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

Fri Apr 20 20:56:32 2007

Contents

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) 2.0 beta Directories

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

Chapter 3

MITK Hierarchical Index

3.1 MITK (Medical Imaging ToolKit) 2.0 beta Class Hierarchy

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

Chapter 4

MITK Class Index

4.1 MITK (Medical Imaging ToolKit) 2.0 beta Class List

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

Chapter 5

MITK Directory Documentation

5.1 E:/MyWorks/Distribute/mitk20betadoc/Include/ Directory Reference

Files

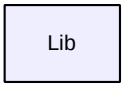
- file **mitkAffineTransform.h**
- file **mitkAmoebaOptimizer.h**
- file **mitkAngleWidgetModel2D.h**
- file **mitkAngleWidgetModel3D.h**
- file **mitkBinaryFilter.h**
- file **mitkBinMarchingCubes.h**
- file **mitkBMPReader.h**
- file **mitkBMPWriter.h**
- file **mitkCacheVolumeReader.h**
- file **mitkCacheVolumeWriter.h**
- file **mitkCamera.h**
- file **mitkClippingPlaneWidgetModel.h**
- file **mitkColorTable.h**
- file **mitkColorTransferFunction.h**
- file **mitkConfig.h**
- file **mitkDataModel.h**
- file **mitkDataObject.h**
- file **mitkDICOMInfoReader.h**
- file **mitkDICOMReader.h**
- file **mitkDICOMStructure.h**
- file **mitkDICOMTags.h**
- file **mitkDICOMWriter.h**
- file **mitkDiffusionFilter.h**
- file **mitkDirectionEncoder.h**

- file **mitkDistanceTransformSaito.h**
- file **mitkEllipseWidgetModel2D.h**
- file **mitkEncodedGradientEstimator.h**
- file **mitkEncodedGradientShader.h**
- file **mitkFilter.h**
- file **mitkFiniteDifferenceFunction.h**
- file **mitkFiniteDifferenceGradientEstimator.h**
- file **mitkFiniteDifferenceImageFilter.h**
- file **mitkFootprint.h**
- file **mitkFootprint1D.h**
- file **mitkFootprint1DGaussian.h**
- file **mitkFootprint2D.h**
- file **mitkFootprint2DGaussian.h**
- file **mitkGarbageCollection.h**
- file **mitkGeometryTypes.h**
- file **mitkGlobal.h**
- file **mitkHalfEdgeStructures.h**
- file **mitkHEICTriangleMesh.h**
- file **mitkHEMesh.h**
- file **mitkHEOoCTriangleMesh.h**
- file **mitkHETriangleMesh.h**
- file **mitkICTriangleMesh.h**
- file **mitkICVolume.h**
- file **mitkIM0Reader.h**
- file **mitkIM0Writer.h**
- file **mitkImageModel.h**
- file **mitkImageView.h**
- file **mitkImageViewManipulatorStandard.h**
- file **mitkImageViewManipulatorWithWidgets.h**
- file **mitkImplementor.h**
- file **mitkInfoReader.h**
- file **mitkInterpolateFilter.h**
- file **mitkJPEGReader.h**
- file **mitkJPEGWriter.h**
- file **mitkLight.h**
- file **mitkLineWidgetModel2D.h**
- file **mitkLineWidgetModel3D.h**
- file **mitkList.h**
- file **mitkLiveWireImageFilter.h**
- file **mitkManipulator.h**
- file **mitkMarchingCubes.h**
- file **mitkMatrix.h**
- file **mitkMatrixD.h**
- file **mitkMCTables.h**
- file **mitkMeanSquaresMetric.h**
- file **mitkMesh.h**
- file **mitkMeshReader.h**
- file **mitkMeshToMeshFilter.h**
- file **mitkMeshViewManipulatorStandard.h**
- file **mitkMeshWriter.h**

- file **mitkMetric.h**
- file **mitkModel.h**
- file **mitkMorphFilter.h**
- file **mitkNearestNeighborInterpolateFilter.h**
- file **mitkNode.h**
- file **mitkNodeHeap.h**
- file **mitkObject.h**
- file **mitkObserver.h**
- file **mitkOoCSurfaceRendererStandard.h**
- file **mitkOoCSurfaceRendererUseVA.h**
- file **mitkOoCTriangleMesh.h**
- file **mitkOoCVolume.h**
- file **mitkOoCVolumeRayCastCompositeFunction.h**
- file **mitkOoCVolumeRendererRayCasting.h**
- file **mitkOoCVolumeRendererTexture3D.h**
- file **mitkOpenGLExam.h**
- file **mitkOptimizer.h**
- file **mitkPickManipulator.h**
- file **mitkPlane.h**
- file **mitkPLYASCIIWriter.h**
- file **mitkPLYBinaryWriter.h**
- file **mitkPLYReader.h**
- file **mitkPolygonWidgetModel2D.h**
- file **mitkProcessObject.h**
- file **mitkPseudocolorWidgetModel.h**
- file **mitkPseudocolorWidgetModelEx.h**
- file **mitkQEMSimplification.h**
- file **mitkQuaternion.h**
- file **mitkRawFilesReader.h**
- file **mitkRawFilesWriter.h**
- file **mitkRawReader.h**
- file **mitkRawWriter.h**
- file **mitkRCPtr.h**
- file **mitkReader.h**
- file **mitkRectWidgetModel2D.h**
- file **mitkRecursiveSphereDirectionEncoder.h**
- file **mitkRegionGrowImageFilter.h**
- file **mitkRegistrationFilter.h**
- file **mitkRenderer.h**
- file **mitkReslicePlaneWidgetModel.h**
- file **mitkRGBToGrayFilter.h**
- file **mitkRigidTransform.h**
- file **mitkSeedFillFilter.h**
- file **mitkSIMD.h**
- file **mitkSobelEdgeDetectFilter.h**
- file **mitkSource.h**
- file **mitkSpeedImageBuilder.h**
- file **mitkSplatCamera.h**
- file **mitkSTLASCIIWriter.h**
- file **mitkSTLBinaryWriter.h**

- file **mitkSurfaceModel.h**
- file **mitkSurfaceProperty.h**
- file **mitkSurfaceRenderer.h**
- file **mitkSurfaceRendererStandard.h**
- file **mitkSurfaceRendererUseVA.h**
- file **mitkSurfaceRendererUseVBO.h**
- file **mitkSystemIncludes.h**
- file **mitkTarget.h**
- file **mitkTemplateExport.h**
- file **mitkThresholdSegmentationFilter.h**
- file **mitkTIFFReader.h**
- file **mitkTIFFWriter.h**
- file **mitkTrackBall.h**
- file **mitkTransferFunction.h**
- file **mitkTransferFunction1D.h**
- file **mitkTransform.h**
- file **mitkTriangleMesh.h**
- file **mitkTriangleMeshSimplification.h**
- file **mitkView.h**
- file **mitkVolume.h**
- file **mitkVolumeCropFilter.h**
- file **mitkVolumeDataTypeConvertor.h**
- file **mitkVolumeModel.h**
- file **mitkVolumeProperty.h**
- file **mitkVolumeRayCastCompositeFunction.h**
- file **mitkVolumeRayCastFunction.h**
- file **mitkVolumeReader.h**
- file **mitkVolumeRenderer.h**
- file **mitkVolumeRendererRayCasting.h**
- file **mitkVolumeRendererRayCastingLoD.h**
- file **mitkVolumeRendererSplatting.h**
- file **mitkVolumeRendererTexture3D.h**
- file **mitkVolumeResizeFilter.h**
- file **mitkVolumeResliceFilter.h**
- file **mitkVolumeSplatFunction.h**
- file **mitkVolumeSplatParallel.h**
- file **mitkVolumeSplatPerspective.h**
- file **mitkVolumeToMeshFilter.h**
- file **mitkVolumeToVolumeFilter.h**
- file **mitkVolumeWriter.h**
- file **mitkWidgetModel.h**
- file **mitkWidgetModel2D.h**
- file **mitkWidgetModel3D.h**
- file **mitkWidgetsViewManipulator.h**
- file **mitkWin32Implementor.h**
- file **mitkWriter.h**

5.2 E:/MyWorks/Distribute/mitk20betadoc/Lib/ Directory Reference



Files

- file **Mitk_dll.dll**
- file **Mitk_dll.lib**

Chapter 6

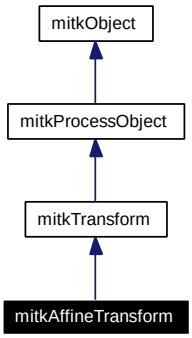
MITK Class Documentation

6.1 mitkAffineTransform Class Reference

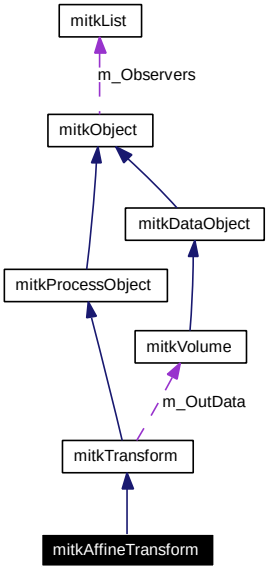
mitkAffineTransform - a concrete transform to perform affine transformation

`#include <mitkAffineTransform.h>`

Inheritance diagram for mitkAffineTransform:



Collaboration diagram for mitkAffineTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- void **Initialize** ()
- virtual bool **Execute** ()
- virtual bool **_transform** (float out[3], float x, float y, float z)

Protected Attributes

- float **m_Matrix** [3][4]

6.1.1 Detailed Description

mitkAffineTransform - a concrete transform to perform affine transformation

mitkAffineTransform is a concrete transform to perform affine transformation.

6.1.2 Member Function Documentation

6.1.2.1 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:

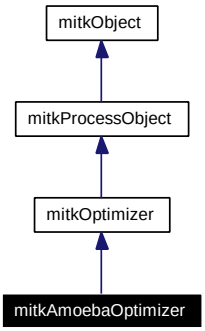
- mitkAffineTransform.h

6.2 mitkAmoebaOptimizer Class Reference

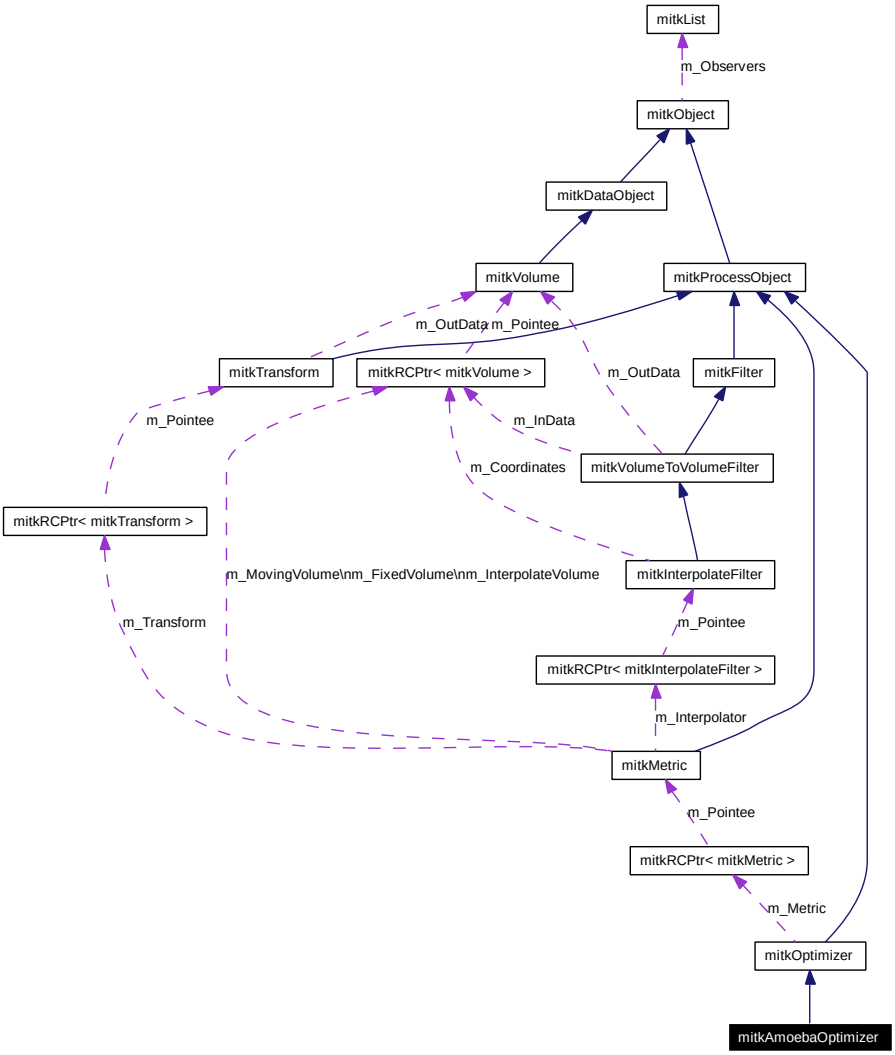
mitkAmoebaOptimizer - a concrete class for implementation of the Nelder-Meade downhill simplex algorithm

`#include <mitkAmoebaOptimizer.h>`

Inheritance diagram for mitkAmoebaOptimizer:



Collaboration diagram for mitkAmoebaOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual bool **Execute** ()

6.2.1 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.

6.2.2 Member Function Documentation

6.2.2.1 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.

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

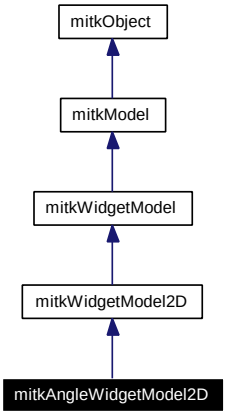
- mitkAmoebaOptimizer.h

6.3 mitkAngleWidgetModel2D Class Reference

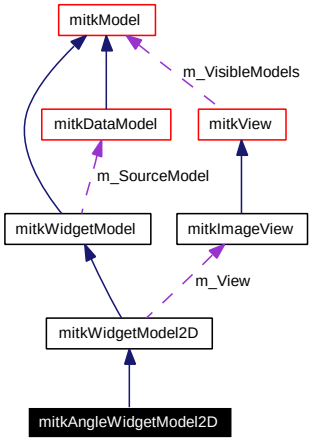
mitkAngleWidgetModel2D - a 2D widget for measuring angles

`#include <mitkAngleWidgetModel2D.h>`

Inheritance diagram for mitkAngleWidgetModel2D:



Collaboration diagram for mitkAngleWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view)
- 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 Types

- enum {

  unknown, **dot**, **arrow0**, **arrow1**,

  line }

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)
- virtual float * **_getBounds** ()
- void **_init** ()

Protected Attributes

- float **m_StartPoint** [2]
- float **m_EndPoint0** [2]
- float **m_EndPoint1** [2]
- float **m_ArrowLength**
- float **m_ArrowColor** [4]
- float **m_DotColor** [4]
- float **m_LineColor** [4]
- float **m_PickedArrowColor** [4]
- float **m_PickedDotColor** [4]
- float **m_PickedLineColor** [4]
- bool **m_StartPointSet**
- bool **m_EndPoint0Set**
- bool **m_EndPoint1Set**

6.3.1 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 view (e.g. mitkImageView), and in other conditions the display could be improper.

6.3.2 Member Function Documentation

6.3.2.1 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.

6.3.2.2 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.

6.3.2.3 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.

6.3.2.4 float mitkAngleWidgetModel2D::GetAngleInDegree ()

Get the value of angle in degree.

Returns:

Return value of this angle in degree.

6.3.2.5 float mitkAngleWidgetModel2D::GetAngleInRadian ()

Get the value of angle in radian.

Returns:
Return value of this angle in radian.

6.3.2.6 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.

6.3.2.7 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.

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

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.3.2.9 virtual int mitkAngleWidgetModel2D::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.3.2.10 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.3.2.11 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.

6.3.2.12 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.

6.3.2.13 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.3.2.14 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.

6.3.2.15 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.

6.3.2.16 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.3.2.17 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.

6.3.2.18 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:

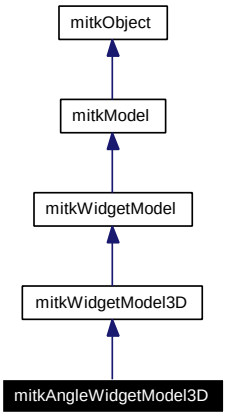
- mitkAngleWidgetModel2D.h

6.4 mitkAngleWidgetModel3D Class Reference

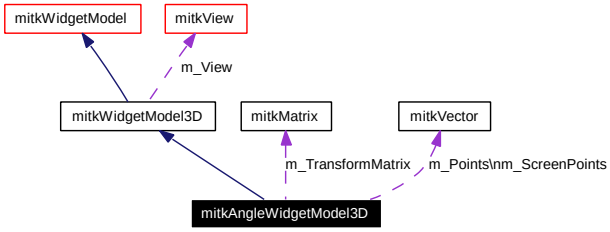
mitkAngleWidgetModel3D - a 3D widget for measuring angles

`#include <mitkAngleWidgetModel3D.h>`

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 (mitkView *view)
- 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 Types

- enum {
  unknown, **ball**, **arrow0**, **arrow1**,
  line }

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)
- virtual float * **_getBounds** ()
- void **_init** ()

Protected Attributes

- mitkVector * **m_Points**
- mitkVector * **m_ScreenPoints**
- mitkMatrix * **m_TransformMatrix**
- float **m_UnitsPerPixel** [3]
- float **m_ArrowLength**
- float **m_ArrowColor** [4]
- float **m_BallColor** [4]
- float **m_LineColor** [4]
- float **m_PickedArrowColor** [4]
- float **m_PickedBallColor** [4]
- float **m_PickedLineColor** [4]

6.4.1 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 view (e.g. mitkView), and in other conditions the display could be improper.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 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

6.4.3 Member Function Documentation

6.4.3.1 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.

6.4.3.2 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.

6.4.3.3 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.

6.4.3.4 float mitkAngleWidgetModel3D::GetAngleInDegree ()

Get the value of angle in degree.

Returns:

Return value of this angle in degree.

6.4.3.5 float mitkAngleWidgetModel3D::GetAngleInRadian ()

Get the value of angle in radian.

Returns:
Return value of this angle in radian.

6.4.3.6 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.

6.4.3.7 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.

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

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.4.3.9 virtual int mitkAngleWidgetModel3D::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.4.3.10 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

6.4.3.11 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

6.4.3.12 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:

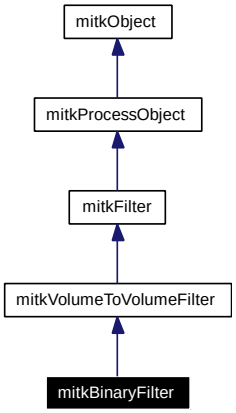
- mitkAngleWidgetModel3D.h

6.5 mitkBinaryFilter Class Reference

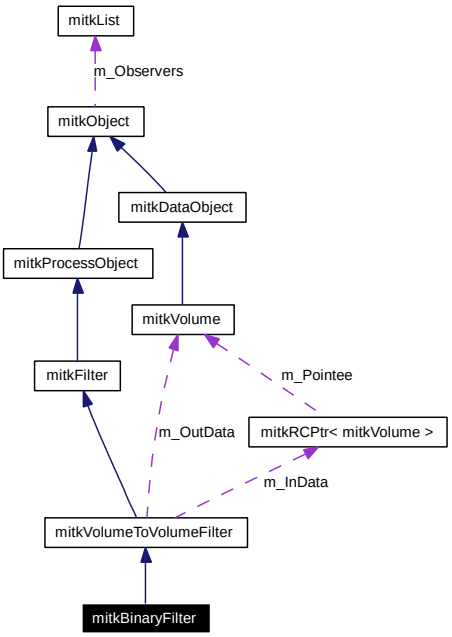
mitkBinaryFilter - a filter to get binary data from a source volume

`#include <mitkBinaryFilter.h>`

Inheritance diagram for mitkBinaryFilter:



Collaboration diagram for mitkBinaryFilter:



Public Member Functions

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

Protected Member Functions

- virtual bool **Execute** ()

6.5.1 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.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 mitkBinaryFilter::mitkBinaryFilter ()

Default constructor.

6.5.3 Member Function Documentation

6.5.3.1 float mitkBinaryFilter::GetCubage ()

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

6.5.3.2 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:

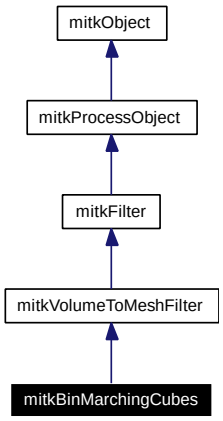
- mitkBinaryFilter.h

6.6 mitkBinMarchingCubes Class Reference

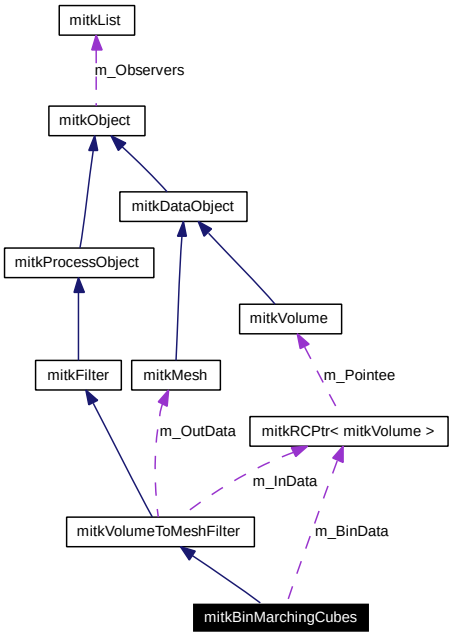
mitkBinMarchingCubes - a marching cubes algorithm using binary data

`#include <mitkBinMarchingCubes.h>`

Inheritance diagram for mitkBinMarchingCubes:



Collaboration diagram for mitkBinMarchingCubes:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkBinMarchingCubes ()
- void SetBinData (mitkVolume *binData)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_BinData**

6.6.1 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.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 mitkBinMarchingCubes::mitkBinMarchingCubes ()

Default constructor.

6.6.3 Member Function Documentation

6.6.3.1 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.

6.6.3.2 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 mitkBinMarching-Cubes 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 mitkBinary-Filter to get the binary data finally.

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

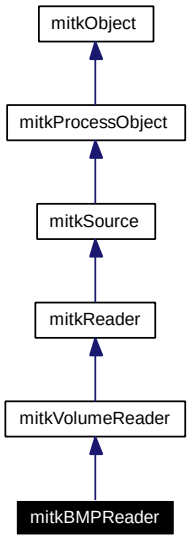
- mitkBinMarchingCubes.h

6.7 mitkBMPReader Class Reference

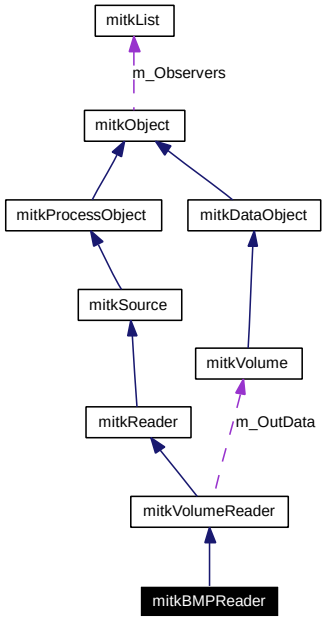
mitkBMPReader - a concrete reader for reading BMP image files

`#include <mitkBMPReader.h>`

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])

Protected Types

- typedef mitkBMPReader::tagBITMAPFILEHEADER **BITMAPFILEHEADER**
- typedef mitkBMPReader::tagBITMAPINFOHEADER **BITMAPINFOHEADER**
- typedef mitkBMPReader::tag_bmp_core_header **BMP_CORE_HEADER**

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- float **m_Spacings** [3]

6.7.1 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.

