void mitkVolumeRendererRayCasting::SetSampleDistance (float *fVal*) [inline]

Set the sample distance in the ray direction.

Parameters:
fVal Specify the sample distance.

8.315.2.31 void mitkVolumeRendererRayCasting::SetVolumeRayCastFunction (mitkVolumeRayCastFunction * *oVal*) [inline]

Set the volume ray cast function. This is used to process values found along the ray to compute a final pixel value.

Parameters:
oVal Represent the volume ray cast function

The documentation for this class was generated from the following file:

- mitkVolumeRendererRayCasting.h

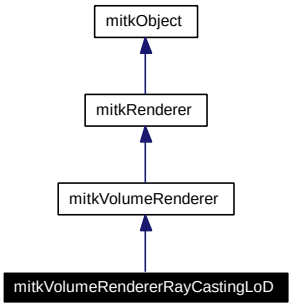
8.316 mitkVolumeRendererRayCastingLoD Class Reference

mitkVolumeRendererRayCastingLoD - a concrete volume renderer with LoD function for rendering a volume

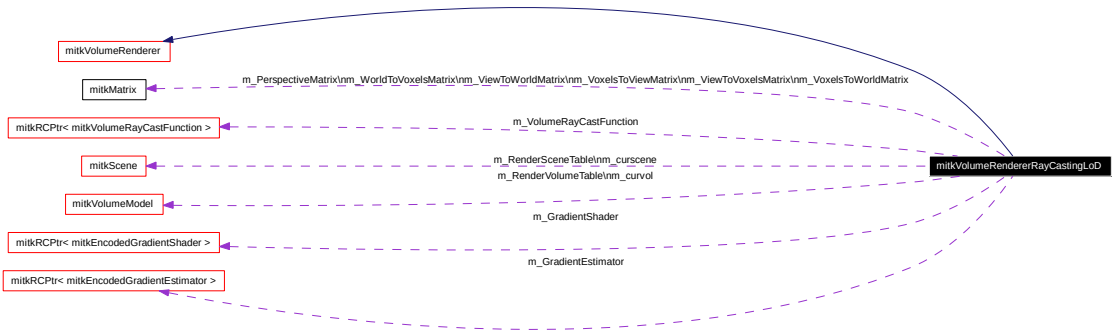
`#include <mitkVolumeRendererRayCastingLoD.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkVolumeRendererRayCastingLoD:



Collaboration diagram for mitkVolumeRendererRayCastingLoD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetSampleDistance (float fVal)
- float GetSampleDistance ()
- void SetVolumeRayCastFunction (mitkVolumeRayCastFunction *oVal)
- mitkVolumeRayCastFunction * GetVolumeRayCastFunction ()
- void SetGradientEstimator (mitkEncodedGradientEstimator *gradest)
- mitkEncodedGradientEstimator * GetGradientEstimator ()
- mitkEncodedGradientShader * GetEncodedGradientShader ()
- float GetImageSampleDistanceMinValue ()
- float GetImageSampleDistanceMaxValue ()

- void SetImageSampleDistance (float fVal)
- float GetImageSampleDistance ()
- float GetMinimumImageSampleDistanceMinValue ()
- float GetMinimumImageSampleDistanceMaxValue ()
- float GetMaximumImageSampleDistanceMinValue ()
- float GetMaximumImageSampleDistanceMaxValue ()
- void SetMinimumImageSampleDistance (float fVal)
- void SetMaximumImageSampleDistance (float fVal)
- float GetMinimumImageSampleDistance ()
- float GetMaximumImageSampleDistance ()
- void SetAutoAdjustSampleDistances (int val)
- int GetAutoAdjustSampleDistances ()
- void AutoAdjustSampleDistancesOn ()
- void AutoAdjustSampleDistancesOff ()
- void SetIntermixIntersectingGeometry (int fVal)
- int GetIntermixIntersectingGeometry ()
- void IntermixIntersectingGeometryOn ()
- void IntermixIntersectingGeometryOff ()
- virtual float GetGradientMagnitudeScale ()
- virtual float GetGradientMagnitudeBias ()

8.316.1 Detailed Description

mitkVolumeRendererRayCastingLoD - a concrete volume renderer with LoD function for rendering a volume

mitkVolumeRendererRayCastingLoD is a concrete volume renderer with LoD function for rendering a volume. It provides two levels of details: Rough and Refined. The implementation of ray casting algorithm is the same as mitkVolumeRendererRayCasting.

8.316.2 Member Function Documentation

8.316.2.1 void mitkVolumeRendererRayCastingLoD::AutoAdjustSampleDistancesOff ()

`[inline]`

Set sample distances auto-adjusting off.

8.316.2.2 void mitkVolumeRendererRayCastingLoD::AutoAdjustSampleDistancesOn ()

`[inline]`

Set sample distances auto-adjusting on.

8.316.2.3 int mitkVolumeRendererRayCastingLoD::GetAutoAdjustSampleDistances ()

`[inline]`

Get whether to auto-adjust the sample distances.

Returns:

Return a value which indicates whether to auto-adjust the sample distances (1 means true, 0 means false).

8.316.2.4 mitkEncodedGradientShader* mitkVolumeRendererRayCastingLoD::GetEncodedGradientShader () [inline]

Get the gradient shader

Returns:
Return the gradient shader

8.316.2.5 mitkEncodedGradientEstimator* mitkVolumeRendererRayCastingLoD::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals

Returns:
Return the gradient estimator used to estimate normals

8.316.2.6 virtual float mitkVolumeRendererRayCastingLoD::GetGradientMagnitudeBias () [virtual]

Get gradient magnitude bias.

Returns:
Return the gradient magnitude bias.

Reimplemented from mitkVolumeRenderer.

8.316.2.7 virtual float mitkVolumeRendererRayCastingLoD::GetGradientMagnitudeScale () [virtual]

Get gradient magnitude scale.

Returns:
Return the gradient magnitude scale.

Reimplemented from mitkVolumeRenderer.

8.316.2.8 float mitkVolumeRendererRayCastingLoD::GetImageSampleDistance () [inline]

Get the sample distance in the image plane.

Returns:
Return the sample distance in the image plane.

8.316.2.9 float mitkVolumeRendererRayCastingLoD::GetImageSampleDistanceMaxValue () [inline]

Get the maximum value of the sample distance in the image plane.

Returns:
Return the maximum value of the sample distance in the image plane.

8.316.2.10 float mitkVolumeRendererRayCastingLoD::GetImageSampleDistanceMinValue () [inline]

Get the minimum value of the sample distance in the image plane.

Returns:
Return the minimum value of the sample distance in the image plane.

8.316.2.11 int mitkVolumeRendererRayCastingLoD::GetIntermixIntersectingGeometry () [inline]

Get the intermix intersecting geometry value

Returns:
Return the intermix intersecting geometry value

8.316.2.12 float mitkVolumeRendererRayCastingLoD::GetMaximumImageSampleDistance () [inline]

Get the maximum sample distance in the image plane.

Returns:
Return the maximum sample distance in the image plane.

8.316.2.13 float mitkVolumeRendererRayCastingLoD::GetMaximumImageSampleDistanceMaxValue () [inline]

Get the maximum value of the maximum sample distance in the image plane.

Returns:
Return the maximum value of the maximum sample distance in the image plane.

8.316.2.14 float mitkVolumeRendererRayCastingLoD::GetMaximumImageSampleDistanceMinValue () [inline]

Get the minimum value of the maximum sample distance in the image plane.

Returns:
Return the minimum value of the maximum sample distance in the image plane.

8.316.2.15 float mitkVolumeRendererRayCastingLoD::GetMinimumImageSampleDistance () [inline]

Get the minimum sample distance in the image plane.

Returns:
Return the minimum sample distance in the image plane.

8.316.2.16 float mitkVolumeRendererRayCastingLoD::GetMinimumImageSampleDistanceMaxValue () [inline]

Get the maximum value of the minimum sample distance in the image plane.

Returns:
Return the maximum value of the minimum sample distance in the image plane.

8.316.2.17 float mitkVolumeRendererRayCastingLoD::GetMinimumImageSampleDistanceMinValue () [inline]

Get the minimum value of the minimum sample distance in the image plane.

Returns:
Return the minimum value of the minimum sample distance in the image plane.

8.316.2.18 float mitkVolumeRendererRayCastingLoD::GetSampleDistance () [inline]

Get the sample distance in the ray direction.

Returns:
Return the sample distance

8.316.2.19 mitkVolumeRayCastFunction* mitkVolumeRendererRayCastingLoD::GetVolumeRayCastFunction () [inline]

Get the volume ray cast function.

Returns:
Return the volume ray cast function

8.316.2.20 void mitkVolumeRendererRayCastingLoD::IntermixIntersectingGeometryOff () [inline]

Set IntermixIntersectingGeometry to off

8.316.2.21 void mitkVolumeRendererRayCastingLoD::IntermixIntersectingGeometryOn () [inline]

Set IntermixIntersectingGeometry to on

8.316.2.22 virtual void mitkVolumeRendererRayCastingLoD::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.316.2.23 virtual int mitkVolumeRendererRayCastingLoD::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.316.2.24 void mitkVolumeRendererRayCastingLoD::SetAutoAdjustSampleDistances (int *val*) [inline]

Set whether to auto-adjust the sample distances.

Parameters:

val indicate whether to auto-adjust the sample distances (>=1 means true, <=0 means false)

8.316.2.25 void mitkVolumeRendererRayCastingLoD::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals

Parameters:

gradest Represent the gradient estimator used to estimate normals

8.316.2.26 void mitkVolumeRendererRayCastingLoD::SetImageSampleDistance (float *fVal*) [inline]

Set the sample distance in the image plane.

Parameters:

fVal Specify the sample distance in the image plane.

8.316.2.27 void mitkVolumeRendererRayCastingLoD::SetIntermixIntersectingGeometry (int *fVal*) [inline]

If IntermixIntersectingGeometry is turned on, the zbuffer will be captured and used to limit the traversal of the rays.

Parameters:

fVal Represent the intermix intersecting geometry value

8.316.2.28 void mitkVolumeRendererRayCastingLoD::SetMaximumImageSampleDistance (float *fVal*) [inline]

Set the maximum value of the sample distance in the image plane.

Parameters:

fVal the maximum value of the sample distance in the image plane.

8.316.2.29 void mitkVolumeRendererRayCastingLoD::SetMinimumImageSampleDistance (float *fVal*) [inline]

Set the minimum value of the sample distance in the image plane.

Parameters:
fVal the minimum value of the sample distance in the image plane

8.316.2.30 void mitkVolumeRendererRayCastingLoD::SetSampleDistance (float *fVal*) [inline]

Set the sample distance in the ray direction.

Parameters:
fVal Specify the sample distance

8.316.2.31 void mitkVolumeRendererRayCastingLoD::SetVolumeRayCastFunction (mitkVolumeRayCastFunction * *oVal*) [inline]

Set the volume ray cast function. This is used to process values found along the ray to compute a final pixel value.