6.7.2 Member Function Documentation

6.7.2.1 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.

6.7.2.2 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.

6.7.2.3 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.

6.7.2.4 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.

6.7.2.5 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:

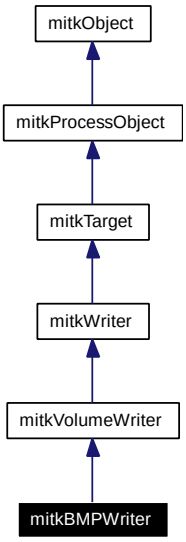
- mitkBMPReader.h

6.8 mitkBMPWriter Class Reference

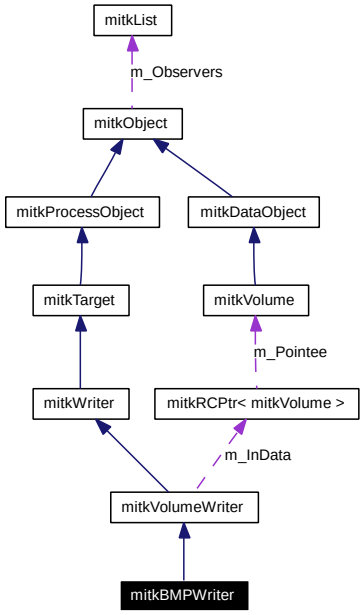
mitkBMPWriter - a concrete writer for writing a volume to BMP image files

`#include <mitkBMPWriter.h>`

Inheritance diagram for mitkBMPWriter:



Collaboration diagram for mitkBMPWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Types

- typedef mitkBMPWriter::tagBITMAPFILEHEADER **BITMAPFILEHEADER**

Protected Member Functions

- virtual bool **Execute** ()

6.8.1 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();
```

6.8.2 Member Function Documentation

6.8.2.1 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:

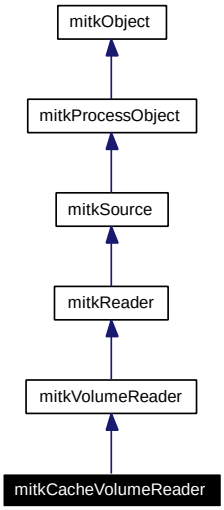
- mitkBMPWriter.h

6.9 mitkCacheVolumeReader Class Reference

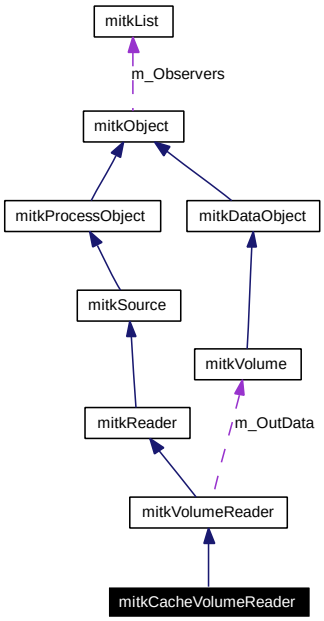
mitkCacheVolumeReader - a volume reader for reading out-of-core volume cache files

`#include <mitkCacheVolumeReader.h>`

Inheritance diagram for mitkCacheVolumeReader:



Collaboration diagram for mitkCacheVolumeReader:



Public Member Functions

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

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- bool **m_IsCfgFileBinary**

6.9.1 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.

6.9.2 Member Function Documentation

6.9.2.1 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.

6.9.2.2 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:

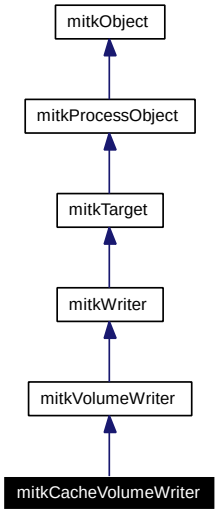
- mitkCacheVolumeReader.h

6.10 mitkCacheVolumeWriter Class Reference

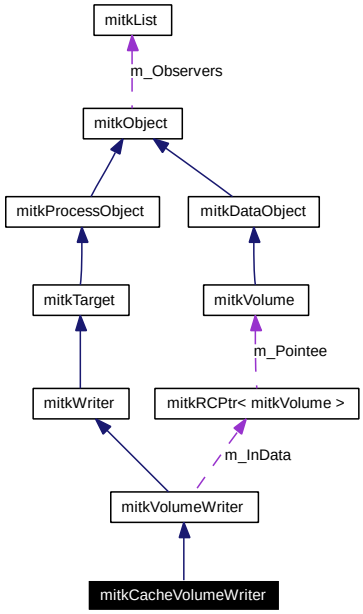
mitkCacheVolumeWriter - a volume writer for reusing out-of-core volume cache files

`#include <mitkCacheVolumeWriter.h>`

Inheritance diagram for mitkCacheVolumeWriter:



Collaboration diagram for mitkCacheVolumeWriter:



Public Member Functions

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

Protected Member Functions

- virtual bool **Execute** ()
- bool **_writeBinaryFile** (char const *filename)

Protected Attributes

- bool **m_IsCfgFileBinary**

6.10.1 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();
```

6.10.2 Member Function Documentation

6.10.2.1 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.

6.10.2.2 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:

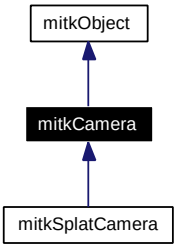
- mitkCacheVolumeWriter.h

6.11 mitkCamera Class Reference

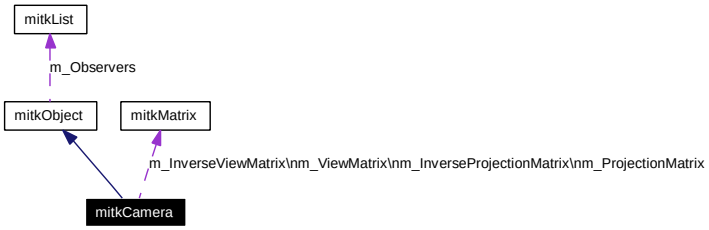
mitkCamera - a camera in 3D view

`#include <mitkCamera.h>`

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])

Protected Attributes

- float **m_FocalPoint** [3]
- float **m_Thickness**
- mitkMatrix * **m_ViewMatrix**
- mitkMatrix * **m_ProjectionMatrix**
- mitkMatrix * **m_InverseProjectionMatrix**
- mitkMatrix * **m_InverseViewMatrix**
- bool **m_ParallelProjection**
- bool **m_Modified**

6.11.1 Detailed Description

mitkCamera - a camera in 3D view

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

6.11.2 Member Function Documentation

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

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

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

Transform a point from camera coordinate to world coordinate.

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

Perspective Projection

Reimplemented in mitkSplatCamera.

6.11.2.4 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

6.11.2.5 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

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

Get the inverse of projection transform matrix.

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

Get the inverse of projection transform matrix.

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

Get the inverse of projection transform matrix.

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

Get the inverse of viewing transform matrix.

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

Get the inverse of viewing transform matrix.

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

Get the inverse of viewing transform matrix.

6.11.2.12 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

6.11.2.13 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

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

Get the projection transform matrix.

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

Get the projection transform matrix.

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

Get the projection transform matrix.

6.11.2.17 float mitkCamera::GetThickness () [inline]

Get the camera thickness.

Returns:

Return the camera thickness

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

Get the viewing transform matrix.

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

Get the viewing transform matrix.

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

Get the viewing transform matrix.

6.11.2.21 bool mitkCamera::IsModified () [inline]

Test whether this camera is modified.

Returns:
Return true if this camera is modified.

6.11.2.22 bool mitkCamera::IsParallelProjection () [inline]

Get the projection mode.

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

6.11.2.23 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.

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

Parallel Projection

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

Perspective Projection

Reimplemented in mitkSplatCamera.

6.11.2.26 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.

6.11.2.27 void mitkCamera::SetProjectionToParallel () [inline]

Set the projection mode to parallel.

6.11.2.28 void mitkCamera::SetProjectionToPerspective () [inline]

Set the projection mode to perspective.

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

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

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

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

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

Transform a point from world coordinate to camera coordinate.

6.11.2.32 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:

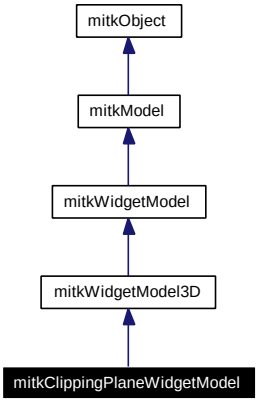
- mitkCamera.h

6.12 mitkClippingPlaneWidgetModel Class Reference

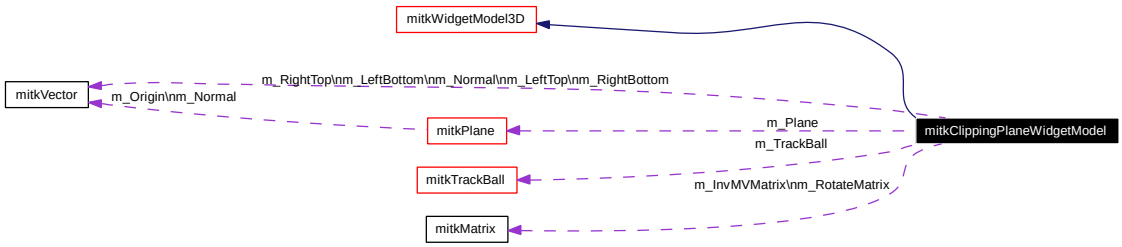
mitkClippingPlaneWidgetModel - a 3D widget for clipping plane

`#include <mitkClippingPlaneWidgetModel.h>`

Inheritance diagram for mitkClippingPlaneWidgetModel:



Collaboration diagram for mitkClippingPlaneWidgetModel:



Public Types

- typedef float **coord_type**

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 (mitkView *view)
- 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 Types

- enum {

  unknown, **lbball**, **ltball**, **rbball**,

  rtball, **lline**, **tline**, **rline**,

  bline, **plane** }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()
- void **_rotate** ()

Protected Attributes

- mitkPlane * **m_Plane**
- mitkVector * **m_LeftBottom**
- mitkVector * **m_RightBottom**
- mitkVector * **m_RightTop**
- mitkVector * **m_LeftTop**
- mitkVector * **m_Normal**
- mitkMatrix * **m_InvMVMatrix**
- mitkMatrix * **m_RotateMatrix**
- mitkTrackBall **m_TrackBall**
- float **m_ArrowLength**
- float **m_ArrowColor** [4]
- float **m_BallColor** [4]
- float **m_LineColor** [4]
- float **m_PickedBallColor** [4]
- float **m_PickedLineColor** [4]
- float **m_Opacity**
- float **m_MoveUnit**

6.12.1 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.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 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 view, the default value is 100.0

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

6.12.2.2 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

6.12.3 Member Function Documentation

6.12.3.1 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.

6.12.3.2 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.

6.12.3.3 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.

6.12.3.4 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.

6.12.3.5 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.

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

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.12.3.7 virtual int mitkClippingPlaneWidgetModel::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:

view the pointer of an mitkView in which this model will be shown

Returns:

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

Reimplemented from mitkModel.

6.12.3.8 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

6.12.3.9 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

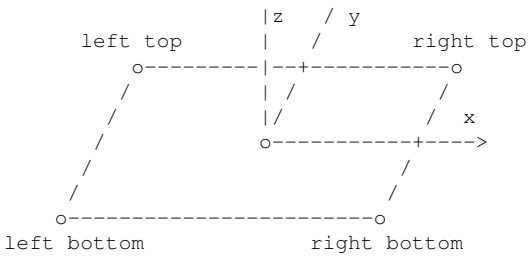
6.12.3.10 void mitkClippingPlaneWidgetModel::RotateDegAroundZAxisOfPlane (float *angle*)

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

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

Rotate the plane around the x axis of the plane.

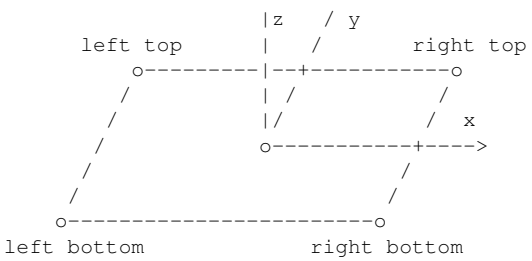


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

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

Rotate the plane around the y axis of the plane.

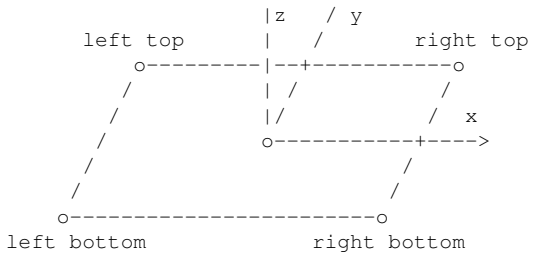


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

6.12.3.13 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

6.12.3.14 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)

6.12.3.15 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

6.12.3.16 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

6.12.3.17 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.

6.12.3.18 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

6.12.3.19 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:

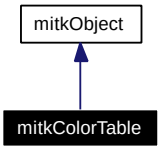
- mitkClippingPlaneWidgetModel.h

6.13 mitkColorTable Class Reference

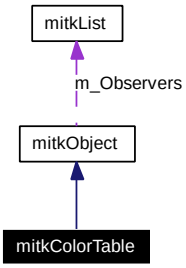
mitkColorTable - a table mapping pixel value to color

`#include <mitkColorTable.h>`

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)

Protected Member Functions

- int **_getIndex** (double pixVal)
- int **_getIndex** (int index)

Protected Attributes

- mitkColorList * **m_ColorList**

6.13.1 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.

6.13.2 Constructor & Destructor Documentation

6.13.2.1 mitkColorTable::mitkColorTable ()

Default constructor.

6.13.3 Member Function Documentation

6.13.3.1 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

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

Copy the entities from another color table.

Parameters:

src the source color table to copy from

6.13.3.3 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.

6.13.3.4 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.

6.13.3.5 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.

6.13.3.6 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.

6.13.3.7 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.

6.13.3.8 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.

6.13.3.9 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.

6.13.3.10 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.

6.13.3.11 int mitkColorTable::GetColorCount ()

Get the number of mapping entities in the table.

6.13.3.12 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.

6.13.3.13 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.

6.13.3.14 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.

6.13.3.15 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.

6.13.3.16 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.

6.13.3.17 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.

6.13.3.18 void mitkColorTable::RemoveAllColors ()

Remove all mapping entities.

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

Remove a mapping entity by index.

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

6.13.3.20 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

6.13.3.21 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.

6.13.3.22 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

6.13.3.23 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.

6.13.3.24 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:

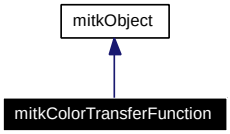
- mitkColorTable.h

6.14 mitkColorTransferFunction Class Reference

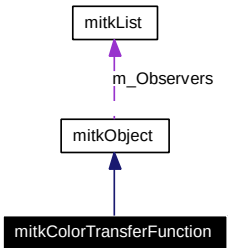
mitkColorTransferFunction - a transfer function to map the scalar value to color

`#include <mitkColorTransferFunction.h>`

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 ()

Protected Member Functions

- void **_buildRFunction** (int x1, float r1, int x2, float r2)
- void **_buildGFunction** (int x1, float g1, int x2, float g2)
- void **_buildBFunction** (int x1, float b1, int x2, float b2)

Protected Attributes

- int **m_MaxX**
- mitkColorPointArray * **m_Points**
- float * **m_RData**
- float * **m_GData**
- float * **m_BData**
- bool **m_Modified**

6.14.1 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.

6.14.2 Member Function Documentation

6.14.2.1 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

6.14.2.2 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
```

.

6.14.2.3 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

6.14.2.4 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.

6.14.2.5 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

6.14.2.6 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
```

.

6.14.2.7 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

6.14.2.8 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.

6.14.2.9 int mitkColorTransferFunction::GetMaxX () [inline]

Get the maximum scalar value of this transfer function.

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

6.14.2.10 int mitkColorTransferFunction::GetPointsCount ()

Get the number of points in this transfer function

Returns:
Return the points number

6.14.2.11 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
```

.

6.14.2.12 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

6.14.2.13 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.

6.14.2.14 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.

6.14.2.15 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.

6.14.2.16 void mitkColorTransferFunction::RemoveAllPoints ()

Remove all points from the color transfer function

6.14.2.17 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.

6.14.2.18 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)

6.14.2.19 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:

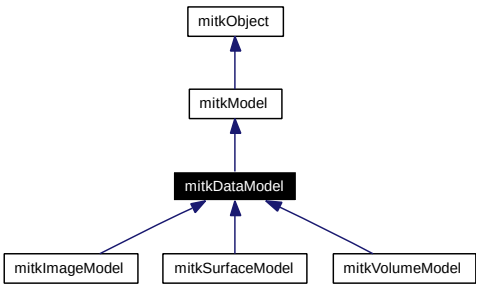
- mitkColorTransferFunction.h

6.15 mitkDataModel Class Reference

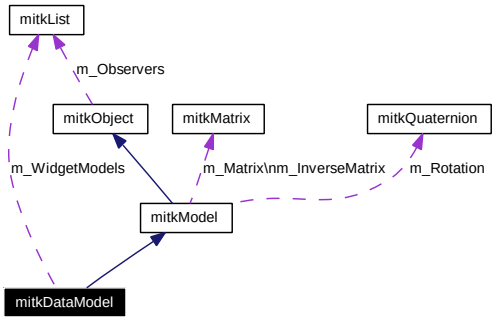
mitkDataModel - abstract class used to represent an data entity in a rendering scene

`#include <mitkDataModel.h>`

Inheritance diagram for mitkDataModel:



Collaboration diagram for mitkDataModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual mitkRenderer * **GetBasicRenderer** ()
- void AddWidget (mitkWidgetModel *widget)
- void UpdateWidgets ()
- void RemoveWidget (mitkWidgetModel *widget)
- void RemoveAllWidgets ()
- int GetWidgetCount ()
- mitkWidgetModel * GetWidgetModel (int index)
- bool IsEmpty ()

Protected Attributes

- mitkList * **m_WidgetModels**
- bool **m_IsEmpty**

6.15.1 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.

6.15.2 Member Function Documentation

6.15.2.1 void mitkDataModel::AddWidget (mitkWidgetModel * *widget*)

Attach a widget to this data model.

Parameters:
widget pointer to an mitkWidgetModel to attach

6.15.2.2 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.

6.15.2.3 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.

6.15.2.4 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.

6.15.2.5 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, mitkSurfaceModel, and mitkVolumeModel.

6.15.2.6 void mitkDataModel::RemoveAllWidgets ()

Detach all widgets from this data model.

6.15.2.7 void mitkDataModel::RemoveWidget (mitkWidgetModel * *widget*)

Detach a widget from this data model.

Parameters:

widget pointer to a mitkWidgetModel to detach

6.15.2.8 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:

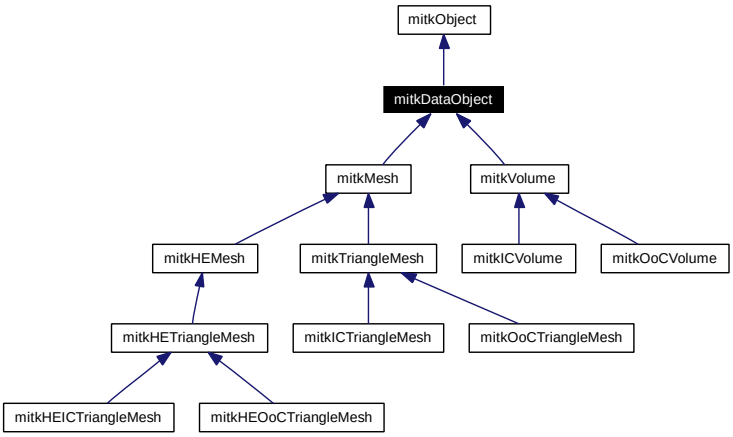
- mitkDataModel.h

6.16 mitkDataObject Class Reference

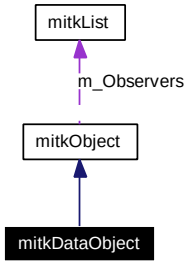
mitkDataObject - an abstract class to represents a data object in MITK

`#include <mitkDataObject.h>`

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 GetActualMemorySize () const =0
- virtual void ShallowCopy (mitkDataObject *src)=0
- virtual void DeepCopy (mitkDataObject *src)=0

6.16.1 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

6.16.2 Member Function Documentation

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

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

Implemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, and mitkVolume.

6.16.2.2 virtual unsigned 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 mitkHEICTriangleMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkOoCTriangleMesh, and mitkOoCVolume.

6.16.2.3 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, and mitkVolume.

6.16.2.4 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 mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkHETriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, and mitkVolume.

6.16.2.5 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, and mitkVolume.

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

Warning:

Internal function. Don't call it directly.

Implemented in mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkMesh, mitkOoCTriangleMesh, mitkOoCVolume, mitkTriangleMesh, and mitkVolume.

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

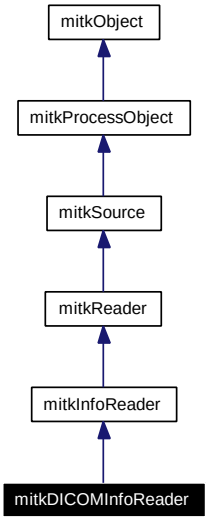
- mitkDataObject.h

6.17 mitkDICOMInfoReader Class Reference

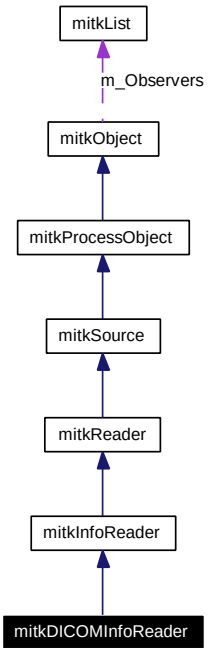
mitkDICOMInfoReader - a concrete reader for reading element information in DICOM files

`#include <mitkDICOMInfoReader.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- DcmFile * **m_DcmFile**

6.17.1 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
    }
}
```

6.17.2 Member Function Documentation

6.17.2.1 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.

6.17.2.2 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.

6.17.2.3 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:

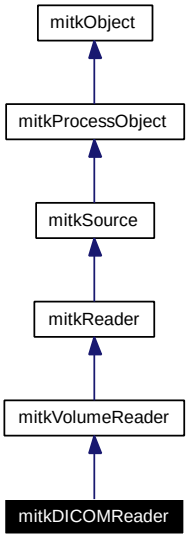
- mitkDICOMInfoReader.h

6.18 mitkDICOMReader Class Reference

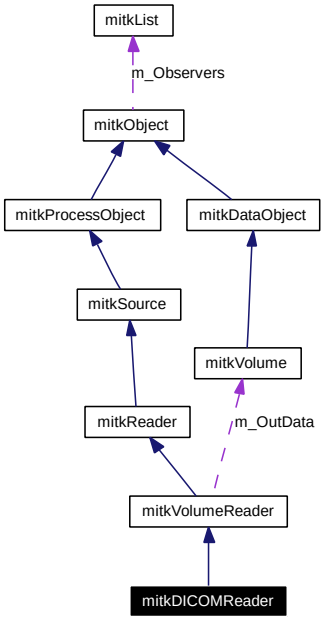
mitkDICOMReader - a concrete reader for reading DICOM image files

`#include <mitkDICOMReader.h>`

Inheritance diagram for mitkDICOMReader:



Collaboration diagram for mitkDICOMReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual bool **Execute** ()
- void **_clear** ()
- int **_preProcess** (VOLUMEINFO &info)
- int **_writeToVolume** (VOLUMEINFO &info)

Protected Attributes

- SliceList * **m_Slices**
- float **m_MinSpacingZ**
- float **m_MaxSpacingZ**
- bool **m_IsMultiFrame**
- bool **m_NeedInsert**

6.18.1 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.

6.18.2 Member Function Documentation

6.18.2.1 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:

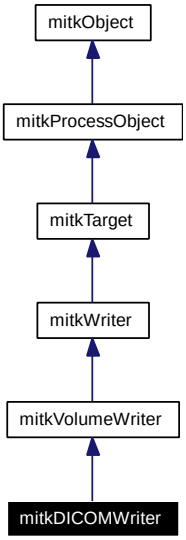
- mitkDICOMReader.h

6.19 mitkDICOMWriter Class Reference

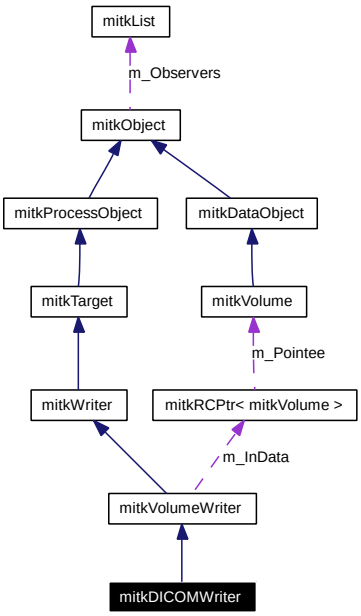
mitkDICOMWriter - a concrete writer for writing a volume to DICOM image files

`#include <mitkDICOMWriter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- char **m_StudyUID** [65]
- char **m_SeriesUID** [65]

6.19.1 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();
```

6.19.2 Member Function Documentation

6.19.2.1 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.

6.19.2.2 void mitkDICOMWriter::SetSeriesUID (const string & *uid*)

Set Series UID.

Parameters:
uid a const reference to a string contains UID

6.19.2.3 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:

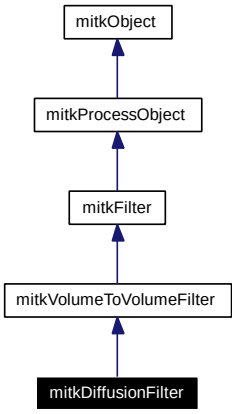
- mitkDICOMWriter.h

6.20 mitkDiffusionFilter Class Reference

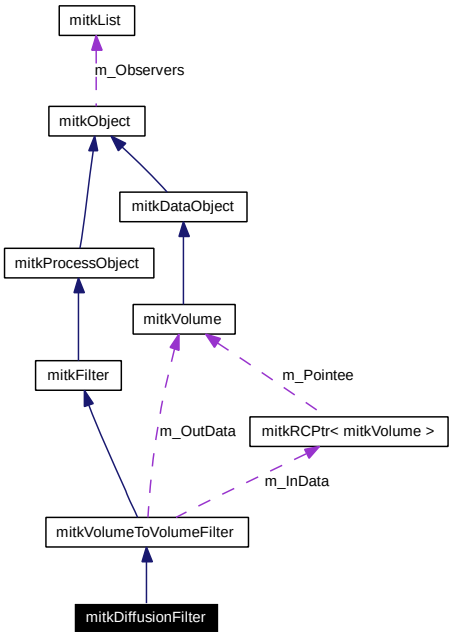
mitkDiffusionFilter - a class used to diffuse the mitkVolume data (2D or 3D)

`#include <mitkDiffusionFilter.h>`

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)

Protected Member Functions

- bool **Execute** ()

Protected Attributes

- double **m_K**
- float **m_TimeInterval**
- bool **m_Enable3D**

6.20.1 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

6.20.2 Member Function Documentation

6.20.2.1 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.

6.20.2.2 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).

6.20.2.3 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.

6.20.2.4 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:

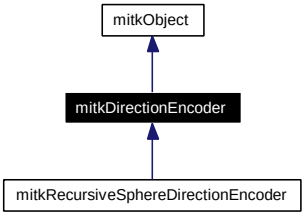
- mitkDiffusionFilter.h

6.21 mitkDirectionEncoder Class Reference

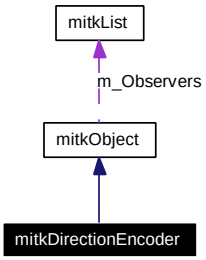
mitkDirectionEncoder - an abstract class to encode a direction into a one or two byte value

`#include <mitkDirectionEncoder.h>`

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

6.21.1 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.

6.21.2 Member Function Documentation

6.21.2.1 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.

6.21.2.2 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.

6.21.2.3 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.

6.21.2.4 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:

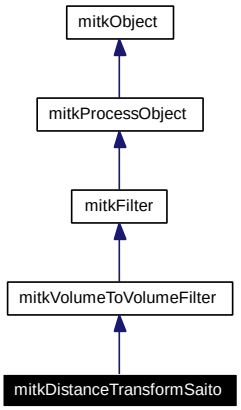
- mitkDirectionEncoder.h

6.22 mitkDistanceTransformSaito Class Reference

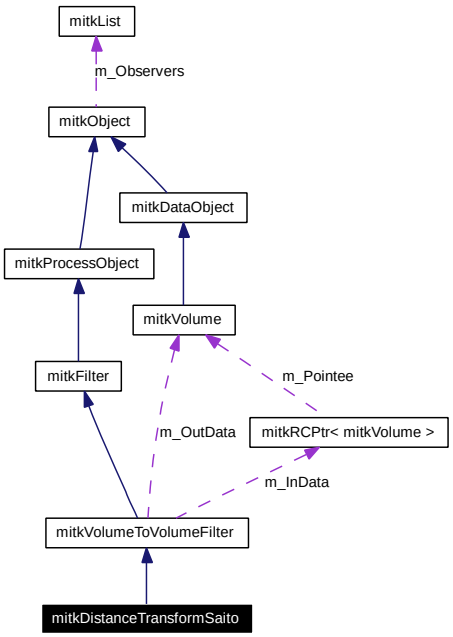
mitkDistanceTransformSaito - computes 3D Euclidean DT

`#include <mitkDistanceTransformSaito.h>`

Inheritance diagram for mitkDistanceTransformSaito:



Collaboration diagram for mitkDistanceTransformSaito:



Public Member Functions

- mitkDistanceTransformSaito ()
- void SetDimension (int d)
- void SetImagePart (int nIp=3)
- int GetImagePart ()

Protected Member Functions

- virtual bool **Execute** ()
- void **PermuteExtent** (mitkVolume *InputVolume, int nExtent[6])
- void **PermuteIncrements** (mitkVolume *InputVolume, int nInc[3])
- float **GetMaximumDistance** (mitkVolume *Volume)

Protected Attributes

- int **m_Dimension**
- int **m_CurrentIteration**
- int **m_Ip**

6.22.1 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.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 mitkDistanceTransformSaito::mitkDistanceTransformSaito ()

Constructor of the class

6.22.3 Member Function Documentation

6.22.3.1 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.

6.22.3.2 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

6.22.3.3 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:

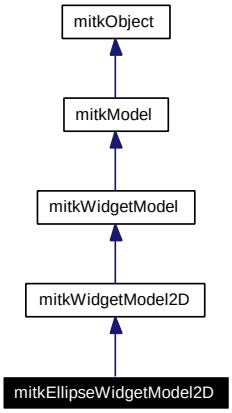
- mitkDistanceTransformSaito.h

6.23 mitkEllipseWidgetModel2D Class Reference

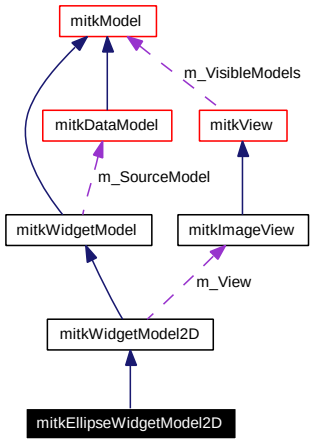
mitkEllipseWidgetModel2D - an ellipse widget used in 2D views

`#include <mitkEllipseWidgetModel2D.h>`

Inheritance diagram for mitkEllipseWidgetModel2D:



Collaboration diagram for mitkEllipseWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view)
- 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 Types

- enum {
  unknown, **ellipse**, **hsline**, **heline**,
  vsline, **veline**, **ssdot**, **esdot**,
  eedot, **sedot** }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()

Protected Attributes

- float **m_DotColor** [4]
- float **m_LineColor** [4]
- float **m_PickedLineColor** [4]
- float **m_PickedDotColor** [4]
- float **m_PickedEllipseColor** [4]
- float **m_PickedCircleColor** [4]
- float **m_CenterPoint** [2]
- float **m_XRadius**
- float **m_YRadius**
- float **m_StartPoint** [2]
- float **m_EndPoint** [2]
- float **m_DotSize**
- bool **m_StartPointSet**
- bool **m_EndPointSet**
- string **m_UnitName**

6.23.1 Detailed Description

mitkEllipseWidgetModel2D - an ellipse widget used in 2D views

mitkEllipseWidgetModel2D is an ellipse widget used in 2D views 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 view (e.g. mitkImageView), or the display could be improper.

6.23.2 Member Function Documentation

6.23.2.1 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.

6.23.2.2 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.

6.23.2.3 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.

6.23.2.4 float mitkEllipseWidgetModel2D::GetArea ()

Get area of this ellipse.

Returns:
Return the area of this ellipse.

6.23.2.5 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

6.23.2.6 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

6.23.2.7 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.

6.23.2.8 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.

6.23.2.9 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.

6.23.2.10 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.

6.23.2.11 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

6.23.2.12 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.

6.23.2.13 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.

6.23.2.14 virtual void mitkEllipseWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.23.2.15 virtual int mitkEllipseWidgetModel2D::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.23.2.16 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

6.23.2.17 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

6.23.2.18 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.23.2.19 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.23.2.20 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.)

6.23.2.21 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.)

6.23.2.22 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.23.2.23 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.)

6.23.2.24 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.)

6.23.2.25 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

6.23.2.26 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

6.23.2.27 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.23.2.28 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.)

6.23.2.29 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.)

6.23.2.30 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:

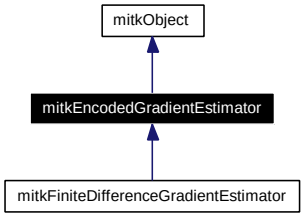
- mitkEllipseWidgetModel2D.h

6.24 mitkEncodedGradientEstimator Class Reference

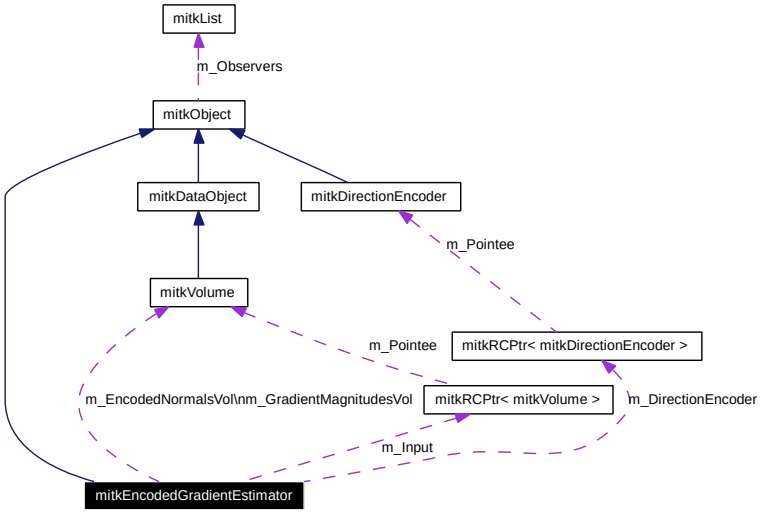
mitkEncodedGradientEstimator - an abstract class for gradient estimation

`#include <mitkEncodedGradientEstimator.h>`

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 ()
- int **GetEncodedNormalIndex** (int xyz_index)
- int **GetEncodedNormalIndex** (int x_index, int y_index, int z_index)
- unsigned char * GetGradientMagnitudes ()
- mitkVolume * GetGradientMagnitudesVol ()
- void SetDirectionEncoder (mitkDirectionEncoder *direnc)
- mitkDirectionEncoder * GetDirectionEncoder ()
- int * **GetInputSize** ()
- void **GetInputSize** (int data[3])
- float * **GetInputAspect** ()
- void **GetInputAspect** (float data[3])

Public Attributes

- mitkRCPtr< mitkVolume > **m_Input**
- unsigned short * **m_EncodedNormals**
- int **m_EncodedNormalsSize** [3]
- unsigned char * **m_GradientMagnitudes**
- mitkVolume * **m_EncodedNormalsVol**
- mitkVolume * **m_GradientMagnitudesVol**

Protected Member Functions

- virtual void **_updateNormals** (void)=0
- void **_computeCircleLimits** (int size)

Protected Attributes

- mitkRCPtr< mitkDirectionEncoder > **m_DirectionEncoder**
- float **m_GradientMagnitudeScale**
- float **m_GradientMagnitudeBias**
- float **m_ZeroNormalThreshold**
- int **m_CylinderClip**
- int * **m_CircleLimits**
- int **m_CircleLimitsSize**
- int **m_UseCylinderClip**
- int **m_BoundsClip**
- int **m_Bounds** [6]
- int **m_InputSize** [3]
- float **m_InputAspect** [3]
- int **m_ComputeGradientMagnitudes**
- int **m_ZeroPad**
- bool **m_EnalbleOoC**

6.24.1 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.

6.24.2 Member Function Documentation

6.24.2.1 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

6.24.2.2 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

6.24.2.3 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

6.24.2.4 virtual int mitkEncodedGradientEstimator::GetBoundsClip () [inline, virtual]

Get the bounds clip value

Returns:

Return the bounds clip value

6.24.2.5 virtual int mitkEncodedGradientEstimator::GetBoundsClipMaxValue () [inline, virtual]

Get the max bounds clip value

Returns:

Return the max bounds clip value

6.24.2.6 virtual int mitkEncodedGradientEstimator::GetBoundsClipMinValue () [inline, virtual]

Get the min bounds clip value

Returns:

Return the min bounds clip value

6.24.2.7 int* mitkEncodedGradientEstimator::GetCircleLimits () [inline]

Get the circle limits value

Returns:

Return the circle limits value

6.24.2.8 int mitkEncodedGradientEstimator::GetComputeGradientMagnitudes () [inline]

Get the compute gradient magnitude value

Returns:
Return the compute gradient magnitude value

6.24.2.9 int mitkEncodedGradientEstimator::GetCylinderClip () [inline]

Get the cylinder clip value

Returns:
Return the cylinder clip value

6.24.2.10 mitkDirectionEncoder* mitkEncodedGradientEstimator::GetDirectionEncoder () [inline]

Get the direction encoder used to encode normal directions to fit within two bytes

Returns:
Return the direction encoder value

6.24.2.11 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.

6.24.2.12 mitkVolume* mitkEncodedGradientEstimator::GetEncodedNormalsVol ()

Get the encoded normals volume (in case of out-of-core data).

Returns:
Return the volume contains the encoded normals.

6.24.2.13 float mitkEncodedGradientEstimator::GetGradientMagnitudeBias () [inline]

Get the scale and bias for the gradient magnitude

Returns:
Return the gradient magnitude

6.24.2.14 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.

6.24.2.15 float mitkEncodedGradientEstimator::GetGradientMagnitudeScale () [inline]

Get the scale and bias for the gradient magnitude

Returns:
Return the scale and bias for the gradient magnitude

6.24.2.16 mitkVolume* mitkEncodedGradientEstimator::GetGradientMagnitudesVol ()

Get the gradient magnitudes volume (in case of out-of-core data).

Returns:
Return the volume contains the gradient magnitude value.

6.24.2.17 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

6.24.2.18 int mitkEncodedGradientEstimator::GetUseCylinderClip () [inline]

Get the use cylinder clip value

Returns:
Return the use cylinder clip value

6.24.2.19 float mitkEncodedGradientEstimator::GetZeroNormalThreshold () [inline]

Get teh zero normal threshold value

Returns:
Return the zero normal threshold value

6.24.2.20 virtual int mitkEncodedGradientEstimator::GetZeroPad () [inline, virtual]

Get the zero pad value

Returns:
Return the zero pad value

6.24.2.21 virtual int mitkEncodedGradientEstimator::GetZeroPadMaxValue () [inline, virtual]

Get the max zero pad value

Returns:
Return the min zero pad value

6.24.2.22 virtual int mitkEncodedGradientEstimator::GetZeroPadMinValue () [inline, virtual]

Get the min zero pad value

Returns:
Return the min zero pad value

6.24.2.23 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

6.24.2.24 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

6.24.2.25 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

6.24.2.26 virtual void mitkEncodedGradientEstimator::SetBoundsClipOff () [inline, virtual]

Turn off the bounding of the normal computation by the this->Bounds bounding box

6.24.2.27 virtual void mitkEncodedGradientEstimator::SetBoundsClipOn () [inline, virtual]

Turn on the bounding of the normal computation by the this->Bounds bounding box

6.24.2.28 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

6.24.2.29 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

6.24.2.30 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

6.24.2.31 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

6.24.2.32 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.

6.24.2.33 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.

6.24.2.34 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

6.24.2.35 void mitkEncodedGradientEstimator::SetGradientMagnitudeBias (float *fGMB*) [inline]

Set the scale and bias for the gradient magnitude

Parameters:
fGMB Represent the gradient magnitude

6.24.2.36 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

6.24.2.37 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

6.24.2.38 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

6.24.2.39 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

6.24.2.40 virtual void mitkEncodedGradientEstimator::SetZeroPadOff () [inline, virtual]

Assume that the data value outside the volume is zero when computing normals.

6.24.2.41 virtual void mitkEncodedGradientEstimator::SetZeroPadOn () [inline, virtual]

Assume that the data value outside the volume is zero when computing normals.

6.24.2.42 void mitkEncodedGradientEstimator::Update ()

Recompute the encoded normals and gradient magnitudes.

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

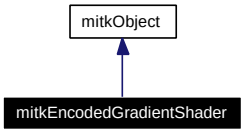
- mitkEncodedGradientEstimator.h

6.25 mitkEncodedGradientShader Class Reference

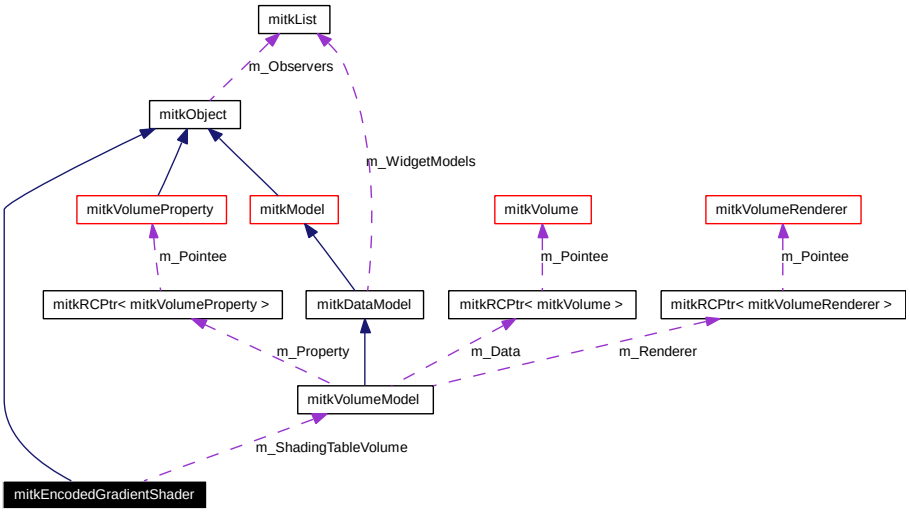
mitkEncodedGradientShader - Computes shading tables for encoded normals

`#include <mitkEncodedGradientShader.h>`

Inheritance diagram for mitkEncodedGradientShader:



Collaboration diagram for mitkEncodedGradientShader:



Public Member Functions

- virtual void **SetZeroNormalDiffuseIntensity** (float fZDiffuseIntensity)
- virtual float **GetZeroNormalDiffuseIntensityMinValue** ()
- virtual float **GetZeroNormalDiffuseIntensityMaxValue** ()
- virtual float **GetZeroNormalDiffuseIntensity** ()
- virtual void **SetZeroNormalSpecularIntensity** (float fZSpecularIntensity)
- virtual float **GetZeroNormalSpecularIntensityMinValue** ()
- virtual float **GetZeroNormalSpecularIntensityMaxValue** ()
- virtual float **GetZeroNormalSpecularIntensity** ()
- void **UpdateShadingTable** (mitkView *view, mitkVolumeModel *vol, mitkEncodedGradientEstimator *gradest)
- float * **GetRedDiffuseShadingTable** (mitkVolumeModel *vol)
- float * **GetGreenDiffuseShadingTable** (mitkVolumeModel *vol)
- float * **GetBlueDiffuseShadingTable** (mitkVolumeModel *vol)
- float * **GetRedSpecularShadingTable** (mitkVolumeModel *vol)
- float * **GetGreenSpecularShadingTable** (mitkVolumeModel *vol)
- float * **GetBlueSpecularShadingTable** (mitkVolumeModel *vol)

Protected Member Functions

- void **_buildShadingTable** (int index, float lightDirection[3], float lightColor[3], float lightIntensity, float viewDirection[3], float material[4], mitkEncodedGradientEstimator ∗gradest)

Protected Attributes

- float ∗ **m_ShadingTable** [MITK_MAX_SHADING_TABLES][6]
- mitkVolumeModel ∗ **m_ShadingTableVolume** [MITK_MAX_SHADING_TABLES]
- int **m_ShadingTableSize** [MITK_MAX_SHADING_TABLES]
- float **m_ZeroNormalDiffuseIntensity**
- float **m_ZeroNormalSpecularIntensity**

6.25.1 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:

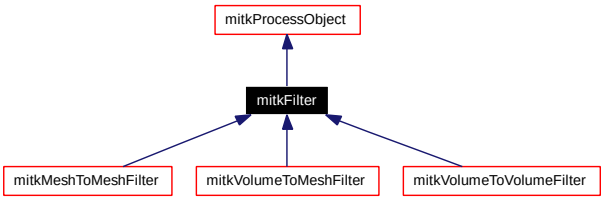
- mitkEncodedGradientShader.h

6.26 mitkFilter Class Reference

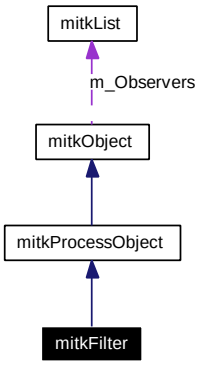
mitkFilter - abstract class specifies interface for filter object

`#include <mitkFilter.h>`

Inheritance diagram for mitkFilter:



Collaboration diagram for mitkFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

6.26.1 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.

6.26.2 Member Function Documentation

6.26.2.1 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, mitkBinMarchingCubes, mitkDiffusionFilter, mitkInterpolateFilter, mitkLiveWireImageFilter, mitkMarchingCubes, mitkMeshToMeshFilter, mitkNearestNeighborInterpolateFilter, mitkQEMSimplification, mitkRegionGrowImageFilter, mitkRegistrationFilter, mitkRGBToGrayFilter, mitkSeedFillFilter, mitkSobelEdgeDetectFilter, mitkThresholdSegmentationFilter, mitkTriangleMeshSimplification, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeResizeFilter, mitkVolumeResliceFilter, mitkVolumeToMeshFilter, and mitkVolumeToVolumeFilter.

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

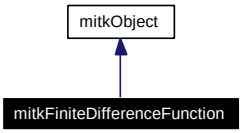
- mitkFilter.h

6.27 mitkFiniteDifferenceFunction Class Reference

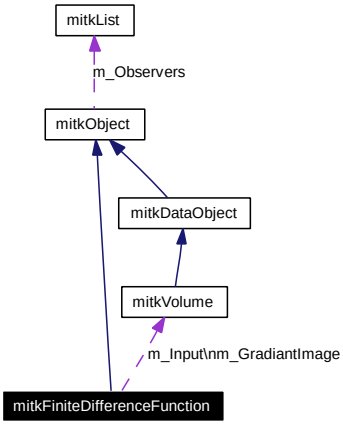
mitkFiniteDifferenceFunction - a component object of the finite difference solver hierarchy

`#include <mitkFiniteDifferenceFunction.h>`

Inheritance diagram for mitkFiniteDifferenceFunction:



Collaboration diagram for mitkFiniteDifferenceFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInput (mitkVolume *volume)
- virtual void Initialize ()
- virtual void Update (mitkVolume *DistanceImage, PointVector *NarrowBand)=0

Protected Attributes

- mitkVolume * **m_GradiantImage**
- mitkVolume * **m_Input**

6.27.1 Detailed Description

mitkFiniteDifferenceFunction - a component object of the finite difference solver hierarchy

mitkFiniteDifferenceFunction This class is a component object of the finite difference solver hierarchy (see mitkFiniteDifferenceImageFilter). It defines a generic interface for a function object that computes a single scalar value from a neighborhood of values.

6.27.2 Member Function Documentation

6.27.2.1 virtual void mitkFiniteDifferenceFunction::Initialize () [inline, virtual]

Do some initialize work

6.27.2.2 virtual void mitkFiniteDifferenceFunction::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.

6.27.2.3 void mitkFiniteDifferenceFunction::SetInput (mitkVolume * *volume*) [inline]

Set the input of the function

Parameters:
volume represent the input

6.27.2.4 virtual void mitkFiniteDifferenceFunction::Update (mitkVolume * *DistanceImage*, PointVector * *NarrowBand*) [pure virtual]

Update the narrowband

Parameters:
DistanceImage represent the input distance image

NarrowBand represent the narrow band

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

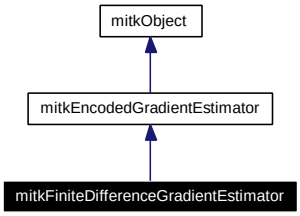
- mitkFiniteDifferenceFunction.h

6.28 mitkFiniteDifferenceGradientEstimator Class Reference

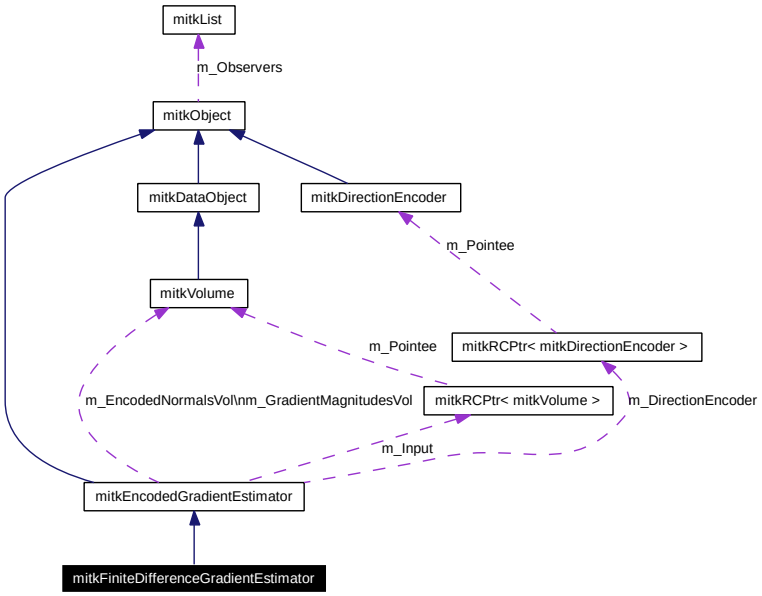
mitkFiniteDifferenceGradientEstimator - a concrete class to use finite differences to estimate gradient

`#include <mitkFiniteDifferenceGradientEstimator.h>`

Inheritance diagram for mitkFiniteDifferenceGradientEstimator:



Collaboration diagram for mitkFiniteDifferenceGradientEstimator:



Public Member Functions

- void SetSampleSpacingInVoxels (int nVal)
- int GetSampleSpacingInVoxels ()

Public Attributes

- int **m_SampleSpacingInVoxels**

Protected Member Functions

- void **_updateNormals** (void)

6.28.1 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.

6.28.2 Member Function Documentation

6.28.2.1 int mitkFiniteDifferenceGradientEstimator::GetSampleSpacingInVoxels () [inline]

Returns:
Return the spacing between samples for the finite differences method used to compute the normal

6.28.2.2 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:

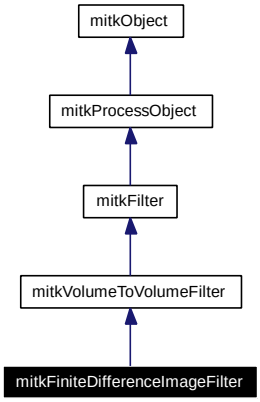
- mitkFiniteDifferenceGradientEstimator.h

6.29 mitkFiniteDifferenceImageFilter Class Reference

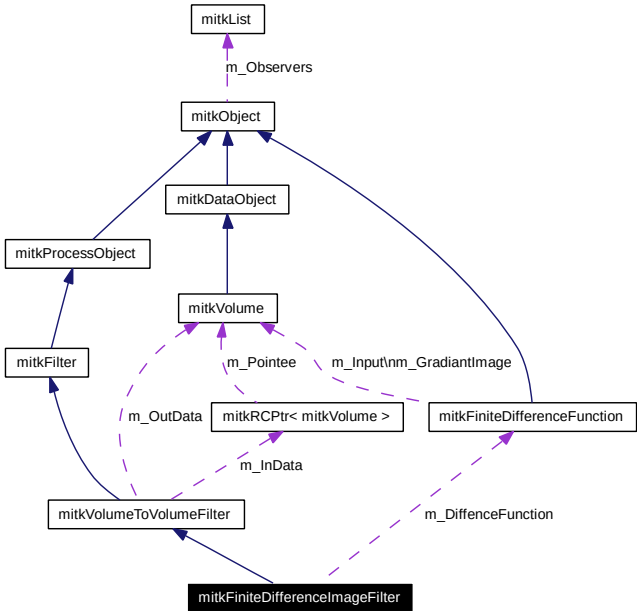
mitkFiniteDifferenceImageFilter - abstract for mitkLevelSetImageFilter

`#include <mitkFiniteDifferenceImageFilter.h>`

Inheritance diagram for mitkFiniteDifferenceImageFilter:



Collaboration diagram for mitkFiniteDifferenceImageFilter:



Public Member Functions

- void **debuguse** ()

Protected Member Functions

- mitkFiniteDifferenceImageFilter ()

- ~mitkFiniteDifferenceImageFilter ()
- virtual void ApplyUpdate ()=0
- virtual void CalculateChange ()=0
- virtual bool Halt ()=0
- virtual void Initialize ()=0
- int GetElapsedIterations ()
- virtual void PostProcess ()=0
- void SetDifferenceFunction (mitkFiniteDifferenceFunction *difFunc)
- mitkFiniteDifferenceFunction * GetDifferenceFunction ()
- virtual bool **Execute** ()

Protected Attributes

- int **m_ElapsedIterations**
- mitkFiniteDifferenceFunction * **m_DiffenceFunction**

6.29.1 Detailed Description

mitkFiniteDifferenceImageFilter - abstract for mitkLevelSetImageFilter

mitkFiniteDifferenceImageFilter is an abstract class for mitkLevelSetImageFilter.

The Finite Difference Solver Hierarchy

This is an overview of the finite difference solver (FDS) framework. The FDS framework is a set of classes for creating filters to solve partial differential equations on images using an iterative, finite difference update scheme.

The high-level algorithm implemented by the framework can be described by the following pseudocode.