Parameters:
oVal Represent the volume ray cast function

The documentation for this class was generated from the following file:

- mitkVolumeRendererRayCastingLoD.h

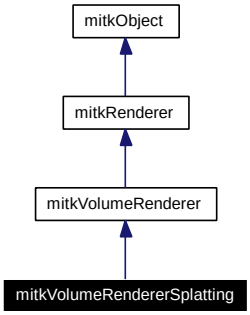
8.317 mitkVolumeRendererSplatting Class Reference

mitkVolumeRendererSplatting - a concrete volume renderer for rendering a volume

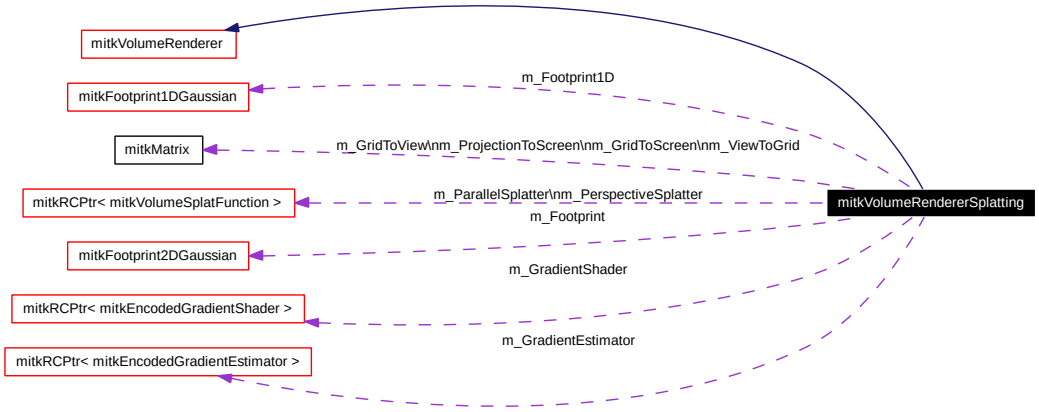
`#include <mitkVolumeRendererSplatting.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkVolumeRendererSplatting:



Collaboration diagram for mitkVolumeRendererSplatting:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetGradientEstimator (mitkEncodedGradientEstimator *gradest)
- void SetPerspectiveSplatter (mitkVolumeSplatFunction *splatter)
- void SetParallelSplatter (mitkVolumeSplatFunction *splatter)
- mitkEncodedGradientEstimator * GetGradientEstimator ()
- mitkVolumeSplatFunction * GetPerspectiveSplatter ()
- mitkVolumeSplatFunction * GetParallelSplatter ()
- mitkEncodedGradientShader * GetEncodedGradientShader ()
- void SetKernelRadii (float radii)

- void SetCoefficient (float coeff)
- void SetAdjustRadii (float adjustRadii)
- float GetKernelRadii ()
- float GetCoefficient ()
- float GetAdjustRadii ()
- void SetImageSampleDistance (float fVal)
- void SetGridSampleDistance (int fVal)
- float GetImageSampleDistance ()
- int GetGridSampleDistance ()
- float GetImageSampleDistanceMinValue ()
- int GetGridSampleDistanceMinValue ()
- float GetImageSampleDistanceMaxValue ()
- int GetGridSampleDistanceMaxValue ()
- void SetMinimumImageSampleDistance (float fVal)
- void SetMinimumGridSampleDistance (int fVal)
- float GetMinimumImageSampleDistanceMinValue ()
- int GetMinimumGridSampleDistanceMinValue ()
- float GetMinimumImageSampleDistanceMaxValue ()
- int GetMinimumGridSampleDistanceMaxValue ()
- float GetMinimumImageSampleDistance ()
- int GetMinimumGridSampleDistance ()
- void SetMaximumImageSampleDistance (float fVal)
- void SetMaximumGridSampleDistance (int fVal)
- float GetMaximumImageSampleDistanceMinValue ()
- int GetMaximumGridSampleDistanceMinValue ()
- float GetMaximumImageSampleDistanceMaxValue ()
- int GetMaximumGridSampleDistanceMaxValue ()
- float GetMaximumImageSampleDistance ()
- int GetMaximumGridSampleDistance ()
- bool GetAutoAdjustImageSampleDistances ()
- void AutoAdjustImageSampleDistancesOn ()
- void AutoAdjustImageSampleDistancesOff ()
- bool GetAutoAdjustGridSampleDistances ()
- void AutoAdjustGridSampleDistancesOn ()
- void AutoAdjustGridSampleDistancesOff ()
- virtual float GetGradientMagnitudeScale ()
- virtual float GetGradientMagnitudeBias ()
- void SetModeIntegral ()
- void SetModeMop ()
- int GetMode ()

8.317.1 Detailed Description

mitkVolumeRendererSplatting - a concrete volume renderer for rendering a volume

mitkVolumeRendererSplatting is a concrete volume renderer for rendering a volume using splatting technique.

Our implemented algorithm references to: [1] Westover L. Interactive volume rendering. Proceedings of the Chapel Hill on Workshop Volume Visualization, 1989.
[2] Westover L. Footprint evolution for volume rendering. Computer Graphics, 1990.
[3] Zwicker M. et al. EWA Volume Splatting. Gross Proceedings of IEEE Visualization, 2001.

Generally, its rendering effect and speed maybe not as good as mitkVolumeRendererRayCasting. However, its rendering advantage emerges when the volume is largely scaled. The default perspective splatter is the object of mitkVolumeSplatPerspective, and the default parallel splatter is the object of mitkVolumeSplatParallel. Particular perspective and parallel splatters are also allowed by calling SetPerspectiveSplatter() and SetParallelSplatter(). This class uses Gaussian kernel as footprint function:

$$footprint(x) = coeff * e^{-(x\mathbf{F}^{-1}x^T]/adjustradii}$$

where $coeff$ is the weight coefficient, $adjustradii$ is the adjust radii coefficient, and $\mathbf{F}$ is the variance matrix:

```
 _                       _
|  vmatrix[0]    vmatrix[2]  |
|_ vmatrix[1]    vmatrix[3] _|.
```

There are two rendering modes based on splatting, one is integral mode, and the other is MOP mode, you can switch the mode using SetModeIntegral() and SetModeMop().

Note:
In order to use this class to render a volume, the mitkScene's camera must be set to be mitkSplatCamera.

8.317.2 Member Function Documentation

8.317.2.1 void mitkVolumeRendererSplatting::AutoAdjustGridSampleDistancesOff ()

`[inline]`

Set grid sample distances auto-adjusting off.

8.317.2.2 void mitkVolumeRendererSplatting::AutoAdjustGridSampleDistancesOn ()

`[inline]`

Set grid sample distances auto-adjusting on.

8.317.2.3 void mitkVolumeRendererSplatting::AutoAdjustImageSampleDistancesOff ()

`[inline]`

Set image sample distances auto-adjusting off.

8.317.2.4 void mitkVolumeRendererSplatting::AutoAdjustImageSampleDistancesOn ()

`[inline]`

Set image sample distances auto-adjusting on.

8.317.2.5 float mitkVolumeRendererSplatting::GetAdjustRadii () [inline]

Get the adjust radii of Gaussian kernel in grid space.

Returns:
Return the adjust radii of Gaussian kernel in grid space.

Note:
The default value is 6.0, and will be clamped to [0.1,20.0].

8.317.2.6 bool mitkVolumeRendererSplatting::GetAutoAdjustGridSampleDistances () [inline]

Get whether to auto-adjust the grid sample distances.

Returns:
Return a value which indicates whether to auto-adjust the grid sample distances (1 means true, 0 means false).

8.317.2.7 bool mitkVolumeRendererSplatting::GetAutoAdjustImageSampleDistances () [inline]

Get whether to auto-adjust the image sample distances.

Returns:
Return a value which indicates whether to auto-adjust the sample distances (1 means true, 0 means false).

8.317.2.8 float mitkVolumeRendererSplatting::GetCoefficient () [inline]

Get the weight coefficient of Gaussian kernel in grid space.

Returns:
Return the weight coefficient of Gaussian kernel in grid space.

Note:
The default value is 1.0, and will be clamped to [0.01,10.0].

8.317.2.9 mitkEncodedGradientShader* mitkVolumeRendererSplatting::GetEncodedGradientShader () [inline]

Get the gradient shader.

Returns:
Return the gradient shader.

8.317.2.10 mitkEncodedGradientEstimator* mitkVolumeRendererSplatting::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals.

Returns:
Return the gradient estimator used to estimate normals.

8.317.2.11 virtual float mitkVolumeRendererSplatting::GetGradientMagnitudeBias () [virtual]

Get gradient magnitude bias.

Returns:
Return the gradient magnitude bias.

Reimplemented from mitkVolumeRenderer.

8.317.2.12 virtual float mitkVolumeRendererSplatting::GetGradientMagnitudeScale () [virtual]

Get gradient magnitude scale.

Returns:
Return the gradient magnitude scale.

Reimplemented from mitkVolumeRenderer.

8.317.2.13 int mitkVolumeRendererSplatting::GetGridSampleDistance () [inline]

Get the sample distance in the grid space.

Returns:
Return the grid sample distance in the grid space.

8.317.2.14 int mitkVolumeRendererSplatting::GetGridSampleDistanceMaxValue () [inline]

Get the maximum value of the grid sample distance in the grid space.

Returns:
Return the maximum value of the grid sample distance in the grid space.

8.317.2.15 int mitkVolumeRendererSplatting::GetGridSampleDistanceMinValue () [inline]

Get the minimum value of the grid sample distance in the grid space.

Returns:
Return the minimum value of the grid sample distance in the grid space.

8.317.2.16 float mitkVolumeRendererSplatting::GetImageSampleDistance () [inline]

Get the sample distance in the image plane.

Returns:
Return the image sample distance in the image plane.

8.317.2.17 float mitkVolumeRendererSplatting::GetImageSampleDistanceMaxValue () [inline]

Get the maximum value of the image sample distance in the image plane.

Returns:
Return the maximum value of the image sample distance in the image plane.

8.317.2.18 float mitkVolumeRendererSplatting::GetImageSampleDistanceMinValue () [inline]

Get the minimum value of the image sample distance in the image plane.

Returns:
Return the minimum value of the image sample distance in the image plane.

8.317.2.19 float mitkVolumeRendererSplatting::GetKernelRadii () [inline]

Get the splat radii of Gaussian kernel in grid space.

Returns:
Return the splat radii of Gaussian kernel in grid space.

Note:
The default value is 1.0, and will be clamped to [0.01,10.0].

8.317.2.20 int mitkVolumeRendererSplatting::GetMaximumGridSampleDistance () [inline]

Get the maximum grid sample distance in the grid space.

Returns:
Return the maximum grid sample distance in the grid space.

8.317.2.21 int mitkVolumeRendererSplatting::GetMaximumGridSampleDistanceMaxValue () [inline]

Get the maximum value of the maximum grid sample distance in the grid space.

Returns:
Return the maximum value of the maximum grid sample distance in the grid space.