```
WHILE NOT convergence:
   FOR ALL pixels i
      time_step = calculate_change(i)
      update(i, time_step)
```

The following equation describes update $n+1$ at pixel i on discrete image u :

$$u_{\mathbf{i}}^{n+1} = u_{\mathbf{i}}^{n} + \Delta u_{\mathbf{i}}^{n} \Delta t$$

Component objects

The FDS hierarchy is comprised of two component object types, variations of which are designed to be plugged together to create filters for different applications. At the process level are the "solver" objects, which are subclasses of mitkFiniteDifferenceImageFilter. Solver objects are filters that take image inputs and produce image outputs. Solver objects require a "finite difference function" object to perform the calculation at each image pixel during iteration. These specialized function objects are subclasses of mitkFiniteDifferenceFunction. FiniteDifferenceFunctions take a neighborhood of pixels as input and produce a scalar valued result.

Filters for different applications are created by defining a function object to handle the numerical calculations and choosing (or creating) a solver object that reflects the requirements and constraints of the application. For example, anisotropic diffusion filters are created by plugging anisotropic diffusion functions into the DenseFiniteDifferenceImageFilter. The separation between function object and solver object allows us to create, for example, sparse-field, dense-field, and narrow-band implementations of a level-set surface evolution filter can all be constructed by plugging the same function object into three different, specialized solvers.

Creating new filters in this hierarchy

The procedure for creating a filter within the FDS hierarchy is to identify all the virtual methods that need to be defined for your particular application. In the simplest case, a filter needs only to instantiate a specific function object and define some halting criteria. For more complicated applications, you may need to define a specialized type of iteration scheme or updating procedure in a higher-level solver object.

Some simple examples are the specific subclasses of AnisotropicDiffusionImageFilter. The leaves of the anisotropic diffusion filter tree only define the function object they use for their particular flavor of diffusion. See CurvatureAnisotropicDiffusionImageFilter and GradientAnisotropicDiffusionImageFilter for details.

mitkFiniteDifferenceImageFilter

This class defines the generic solver API at the top level of the FDS framework. FiniteDifferenceImageFilter is an abstract class that implements the generic, high-level algorithm (described above).

Inputs and Outputs

This filter is an Image to Image filter. Depending on the specific subclass implementation, finite difference image filters may process a variety of image types. The input to the filter is the initial value of u and the output of the filter is the solution to the p.d.e.

How to use this class

Execute() relies on several virtual methods that must be defined by a subclass. Specifically: *ApplyUpdate() CalculateChange()* and *Halt()*. To create a finite difference solver, implement a subclass to define these methods.

Note that there is no fixed container type for the buffer used to hold the update Δ . The container might be another image, or simply a list of values. The member variable m_DiffenceFunction (a pointer to mitkLevelSetFunction) is responsible for creating the Δ container. CalculateChange populates this buffer and ApplyUpdate adds the buffer values to the output image (solution). The boolean Halt() method returns a true value to stop iteration.

Some codes are borrowed from ITK, and please see the copyright at end.

Note:

Datatype of both the input and output should be double.

See also:

mitkLevelSetImageFilter
mitkFiniteDifferenceFunction

6.29.2 Constructor & Destructor Documentation

6.29.2.1 mitkFiniteDifferenceImageFilter::mitkFiniteDifferenceImageFilter () [inline, protected]

Constructor of the class

6.29.2.2 mitkFiniteDifferenceImageFilter::~mitkFiniteDifferenceImageFilter () [inline, protected]

Destructor of the class

6.29.3 Member Function Documentation

6.29.3.1 virtual void mitkFiniteDifferenceImageFilter::ApplyUpdate () [protected, pure virtual]

This method is defined by a subclass to apply changes to the output

6.29.3.2 virtual void mitkFiniteDifferenceImageFilter::CalculateChange () [protected, pure virtual]

This method is defined by a subclass to populate an update buffer with changes for the pixels in the output.

6.29.3.3 mitkFiniteDifferenceFunction* mitkFiniteDifferenceImageFilter::GetDifferenceFunction () [inline, protected]

Get the difference function of the class

Returns:
Return the difference function.

6.29.3.4 int mitkFiniteDifferenceImageFilter::GetElapsedIterations () [inline, protected]

Get the number of elapsed iteration.

Returns:
numbers of the elapsed iterations

6.29.3.5 virtual bool mitkFiniteDifferenceImageFilter::Halt () [protected, pure virtual]

This method returns true when the current iterative solution of the equation has met the criteria to stop solving. Defined by a subclass.

Returns:
return true if the iterations should be stopped, otherwize false.

6.29.3.6 virtual void mitkFiniteDifferenceImageFilter::Initialize () [protected, pure virtual]

This method is optionally defined by a subclass and is called before the loop of iterations of calculate_-change & upate.

6.29.3.7 virtual void mitkFiniteDifferenceImageFilter::PostProcess () [protected, pure virtual]

This method is optionally defined by a subclass to do some post process work.

6.29.3.8 void mitkFiniteDifferenceImageFilter::SetDifferenceFunction (mitkFiniteDifferenceFunction * *difFunc*) [inline, protected]

Set the difference function of the class

Parameters:

difFunc Represent the difference function.

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

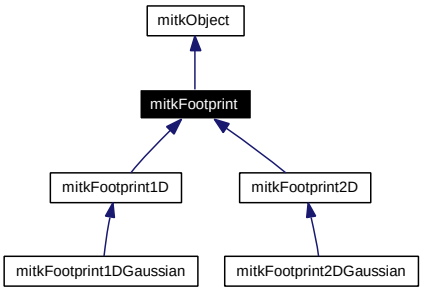
- mitkFiniteDifferenceImageFilter.h

6.30 mitkFootprint Class Reference

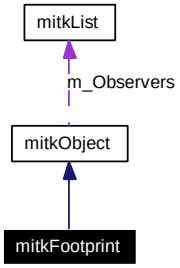
mitkFootprint - abstract class defines common interface for footprint

`#include <mitkFootprint.h>`

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)

Protected Attributes

- float * **m_FootprintTable**
- bool **m_NeedRebuild**

6.30.1 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.

6.30.2 Member Function Documentation

6.30.2.1 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.

6.30.2.2 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.

6.30.2.3 bool mitkFootprint::GetRebuild () [inline]

Get the status of building.

Returns:

Return true, the building is enabled. Return false, the building is disabled.

6.30.2.4 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.

6.30.2.5 void mitkFootprint::SetRebuild (bool *build*) [inline]

Set the status of building.

Parameters:

build build = true163172enable the rebuilding. build = false163172disable the rebuilding.

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

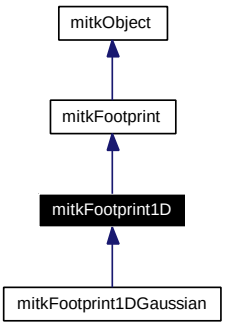
- mitkFootprint.h

6.31 mitkFootprint1D Class Reference

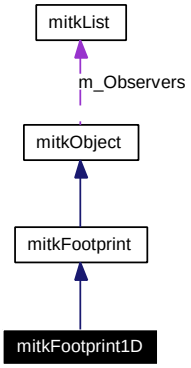
mitkFootprint1D - abstract class specifies interface for one dimensional footprint

`#include <mitkFootprint1D.h>`

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 ()

Protected Attributes

- float **m_RS**
- float **m_OldRS**
- int **m_N**
- int **m_OldN**

6.31.1 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.

6.31.2 Member Function Documentation

6.31.2.1 float mitkFootprint1D::GetRadiiSquare () [inline]

Get the footprint extent (radii square) in radial direction.

Returns:
Return the Radii square in radial direction.

6.31.2.2 int mitkFootprint1D::GetSampleNumber () [inline]

Get the sample number in radial direction.

Returns:
Return the sample number in radial direction.

6.31.2.3 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.

6.31.2.4 void mitkFootprint1D::SetRadiiSquare (float *rs*) [inline]

Set the footprint extent (radii square) in radial direction.

Parameters:
rs Radii square in radial direction.

6.31.2.5 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:

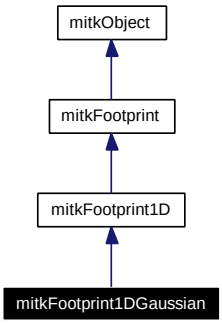
- mitkFootprint1D.h

6.32 mitkFootprint1DGaussian Class Reference

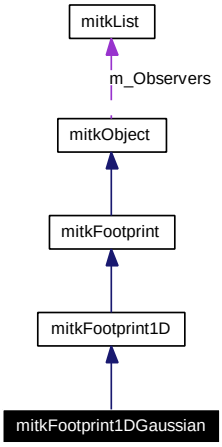
mitkFootprint1DGaussian - concrete class for one dimensional footprint with Gaussian kernel

`#include <mitkFootprint1DGaussian.h>`

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)

Protected Member Functions

- void **_buildTable** ()

Protected Attributes

- float **m_Coeff**
- float **m_AdjustRadii**
- float **m_VMatrix** [4]
- float **m_InvMatrix** [4]

6.32.1 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); }
```

6.32.2 Constructor & Destructor Documentation

6.32.2.1 mitkFootprint1DGaussian::mitkFootprint1DGaussian ()

Constructor of the class

6.32.3 Member Function Documentation

6.32.3.1 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.

6.32.3.2 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.

6.32.3.3 virtual float* mitkFootprint1DGaussian::EncodeFootprintTable () [virtual]

Encode the footprint table.

Returns:

Return the entry of the footprint table.

Reimplemented from mitkFootprint.

6.32.3.4 float mitkFootprint1DGaussian::GetAdjustRadii () [inline]

Get the adjust radii coefficient of Gaussian kernel.

Returns:

Return the adjust radii coefficient of Gaussian kernel.

6.32.3.5 float mitkFootprint1DGaussian::GetCoefficient () [inline]

Get the weight coefficient of Gaussian kernel.

Returns:

Return the weight coefficient of Gaussian kernel.

6.32.3.6 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).

6.32.3.7 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.

6.32.3.8 void mitkFootprint1DGaussian::SetAdjustRadii (float *adjustradii*) [inline]

Set the adjust radii coefficient of Gaussian kernel.

Parameters:
adjustradii The adjust radii coefficient of Gaussian kernel.

6.32.3.9 void mitkFootprint1DGaussian::SetCoefficient (float *coeff*) [inline]

Set the weight coefficient of Gaussian kernel.

Parameters:
coeff The weight coefficient of Gaussian kernel.

6.32.3.10 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:

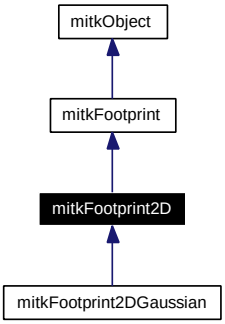
- mitkFootprint1DGaussian.h

6.33 mitkFootprint2D Class Reference

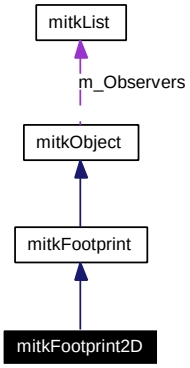
mitkFootprint2D - abstract class specifies interface for two dimensional footprint

`#include <mitkFootprint2D.h>`

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)

Protected Attributes

- float **m_RX**
- float **m_RY**
- float **m_OldRX**
- float **m_OldRY**
- int **m_NX**
- int **m_NY**

- int **m_OldNX**
- int **m_OldNY**
- int **m_NumX**
- int **m_NumY**

6.33.1 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.

6.33.2 Member Function Documentation

6.33.2.1 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.

6.33.2.2 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.

6.33.2.3 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.

6.33.2.4 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.