8.317.2.22 int mitkVolumeRendererSplatting::GetMaximumGridSampleDistanceMinValue () [inline]

Get the minimum value of the maximum grid sample distance in the grid space.

Returns:
Return the minimum value of the maximum grid sample distance in the grid space.

8.317.2.23 float mitkVolumeRendererSplatting::GetMaximumImageSampleDistance () [inline]

Get the maximum image sample distance in the image plane.

Returns:
Return the maximum sample distance in the image plane.

8.317.2.24 float mitkVolumeRendererSplatting::GetMaximumImageSampleDistanceMaxValue () [inline]

Get the maximum value of the maximum image sample distance in the image plane.

Returns:
Return the maximum value of the maximum image sample distance in the image plane.

8.317.2.25 float mitkVolumeRendererSplatting::GetMaximumImageSampleDistanceMinValue () [inline]

Get the minimum value of the maximum image sample distance in the image plane.

Returns:
Return the minimum value of the maximum image sample distance in the image plane.

8.317.2.26 int mitkVolumeRendererSplatting::GetMinimumGridSampleDistance () [inline]

Get the minimum grid sample distance in the grid space.

Returns:
Return the minimum grid sample distance in the grid space.

8.317.2.27 int mitkVolumeRendererSplatting::GetMinimumGridSampleDistanceMaxValue () [inline]

Get the maximum value of the minimum grid sample distance in the grid space.

Returns:
Return the maximum value of the minimum grid sample distance in the grid space.

8.317.2.28 int mitkVolumeRendererSplatting::GetMinimumGridSampleDistanceMinValue () [inline]

Get the minimum value of the minimum grid sample distance in the grid space.

Returns:
Return the minimum value of the minimum grid sample distance in the grid space.

8.317.2.29 float mitkVolumeRendererSplatting::GetMinimumImageSampleDistance () [inline]

Get the minimum image sample distance in the image plane.

Returns:
Return the minimum sample distance in the image plane.

8.317.2.30 float mitkVolumeRendererSplatting::GetMinimumImageSampleDistanceMaxValue () [inline]

Get the maximum value of the minimum image sample distance in the image plane.

Returns:
Return the maximum value of the minimum image sample distance in the image plane.

8.317.2.31 float mitkVolumeRendererSplatting::GetMinimumImageSampleDistanceMinValue () [inline]

Get the minimum value of the minimum image sample distance in the image plane.

Returns:
Return the minimum value of the minimum image sample distance in the image plane.

8.317.2.32 int mitkVolumeRendererSplatting::GetMode () [inline]

Get the volume rendering mode.

Returns:
Return 0 if the volume rendering mode is integral projection. Return 1 if the volume rendering mode is MOP(maximum opacity projection).

8.317.2.33 mitkVolumeSplatFunction* mitkVolumeRendererSplatting::GetParallelSplatter ()

Get the Parallel splatter.

Returns:
Return the current Parallel splatter.

8.317.2.34 mitkVolumeSplatFunction* mitkVolumeRendererSplatting::GetPerspectiveSplatter ()

Get the perspective splatter.

Returns:
Return the current perspective splatter.

8.317.2.35 virtual void mitkVolumeRendererSplatting::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.317.2.36 virtual int mitkVolumeRendererSplatting::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.317.2.37 void mitkVolumeRendererSplatting::SetAdjustRadii (float *adjustRadii*) [inline]

Set the adjust radii of Gaussian kernel in grid space.

Parameters:
adjustRadii Specify the adjust radii of Gaussian kernel in grid space.

8.317.2.38 void mitkVolumeRendererSplatting::SetCoefficient (float *coeff*) [inline]

Set the weight coefficient of Gaussian kernel in grid space.

Parameters:
coeff Specify the weight coefficient of Gaussian kernel in grid space.

8.317.2.39 void mitkVolumeRendererSplatting::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals.

Parameters:
gradest Represent the gradient estimator used to estimate normals.

8.317.2.40 void mitkVolumeRendererSplatting::SetGridSampleDistance (int *fVal*) [inline]

Set the sample distance in the grid space.

Parameters:
fVal Specify the sample distance in the grid space.

8.317.2.41 void mitkVolumeRendererSplatting::SetImageSampleDistance (float *fVal*) [inline]

Set the sample distance in the image plane.

Parameters:
fVal Specify the sample distance in the image plane.

8.317.2.42 void mitkVolumeRendererSplatting::SetKernelRadii (float *radii*) [inline]

Set the splat radii of Gaussian kernel in grid space.

Parameters:
radii Specify the splat radii of Gaussian kernel in grid space.

8.317.2.43 void mitkVolumeRendererSplatting::SetMaximumGridSampleDistance (int *fVal*) [inline]

Set the maximum value of the grid sample distance in the grid space.

Parameters:
fVal the maximum value of the grid sample distance in the grid space.

8.317.2.44 void mitkVolumeRendererSplatting::SetMaximumImageSampleDistance (float *fVal*) [inline]

Set the maximum value of the image sample distance in the image plane.

Parameters:
fVal the maximum value of the sample distance in the image plane.

8.317.2.45 void mitkVolumeRendererSplatting::SetMinimumGridSampleDistance (int *fVal*) [inline]

Set the minimum value of the grid sample distance in the grid space.

Parameters:
fVal the minimum value of the grid sample distance in the grid space.

8.317.2.46 void mitkVolumeRendererSplatting::SetMinimumImageSampleDistance (float *fVal*) [inline]

Set the minimum value of the image sample distance in the image plane.

Parameters:
fVal the minimum value of the sample distance in the image plane

8.317.2.47 void mitkVolumeRendererSplatting::SetModeIntegral () [inline]

Set volume rendering mode to be integral projection.

8.317.2.48 void mitkVolumeRendererSplatting::SetModeMop () [inline]

Set volume rendering mode to be MOP(maximum opacity projection).

Note:
Each image pixel stands for maximum opacity element in the ray direction, and when opacity is linear with intensity, MOP is equal to MIP(maximum intensity projection).

8.317.2.49 void mitkVolumeRendererSplatting::SetParallelSplatter (mitkVolumeSplatFunction * *splatter*)

Set Parallel splatter to splat the volume.

Parameters:
splatter Represent the Parallel splatter.

8.317.2.50 void mitkVolumeRendererSplatting::SetPerspectiveSplatter (mitkVolumeSplatFunction * *splatter*)

Set perspective splatter to splat the volume.

Parameters:
splatter Represent the perspective splatter.

The documentation for this class was generated from the following file:

- mitkVolumeRendererSplatting.h

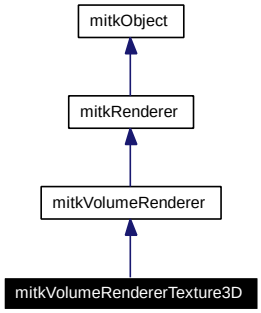
8.318 mitkVolumeRendererTexture3D Class Reference

mitkVolumeRendererTexture3D - a concrete volume renderer for rendering a volume

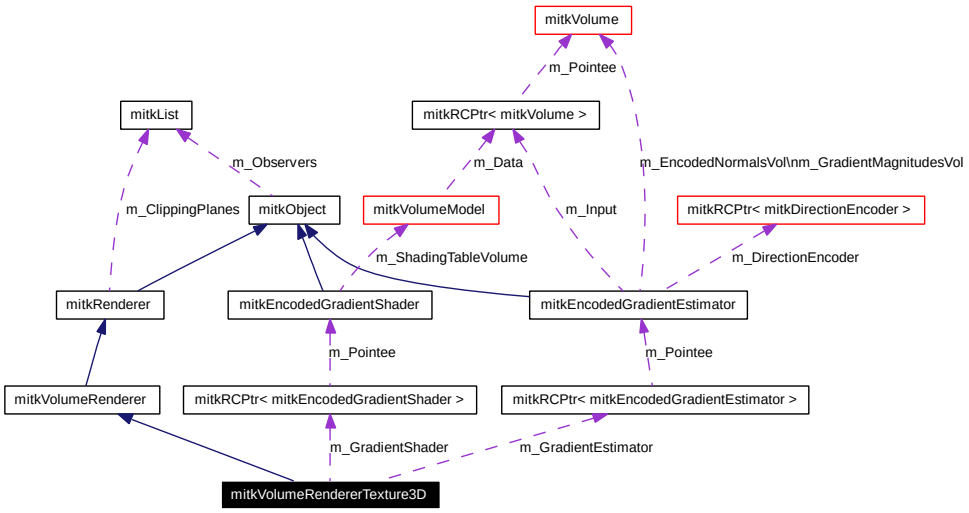
`#include <mitkVolumeRendererTexture3D.h>`

Inherits mitkVolumeRenderer.

Inheritance diagram for mitkVolumeRendererTexture3D:



Collaboration diagram for mitkVolumeRendererTexture3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkScene *scene, mitkVolumeModel *vol)
- void SetSampleDistance (float sd)
- void EnableAutoSampleDistance ()
- void SetGradientEstimator (mitkEncodedGradientEstimator *gradest)
- mitkEncodedGradientEstimator * GetGradientEstimator ()

8.318.1 Detailed Description

mitkVolumeRendererTexture3D - a concrete volume renderer for rendering a volume

mitkVolumeRendererTexture3D is a concrete volume renderer for rendering a volume by using 3D texture acceleration. (Currently, it does not support shading.)

Note:
To use this class, your graphics card should support 3D texture.

8.318.2 Member Function Documentation

8.318.2.1 void mitkVolumeRendererTexture3D::EnableAutoSampleDistance () [inline]

Enable automatic calculation of the sample distance.

8.318.2.2 mitkEncodedGradientEstimator* mitkVolumeRendererTexture3D::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals

Returns:
Return the gradient estimator used to estimate normals

8.318.2.3 virtual void mitkVolumeRendererTexture3D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeRenderer.

8.318.2.4 virtual int mitkVolumeRendererTexture3D::Render (mitkScene * *scene*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

8.318.2.5 void mitkVolumeRendererTexture3D::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals

Parameters:
gradest Represent the gradient estimator used to estimate normals

8.318.2.6 void mitkVolumeRendererTexture3D::SetSampleDistance (float *sd*) [inline]

Set sample distance.

Parameters:
sd sample distance

Note:
Once the user specified sample distance was set, automatic calculation of the sample distance will be disabled.