6.33.2.5 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:

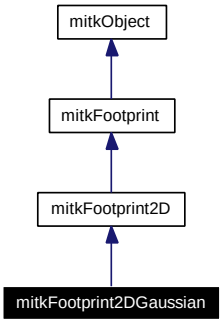
- mitkFootprint2D.h

6.34 mitkFootprint2DGaussian Class Reference

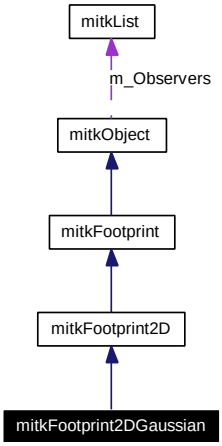
mitkFootprint2DGaussian - concrete class for two dimensional footprint with Gaussian kernel

`#include <mitkFootprint2DGaussian.h>`

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)

Protected Member Functions

- void **_buildTable** ()

Protected Attributes

- float **m_Coeff**
- float **m_AdjustRadii**
- float **m_VMatrix** [4]

6.34.1 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); }
```

6.34.2 Constructor & Destructor Documentation

6.34.2.1 mitkFootprint2DGaussian::mitkFootprint2DGaussian ()

Constructor of the class

6.34.3 Member Function Documentation

6.34.3.1 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.

6.34.3.2 virtual float* mitkFootprint2DGaussian::EncodeFootprintTable () [virtual]

Encode the footprint table.

Returns:
Return the entry of the footprint table.

Reimplemented from mitkFootprint.

6.34.3.3 float mitkFootprint2DGaussian::GetAdjustRadii () [inline]

Get the adjust radii coefficient of Gaussian kernel.

Returns:
Return the adjust radii coefficient of Gaussian kernel.

6.34.3.4 float mitkFootprint2DGaussian::GetCoefficient () [inline]

Get the weight coefficient of Gaussian kernel.

Returns:
Return the weight coefficient of Gaussian kernel.

6.34.3.5 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).

6.34.3.6 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.

6.34.3.7 void mitkFootprint2DGaussian::SetAdjustRadii (float *adjustradii*) [inline]

Set the adjust radii coefficient of Gaussian kernel.

Parameters:
adjustradii The adjust radii coefficient of Gaussian kernel.

6.34.3.8 void mitkFootprint2DGaussian::SetCoefficient (float *coeff*) [inline]

Set the weight coefficient of Gaussian kernel.

Parameters:

coeff The weight coefficient of Gaussian kernel.

6.34.3.9 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:

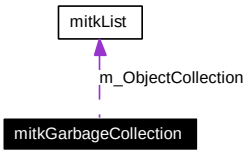
- mitkFootprint2DGaussian.h

6.35 mitkGarbageCollection Class Reference

mitkGarbageCollection - a simple implementation of garbage collection

`#include <mitkGarbageCollection.h>`

Collaboration diagram for mitkGarbageCollection:



Public Member Functions

- void **AddObject** (mitkObject *obj)
- void **RemoveObject** (mitkObject *obj)

6.35.1 Detailed Description

mitkGarbageCollection - a simple implementation of garbage collection

mitkGarbageCollection implement a simple garbage collection mechanism. And it can ensure all MITK objects are deleted when the application which uses MITK objects exits. Users needn't call its member function directly.

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

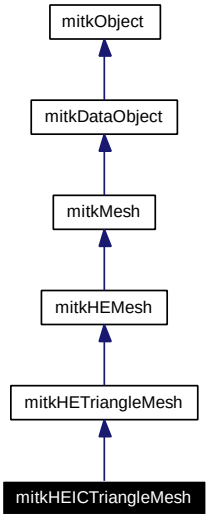
- mitkGarbageCollection.h

6.36 mitkHEICTriangleMesh Class Reference

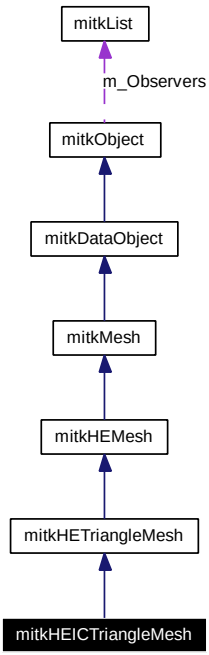
mitkHEICTriangleMesh - a concrete class for in-core triangle meshes represented by Half Edges

`#include <mitkHEICTriangleMesh.h>`

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 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 ()

Protected Member Functions

- virtual size_type **_getVertices** (index_type startIdx, size_type num, Vertex3f *verts)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, TriangleFace *faces)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, Vertex3f *verts)
- virtual VertexHandle **_addNewVertex** (float x, float y, float z, float nx=0.0f, float ny=0.0f, float nz=0.0f, HalfEdgeHandle edge=HalfEdgeHandle())
- virtual HalfEdgeHandle **_addNewEdge** (VertexHandle start, VertexHandle end)
- virtual FaceHandle **_addNewFace** (HalfEdgeHandle halfedge=HalfEdgeHandle())
- virtual void **_clearDeletedVertices** ()
- virtual void **_clearDeletedEdges** ()
- virtual void **_clearDeletedFaces** ()

Protected Attributes

- size_type **m_Increment**
- Vertex * **m_Vertices**
- Edge * **m_Edges**
- Face * **m_Faces**
- Vertex3f * **m_VertArray**
- TriangleFace * **m_FaceArray**

6.36.1 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

6.36.2 Constructor & Destructor Documentation

6.36.2.1 mitkHEICTriangleMesh::mitkHEICTriangleMesh ()

Default constructor.

6.36.3 Member Function Documentation

6.36.3.1 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();

6.36.3.2 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();

6.36.3.3 virtual void mitkHEICTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkHEMesh.

6.36.3.4 virtual unsigned 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.

6.36.3.5 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.

6.36.3.6 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.

6.36.3.7 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.

6.36.3.8 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.

6.36.3.9 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.

6.36.3.10 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.

6.36.3.11 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.

6.36.3.12 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.

6.36.3.13 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.

6.36.3.14 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.

6.36.3.15 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.

6.36.3.16 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.

6.36.3.17 virtual void mitkHEICTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkHETriangleMesh.

6.36.3.18 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.

6.36.3.19 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.

6.36.3.20 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.

6.36.3.21 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:

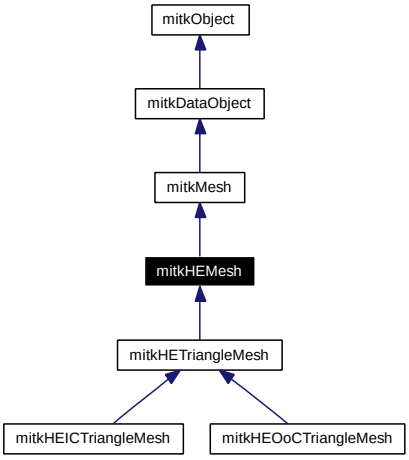
- mitkHEICTriangleMesh.h

6.37 mitkHEMesh Class Reference

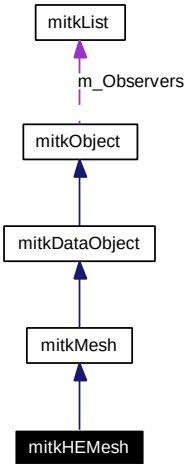
mitkHEMesh - a concrete class for polygon meshes represented by Half Edges

`#include <mitkHEMesh.h>`

Inheritance diagram for mitkHEMesh:



Collaboration diagram for mitkHEMesh:



Public Types

- typedef HEVertex **Vertex**
- typedef HEHalfEdge **HalfEdge**
- typedef HEEdge **Edge**
- typedef HEFace **Face**
- typedef HEVertexPtr **VertexPtr**
- typedef HEHalfEdgePtr **HalfEdgePtr**
- typedef HEEdgePtr **EdgePtr**
- typedef HEFacePtr **FacePtr**

- typedef HEVertexHandle **VertexHandle**
- typedef HEHalfEdgeHandle **HalfEdgeHandle**
- typedef HEEdgeHandle **EdgeHandle**
- typedef HEFaceHandle **FaceHandle**
- 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 ()

Protected Member Functions

- void **_init** ()
- virtual index_type **_addVertex** (Vertex3f &vert)
- virtual index_type **_addFace** (unsigned int num, index_type *verts)
- virtual Vertex3f const * **_getVertex** (index_type vertIdx)
- virtual index_type const * **_getFace** (index_type faceIdx)
- virtual Vertex3f * **_getVertexForWrite** (index_type vertIdx)
- virtual index_type * **_getFaceForWrite** (index_type faceIdx)
- virtual bool **_setFace** (index_type faceIdx, unsigned int num, index_type *verts)
- virtual VertexHandle **_addNewVertex** (float x, float y, float z, float nx=0.0f, float ny=0.0f, float nz=0.0f, HalfEdgeHandle edge=HalfEdgeHandle())=0
- virtual HalfEdgeHandle **_addNewEdge** (VertexHandle start, VertexHandle end)=0
- virtual FaceHandle **_addNewFace** (HalfEdgeHandle halfedge=HalfEdgeHandle())=0
- void **_adjustOutHalfEdge** (VertexHandle vert)
- void **_deleteVert** (VertexHandle v)
- void **_deleteEdge** (EdgeHandle e)
- void **_deleteFace** (FaceHandle f)
- virtual void **_clearDeletedVertices** ()=0
- virtual void **_clearDeletedEdges** ()=0
- virtual void **_clearDeletedFaces** ()=0
- VertexPtr **_getVertexPtr** (index_type idx) const
- EdgePtr **_getEdgePtr** (index_type idx) const
- HalfEdgePtr **_getHalfEdgePtr** (index_type idx) const
- FacePtr **_getFacePtr** (index_type idx) const
- Orientation **_testOrientation** (FaceHandle fh)
- void **_clearEdgeFlags** ()
- void **_clearVertFlags** ()
- void **_clearFaceFlags** ()

Protected Attributes

- size_type **m_VertNum**
- size_type **m_EdgeNum**
- size_type **m_FaceNum**
- size_type **m_ValidVertNum**
- size_type **m_ValidEdgeNum**
- size_type **m_ValidFaceNum**
- size_type **m_MaxVertNum**
- size_type **m_MaxEdgeNum**
- size_type **m_MaxFaceNum**
- index_type * **m_TempFace**
- bool **m_NeedClearVert**
- bool **m_NeedClearEdge**
- bool **m_NeedClearFace**

Friends

- class **VertexVertexIterT< mitkHEMesh >**
- class **VertexOHalfedgeIterT< mitkHEMesh >**
- class **VertexIHalfedgeIterT< mitkHEMesh >**
- class **VertexEdgeIterT< mitkHEMesh >**
- class **FaceVertexIterT< mitkHEMesh >**
- class **FaceHalfedgeIterT< mitkHEMesh >**
- class **FaceEdgeIterT< mitkHEMesh >**
- class **FaceFaceIterT< mitkHEMesh >**
- class **VertexIterT< mitkHEMesh >**
- class **EdgeIterT< mitkHEMesh >**
- class **FaceIterT< mitkHEMesh >**

6.37.1 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.

6.37.2 Member Typedef Documentation

6.37.2.1 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();
}
```

6.37.2.2 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();
}
```

6.37.2.3 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();
}
```

6.37.2.4 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();
}
```

6.37.2.5 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();
}
```

6.37.2.6 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();
}
```

6.37.2.7 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();
}
```

6.37.2.8 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();
}
```

6.37.2.9 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();
}
```

6.37.2.10 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();
}
```

6.37.2.11 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();
}
```

6.37.2.12 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();
}
```

6.37.3 Constructor & Destructor Documentation

6.37.3.1 mitkHEMesh::mitkHEMesh ()

Default constructor.

6.37.4 Member Function Documentation

6.37.4.1 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).

6.37.4.2 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.

6.37.4.3 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.

6.37.4.4 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.

6.37.4.5 VertexHandle mitkHEMesh::AddVertex (Vertex const & *vert*) `[inline]`

Add vertex.

Parameters:
vert the vertex to add

Returns:
Return the handle of the vertex added.

6.37.4.6 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.

6.37.4.7 void mitkHEMesh::ClearFlags ()

Clear all the user-set flags except MITK_FLAG_DELETED.

6.37.4.8 void mitkHEMesh::ClearGarbage ()

Clear the garbage generated by deletion.

Note:
Call this function before any kinds of iteration.

6.37.4.9 virtual void mitkHEMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

6.37.4.10 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

6.37.4.11 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

6.37.4.12 void mitkHEMesh::DeleteVertex (VertexHandle *vert*)

Delete vertex.

Parameters:
vert the handle of the vertex to be deleted

6.37.4.13 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.

6.37.4.14 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.

6.37.4.15 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.

6.37.4.16 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.

6.37.4.17 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.

6.37.4.18 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.

6.37.4.19 size_type mitkHEMesh::GetEdgeNumber () const [inline]

Get edge number.

Returns:
Return the number of edges.

6.37.4.20 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.

6.37.4.21 virtual size_type mitkHEMesh::GetFaceNumber () const [inline, virtual]

Get face number.

Returns:
Return the number of faces.

Implements mitkMesh.

6.37.4.22 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.

6.37.4.23 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.

6.37.4.24 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.

6.37.4.25 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.

6.37.4.26 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.

6.37.4.27 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.

6.37.4.28 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.

6.37.4.29 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.

6.37.4.30 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.

6.37.4.31 virtual size_type mitkHEMesh::GetVertexNumber () const [inline, virtual]

Get vertex number.

Returns:
Return the number of vertices.

Implements mitkMesh.

6.37.4.32 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.

6.37.4.33 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.

6.37.4.34 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.

6.37.4.35 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.

6.37.4.36 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.

6.37.4.37 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.

6.37.4.38 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.

6.37.4.39 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.

6.37.4.40 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.

6.37.4.41 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.

6.37.4.42 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.

6.37.4.43 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.

6.37.4.44 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.

6.37.4.45 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.

6.37.4.46 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.

6.37.4.47 virtual void mitkHEMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

6.37.4.48 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:

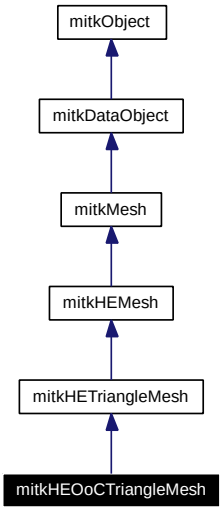
- mitkHEMesh.h

6.38 mitkHEOoCTriangleMesh Class Reference

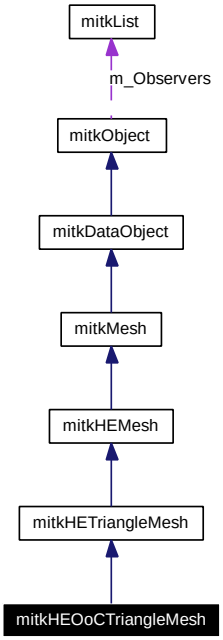
mitkHEOoCTriangleMesh - a concrete class for out-of-core triangle meshes represented by Half Edges

`#include <mitkHEOoCTriangleMesh.h>`

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 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 ()

Protected Member Functions

- virtual size_type **_getVertices** (index_type startIdx, size_type num, Vertex3f *verts)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, TriangleFace *faces)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, Vertex3f *verts)
- virtual VertexHandle **_addNewVertex** (float x, float y, float z, float nx=0.0f, float ny=0.0f, float nz=0.0f, HalfEdgeHandle edge=HalfEdgeHandle())
- virtual HalfEdgeHandle **_addNewEdge** (VertexHandle start, VertexHandle end)
- virtual FaceHandle **_addNewFace** (HalfEdgeHandle halfedge=HalfEdgeHandle())
- virtual void **_clearDeletedVertices** ()
- virtual void **_clearDeletedEdges** ()

- virtual void **_clearDeletedFaces** ()
- size_type **_getVertexBlock** (index_type blkIdx, Vertex3f *verts)
- size_type **_getTriangleFaceBlock** (index_type blkIdx, TriangleFace *faces)
- size_type **_getTriangleFaceBlock** (index_type blkIdx, Vertex3f *verts)

Protected Attributes

- HEVertexStorage * **m_VertStor**
- HEEdgeStorage * **m_EdgeStor**
- HEFaceStorage * **m_FaceStor**
- size_type **m_VertBlkSize**
- size_type **m_EdgeBlkSize**
- size_type **m_FaceBlkSize**
- index_type **m_VertIdxMask**
- unsigned int **m_VertIdxLen**
- index_type **m_VertBlkMask**
- index_type **m_EdgeIdxMask**
- unsigned int **m_EdgeIdxLen**
- index_type **m_EdgeBlkMask**
- index_type **m_FaceIdxMask**
- unsigned int **m_FaceIdxLen**
- index_type **m_FaceBlkMask**

6.38.1 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

6.38.2 Constructor & Destructor Documentation

6.38.2.1 mitkHEOoCTriangleMesh::mitkHEOoCTriangleMesh ()

Default constructor.

6.38.3 Member Function Documentation

6.38.3.1 virtual void mitkHEOoCTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkHEMesh.

6.38.3.2 virtual unsigned 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.

6.38.3.3 unsigned int mitkHEOoCTriangleMesh::GetBufferedEdgeBlockNum ()

Get the number of buffered edge blocks in memory.

Returns:
Return the number of buffered vertex blocks.

6.38.3.4 unsigned int mitkHEOoCTriangleMesh::GetBufferedFaceBlockNum ()

Get the number of buffered face blocks in memory.

Returns:
Return the number of buffered face blocks.

6.38.3.5 unsigned int mitkHEOoCTriangleMesh::GetBufferedVertexBlockNum ()

Get the number of buffered vertex blocks in memory.

Returns:
Return the number of buffered vertex blocks.

6.38.3.6 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.

6.38.3.7 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.

6.38.3.8 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.

6.38.3.9 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.

6.38.3.10 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.

6.38.3.11 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.

6.38.3.12 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.

6.38.3.13 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.

6.38.3.14 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.

6.38.3.15 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.

6.38.3.16 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.

6.38.3.17 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.

6.38.3.18 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.

6.38.3.19 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.

6.38.3.20 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.

6.38.3.21 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.

6.38.3.22 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.

6.38.3.23 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.

6.38.3.24 virtual void mitkHEOoCTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkHETriangleMesh.

6.38.3.25 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.

6.38.3.26 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.

6.38.3.27 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.

6.38.3.28 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.

6.38.3.29 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

6.38.3.30 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.

6.38.3.31 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

6.38.3.32 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.

6.38.3.33 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.

6.38.3.34 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

6.38.3.35 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

6.38.3.36 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.

6.38.3.37 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.

6.38.3.38 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:

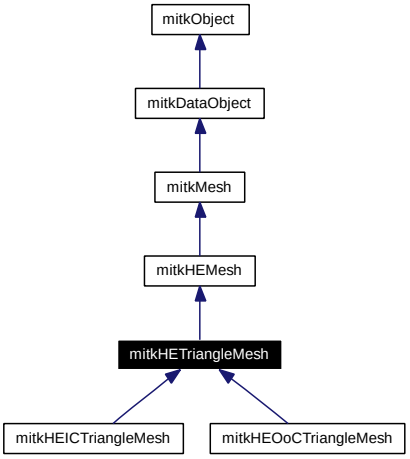
- mitkHEOoCTriangleMesh.h

6.39 mitkHETriangleMesh Class Reference

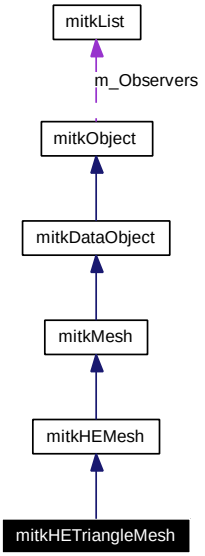
mitkHETriangleMesh - an abstract class for triangle meshes represented by Half Edges

`#include <mitkHETriangleMesh.h>`

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)

Protected Member Functions

- void **_removeEdge** (HalfEdgeHandle he)
- void **_removeLoop** (HalfEdgeHandle he)
- HalfEdgeHandle **_insertLoop** (HalfEdgeHandle he)
- HalfEdgeHandle **_insertEdge** (VertexHandle v, HalfEdgeHandle h0, HalfEdgeHandle h1)

6.39.1 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)

6.39.2 Constructor & Destructor Documentation

6.39.2.1 mitkHETriangleMesh::mitkHETriangleMesh ()

Default constructor.

6.39.3 Member Function Documentation

6.39.3.1 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).

6.39.3.2 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).

6.39.3.3 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.

6.39.3.4 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.

6.39.3.5 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.

6.39.3.6 void mitkHETriangleMesh::Flip (EdgeHandle *edge*)

Flip edge.

Parameters:
edge the handle of the edge to be flipped

6.39.3.7 virtual int mitkHETriangleMesh::GetDataObjectType () [inline, virtual]

Return what type of data object this is.

Returns:
Return the type of this data object.

6.39.3.8 virtual void mitkHETriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkHEMesh.

Reimplemented in mitkHEICTriangleMesh, and mitkHEOoCTriangleMesh.

6.39.3.9 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.

6.39.3.10 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.

6.39.3.11 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.

6.39.3.12 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.

6.39.3.13 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:

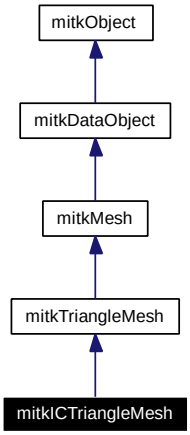
- mitkHETriangleMesh.h

6.40 mitkICTriangleMesh Class Reference

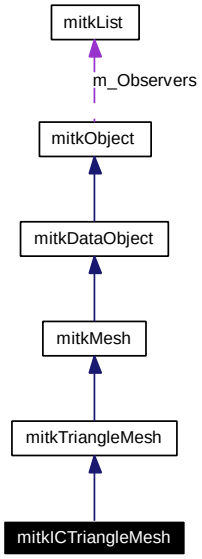
mitkICTriangleMesh - a concrete class for in-core triangle meshes

`#include <mitkICTriangleMesh.h>`

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 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 ()

Protected Member Functions

- virtual index_type **_addVertex** (Vertex3f &vert)
- virtual index_type **_addFace** (unsigned int num, index_type *verts)
- virtual Vertex3f const * **_getVertex** (index_type vertIdx)
- virtual index_type const * **_getFace** (index_type faceIdx)
- virtual Vertex3f * **_getVertexForWrite** (index_type vertIdx)
- virtual index_type * **_getFaceForWrite** (index_type faceIdx)
- Orientation **_testOrientation** (index_type faceIdx) const

Protected Attributes

- Vertex3f * **m_VertexData**
- TriangleFace * **m_FaceData**

6.40.1 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

6.40.2 Constructor & Destructor Documentation

6.40.2.1 mitkICTriangleMesh::mitkICTriangleMesh ()

Default constructor of this class.

6.40.3 Member Function Documentation

6.40.3.1 virtual void mitkICTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

6.40.3.2 virtual unsigned 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.

6.40.3.3 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.

6.40.3.4 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.

6.40.3.5 virtual float* mitkICTriangleMesh::GetVertexData () [inline, virtual]

Get data pointer of this vertex data.

Returns:
Return a float pointer to the vertex data.

Implements mitkMesh.

6.40.3.6 virtual void mitkICTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkTriangleMesh.

6.40.3.7 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.

6.40.3.8 virtual void mitkICTriangleMesh::ReverseNormals () [virtual]

Reverse normals.

Reimplemented from mitkMesh.

6.40.3.9 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.

6.40.3.10 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.

6.40.3.11 virtual void mitkICTriangleMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

6.40.3.12 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:

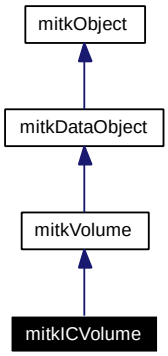
- mitkICTriangleMesh.h

6.41 mitkICVolume Class Reference

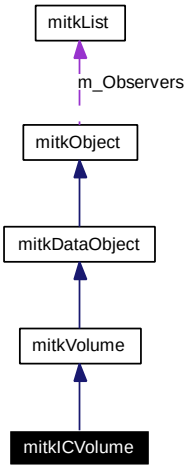
mitkICVolume - a concrete data object to represent an in-core multi-dimensional medical image dataset

`#include <mitkICVolume.h>`

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 GetActualMemorySize () const
- virtual void Initialize ()
- virtual void ShallowCopy (mitkDataObject *src)
- virtual void DeepCopy (mitkDataObject *src)

Protected Attributes

- void * **m_Data**

6.41.1 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.

6.41.2 Constructor & Destructor Documentation

6.41.2.1 mitkICVolume::mitkICVolume ()

Default constructor.

6.41.3 Member Function Documentation

6.41.3.1 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.

6.41.3.2 virtual void mitkICVolume::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Reimplemented from mitkVolume.

6.41.3.3 virtual unsigned 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.

6.41.3.4 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.

6.41.3.5 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.

6.41.3.6 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.

6.41.3.7 virtual int mitkICVolume::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_IN_CORE_VOLUME

Reimplemented from mitkVolume.

6.41.3.8 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.

6.41.3.9 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.

6.41.3.10 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.

6.41.3.11 virtual void mitkICVolume::Initialize () [virtual]

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

Reimplemented from mitkVolume.

6.41.3.12 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.

6.41.3.13 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.

6.41.3.14 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.

6.41.3.15 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.

6.41.3.16 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.

6.41.3.17 virtual void mitkICVolume::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:

Don't call this function directly.

Reimplemented from mitkVolume.

6.41.3.18 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.

6.41.3.19 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:

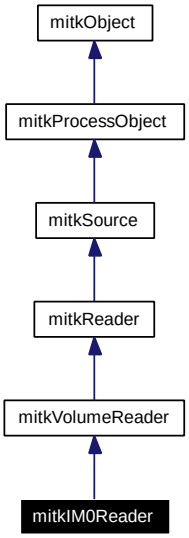
- mitkICVolume.h

6.42 mitkIM0Reader Class Reference

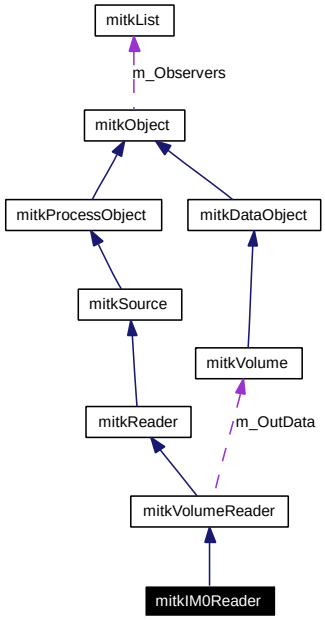
mitkIM0Reader - a concrete reader for reading IM0 volume file

`#include <mitkIM0Reader.h>`

Inheritance diagram for mitkIM0Reader:



Collaboration diagram for mitkIM0Reader:



Protected Member Functions

- virtual bool **Execute** ()

6.42.1 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:

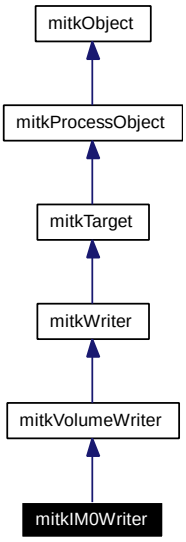
- mitkIM0Reader.h

6.43 mitkIM0Writer Class Reference

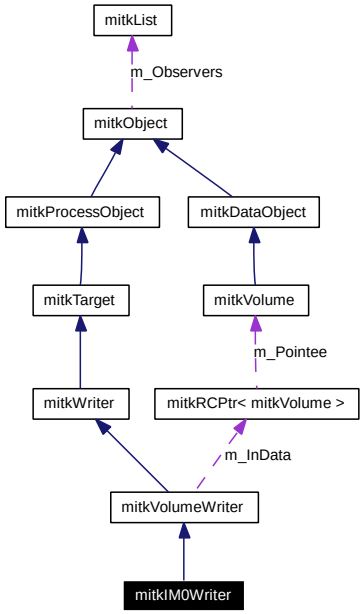
mitkIM0Writer - a concrete writer for writing a volume to IM0 volume file

`#include <mitkIM0Writer.h>`

Inheritance diagram for mitkIM0Writer:



Collaboration diagram for mitkIM0Writer:



Protected Member Functions

- virtual bool **Execute** ()

6.43.1 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:

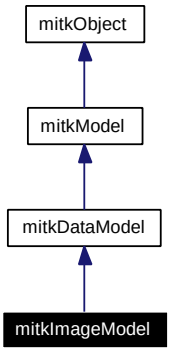
- mitkIM0Writer.h

6.44 mitkImageModel Class Reference

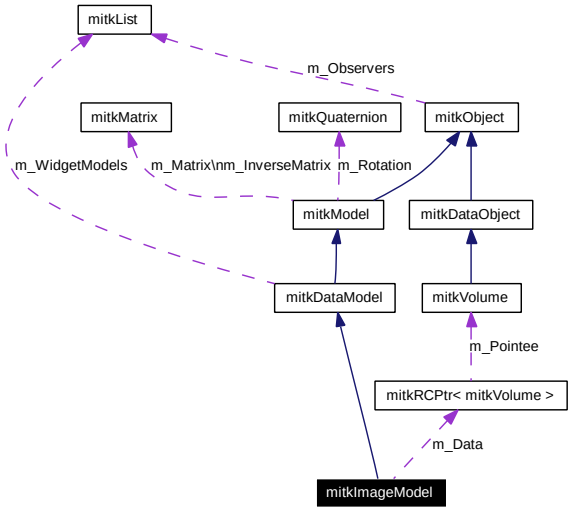
mitkImageModel - a concrete model class used to display a 2D image in mitkImageView

`#include <mitkImageModel.h>`

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 (mitkView *view)
- 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 &r-Value, 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)
- void **SetSliceChanged** ()

Protected Member Functions

- virtual float * **_getBounds** ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_Data**
- float **m_Opacity**
- ViewMode **m_ViewMode**
- unsigned int **m_TexId**
- int **m_TexImageWidth**
- int **m_TexImageHeight**
- float **m_RectLeft**
- float **m_RectRight**
- float **m_RectTop**
- float **m_RectBottom**
- int **m_CurrentSlice**
- int **m_TotalSlice**
- int **m_CurrentTex**
- float **m_Spacings** [3]
- int **m_Incs** [3]
- int **m_DummyTexWidth**
- int **m_DummyTexHeight**
- float **m_WindowCenter**
- float **m_WindowWidth**
- float **m_DefaultWindowCenter**
- float **m_DefaultWindowWidth**
- bool **m_NeedSetDummyTexture**
- bool **m_EnablePseudocolor**

6.44.1 Detailed Description

mitkImageModel - a concrete model class used to display a 2D image in mitkImageView

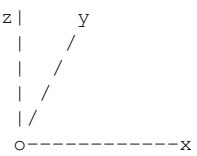
mitkImageModel is a concrete model class used to display a 2D image in mitkImageView. It can access a volume as input and display it in image view. To use it, the code snippet is:

```
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
aImageView->AddModel(aModel);
aImageView->Update();
aModel->NextSlice();  //Display the next slice
aImageView->Update();
```

6.44.2 Member Enumeration Documentation

6.44.2.1 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

6.44.3 Member Function Documentation

6.44.3.1 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

6.44.3.2 void mitkImageModel::CleanUp ()

Internal function, Don't call it directly.

6.44.3.3 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.

6.44.3.4 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

6.44.3.5 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

6.44.3.6 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

6.44.3.7 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

6.44.3.8 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.

6.44.3.9 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.

6.44.3.10 int mitkImageModel::GetCurrentSliceNumber () const [inline]

Get the current slice number displayed in ImageView

Returns:
Return the current slice number

6.44.3.11 mitkVolume* mitkImageModel::GetData ()

Get the volume data.

Returns:
Return pointer to the volume data.

6.44.3.12 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

6.44.3.13 int mitkImageModel::GetHeight () const [inline]

Internal function, Don't call it directly.

6.44.3.14 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

6.44.3.15 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

6.44.3.16 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

6.44.3.17 float mitkImageModel::GetOpacity () const [inline]

Get the opacity of this model.

Returns:

Return the opacity of this model.

6.44.3.18 float mitkImageModel::GetSpacingX () const [inline]

Internal function, Don't call it directly.

6.44.3.19 float mitkImageModel::GetSpacingY () [inline]

Internal function, Don't call it directly.

6.44.3.20 float mitkImageModel::GetSpacingZ () const [inline]

Internal function, Don't call it directly.

6.44.3.21 unsigned int mitkImageModel::GetTextureID () const [inline]

Get the texture ID.

Returns:

Return the texture ID.

6.44.3.22 int mitkImageModel::GetTotalSliceNumber () const [inline]

Display previous slice. If current slice is the first slice in the volume, then display the last slice.

6.44.3.23 ViewMode mitkImageModel::GetViewMode () const [inline]

Get the view mode of the volume.

See also:
ViewMode

Returns:
Return the view mode used by this ImageModel

6.44.3.24 int mitkImageModel::GetWidth () const [inline]

Internal function, Don't call it directly.

6.44.3.25 float mitkImageModel::GetWindowCenter () const [inline]

Get the window center of this image model

Returns:
Return the window center of this image model

6.44.3.26 float mitkImageModel::GetWindowWidth () const [inline]

Get the window width of this image model

Returns:
Return the window width of this image model

6.44.3.27 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.

6.44.3.28 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.

6.44.3.29 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

6.44.3.30 virtual bool mitkImageModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:

Return true if this model is opaque.

Implements mitkModel.

6.44.3.31 void mitkImageModel::NextSlice ()

Display next slice. If current slice is the last slice in the volume, then display the first slice.

6.44.3.32 void mitkImageModel::PrevSlice ()

Display previous slice. If current slice is the first slice in the volume, then display the last slice.

6.44.3.33 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.

6.44.3.34 virtual int mitkImageModel::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView 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.

6.44.3.35 void mitkImageModel::ResetWindowWidthCenter ()

Reset the window width and center to the default value.

6.44.3.36 void mitkImageModel::SetCurrentSliceNumber (int *sliceNo*)

Set the current slice number for displaying.

Parameters:
sliceNo Specify the current slice number

6.44.3.37 void mitkImageModel::SetData (mitkVolume * *data*)

Set the volume data.

Parameters:
data pointer to an mitkVolume

6.44.3.38 void mitkImageModel::SetOpacity (float *opacity*) [inline]

Set the opacity of this model.

Parameters:
opacity Specify the opacity of this model.

6.44.3.39 void mitkImageModel::SetViewMode (ViewMode *viewMode*)

Set the view mode of the volume.

See also:
ViewMode

Parameters:
viewMode Specify the view mode

6.44.3.40 void mitkImageModel::SetWindowCenter (float *winCenter*)

Set the window center of this image model

Parameters:
winCenter Specify the new window center of this image model

6.44.3.41 void mitkImageModel::SetWindowWidth (float *winWidth*)

Set the window width of this image model

Parameters:
winWidth Specify the new window width of this image model

6.44.3.42 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:

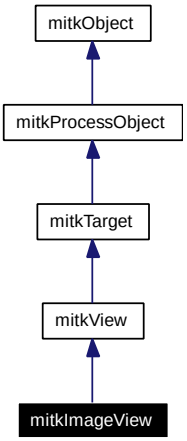
- mitkImageModel.h

6.45 mitkImageView Class Reference

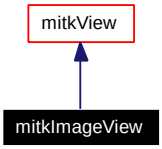
mitkImageView - a view to display 2D images

`#include <mitkImageView.h>`

Inheritance diagram for mitkImageView:



Collaboration diagram for mitkImageView:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void OnCreate ()
- virtual void OnSize (int newX, int newY)
- virtual void OnDraw ()
- 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 Scale (float delt)
- 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)

Protected Member Functions

- void **_drawCrossArrow** ()

Protected Attributes

- bool **m_CrossArrowEnabled**
- int **m_CrossArrowWidth**
- int **m_CrossArrowHeight**
- float **m_CrossArrowPosition** [2]

6.45.1 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.

6.45.2 Member Function Documentation

6.45.2.1 void mitkImageView::AdjustWidthCenter (float *deltX*, float *deltY*)

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

6.45.2.2 virtual mitkManipulator* mitkImageView::CreateManipulator () [virtual]

Override the base class function. Create a proper manipulator for the mouse events processing. You needn't call this function directly.

Reimplemented from mitkView.

6.45.2.3 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.

6.45.2.4 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.

6.45.2.5 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.

6.45.2.6 int mitkImageView::GetCrossArrowHeight () [inline]

Get the height of the rectangle associated with the cross arrow.

Returns:

Return the height of the rectangle.

6.45.2.7 float mitkImageView::GetCrossArrowPositionX () [inline]

Get the x coordinate of the cross arrow center.

Returns:
Return the x coordinate of the cross arrow center.

6.45.2.8 float mitkImageView::GetCrossArrowPositionY () [inline]

Get the y coordinate of the cross arrow center.

Returns:
Return the y coordinate of the cross arrow center.

6.45.2.9 int mitkImageView::GetCrossArrowWidth () [inline]

Get the width of the rectangle associated with the cross arrow.

Returns:
Return the width of the rectangle.

6.45.2.10 void mitkImageView::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 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

6.45.2.11 void mitkImageView::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 view

y return the y coordinate in this view

6.45.2.12 void mitkImageView::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 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.

6.45.2.13 float mitkImageView::GetWindowCenter ()

Get the window center of this view

Returns:

Return the window center of this view

6.45.2.14 float mitkImageView::GetWindowWidth ()

Get the window width of this view

Returns:

Return the window width of this view

6.45.2.15 virtual void mitkImageView::OnCreate () [virtual]

OS dependent function, don't call it.

Reimplemented from mitkView.

6.45.2.16 virtual void mitkImageView::OnDraw () [virtual]

OS dependent function, don't call it.

Reimplemented from mitkView.

6.45.2.17 virtual void mitkImageView::OnSize (int *newX*, int *newY*) [virtual]

OS dependent function, don't call it.

Reimplemented from mitkView.

6.45.2.18 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.

6.45.2.19 void mitkImageView::ResetCrossArrowPosition ()

Reset the center position and the rectangle size to the default value.

6.45.2.20 virtual void mitkImageView::ResetScene () [virtual]

Reset the geometric position of all of the models in this view.

Reimplemented from mitkView.

6.45.2.21 void mitkImageView::ResetWindowWidthCenter ()

Reset the window width and center to the default value.

6.45.2.22 virtual void mitkImageView::Rotate (const mitkQuaternion & *q*) [inline, virtual]

Do nothing. Donot call this function for rotation of a image.

Reimplemented from mitkView.

6.45.2.23 virtual void mitkImageView::Rotate (float *ax*, float *ay*, float *az*, float *angle*) [inline, virtual]

Do nothing. Donot call this function for rotation of a image.

Reimplemented from mitkView.

6.45.2.24 virtual void mitkImageView::Rotate (float *deltX*, float *deltY*, float *deltZ*) [virtual]

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. deltX and deltY will be modified to make sure that they are multiple 180 degrees using the formula (float)((int)(delt / 180.0) $*$ 180.0).

Reimplemented from mitkView.

6.45.2.25 virtual void mitkImageView::Scale (float *delt*) [virtual]

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.

Reimplemented from mitkView.

6.45.2.26 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.

6.45.2.27 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.

6.45.2.28 void mitkImageView::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

6.45.2.29 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.

6.45.2.30 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.

6.45.2.31 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.

6.45.2.32 virtual void mitkImageView::SetRotation (const mitkQuaternion & *q*) [inline, virtual]

Do nothing. Donot call this function for rotation of a image.

Reimplemented from mitkView.

6.45.2.33 virtual void mitkImageView::SetRotation (float *ax*, float *ay*, float *az*, float *angle*) [inline, virtual]

Do nothing. Donot call this function for rotation of a image.

Reimplemented from mitkView.

6.45.2.34 virtual void mitkImageView::SetRotation (float *x*, float *y*, float *z*) [virtual]

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.

Note:
x and y will be modified to make sure that they are multiple 180 degrees using the formula (float)((int)(x / 180.0) ∗ 180.0).

Reimplemented from mitkView.

6.45.2.35 void mitkImageView::SetWindowCenter (float *winCenter*)

Set the window center of this view

Parameters:

winCenter Specify the new window center of this view

6.45.2.36 void mitkImageView::SetWindowWidth (float *winWidth*)

Set the window width of this view

Parameters:

winWidth Specify the new window width of this view

6.45.2.37 virtual void mitkImageView::Translate (float *deltX*, float *deltY*, float *deltZ* = 0) [virtual]

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

Reimplemented from mitkView.

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

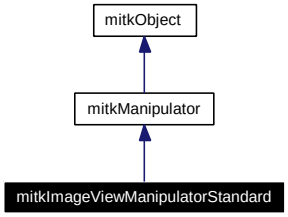
- mitkImageView.h

6.46 mitkImageViewManipulatorStandard Class Reference

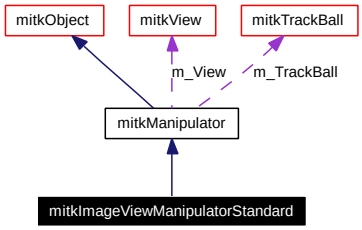
mitkImageViewManipulatorStandard - a concrete manipulator to process mouse events in an image view

`#include <mitkImageViewManipulatorStandard.h>`

Inheritance diagram for mitkImageViewManipulatorStandard:



Collaboration diagram for mitkImageViewManipulatorStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- **mitkImageViewManipulatorStandard** (mitkView *view)

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)
- virtual void **_onMouseWheel** (bool ctrlDown, bool shiftDown, int xPos, int yPos, int delta)

6.46.1 Detailed Description

mitkImageViewManipulatorStandard - a concrete manipulator to process mouse events in an image view

mitkImageViewManipulatorStandard is a concrete manipulator to process mouse events in an image view. It is the default manipulator of a image view. 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

6.46.2 Member Function Documentation

6.46.2.1 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.

6.46.2.2 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.

6.46.2.3 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.

6.46.2.4 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:

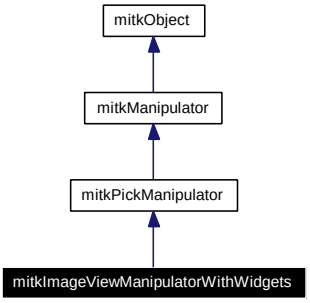
- mitkImageViewManipulatorStandard.h

6.47 mitkImageViewManipulatorWithWidgets Class Reference

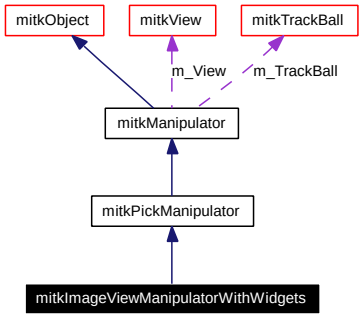
mitkImageViewManipulatorWithWidgets - manipulator of an image view contains widgets

`#include <mitkImageViewManipulatorWithWidgets.h>`

Inheritance diagram for mitkImageViewManipulatorWithWidgets:



Collaboration diagram for mitkImageViewManipulatorWithWidgets:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- **mitkImageViewManipulatorWithWidgets** (mitkView *view)

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)

6.47.1 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.

6.47.2 Member Function Documentation

6.47.2.1 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.

6.47.2.2 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.

6.47.2.3 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.

6.47.2.4 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:

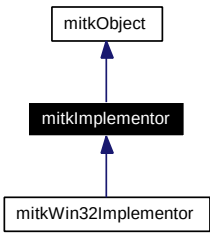
- mitkImageViewManipulatorWithWidgets.h

6.48 mitkImplementor Class Reference

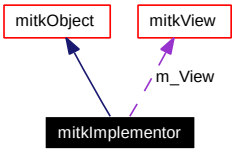
mitkImplementor - an abstract class to define an interface to implement some OS dependent operations

`#include <mitkImplementor.h>`

Inheritance diagram for mitkImplementor:



Collaboration diagram for mitkImplementor:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void * **GetWindowId** (void)=0
- virtual void * **GetParent** (void)=0
- virtual void * **GetApplicationId** (void)=0
- virtual void * **GetDeviceContext** (void)=0
- virtual void * **GetRenderContext** (void)=0
- virtual void **SetParent** (void *parentId)=0
- virtual void **SetPosition** (int leftPos, int topPos)=0
- virtual void **SetSize** (int widthSize, int heightSize)=0
- virtual void **SetWindowName** (const char *winName)=0
- virtual void **Show** ()=0
- virtual void **Hide** ()=0
- virtual void **Update** ()=0
- virtual void **MakeCurrent** ()=0
- virtual void **SwapBuffers** ()=0
- virtual bool **HasUnprocessedMouseMessage** ()=0
- double **GetFrameSpeed** ()
- double **GetRenderTime** ()
- void **EnableCalculateFrameSpeed** ()
- void **DisableCalculateFrameSpeed** ()

Protected Member Functions

- **mitkImplementor** (mitkView *view)

Protected Attributes

- mitkView * **m_View**
- double **m_RenderTime**
- bool **m_CalcFrameSpeed**

6.48.1 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.

6.48.2 Member Function Documentation

6.48.2.1 virtual void mitkImplementor::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 mitkWin32Implementor.

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

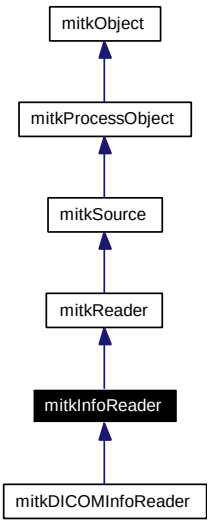
- mitkImplementor.h

6.49 mitkInfoReader Class Reference

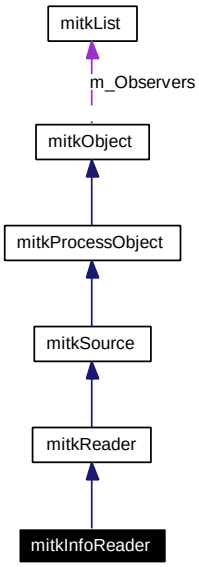
mitkInfoReader - an abstract class represents an information reader

`#include <mitkInfoReader.h>`

Inheritance diagram for mitkInfoReader:



Collaboration diagram for mitkInfoReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

6.49.1 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).

6.49.2 Member Function Documentation

6.49.2.1 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:

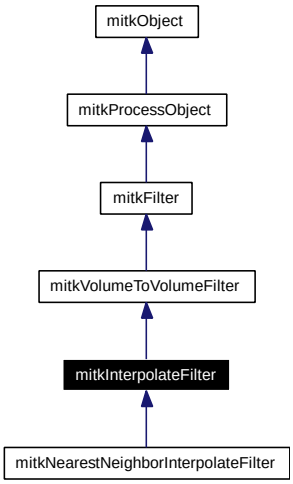
- mitkInfoReader.h

6.50 mitkInterpolateFilter Class Reference

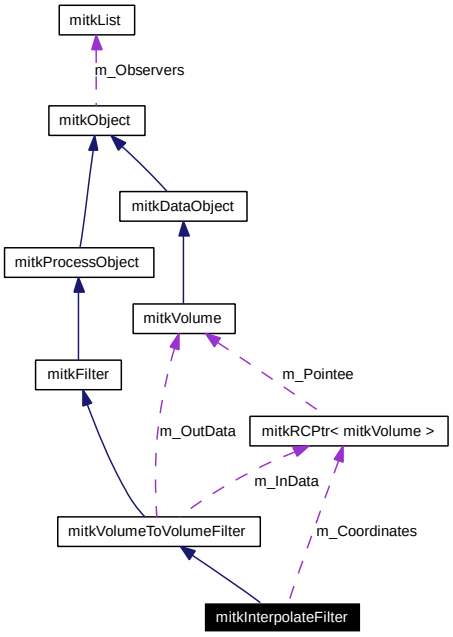
mitkInterpolateFilter - an abstract class specifies interface for interpolating values when objects are resampled through the Transform

`#include <mitkInterpolateFilter.h>`

Inheritance diagram for mitkInterpolateFilter:



Collaboration diagram for mitkInterpolateFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- void SetCorrdinates (mitkVolume *coordinates)
- mitkVolume * GetCoordinates ()
- void SetRegion (int r[6])
- void GetRegion (int r[6])
- void SetCentreTransformFlag (bool flag)
- void Update ()

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_Coordinates**
- int **m_Region** [6]
- int **m_Dimensions** [3]
- float **m_Spacings** [3]
- bool **m_IsCentreTransform**

6.50.1 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.

Note:
datatype of the output is float.

6.50.2 Member Function Documentation

6.50.2.1 mitkVolume* mitkInterpolateFilter::GetCoordinates ()

Get coordinates for interpolating pixels.

Returns:
Return the pointer to the coodinates.

6.50.2.2 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.

6.50.2.3 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 mitkNearestNeighborInterpolateFilter.

6.50.2.4 void mitkInterpolateFilter::SetCentreTransformFlag (bool *flag*) [inline]

Set the flag of centre transform.

Parameters:
flag The flag of centre transform.

6.50.2.5 void mitkInterpolateFilter::SetCorrdinates (mitkVolume * *coordinates*) [inline]

Set coordinates for interpolating pixels.

Parameters:
coordinates The pointer to the coordinates for interpolating pixels.

6.50.2.6 void mitkInterpolateFilter::SetRegion (int *r*[6]) [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.

6.50.2.7 void mitkInterpolateFilter::Update ()

Initialize the Interpolator . Perform before Run() function.

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

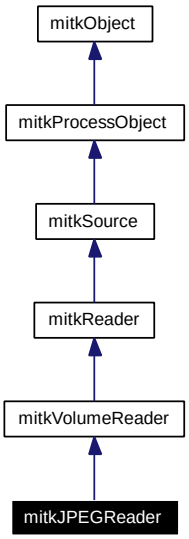
- mitkInterpolateFilter.h

6.51 mitkJPEGReader Class Reference

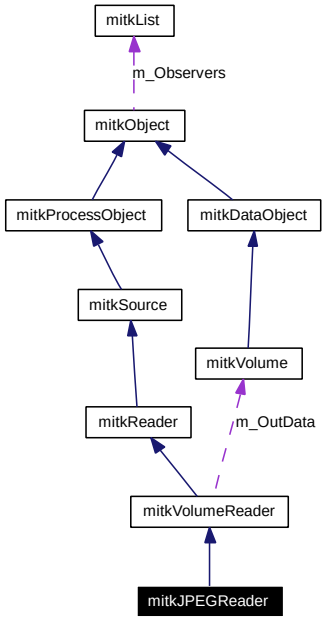
mitkJPEGReader - a concrete reader for reading JPEG image files

`#include <mitkJPEGReader.h>`

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])

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- float **m_Spacings** [3]

6.51.1 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.

6.51.2 Member Function Documentation

6.51.2.1 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.

6.51.2.2 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.

6.51.2.3 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.

6.51.2.4 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.

6.51.2.5 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:

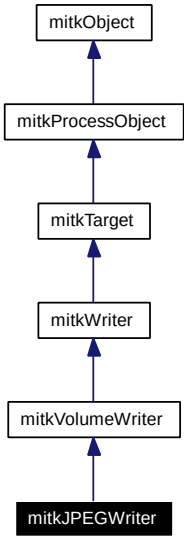
- mitkJPEGReader.h

6.52 mitkJPEGWriter Class Reference

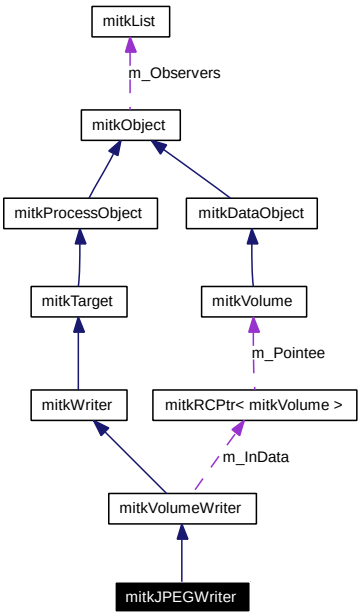
mitkJPEGWriter - a concrete writer for writing a volume to JPEG image files

`#include <mitkJPEGWriter.h>`

Inheritance diagram for mitkJPEGWriter:



Collaboration diagram for mitkJPEGWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- void SetQuality (int quality)

Protected Member Functions

- virtual bool **Execute** ()

6.52.1 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();
```

6.52.2 Member Function Documentation

6.52.2.1 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.