The documentation for this class was generated from the following file:

- mitkVolumeRendererTexture3D.h

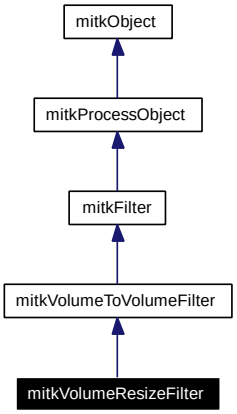
8.319 mitkVolumeResizeFilter Class Reference

mitkVolumeResizeFilter - a concrete filter class to zoom the specified volume

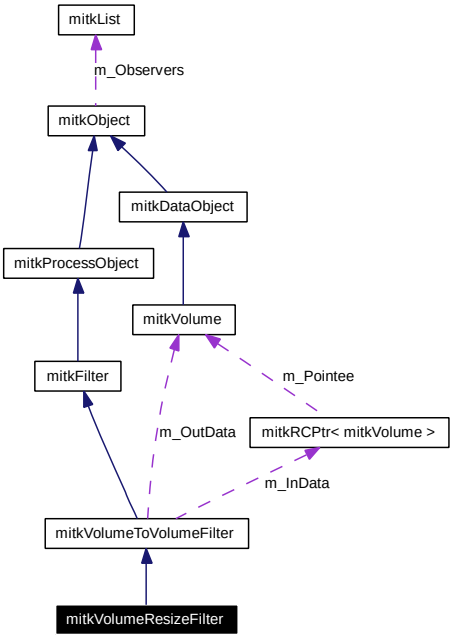
`#include <mitkVolumeResizeFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkVolumeResizeFilter:



Collaboration diagram for mitkVolumeResizeFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetZoomSize (int const xzoomsize, int const yzoomsize, int const zzoomsize)
- void SetZoomRate (float const xzoomrate, float const yzoomrate, float const zzoomrate)

8.319.1 Detailed Description

mitkVolumeResizeFilter - a concrete filter class to zoom the specified volume

mitkVolumeResizeFilter is a concrete filter class which inherits from volume to volume filter to zoom the specified volume. This filter needs a volume input and generates a volume output. So you should first input a volume using public member function SetInput(), then you should set zoom parameters using SetZoomSize(const int, const int, const int)(represents demanded width,height and image number respectively) or SetZoomRate(const float, const float, const float) (represents zoom rate of demanded width, height and image number respectively, if the corresponding parameter value is bigger than 1.0, it means "zoom in", and if the corresponding parameter value is smaller than 1.0,it means "zoom out". Two examples using mitkVolumeResizeFilter are given below.

Example 1: If you want to zoom volume using SetZoomSize() the code snippet is:

```
mitkVolumeResizeFilter *filter = new mitkVolumeResizeFilter;
filter->SetInput(inVolume);
filter->SetZoomSize(targetwidth,targetheight,targetimagenumber);
if (filter->Run())
{
      mitkVolume * outVolume = filter->GetOutput();
       Using  outVolume...
}
```

Example 2: If you want to zoom volume using SetZoomRate() the code snippet is:

```
mitkVolumeResizeFilter *filter = new mitkVolumeResizeFilter;
filter->SetInput(inVolume);
filter->SetZoomRate(widthzoomrate,heightzoomrate,imagenumberzoomrate);
if (filter->Run())
{
      mitkVolume * outVolume = filter->GetOutput();
      Using outVolume...
}
```

Note:
The inpute size must >= 2x2x2!

8.319.2 Member Function Documentation

8.319.2.1 virtual void mitkVolumeResizeFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.319.2.2 void mitkVolumeResizeFilter::SetZoomRate (float const *xzoomrate*, float const *yzoomrate*, float const *zzoomrate*)

Call SetProperty() to set property of the target volume(the output volume) with the specified zoom rate in each direction which is demanded by user.

Parameters:
xzoomrate The specified zoom rate in direction x demanded by user.

yzoomrate The specified zoom rate in direction y demanded by user.

zzoomrate The specified zoom rate in direction z demanded by user.

Returns:
Return true if this setting process is successful,otherwise return false.

8.319.2.3 void mitkVolumeResizeFilter::SetZoomSize (int const *xzoomsize*, int const *yzoomsize*, int const *zzoomsize*)

Call SetProperty() to set property of the target volume(the output volume) with the specified width,height and image numbers which is demanded by user.

Parameters:

xzoomsize The specified target width demanded by user

yzoomsize The specified target height demanded by user

zzoomsize The specified target image numbers demanded by user

Returns:
Return true if this setting process is successful,otherwise return false.

The documentation for this class was generated from the following file:

- mitkVolumeResizeFilter.h

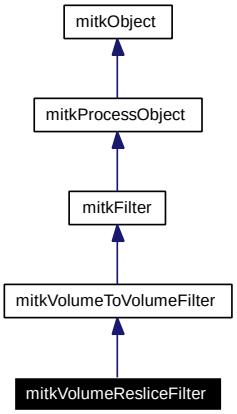
8.320 mitkVolumeResliceFilter Class Reference

mitkVolumeResliceFilter - a volume re-slice filter

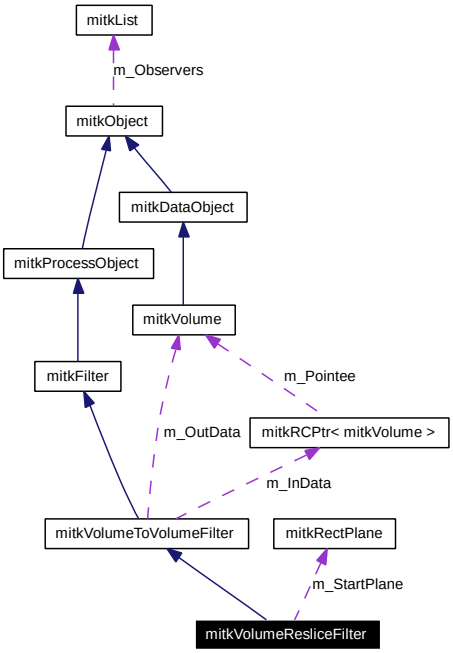
`#include <mitkVolumeResliceFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkVolumeResliceFilter:



Collaboration diagram for mitkVolumeResliceFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkVolumeResliceFilter ()

- void SetStartPlane (mitkRectPlane const &p)
- void SetStopPoint (double x, double y, double z)
- void SetSliceWidth (int w)
- void SetSliceHeight (int h)
- void SetSliceNumber (int num)
- void SetSpacingX (double sx)
- void SetSpacingY (double sy)
- void SetSpacingZ (double sz)

8.320.1 Detailed Description

mitkVolumeResliceFilter - a volume re-slice filter

mitkVolumeResliceFilter is a volume re-slice filter. It can reconstruct one slice or all slices of the input volume data along an arbitrary direction. It use a rectangle plane to scan the volume data from a start position (specified by a start plane) to a stop position (specified by a stop point), and reconstruct the slices equidistantly.

8.320.2 Constructor & Destructor Documentation

8.320.2.1 mitkVolumeResliceFilter::mitkVolumeResliceFilter ()

The default constructor.

8.320.3 Member Function Documentation

8.320.3.1 virtual void mitkVolumeResliceFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeToVolumeFilter.

8.320.3.2 void mitkVolumeResliceFilter::SetSliceHeight (int *h*) [inline]

Set the image height of the reconstructed slices.

Parameters:
h the image height of the reconstructed slices

8.320.3.3 void mitkVolumeResliceFilter::SetSliceNumber (int *num*) [inline]

Set the number of the reconstructed slices.

Parameters:
num the number of the reconstructed slices

8.320.3.4 void mitkVolumeResliceFilter::SetSliceWidth (int *w*) [inline]

Set the image width of the reconstructed slices.

Parameters:
w the image width of the reconstructed slices

8.320.3.5 void mitkVolumeResliceFilter::SetSpacingX (double *sx*) [inline]

Set spacing information in x axis, the unit is mm.

Parameters:
sx the spacing (mm) in two adjacent voxels in x axis.

8.320.3.6 void mitkVolumeResliceFilter::SetSpacingY (double *sy*) [inline]

Set spacing information in y axis, the unit is mm.

Parameters:
sy the spacing (mm) in two adjacent voxels in y axis.

8.320.3.7 void mitkVolumeResliceFilter::SetSpacingZ (double *sz*) [inline]

Set spacing information in z axis, the unit is mm.

Parameters:
sz the spacing (mm) in two adjacent voxels in z axis.

8.320.3.8 void mitkVolumeResliceFilter::SetStartPlane (mitkRectPlane const & *p*) [inline]

Set the plane where the reconstruction start.

Parameters:
p the reference to a const mitkRectPlane object which specify the start rectangle plane.

8.320.3.9 void mitkVolumeResliceFilter::SetStopPoint (double *x*, double *y*, double *z*) [inline]

Set the point where the reconstruction stop.

Parameters:
x the x-coordinate of the stop point
y the y-coordinate of the stop point
z the z-coordinate of the stop point

The documentation for this class was generated from the following file:

- mitkVolumeResliceFilter.h

8.321 mitkVolumeSplatFunction Class Reference

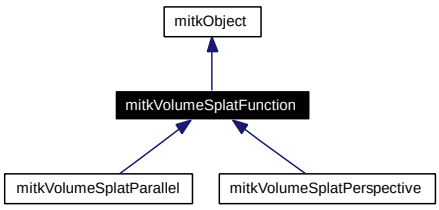
mitkVolumeSplatFunction - abstract class defines interface for volume splatting

`#include <mitkVolumeSplatFunction.h>`

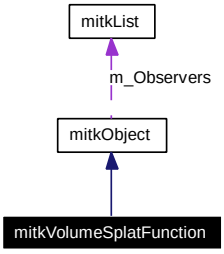
Inherits mitkObject.

Inherited by mitkVolumeSplatParallel, and mitkVolumeSplatPerspective.

Inheritance diagram for mitkVolumeSplatFunction:



Collaboration diagram for mitkVolumeSplatFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkVolumeSplatFunction ()
- virtual bool IsParallel ()
- virtual void Splat (mitkSplat *splatInform)=0

8.321.1 Detailed Description

mitkVolumeSplatFunction - abstract class defines interface for volume splatting

mitkVolumeSplatFunction is an abstract class that defines interface for both parallel and perspective splatting. The mitkVolumeRendererSplatting class provides common information for splatting, and then calls "Splat()" to implement volume splatting progress concretely.

8.321.2 Constructor & Destructor Documentation

8.321.2.1 mitkVolumeSplatFunction::mitkVolumeSplatFunction ()

The default constructor.

8.321.3 Member Function Documentation

8.321.3.1 virtual bool mitkVolumeSplatFunction::IsParallel () [inline, virtual]

Get the projection status of the class.

Returns:
Return true, means that this class is used for parallel splatting. Return false, means that this class is used for perspective splatting.

Note:
All of its subclasses must implement this virtual to indicate its status.

Reimplemented in mitkVolumeSplatParallel, and mitkVolumeSplatPerspective.

8.321.3.2 virtual void mitkVolumeSplatFunction::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkObject.

Reimplemented in mitkVolumeSplatParallel, and mitkVolumeSplatPerspective.

8.321.3.3 virtual void mitkVolumeSplatFunction::Splat (mitkSplat * *splatInform*) [pure virtual]

The real function to splat the volume.

Note:
All of its subclasses must implement this pure virtual to splat the volume.

Implemented in mitkVolumeSplatParallel, and mitkVolumeSplatPerspective.

The documentation for this class was generated from the following file:

- mitkVolumeSplatFunction.h

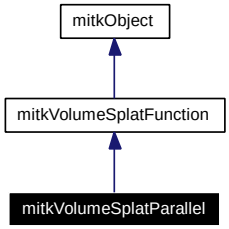
8.322 mitkVolumeSplatParallel Class Reference

mitkVolumeSplatParallel - concrete class for parallel volume splatting

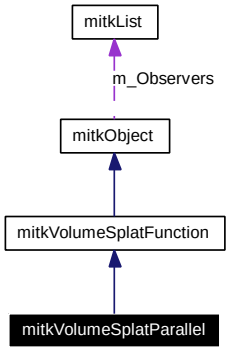
`#include <mitkVolumeSplatParallel.h>`

Inherits mitkVolumeSplatFunction.

Inheritance diagram for mitkVolumeSplatParallel:



Collaboration diagram for mitkVolumeSplatParallel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkVolumeSplatParallel ()
- virtual bool IsParallel ()
- virtual void Splat (mitkSplat *splatInform)

8.322.1 Detailed Description

mitkVolumeSplatParallel - concrete class for parallel volume splatting

mitkVolumeSplatParallel is a concrete class that splats the volume with parallel projection. This is the default parallel splatting class.

8.322.2 Constructor & Destructor Documentation

8.322.2.1 mitkVolumeSplatParallel::mitkVolumeSplatParallel ()

The default constructor.

8.322.3 Member Function Documentation

8.322.3.1 virtual bool mitkVolumeSplatParallel::IsParallel () [inline, virtual]

Get the projection status of the class.

Returns:
Return true, means that this class is used for parallel splatting. Return false, means that this class is used for perspective splatting.

Reimplemented from mitkVolumeSplatFunction.

8.322.3.2 virtual void mitkVolumeSplatParallel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeSplatFunction.

8.322.3.3 virtual void mitkVolumeSplatParallel::Splat (mitkSplat * *splatInform*) [virtual]

The real function to splat the volume.

Implements mitkVolumeSplatFunction.

The documentation for this class was generated from the following file:

- mitkVolumeSplatParallel.h

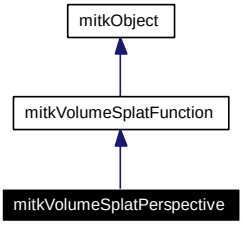
8.323 mitkVolumeSplatPerspective Class Reference

mitkVolumeSplatCompositeFunction - concrete class for perspective volume splatting

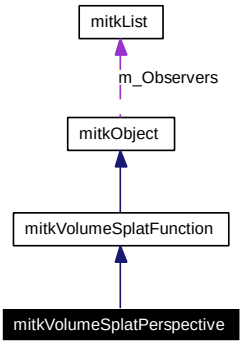
`#include <mitkVolumeSplatPerspective.h>`

Inherits mitkVolumeSplatFunction.

Inheritance diagram for mitkVolumeSplatPerspective:



Collaboration diagram for mitkVolumeSplatPerspective:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkVolumeSplatPerspective ()
- virtual bool IsParallel ()
- virtual void Splat (mitkSplat *splatInform)

8.323.1 Detailed Description

mitkVolumeSplatCompositeFunction - concrete class for perspective volume splatting

mitkVolumeSplatCompositeFunction is a concrete class that splats the volume with perspective projection. This is the default perspective splatting class.

8.323.2 Constructor & Destructor Documentation

8.323.2.1 mitkVolumeSplatPerspective::mitkVolumeSplatPerspective ()

The default constructor.

8.323.3 Member Function Documentation

8.323.3.1 virtual bool mitkVolumeSplatPerspective::IsParallel () [inline, virtual]

Get the projection status of the class.

Returns:
Return true, means that this class is used for parallel splatting. Return false, means that this class is used for perspective splatting.

Reimplemented from mitkVolumeSplatFunction.

8.323.3.2 virtual void mitkVolumeSplatPerspective::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkVolumeSplatFunction.

8.323.3.3 virtual void mitkVolumeSplatPerspective::Splat (mitkSplat * *splatInform*) [virtual]

The real function to splat the volume.

Implements mitkVolumeSplatFunction.

The documentation for this class was generated from the following file:

- mitkVolumeSplatPerspective.h

8.324 mitkVolumeToMeshFilter Class Reference

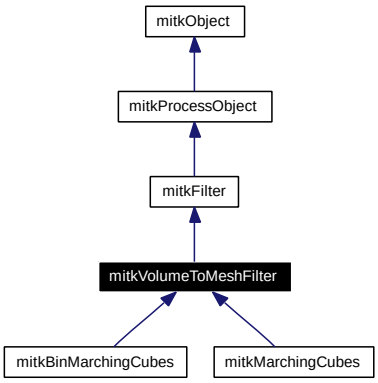
mitkVolumeToMeshFilter - abstract class specifies interface for volume to mesh filter

`#include <mitkVolumeToMeshFilter.h>`

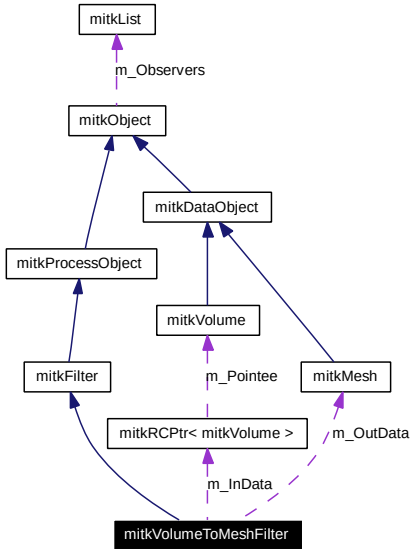
Inherits mitkFilter.

Inherited by mitkBinMarchingCubes, and mitkMarchingCubes.

Inheritance diagram for mitkVolumeToMeshFilter:



Collaboration diagram for mitkVolumeToMeshFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInput (mitkVolume *inData)
- mitkVolume * GetInput ()
- void SetOoCSupport (char const *diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufBlockNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)

- mitkMesh * GetOutput ()
- void SetOutputMesh (mitkMesh *mesh)

8.324.1 Detailed Description

mitkVolumeToMeshFilter - abstract class specifies interface for volume to mesh filter

mitkVolumeToMeshFilter is an abstract class specifies interface for volume to mesh filter. This type of filter has a volume input and generates a mesh as output.

8.324.2 Member Function Documentation

8.324.2.1 mitkVolume* mitkVolumeToMeshFilter::GetInput () [inline]

Get the input volume

Returns:
Return the input volume

8.324.2.2 mitkMesh* mitkVolumeToMeshFilter::GetOutput ()

Get the output mesh

Returns:
Return the output mesh

8.324.2.3 virtual void mitkVolumeToMeshFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFilter.

Reimplemented in mitkBinMarchingCubes, and mitkMarchingCubes.

8.324.2.4 void mitkVolumeToMeshFilter::SetInput (mitkVolume * *inData*) [inline]

Set the input volume

Parameters:
inData Specify the input volume

8.324.2.5 void mitkVolumeToMeshFilter::SetOoCSupport (char const * *diskPath* = mitkOoCGlobalSettings::DiskCachePath, unsigned int *bufBlockNum* = mitkOoCGlobalSettings::BufferedSliceNumber, bool *supportOoC* = true)

Let the reader support out-of-core mesh data.

Parameters:

diskPath the path in the disk to cache the mesh data

bufSliceNum the number of slices to cache in the main memory

supportOoC whether to turn on out-of-core support

Note:

The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

8.324.2.6 void mitkVolumeToMeshFilter::SetOutputMesh (mitkMesh * *mesh*)

Set user defined output mesh to replace the default generated mesh.

Parameters:

mesh the new output mesh object to be set

The documentation for this class was generated from the following file:

- mitkVolumeToMeshFilter.h

8.325 mitkVolumeToVolumeFilter Class Reference

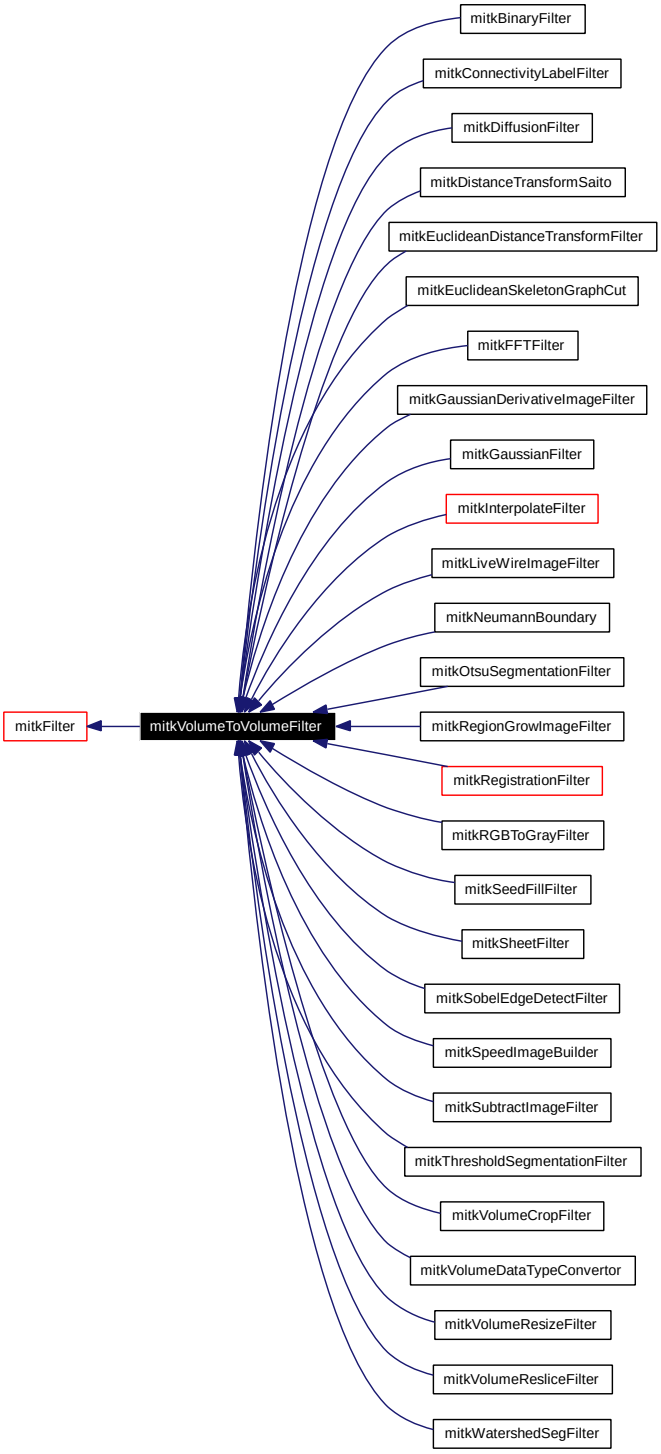
mitkVolumeToVolumeFilter - abstract class specifies interface for volume to volume filter

`#include <mitkVolumeToVolumeFilter.h>`

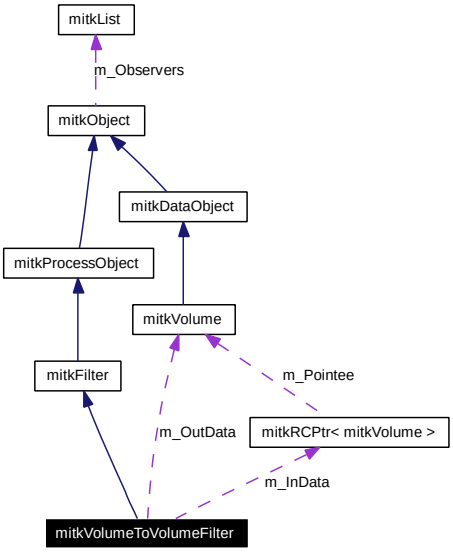
Inherits mitkFilter.