6.52.2.2 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:

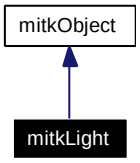
- mitkJPEGWriter.h

6.53 mitkLight Class Reference

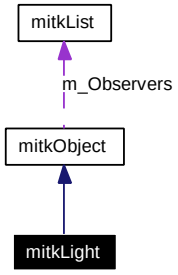
mitkLight - a light object in 3D view

`#include <mitkLight.h>`

Inheritance diagram for mitkLight:



Collaboration diagram for mitkLight:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void **SetColor** (float r, float g, float b)
- void **SetColor** (float color[3])
- void **GetColor** (float &r, float &g, float &b)
- void **GetColor** (float color[3])
- float * **GetColor** ()
- void **SetPosition** (float x, float y, float z)
- void **SetPosition** (float position[3])
- void **GetPosition** (float &x, float &y, float &z)
- void **GetPosition** (float position[3])
- float * **GetPosition** ()
- void **SetIntensity** (float intensity)
- float **GetIntensity** (void)
- void **SetSwitch** (bool isOn)
- void **TurnOn** ()
- void **TurnOff** ()
- bool **IsOn** ()
- void **SetPositional** (bool isPositional)
- void **SetPositionalOn** ()
- void **SetPositionalOff** ()
- bool **IsPositional** ()
- void **SetExponent** (float exponent)

- float **GetExponent** ()
- void **SetConeAngle** (float angle)
- float **GetConeAngle** ()
- void **SetAttenuationValues** (float val1, float val2, float val3)
- void **SetAttenuationValues** (float val[3])
- void **GetAttenuationValues** (float &val1, float &val2, float &val3)
- void **GetAttenuationValues** (float val[3])
- float * **GetAttenuationValues** ()
- void **SetFocalPoint** (float x, float y, float z)
- void **SetFocalPoint** (float focalPoint[3])
- void **GetFocalPoint** (float &x, float &y, float &z)
- void **GetFocalPoint** (float facalPoint[3])
- float * **GetFocalPoint** ()
- bool **IsModified** ()
- void **Render** (int lightIndex)

Protected Attributes

- float **m_Position** [3]
- float **m_Intensity**
- float **m_Color** [3]
- float **m_Exponent**
- float **m_ConeAngle**
- float **m_AttenuationValues** [3]
- float **m_FocalPoint** [3]
- bool **m_Switch**
- bool **m_Positional**
- bool **m_Modified**

6.53.1 Detailed Description

mitkLight - a light object in 3D view

mitkLight is a light object in 3D view. The only way you can access it is to get a pointer to it by using the member function GetLight() of mitkView. Then you can control it through its interface, and the change will affect the image rendered in the view.

6.53.2 Member Function Documentation

6.53.2.1 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:

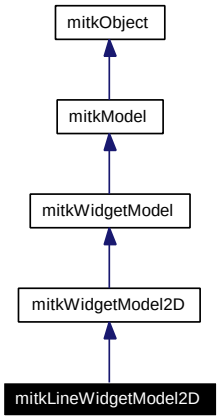
- mitkLight.h

6.54 mitkLineWidgetModel2D Class Reference

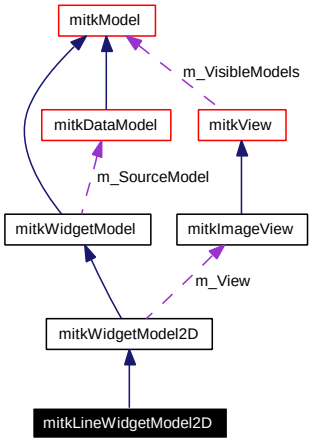
mitkLineWidgetModel2D - a line widget used in 2D view

`#include <mitkLineWidgetModel2D.h>`

Inheritance diagram for mitkLineWidgetModel2D:



Collaboration diagram for mitkLineWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkLineWidgetModel2D ()
- mitkLineWidgetModel2D (float startPoint[2])
- virtual int Render (mitkView *view)
- 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 Types

- enum { **unknown**, **arrow0**, **arrow1**, **line** }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()

Protected Attributes

- float **m_Point0** [2]
- float **m_Point1** [2]
- float **m_ArrowLength**
- float **m_ArrowColor** [4]
- float **m_LineColor** [4]
- float **m_PickedArrowColor** [4]
- float **m_PickedLineColor** [4]
- int **m_PointsSetNum**
- string **m_UnitName**

6.54.1 Detailed Description

mitkLineWidgetModel2D - a line widget used in 2D view

mitkLineWidgetModel2D is a line widget used in 2D view 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 view (e.g. mitkImageView), and in other conditions the display could be improper.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 mitkLineWidgetModel2D::mitkLineWidgetModel2D ()

The default constructor.

6.54.2.2 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.)

6.54.3 Member Function Documentation

6.54.3.1 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.

6.54.3.2 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.

6.54.3.3 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.

6.54.3.4 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

6.54.3.5 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

6.54.3.6 float mitkLineWidgetModel2D::GetLineLength ()

Get length of this line.

Returns:

Return the length of the line.

6.54.3.7 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.

6.54.3.8 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

6.54.3.9 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

6.54.3.10 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

6.54.3.11 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.

6.54.3.12 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.

6.54.3.13 virtual void mitkLineWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.54.3.14 virtual int mitkLineWidgetModel2D::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:

view the pointer of an mitkView in which this model will be shown

Returns:

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

Reimplemented from mitkModel.

6.54.3.15 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.54.3.16 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.

6.54.3.17 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.

6.54.3.18 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.54.3.19 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.

6.54.3.20 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.

6.54.3.21 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.54.3.22 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.

6.54.3.23 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.

6.54.3.24 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:

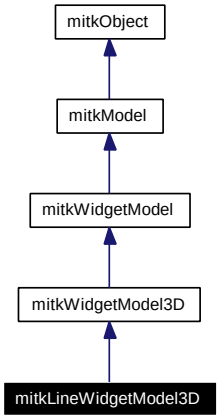
- mitkLineWidgetModel2D.h

6.55 mitkLineWidgetModel3D Class Reference

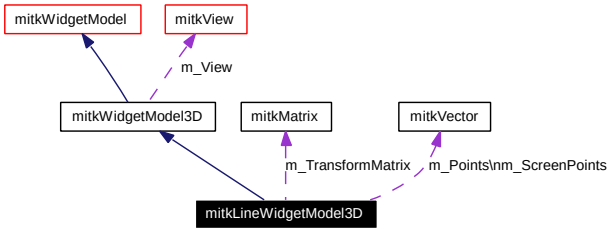
mitkLineWidgetModel3D - a line widget used in a 3D scene

`#include <mitkLineWidgetModel3D.h>`

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 (mitkView *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 Types

- enum { **unknown**, **arrow0**, **arrow1**, **line** }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()

Protected Attributes

- mitkVector * **m_Points**
- mitkVector * **m_ScreenPoints**
- mitkMatrix * **m_TransformMatrix**
- float **m_UnitsPerPixel** [3]
- float **m_ArrowLength**
- float **m_ArrowColor** [4]
- float **m_LineColor** [4]
- float **m_PickedArrowColor** [4]
- float **m_PickedLineColor** [4]
- string **m_UnitName**

6.55.1 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. mitkView), and in other conditions the display could be improper.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 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

6.55.3 Member Function Documentation

6.55.3.1 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.

6.55.3.2 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.

6.55.3.3 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.

6.55.3.4 float mitkLineWidgetModel3D::GetLineLength ()

Get length of this line.

Returns:

Return the length of the line.

6.55.3.5 const string& mitkLineWidgetModel3D::GetUnitName () [inline]

Get name string of unit.

Returns:
Return a constant reference to a string contains the name

6.55.3.6 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.

6.55.3.7 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.

6.55.3.8 virtual void mitkLineWidgetModel3D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.55.3.9 virtual int mitkLineWidgetModel3D::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.55.3.10 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

6.55.3.11 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

6.55.3.12 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

6.55.3.13 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

6.56 mitkList Class Reference

mitkList - a utility class for a list of mitkObject

`#include <mitkList.h>`

Public Member Functions

- void **Delete** ()
- 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 ()
- void **Clear** ()
- int Find (mitkObject *itemFinding)
- mitkObject * GetItem (int i)
- mitkObject * **Item** (int i)
- int Count ()
- int **GetNumberOfItems** ()
- void InitTraversal ()
- mitkObject * **GetNextItem** (void)

Protected Attributes

- int **m_Count**
- mitkListNode * **m_Top**
- mitkListNode * **m_Bottom**
- mitkListNode * **m_Current**

6.56.1 Detailed Description

mitkList - a utility class for a list of mitkObject

mitkList is a utility class for a list of mitkObject

6.56.2 Member Function Documentation

6.56.2.1 void mitkList::Add (mitkObject * *itemAdding*)

Add an object to the list.

6.56.2.2 int mitkList::Count () `[inline]`

Return the number of objects in the list.

6.56.2.3 int mitkList::Find (mitkObject * *itemFinding*)

Search for an object and return location in list. If location == -1, object was not found.

6.56.2.4 mitkObject* mitkList::GetItem (int *i*)

Get an object in location i

6.56.2.5 void mitkList::InitTraversal () [inline]

Traversal the list

6.56.2.6 void mitkList::Insert (mitkObject * *itemInserting*, int *pos*)

Insert an object to the list.

6.56.2.7 void mitkList::Remove (mitkObject * *itemRemove*)

Remove an object from the list.

6.56.2.8 void mitkList::Remove (int *i*)

Remove the i'th item in the list.

6.56.2.9 void mitkList::RemoveAll ()

Remove all objects from the list.

6.56.2.10 void mitkList::Replace (int *i*, mitkObject * *itemNew*)

Replace the i'th item

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

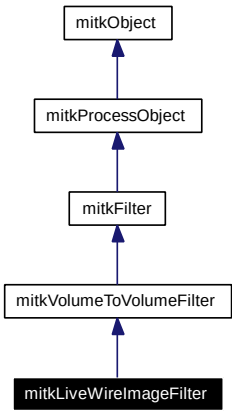
- mitkList.h

6.57 mitkLiveWireImageFilter Class Reference

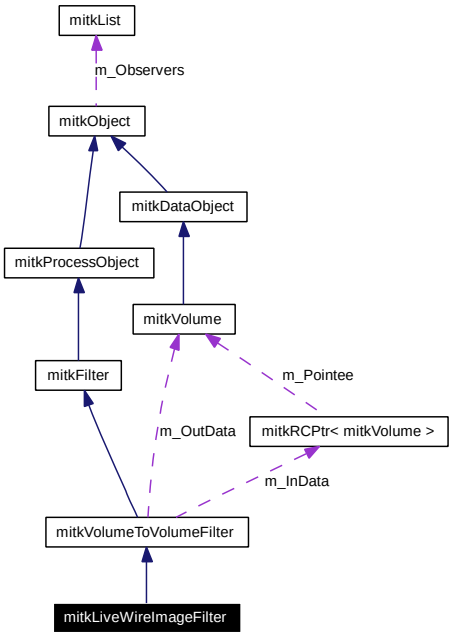
mitkLiveWireImageFilter - A class that implement livewire segmentation arithmetic

`#include <mitkLiveWireImageFilter.h>`

Inheritance diagram for mitkLiveWireImageFilter:



Collaboration diagram for mitkLiveWireImageFilter:



Public Member Functions

- mitkLiveWireImageFilter ()
- virtual void PrintSelf (ostream &os)
- virtual ~mitkLiveWireImageFilter ()
- void **Initialize** ()

- void **SetStartPoint** (int x, int y)
- void **SetEndPoint** (int x, int y)
- Contour * **GetContour** ()

Protected Member Functions

- virtual bool **Execute** ()
- void **ClearDir** ()

Protected Attributes

- double ** **m_CRow**
- double ** **m_CCol**
- double * **m_GradientImage**
- double * **m_DirVal**
- int * **m_DirX**
- int * **m_DirY**
- point **m_StartPoint**
- point **m_EndPoint**

6.57.1 Detailed Description

mitkLiveWireImageFilter - A class that implement livewire segmentation arithmetic

mitkLiveWireImageFilter - A class that implement livewire segmentation arithmetic

6.57.2 Constructor & Destructor Documentation

6.57.2.1 mitkLiveWireImageFilter::mitkLiveWireImageFilter ()

Constructor of the class.

6.57.2.2 virtual mitkLiveWireImageFilter::~mitkLiveWireImageFilter () [virtual]

Constructor of the class.

6.57.3 Member Function Documentation

6.57.3.1 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:

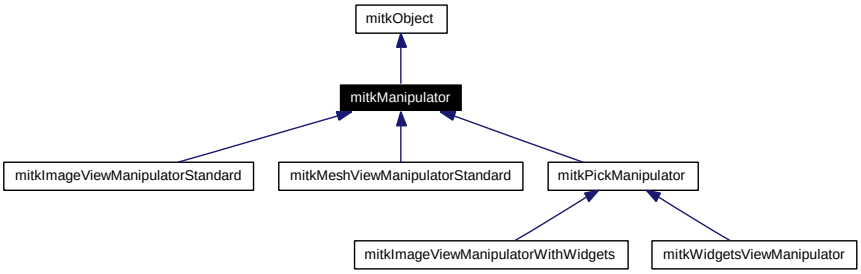
- mitkLiveWireImageFilter.h

6.58 mitkManipulator Class Reference

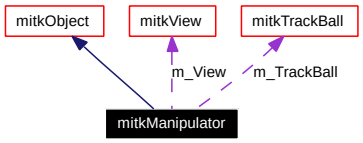
mitkManipulator - an abstract class to define an interface to implement mouse events processing in a view

`#include <mitkManipulator.h>`

Inheritance diagram for mitkManipulator:



Collaboration diagram for mitkManipulator:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- **mitkManipulator** (mitkView *view)
- 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 SetView (mitkView *view)
- void EnableLoD (bool enable=true)
- bool IsLoDEnabled ()

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)=0
- virtual void **_onMouseWheel** (bool ctrlDown, bool shiftDown, int xPos, int yPos, int delta)

Protected Attributes

- mitkTrackBall **m_TrackBall**
- mitkView * **m_View**

- int **m_OldX** [3]
- int **m_OldY** [3]
- bool **m_ButtonDown** [3]
- bool **m_EnableLoD**

6.58.1 Detailed Description

mitkManipulator - an abstract class to define an interface to implement mouse events processing in a view

mitkManipulator is an abstract class to define an interface to implement mouse events processing in a view. 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 view properly.

6.58.2 Member Function Documentation

6.58.2.1 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.

6.58.2.2 bool mitkManipulator::IsLoDEnabled () `[inline]`

Test if the LoD mode is enabled.

Returns:

Return true if the LoD mode is enabled, otherwise return false.

6.58.2.3 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

6.58.2.4 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

6.58.2.5 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

6.58.2.6 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

6.58.2.7 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.

6.58.2.8 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:

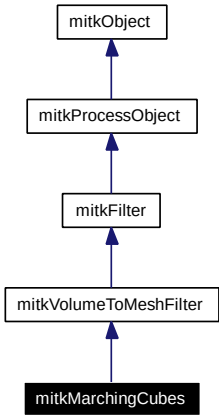
- mitkManipulator.h

6.59 mitkMarchingCubes Class Reference

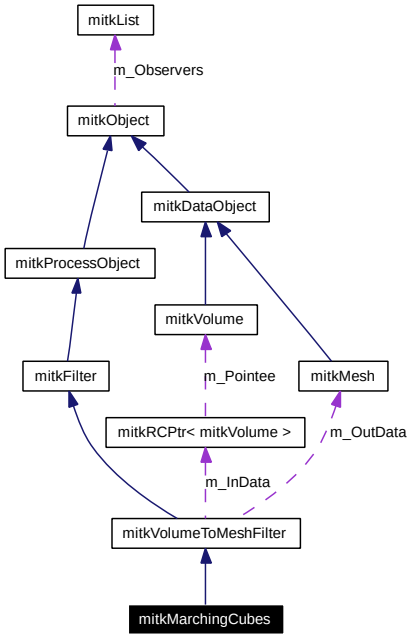
mitkMarchingCubes - implementing 3d reconstruction algorithm

`#include <mitkMarchingCubes.h>`

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 ()

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- float **m_LowThreshold**
- float **m_HighThreshold**

6.59.1 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.

6.59.2 Constructor & Destructor Documentation

6.59.2.1 mitkMarchingCubes::mitkMarchingCubes ()

Default constructor.

6.59.3 Member Function Documentation

6.59.3.1 float mitkMarchingCubes::GetHighThreshold () [inline]

Get low and high threshold.

Returns:
Return the high threshold.

6.59.3.2 float mitkMarchingCubes::GetLowThreshold () [inline]

Get low threshold.

Returns:
Retrurn the low threshold.

6.59.3.3 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.

6.59.3.4 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

6.60 mitkMatrix Class Reference

mitkMatrix - an encapsulation of a matrix

`#include <mitkMatrix.h>`

Public Member Functions

- **mitkMatrix** (const mitkMatrix &m)
- **mitkMatrix** (float e0, float e1, float e2, float e3, float e4, float e5, float e6, float e7, float e8, float e9, float e10, float e11, float e12, float e13, float e14, float e15)
- mitkMatrix & **operator=** (const mitkMatrix &m)
- mitkMatrix & **operator** *= (const mitkMatrix &)
- mitkMatrix & **operator** *= (const float)
- mitkMatrix & **operator+=** (const mitkMatrix &)
- mitkMatrix & **operator-=** (const mitkMatrix &)
- **operator const float** * () const
- **operator float** * ()
- 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)

Public Attributes

- float **ele** [16]

Friends

- mitkMatrix **operator** * (const mitkMatrix &, const mitkMatrix &)
- mitkMatrix **operator+** (const mitkMatrix &, const mitkMatrix &)
- mitkMatrix **operator-** (const mitkMatrix &, const mitkMatrix &)
- mitkMatrix **operator+** (const mitkMatrix &)
- mitkMatrix **operator-** (const mitkMatrix &)
- mitkMatrix **operator** * (const mitkMatrix &, const float)
- mitkMatrix **operator** * (const float, const mitkMatrix &)

6.60.1 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.

6.60.2 Member Function Documentation

6.60.2.1 void mitkMatrix::Adjoint ()

Adjoint matrix

6.60.2.2 float mitkMatrix::Determinant ()

Returns the determinant

6.60.2.3 void mitkMatrix::IdentityMatrix () [inline]

Set the matrix to identity matrix

6.60.2.4 float mitkMatrix::Inverse ()

Inverses the matrix and returns the determinant

6.60.2.5 float mitkMatrix::MaxValue ()

Returns the maximum absolute value of the matrix

6.60.2.6 float mitkMatrix::MinValue ()

Returns the minimum absolute value of the matrix

6.60.2.7 void mitkMatrix::Rotate (const float *angle*, const float *x*, const float *y*, const float *z*)

Rotate current matrix around arbitrary axis

6.60.2.8 void mitkMatrix::RotateX (const float *angle*)

Rotate current matrix around x axis

6.60.2.9 void mitkMatrix::RotateXMatrix (const float *angle*)

Rotation around x axis

6.60.2.10 void mitkMatrix::RotateY (const float *angle*)

Rotate current matrix around y axis

6.60.2.11 void mitkMatrix::RotateYMatrix (const float *angle*)

Rotation around y axis

6.60.2.12 void mitkMatrix::RotateZ (const float *angle*)

Rotate current matrix around z axis

6.60.2.13 void mitkMatrix::RotateZMatrix (const float *angle*)

Rotation around z axis

6.60.2.14 void mitkMatrix::Scale (const float *a*, const float *b*, const float *c*)

Scale current matrix

6.60.2.15 void mitkMatrix::ScaleMatrix (const float *a*)

Uniform scale transformation

6.60.2.16 void mitkMatrix::ScaleMatrix (const float *a*, const float *b*, const float *c*)

Scale transformation

6.60.2.17 void mitkMatrix::Translate (const float *dx*, const float *dy*, const float *dz*)

Translate current matrix

6.60.2.18 void mitkMatrix::TranslateMatrix (const float *dx*, const float *dy*, const float *dz*)

Translate transformation

6.60.2.19 void mitkMatrix::Transpose ()

Transposes the matrix

6.60.2.20 void mitkMatrix::ZeroMatrix () [inline]

Clean the matrix to zero

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

- mitkMatrix.h

6.61 mitkMatrixD Class Reference

mitkMatrixD - an encapsulation of a matrix

`#include <mitkMatrixD.h>`

Public Member Functions

- **mitkMatrixD** (const mitkMatrixD &m)
- **mitkMatrixD** (double e0, double e1, double e2, double e3, double e4, double e5, double e6, double e7, double e8, double e9, double e10, double e11, double e12, double e13, double e14, double e15)
- mitkMatrixD & **operator=** (const mitkMatrixD &m)
- mitkMatrixD & **operator** *= (const mitkMatrixD &)
- mitkMatrixD & **operator** *= (const double)
- mitkMatrixD & **operator+=** (const mitkMatrixD &)
- mitkMatrixD & **operator-=** (const mitkMatrixD &)
- **operator const double** * () const
- **operator double** * ()
- 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)

Public Attributes

- double **ele** [16]

Friends

- mitkMatrixD **operator** * (const mitkMatrixD &, const mitkMatrixD &)
- mitkMatrixD **operator+** (const mitkMatrixD &, const mitkMatrixD &)
- mitkMatrixD **operator-** (const mitkMatrixD &, const mitkMatrixD &)
- mitkMatrixD **operator+** (const mitkMatrixD &)
- mitkMatrixD **operator-** (const mitkMatrixD &)
- mitkMatrixD **operator** * (const mitkMatrixD &, const double)
- mitkMatrixD **operator** * (const double, const mitkMatrixD &)

6.61.1 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.