Inherited by mitkBinaryFilter, mitkCannyEdgeDetectFilter, mitkConnectivityLabelFilter, mitkDiffusionFilter, mitkDistanceTransformSaito, mitkEuclideanDistanceTransformFilter, mitkEuclideanSkeletonGraphCut, mitkFFTFilter, mitkGaussianDerivativeImageFilter, mitkGaussianFilter, mitkGraphCutImageFilter, mitkInterpolateFilter, mitkIsosurfaceCurvatureFilter, mitkLiveWireImageFilter, mitkNeumannBoundary, mitkOtsuSegmentationFilter, mitkPalagyiShrinking_6, mitkPalagyiThinning_6, mitkRegionGrowImageFilter, mitkRegistrationFilter, mitkRGBToGrayFilter, mitkSeedFillFilter, mitkSheetFilter, mitkSobelEdgeDetectFilter, mitkSpeedImageBuilder, mitkSubtractImageFilter, mitkThresholdSegmentationFilter, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeResizeFilter, mitkVolumeResliceFilter, and mitkWatershedSegFilter.

Inheritance diagram for mitkVolumeToVolumeFilter:



Collaboration diagram for mitkVolumeToVolumeFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInput (mitkVolume *inData)
- mitkVolume * GetInput ()
- void SetOoCSupport (char const *diskPath=mitkOoCGlobalSettings::DiskCachePath, unsigned int bufSliceNum=mitkOoCGlobalSettings::BufferedSliceNumber, bool supportOoC=true)
- mitkVolume * GetOutput ()

8.325.1 Detailed Description

mitkVolumeToVolumeFilter - abstract class specifies interface for volume to volume filter

mitkVolumeToVolumeFilter is an abstract class specifies interface for volume to volume filter. This type of filter has a volume input and generates a volume as output.

8.325.2 Member Function Documentation

8.325.2.1 mitkVolume* mitkVolumeToVolumeFilter::GetInput () [inline]

Get the input volume

Returns:
Return the input volume

8.325.2.2 mitkVolume* mitkVolumeToVolumeFilter::GetOutput ()

Get the output volume

Returns:
Return the output volume

Reimplemented in mitkRegistrationFilter.

8.325.2.3 virtual void mitkVolumeToVolumeFilter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkFilter.

Reimplemented in mitkBinaryFilter, mitkDiffusionFilter, mitkFFTFilter, mitkGaussianDerivativeImageFilter, mitkGaussianFilter, mitkNeumannBoundary, mitkRGBToGrayFilter, mitkSheetFilter, mitkSobelEdgeDetectFilter, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeResizeFilter, mitkVolumeResliceFilter, mitkBSplineInterpolateFilter, mitkCubicInterpolateFilter, mitkInterpolateFilter, mitkLandmarkRegistrationFilter, mitkLinearInterpolateFilter, mitkNearestNeighborInterpolateFilter, mitkPointSetRegistrationFilter, mitkRegistrationFilter, mitkSubtractImageFilter, mitkConnectivityLabelFilter, mitkLiveWireImageFilter, mitkOtsuSegmentationFilter, mitkRegionGrowImageFilter, mitkSeedFillFilter, and mitkThresholdSegmentationFilter.

8.325.2.4 void mitkVolumeToVolumeFilter::SetInput (mitkVolume * *inData*) [inline]

Set the input volume

Parameters:
inData Specify the input volume

8.325.2.5 void mitkVolumeToVolumeFilter::SetOoCSupport (char const * *diskPath* = mitkOoCGlobalSettings::DiskCachePath, unsigned int *bufSliceNum* = mitkOoCGlobalSettings::BufferedSliceNumber, bool *supportOoC* = true)

Let the reader support out-of-core volume data.

Parameters:
diskPath the path in the disk to cache the volume data

bufSliceNum the number of slices to cache in the main memory

supportOoC whether to turn on out-of-core support

Note:
The parameter diskPath must be specified (not NULL) if you really want to turn on out-of-core support, if not, the value of supportOoC will be ignored even if it is set to true.

The documentation for this class was generated from the following file:

- mitkVolumeToVolumeFilter.h

8.326 mitkVolumeWriter Class Reference

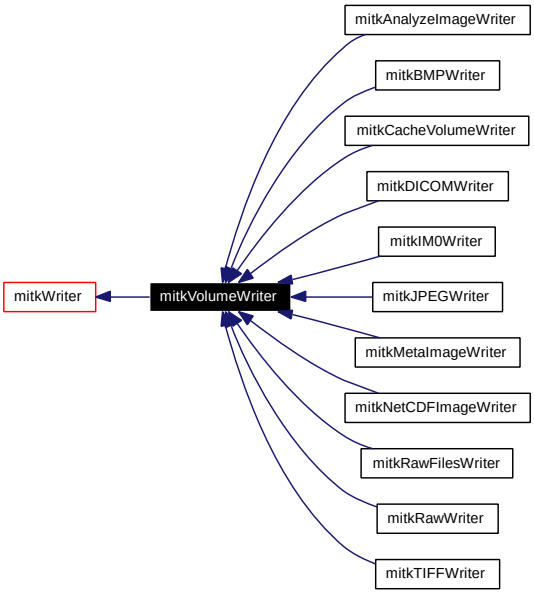
mitkVolumeWriter - an abstract class represents a volume writer for writing image/volume files to disk

`#include <mitkVolumeWriter.h>`

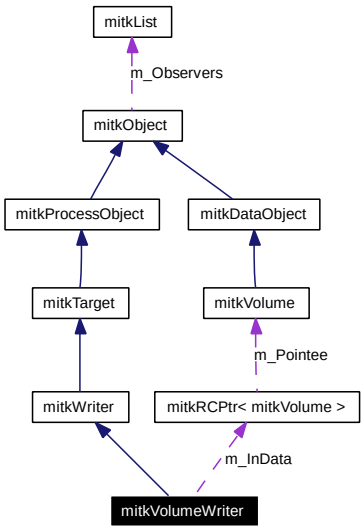
Inherits mitkWriter.

Inherited by mitkAnalyzeImageWriter, mitkBMPWriter, mitkCacheVolumeWriter, mitkDICOMWriter, mitkIM0Writer, mitkJPEGWriter, mitkMetaImageWriter, mitkNetCDFImageWriter, mitkRawFilesWriter, mitkRawWriter, and mitkTIFFWriter.

Inheritance diagram for mitkVolumeWriter:



Collaboration diagram for mitkVolumeWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInput (mitkVolume *inData)
- mitkVolume * GetInput ()

8.326.1 Detailed Description

mitkVolumeWriter - an abstract class represents a volume writer for writing image/volume files to disk

mitkVolumeWriter defines the interface of all of the volume writers. To use a concrete writer, for example, mitkBMPWriter, the code snippet is:

```
mitkBMPWriter *aWriter = new mitkBMPWriter;
aWriter->SetInput(aVolume);
int imageNum = aVolume->GetImageNum();
Gerenate file names into files[imageNum];
for(int i = 0; i < imageNum; i++)
    aWriter->AddFileName(files[i]);
aWriter->Run();
```

8.326.2 Member Function Documentation

8.326.2.1 mitkVolume* mitkVolumeWriter::GetInput () [inline]

Get input volume.

Returns:
Return the input volume

8.326.2.2 virtual void mitkVolumeWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWriter.

Reimplemented in mitkAnalyzeImageWriter, mitkBMPWriter, mitkCacheVolumeWriter, mitkDICOMWriter, mitkJPEGWriter, mitkMetaImageWriter, mitkNetCDFImageWriter, mitkRawFilesWriter, mitkRawWriter, and mitkTIFFWriter.

8.326.2.3 void mitkVolumeWriter::SetInput (mitkVolume * *inData*) [inline]

Set input volume to write to disk file.

Parameters:
inData Input volume

The documentation for this class was generated from the following file:

- mitkVolumeWriter.h

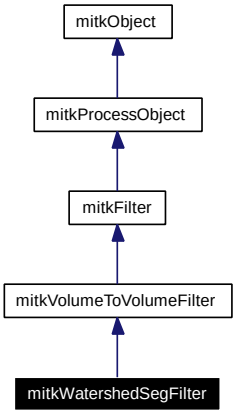
8.327 mitkWatershedSegFilter Class Reference

mitkWatershedSegFilter - a class implement watershed segmentation arithmetic

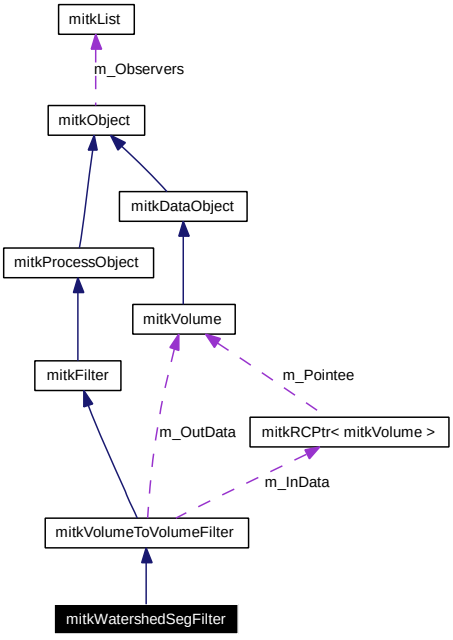
`#include <mitkWatershedSegFilter.h>`

Inherits mitkVolumeToVolumeFilter.

Inheritance diagram for mitkWatershedSegFilter:



Collaboration diagram for mitkWatershedSegFilter:



Public Member Functions

- mitkWatershedSegFilter ()
- void setThrValue (int value)

8.327.1 Detailed Description

mitkWatershedSegFilter - a class implement watershed segmentation arithmetic

mitkWatershedSegFilter implement watershed segmentation arithmetic. The output is of the same datatype of the input.

8.327.2 Constructor & Destructor Documentation

8.327.2.1 mitkWatershedSegFilter::mitkWatershedSegFilter ()

Constructor of the class

8.327.3 Member Function Documentation

8.327.3.1 void mitkWatershedSegFilter::setThrValue (int *value*) [inline]

Set the threshold value.

Parameters:

value Represent the thresh value

The documentation for this class was generated from the following file:

- mitkWatershedSegFilter.h

8.328 mitkWidgetModel Class Reference

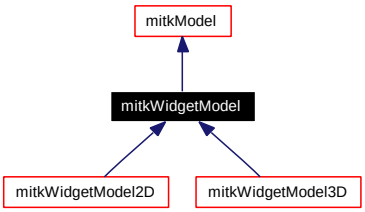
mitkWidgetModel - abstract class used to represent a widget entity (e.g. a line or an angle) in a rendering scene

`#include <mitkWidgetModel.h>`

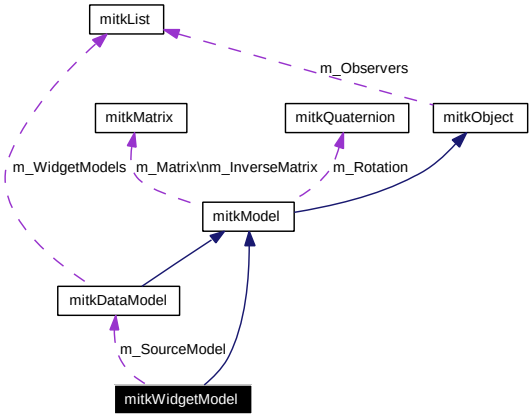
Inherits mitkModel.

Inherited by mitkWidgetModel2D, and mitkWidgetModel3D.

Inheritance diagram for mitkWidgetModel:



Collaboration diagram for mitkWidgetModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Pick (const WidgetNames &names)=0
- virtual void Release ()=0
- virtual void Select (mitkScene *scene)
- void OnMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void OnMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- void OnMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)
- virtual void SetSourceModel (mitkDataModel *model)
- mitkDataModel * GetSourceModel ()
- virtual void SetScene (mitkScene *scene)=0
- virtual void SetView (mitkView *view)
- virtual bool IsOpaque ()

- virtual bool IsActive ()
- bool IsMouseLeftButtonDown ()
- bool IsMouseRightButtonDown ()
- bool IsMouseMiddleButtonDown ()
- void UpdateObservers ()
- virtual void Update ()=0

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)=0
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)=0
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos, int deltaX, int deltaY)=0

8.328.1 Detailed Description

mitkWidgetModel - abstract class used to represent a widget entity (e.g. a line or an angle) in a rendering scene

mitkWidgetModel is an abstract class used to represent a widget entity (e.g. a line or an angle) in a rendering scene. It can make responses to the mouse events and manipulate the mitkDataModel to which it attaches.

8.328.2 Member Function Documentation

8.328.2.1 virtual void mitkWidgetModel::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, pure virtual]

Deal with mouse down event. Should be implemented in the concrete classes.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, and mitkReslicePlaneWidgetModel.

8.328.2.2 virtual void mitkWidgetModel::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*) [protected, pure virtual]

Deal with mouse move event. Should be implemented in the concrete classes.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

deltaX movement along x-axis of the mouse when mouse move event occurs

deltaY movement along y-axis of the mouse when mouse move event occurs

Implemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, and mitkReslicePlaneWidgetModel.

8.328.2.3 virtual void mitkWidgetModel::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, pure virtual]

Deal with mouse up event. Should be implemented in the concrete classes.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, and mitkReslicePlaneWidgetModel.

8.328.2.4 mitkDataModel* mitkWidgetModel::GetSourceModel () [inline]

Get the source model to which this widget attach.

Returns:

Return the pointer to the source model.

8.328.2.5 virtual bool mitkWidgetModel::IsActive () [inline, virtual]

Whether this widget is active.

Returns:

Return true if this widget is active (been selected).

8.328.2.6 bool mitkWidgetModel::IsMouseLeftButtonDown () [inline]

Whether the left button of the mouse is down.

Returns:

Return true if the left button of the mouse is down.

8.328.2.7 bool mitkWidgetModel::IsMouseMiddleButtonDown () [inline]

Whether the middle button of the mouse is down.

Returns:
Return true if the left button of the mouse is down.

8.328.2.8 bool mitkWidgetModel::IsMouseRightButtonDown () [inline]

Whether the right button of the mouse is down.

Returns:
Return true if the left button of the mouse is down.

8.328.2.9 virtual bool mitkWidgetModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Implements mitkModel.

8.328.2.10 void mitkWidgetModel::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

Deal with mouse down event.

Parameters:
mouseButton indicates which mouse button is pressed
ctrlDown indicates if the key "Ctrl" is pressed
shiftDown indicates if the key "Shift" is pressed
xPos x-coordinate of the mouse position when mouse down event occurs
yPos y-coordinate of the mouse position when mouse down event occurs

8.328.2.11 void mitkWidgetModel::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *deltaX*, int *deltaY*)

Deal with mouse move event.

Parameters:
ctrlDown indicates if the key "Ctrl" is pressed
shiftDown indicates if the key "Shift" is pressed
xPos x-coordinate of the mouse position when mouse move event occurs
yPos y-coordinate of the mouse position when mouse move event occurs
deltaX movement along x-axis of the mouse when mouse move event occurs
deltaY movement along y-axis of the mouse when mouse move event occurs

8.328.2.12 void mitkWidgetModel::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

8.328.2.13 virtual void mitkWidgetModel::Pick (const WidgetNames & *names*) [pure virtual]

Maintain the selection status when this widget is picked.

Parameters:

names a constant reference to an WidgerNames which contains the names of selected parts of this widget.

Implemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, and mitkReslicePlaneWidgetModel.

8.328.2.14 virtual void mitkWidgetModel::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkModel.

Reimplemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, mitkReslicePlaneWidgetModel, mitkWidgetModel2D, and mitkWidgetModel3D.

8.328.2.15 virtual void mitkWidgetModel::Release () [pure virtual]

Maintain the selection status when this widget is released.

Implemented in mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, and mitkReslicePlaneWidgetModel.

8.328.2.16 virtual void mitkWidgetModel::Select (mitkScene * *scene*) [virtual]

Use select mode to render this model.

Parameters:
scene the pointer of an mitkScene in which this model is contained.

Reimplemented from mitkModel.

8.328.2.17 virtual void mitkWidgetModel::SetScene (mitkScene * *scene*) [pure virtual]

Set the scene which contains this model.

Parameters:
scene pointer to the mitkScene contains this model

Note:
This is a pure virtual function and this class dose not has data member of scene class. The concrete class derived form this class must implement this function and does all necessary work itself including specifying the data member for the scene.

Implemented in mitkWidgetModel2D, and mitkWidgetModel3D.

8.328.2.18 virtual void mitkWidgetModel::SetSourceModel (mitkDataModel * *model*) [virtual]

Associate this widget with a model.

Parameters:
model pointer to an miekDataModel to which this widget attach

Reimplemented in mitkClippingPlaneWidgetModel, mitkPseudocolorWidgetModelEx, mitkReslicePlaneWidgetModel, mitkWidgetModel2D, and mitkWidgetModel3D.

8.328.2.19 virtual void mitkWidgetModel::SetView (mitkView * *view*) [virtual]

Set the view which contains this model.

Parameters:
view pointer to the mitkView contains this model

8.328.2.20 virtual void mitkWidgetModel::Update () [pure virtual]

Update the parameters of the widget.

Implemented in mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkLineWidgetModel3D, mitkPseudocolorWidgetModelEx, mitkReslicePlaneWidgetModel, mitkWidgetModel2D, and mitkWidgetModel3D.

8.328.2.21 void mitkWidgetModel::UpdateObservers () [inline]

Update the observers for calling from outside (e.g. by a manipulator).

The documentation for this class was generated from the following file:

- mitkWidgetModel.h

8.329 mitkWidgetModel2D Class Reference

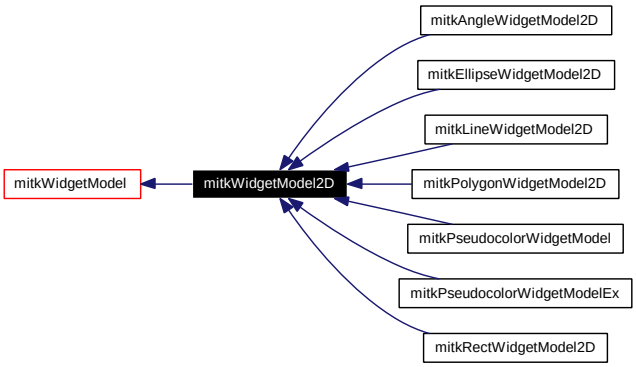
mitkWidgetModel2D - abstract class used to represent a 2D widget entity

`#include <mitkWidgetModel2D.h>`

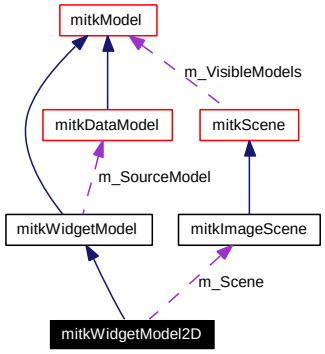
Inherits mitkWidgetModel.

Inherited by mitkAngleWidgetModel2D, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, and mitkRectWidgetModel2D.

Inheritance diagram for mitkWidgetModel2D:



Collaboration diagram for mitkWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void SetScene (mitkScene *scene)
- virtual void SetSourceModel (mitkDataModel *model)
- void SetUnits (float ux, float uy)
- void SetUnits (float units[2])
- void SetColor (float r, float g, float b, float a=1.0)
- void SetColor (int r, int g, int b, int a=255)
- virtual void Update ()
- virtual mitkVolume * GetCurrentImage ()
- virtual mitkVolume * GetRegionMask ()

8.329.1 Detailed Description

mitkWidgetModel2D - abstract class used to represent a 2D widget entity

mitkWidgetModel2D is an abstract class used to represent a 2D widget entity (e.g. a line or an angle) in a rendering scene. It can make responses to the mouse events and manipulate the mitkDataModel to which it attaches.

Note:
2D widgets are only supposed to be attached to 2D data models (e.g. mitkImageModel). Attaching them to a 3D data model could induce improper display.

8.329.2 Member Function Documentation

8.329.2.1 virtual mitkVolume* mitkWidgetModel2D::GetCurrentImage () [virtual]

Get the slice data which currently displayed by the source model (a mitkImageModel).

Returns:
Return a pointer of mitkVolume object contains the slice data.

Note:
The returned object pointer should be deleted properly by yourself.

8.329.2.2 virtual mitkVolume* mitkWidgetModel2D::GetRegionMask () [inline, virtual]

Get mask image where the value of widget region is 255 and the rest is 0.

Returns:
Return a pointer of mitkVolume object contains the mask image.

Note:
The returned object pointer should be deleted properly by yourself.

Reimplemented in mitkEllipseWidgetModel2D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, and mitkRectWidgetModel2D.

8.329.2.3 virtual void mitkWidgetModel2D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel.

Reimplemented in mitkAngleWidgetModel2D, mitkEllipseWidgetModel2D, mitkLineWidgetModel2D, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, and mitkRectWidgetModel2D.

8.329.2.4 void mitkWidgetModel2D::SetColor (int *r*, int *g*, int *b*, int *a* = 255)

Set the color of the widget when it is not active. The value range is from 0 to 255.

Parameters:

r red value of the color

g green value of the color

b blue value of the color

a alpha value of the color (the default value is 255)

8.329.2.5 void mitkWidgetModel2D::SetColor (float *r*, float *g*, float *b*, float *a* = 1.0)

Set the color of the widget when it is not active. The value range is from 0.0 to 1.0.

Parameters:

r red value of the color

g green value of the color

b blue value of the color

a alpha value of the color (the default value is 1.0)

8.329.2.6 virtual void mitkWidgetModel2D::SetScene (mitkScene * *scene*) [virtual]

Set the scene which contains this model.

Parameters:

scene pointer to the mitkScene contains this model

Implements mitkWidgetModel.

8.329.2.7 virtual void mitkWidgetModel2D::SetSourceModel (mitkDataModel * *model*) [virtual]

Associate this widget with a model.

Parameters:

model pointer to an miekDataModel to which this widget attach

Reimplemented from mitkWidgetModel.

Reimplemented in mitkPseudocolorWidgetModelEx.

8.329.2.8 void mitkWidgetModel2D::SetUnits (float *units*[2]) [inline]

Set unit length on axes.

Parameters:

units[0] unit length in x-axis

units[1] unit length in y-axis

8.329.2.9 void mitkWidgetModel2D::SetUnits (float *ux*, float *uy*) [inline]

Set unit length on axes.

Parameters:

ux unit length in x-axis

uy unit length in y-axis

8.329.2.10 virtual void mitkWidgetModel2D::Update () [virtual]

Update the parameters of the widget.

Implements mitkWidgetModel.

Reimplemented in mitkPseudocolorWidgetModelEx.

The documentation for this class was generated from the following file:

- mitkWidgetModel2D.h

8.330 mitkWidgetModel3D Class Reference

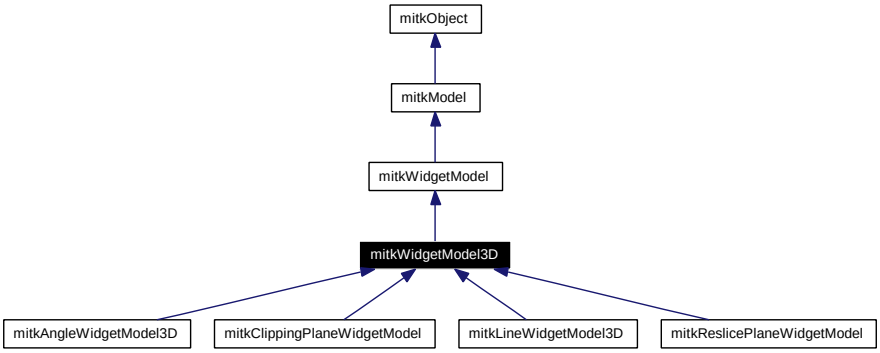
mitkWidgetModel3D - abstract class used to represent a 3D widget entity

`#include <mitkWidgetModel3D.h>`

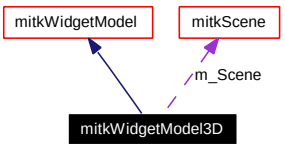
Inherits mitkWidgetModel.

Inherited by mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkLineWidgetModel3D, and mitkReslicePlaneWidgetModel.

Inheritance diagram for mitkWidgetModel3D:



Collaboration diagram for mitkWidgetModel3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void SetScene (mitkScene *scene)
- virtual void SetSourceModel (mitkDataModel *model)
- virtual void Update ()

8.330.1 Detailed Description

mitkWidgetModel3D - abstract class used to represent a 3D widget entity

mitkWidgetModel3D is an abstract class used to represent a 3D widget entity (e.g. a line or an angle) in a rendering scene. It can make responses to the mouse events and manipulate the mitkDataModel to which it attaches.

Note:

3D widgets are only supposed to be attached to 3D data models (e.g. mitkVolumeModel, mitkSurfaceModel). Attaching them to a 2D data model could induce improper display.

8.330.2 Member Function Documentation

8.330.2.1 virtual void mitkWidgetModel3D::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkWidgetModel.

Reimplemented in mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkLineWidgetModel3D, and mitkReslicePlaneWidgetModel.

8.330.2.2 virtual void mitkWidgetModel3D::SetScene (mitkScene * *scene*) [inline, virtual]

Set the scene which contains this model.

Parameters:
scene pointer to the mitkScene contains this model

Implements mitkWidgetModel.

8.330.2.3 virtual void mitkWidgetModel3D::SetSourceModel (mitkDataModel * *model*) [virtual]

Associate this widget with a model.

Parameters:
model pointer to an miekDataModel to which this widget attach

Reimplemented from mitkWidgetModel.

Reimplemented in mitkClippingPlaneWidgetModel, and mitkReslicePlaneWidgetModel.

8.330.2.4 virtual void mitkWidgetModel3D::Update () [virtual]

Update the parameters of the widget.

Implements mitkWidgetModel.

Reimplemented in mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkLineWidgetModel3D, and mitkReslicePlaneWidgetModel.

The documentation for this class was generated from the following file:

- mitkWidgetModel3D.h

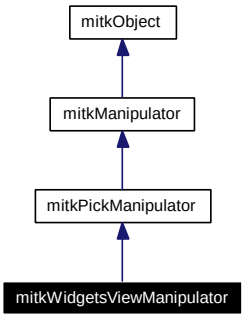
8.331 mitkWidgetsViewManipulator Class Reference

mitkWidgetsViewManipulator - manipulator of a view contains widgets

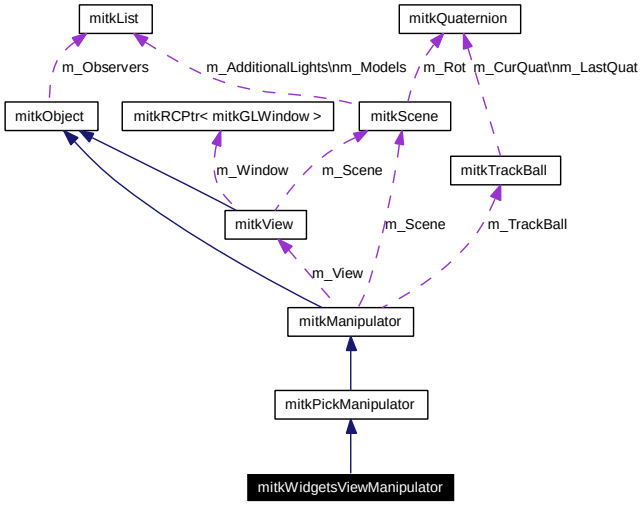
`#include <mitkWidgetsViewManipulator.h>`

Inherits mitkPickManipulator.

Inheritance diagram for mitkWidgetsViewManipulator:



Collaboration diagram for mitkWidgetsViewManipulator:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual void _onMouseDown (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseUp (int mouseButton, bool ctrlDown, bool shiftDown, int xPos, int yPos)
- virtual void _onMouseMove (bool ctrlDown, bool shiftDown, int xPos, int yPos)

8.331.1 Detailed Description

mitkWidgetsViewManipulator - manipulator of a view contains widgets

mitkWidgetsViewManipulator is a manipulator of a view contains widgets. It can select a widget in the scene and drive the widget to manipulate the object in the scene.

8.331.2 Member Function Documentation

8.331.2.1 virtual void mitkWidgetsViewManipulator::_onMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse down event.

Parameters:

mouseButton indicates which mouse button is pressed

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse down event occurs

yPos y-coordinate of the mouse position when mouse down event occurs

Implements mitkManipulator.

8.331.2.2 virtual void mitkWidgetsViewManipulator::_onMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse move event.

Parameters:

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse move event occurs

yPos y-coordinate of the mouse position when mouse move event occurs

Implements mitkManipulator.

8.331.2.3 virtual void mitkWidgetsViewManipulator::_onMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*) [protected, virtual]

Deal with mouse up event.

Parameters:

mouseButton indicates which mouse button was pressed and now released

ctrlDown indicates if the key "Ctrl" is pressed

shiftDown indicates if the key "Shift" is pressed

xPos x-coordinate of the mouse position when mouse up event occurs

yPos y-coordinate of the mouse position when mouse up event occurs

Implements mitkManipulator.

8.331.2.4 virtual void mitkWidgetsViewManipulator::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:

os The specified ostream to output information.

Reimplemented from mitkPickManipulator.

The documentation for this class was generated from the following file:

- mitkWidgetsViewManipulator.h

8.332 mitkWriter Class Reference

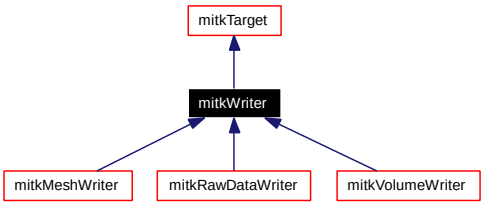
mitkWriter - an abstract class represents a writer

`#include <mitkWriter.h>`

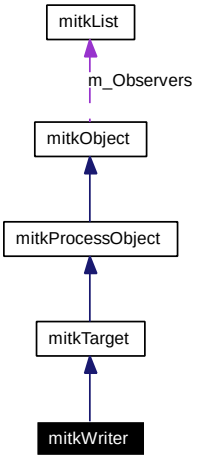
Inherits mitkTarget.

Inherited by mitkMeshWriter, mitkRawDataWriter, and mitkVolumeWriter.

Inheritance diagram for mitkWriter:



Collaboration diagram for mitkWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddFileName (const char *inFileName)
- void SortFileNames ()

8.332.1 Detailed Description

mitkWriter - an abstract class represents a writer

mitkWriter defines the interface of all of the writers. To use a concrete writer, for example, mitkBMPWriter, the code snippet is:

```
mitkBMPWriter *aWriter = new mitkBMPWriter;
aWriter->SetInput(aVolume);
```

```
int imageNum = aVolume->GetImageNum();
Gerenate file names into files[imageNum];
for(int i = 0; i < imageNum; i++)
    aWriter->AddFileName(files[i]);
aWriter->Run();
```

8.332.2 Member Function Documentation

8.332.2.1 void mitkWriter::AddFileName (const char * *inFileName*)

Add a file into the internal list for writing these files.

Parameters:
inFileName C string for the file name.

8.332.2.2 virtual void mitkWriter::PrintSelf (ostream & *os*) [virtual]

Print the necessary information about this object for the debugging purpose.

Parameters:
os The specified ostream to output information.

Reimplemented from mitkTarget.

Reimplemented in mitkAnalyzeImageWriter, mitkBMPWriter, mitkCacheVolumeWriter, mitkDICOMWriter, mitkJPEGWriter, mitkMeshWriter, mitkMetaImageWriter, mitkNetCDFImageWriter, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkRawFilesWriter, mitkRawWriter, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkTIFFWriter, mitkVolumeWriter, mitkRawDataWriter, and mitkTrackedBscanDataWriter.

8.332.2.3 void mitkWriter::SortFileNames ()

Sort all the file names alphabetically.

The documentation for this class was generated from the following file:

- mitkWriter.h