6.61.2 Member Function Documentation

6.61.2.1 void mitkMatrixD::Adjoint ()

Adjoint matrix

6.61.2.2 double mitkMatrixD::Determinant ()

Returns the determinant

6.61.2.3 void mitkMatrixD::IdentityMatrix () [inline]

Set the matrix to identity matrix

6.61.2.4 double mitkMatrixD::Inverse ()

Inverses the matrix and returns the determinant

6.61.2.5 double mitkMatrixD::MaxValue ()

Returns the maximum absolute value of the matrix

6.61.2.6 double mitkMatrixD::MinValue ()

Returns the minimum absolute value of the matrix

6.61.2.7 void mitkMatrixD::Rotate (const double *angle*, const double *x*, const double *y*, const double *z*)

Rotate current matrix around arbitrary axis

6.61.2.8 void mitkMatrixD::RotateX (const double *angle*)

Rotate current matrix around x axis

6.61.2.9 void mitkMatrixD::RotateXMatrix (const double *angle*)

Rotation around x axis

6.61.2.10 void mitkMatrixD::RotateY (const double *angle*)

Rotate current matrix around y axis

6.61.2.11 void mitkMatrixD::RotateYMatrix (const double *angle*)

Rotation around y axis

6.61.2.12 void mitkMatrixD::RotateZ (const double *angle*)

Rotate current matrix around z axis

6.61.2.13 void mitkMatrixD::RotateZMatrix (const double *angle*)

Rotation around z axis

6.61.2.14 void mitkMatrixD::Scale (const double *a*, const double *b*, const double *c*)

Scale current matrix

6.61.2.15 void mitkMatrixD::ScaleMatrix (const double *a*)

Uniform scale transformation

6.61.2.16 void mitkMatrixD::ScaleMatrix (const double *a*, const double *b*, const double *c*)

Scale transformation

6.61.2.17 void mitkMatrixD::Translate (const double *dx*, const double *dy*, const double *dz*)

Translate current matrix

6.61.2.18 void mitkMatrixD::TranslateMatrix (const double *dx*, const double *dy*, const double *dz*)

Translate transformation

6.61.2.19 void mitkMatrixD::Transpose ()

Transposes the matrix

6.61.2.20 void mitkMatrixD::ZeroMatrix () [inline]

Clean the matrix to zero

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

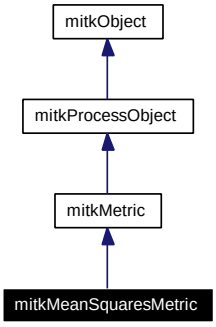
- mitkMatrixD.h

6.62 mitkMeanSquaresMetric Class Reference

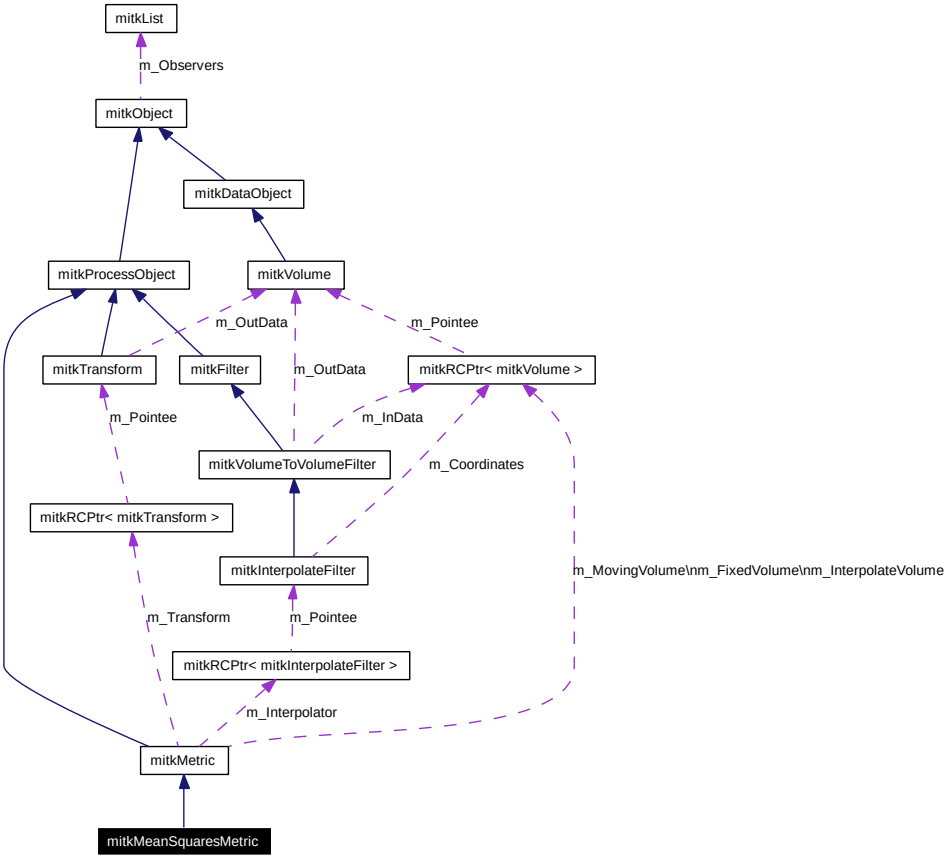
mitkMeanSquaresMetric - a concrete class computing similarity between two objects to be registered

`#include <mitkMeanSquaresMetric.h>`

Inheritance diagram for mitkMeanSquaresMetric:



Collaboration diagram for mitkMeanSquaresMetric:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- virtual float GetSimilarity (vector< float > *transformParameters)

Protected Member Functions

- virtual bool **Execute** ()

6.62.1 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.

6.62.2 Member Function Documentation

6.62.2.1 virtual float mitkMeanSquaresMetric::GetSimilarity (vector< float > * *transformParameters*) [virtual]

Get Similarity between regions of two volumes

Reimplemented from mitkMetric.

6.62.2.2 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:

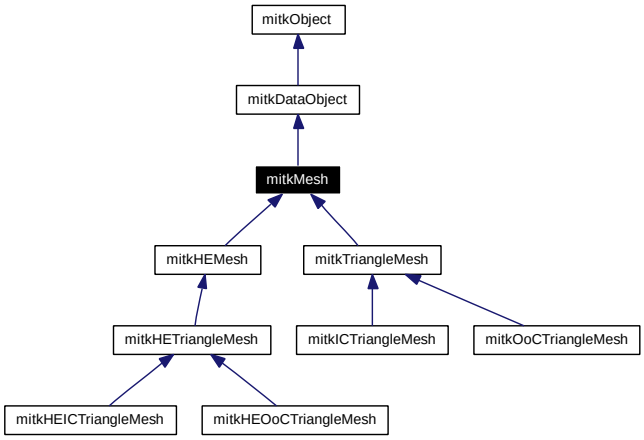
- mitkMeanSquaresMetric.h

6.63 mitkMesh Class Reference

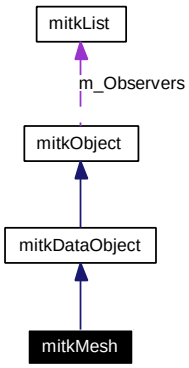
mitkMesh - an abstract class for mesh types

`#include <mitkMesh.h>`

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)

Protected Types

- enum **Orientation** { **CLOCKWISE**, **ANTICLOCKWISE**, **DEGENERATIVE** }

Protected Member Functions

- virtual index_type **_addVertex** (Vertex3f &vert)=0
- virtual index_type **_addFace** (unsigned int num, index_type *verts)=0
- virtual Vertex3f const * **_getVertex** (index_type vertIdx)=0
- virtual index_type const * **_getFace** (index_type faceIdx)=0
- virtual Vertex3f * **_getVertexForWrite** (index_type vertIdx)=0
- virtual index_type * **_getFaceForWrite** (index_type faceIdx)=0
- virtual size_type **_getVertices** (index_type startIdx, size_type num, Vertex3f *verts)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, TriangleFace *faces)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, Vertex3f *verts)
- virtual bool **_setFace** (index_type faceIdx, unsigned int num, index_type *verts)

Protected Attributes

- float **m_BoundingBox** [6]
- bool **m_IsClockwise**
- bool **m_IsNormalReversed**

6.63.1 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

6.63.2 Member Function Documentation

6.63.2.1 index_type mitkMesh::AddFace (TriangleFace & *face*) [inline]

An explicit instantiation of AddFace().

Reimplemented in mitkTriangleMesh.

6.63.2.2 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.

6.63.2.3 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

6.63.2.4 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.

6.63.2.5 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.

6.63.2.6 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

6.63.2.7 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

6.63.2.8 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.

6.63.2.9 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.

6.63.2.10 bool mitkMesh::GetFace (index_type *faceIdx*, TriangleFace & *face*) [inline]

An explicit instantiation of GetFace().

6.63.2.11 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.

6.63.2.12 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.

6.63.2.13 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.

6.63.2.14 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.

6.63.2.15 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.

6.63.2.16 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.

6.63.2.17 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.

6.63.2.18 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.

6.63.2.19 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.

6.63.2.20 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.

6.63.2.21 bool mitkMesh::IsClockwise () const [inline]

Get the orientation of front-facing polygons.

Returns:
Return true if selects clockwise polygons as front-facing.

6.63.2.22 bool mitkMesh::IsNormalReversed () const [inline]

Indicate if the normals are reversed.

Returns:
Return true if the normals are reversed, otherwise return false.

6.63.2.23 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.

6.63.2.24 virtual void mitkMesh::ReverseNormals () [inline, virtual]

Reverse normals.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

6.63.2.25 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

6.63.2.26 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

6.63.2.27 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

6.63.2.28 bool mitkMesh::SetFace (index_type *faceIdx*, TriangleFace & *face*) [inline]

An explicit instantiation of SetFace().

6.63.2.29 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.

6.63.2.30 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.

6.63.2.31 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.

6.63.2.32 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.

6.63.2.33 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.

6.63.2.34 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:

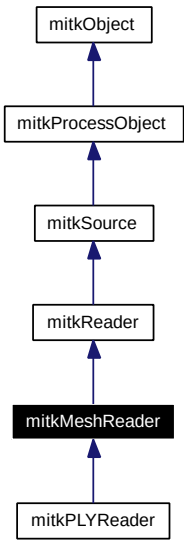
- mitkMesh.h

6.64 mitkMeshReader Class Reference

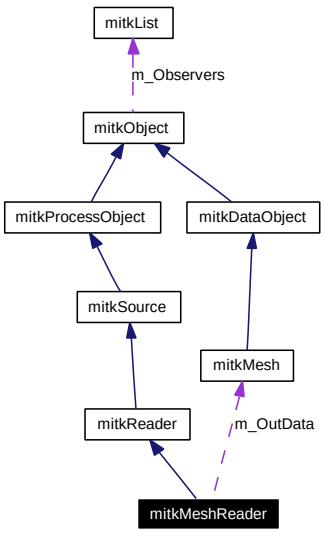
mitkMeshReader - an abstract class represents a mesh reader to read mesh files

`#include <mitkMeshReader.h>`

Inheritance diagram for mitkMeshReader:



Collaboration diagram for mitkMeshReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetOoCSupport (char const *diskPath, unsigned int bufBlockNum=32, bool supportOoC=true)
- mitkMesh * GetOutput ()
- void **SetOutputMesh** (mitkMesh *mesh)

Protected Attributes

- mitkMesh * **m_OutData**
- mitkString * **m_DiskPath**
- unsigned int **m_BufferedBlockNum**
- bool **m_NeedOoC**

6.64.1 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
}
```

6.64.2 Member Function Documentation

6.64.2.1 mitkMesh* mitkMeshReader::GetOutput ()

Get the output mesh the reader has read.

Returns:
the output volume.

6.64.2.2 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.

6.64.2.3 void mitkMeshReader::SetOoCSupport (char const * *diskPath*, unsigned int *bufBlockNum* = 32, 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:

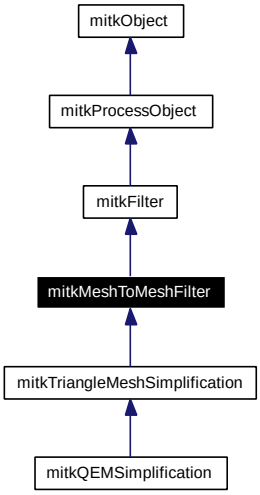
- mitkMeshReader.h

6.65 mitkMeshToMeshFilter Class Reference

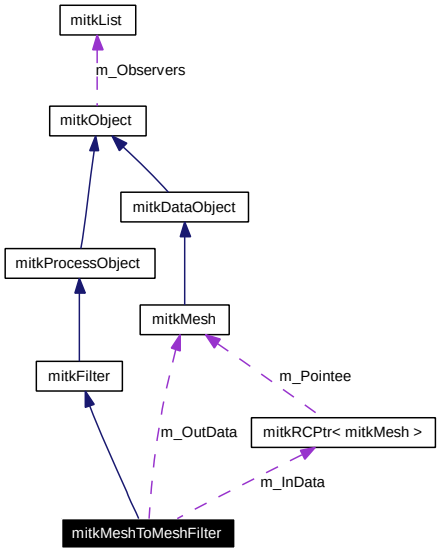
mitkMeshToMeshFilter - abstract class specifies interface for mesh to mesh filter

`#include <mitkMeshToMeshFilter.h>`

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, unsigned int bufBlockNum=32, bool supportOoC=true)

- mitkMesh * GetOutput ()
- void SetOutputMesh (mitkMesh *mesh)

Protected Member Functions

- virtual mitkMesh * **_createOutMesh** ()=0

Protected Attributes

- mitkRCPtr< mitkMesh > **m_InData**
- mitkMesh * **m_OutData**
- mitkString * **m_DiskPath**
- unsigned int **m_BufferedBlockNum**
- bool **m_NeedOoC**

6.65.1 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.

6.65.2 Member Function Documentation

6.65.2.1 mitkMesh* mitkMeshToMeshFilter::GetInput () [inline]

Get the input mesh

Returns:
Return the input mesh.

6.65.2.2 mitkMesh* mitkMeshToMeshFilter::GetOutput () [inline]

Get the output mesh

Returns:
Return the output mesh.

6.65.2.3 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.

6.65.2.4 void mitkMeshToMeshFilter::SetInput (mitkMesh * *mesh*) [inline]

Set the input mesh.

Parameters:
mesh specify the input mesh

6.65.2.5 void mitkMeshToMeshFilter::SetOoCSupport (char const * *diskPath*, unsigned int *bufBlockNum* = 32, 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.

6.65.2.6 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:

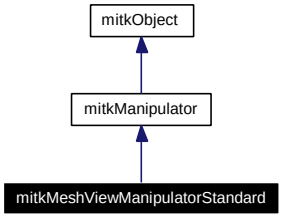
- mitkMeshToMeshFilter.h

6.66 mitkMeshViewManipulatorStandard Class Reference

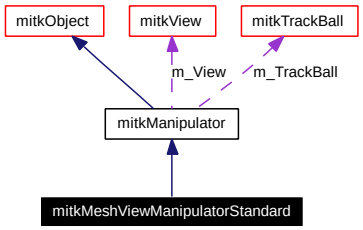
mitkMeshViewManipulatorStandard - a concrete manipulator to process mouse events in an 3D view

`#include <mitkMeshViewManipulatorStandard.h>`

Inheritance diagram for mitkMeshViewManipulatorStandard:



Collaboration diagram for mitkMeshViewManipulatorStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- **mitkMeshViewManipulatorStandard** (mitkView *view)

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)

6.66.1 Detailed Description

mitkMeshViewManipulatorStandard - a concrete manipulator to process mouse events in an 3D view

mitkMeshViewManipulatorStandard is a concrete manipulator to process mouse events in an 3D view. It is the default manipulator of a mitkView. The behavior of this manipulator is shown as follows:

Mouse Left Key — Rotation

Mouse Middle Key — Translation

Mouse Right Key — Zoom in / out

6.66.2 Member Function Documentation

6.66.2.1 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.

6.66.2.2 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.

6.66.2.3 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.

6.66.2.4 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:

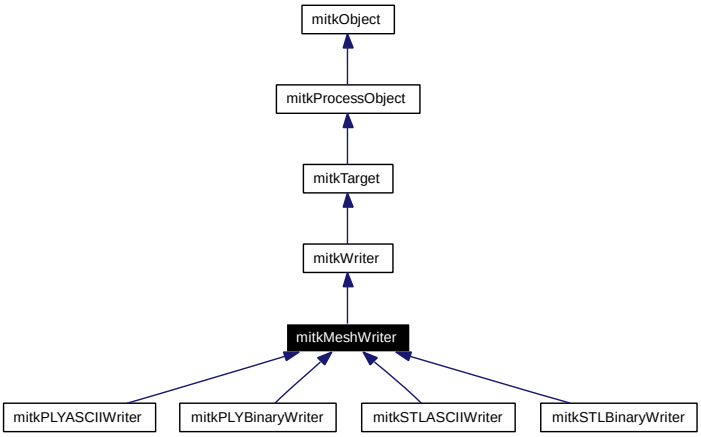
- mitkMeshViewManipulatorStandard.h

6.67 mitkMeshWriter Class Reference

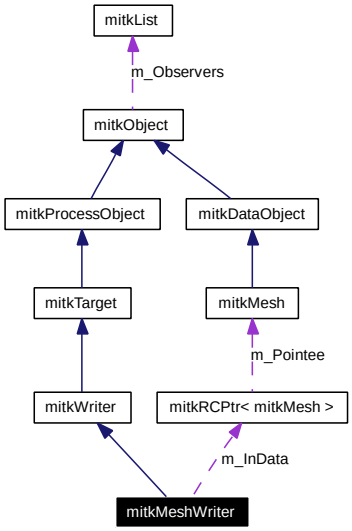
mitkMeshWriter - an abstract class represents a mesh writer for writing mesh data to disk

`#include <mitkMeshWriter.h>`

Inheritance diagram for mitkMeshWriter:



Collaboration diagram for mitkMeshWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInput (mitkMesh *inData)
- mitkMesh * GetInput ()

Protected Attributes

- mitkRCPtr< mitkMesh > **m_InData**

6.67.1 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()
```

6.67.2 Member Function Documentation

6.67.2.1 mitkMesh* mitkMeshWriter::GetInput () [inline]

Get input mesh.

Returns:
Return the input volume

6.67.2.2 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.

6.67.2.3 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:

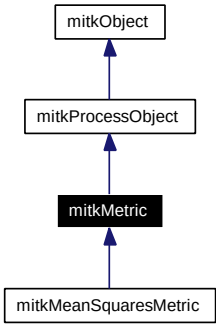
- mitkMeshWriter.h

6.68 mitkMetric Class Reference

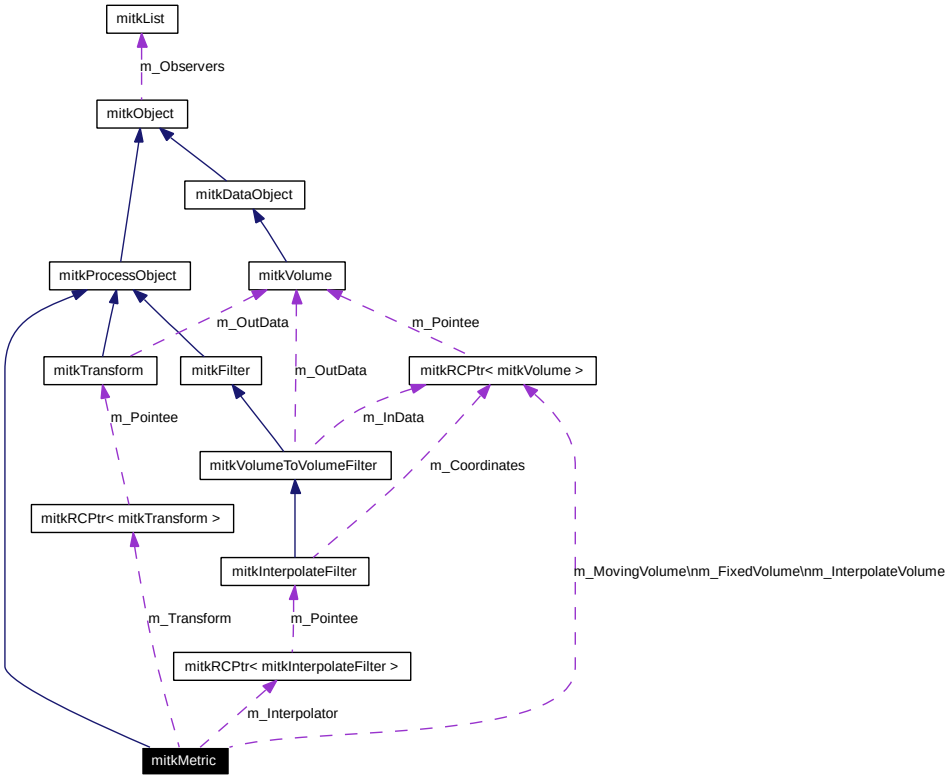
mitkMetric - an abstract class specifies interface for computing similarity between regions of two volumes

`#include <mitkMetric.h>`

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 (vector< float > *parameters)
- vector< float > * GetTransformParameters ()
- void Update ()
- virtual float GetSimilarity (vector< float > *transformParameters)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_FixedVolume**
- mitkRCPtr< mitkVolume > **m_MovingVolume**
- mitkRCPtr< mitkVolume > **m_InterpolateVolume**
- float **m_Similarity**
- bool **m_FirstRun**
- int **m_Region** [6]
- int **m_Dimensions** [3]
- void * **m_FixedData**
- float * **m_InterpolateData**
- mitkRCPtr< mitkTransform > **m_Transform**
- mitkRCPtr< mitkInterpolateFilter > **m_Interpolator**
- vector< float > * **m_TransformParameters**

6.68.1 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.

6.68.2 Member Function Documentation

6.68.2.1 mitkVolume* mitkMetric::GetFixedVolume () [inline]

Get the Fixed Volume.

Returns:
Return the pointer to the fixed volume.

6.68.2.2 mitkInterpolateFilter* mitkMetric::GetInterpolator ()

Get the Interpolator.

Returns:
Return the pointer to the Interpolator.

6.68.2.3 mitkVolume* mitkMetric::GetMovingVolume () [inline]

Get the Moving Volume.

Returns:
Return the pointer to the moving volume.

6.68.2.4 virtual float mitkMetric::GetSimilarity (vector< float > * *transformParameters*) [virtual]

Get Similarity between regions of two volumes.

Parameters:
transformParameters The vector pointer to the transform parameters.

Returns:
Return the similarity.

Reimplemented in mitkMeanSquaresMetric.

6.68.2.5 mitkTransform* mitkMetric::GetTransform ()

Get the Transform.

Returns:
Return the pointer to the Transform.

6.68.2.6 vector<float>* mitkMetric::GetTransformParameters ()

Get the transform parameters.

Returns:
Return the vector pointer to the transform parameters.

6.68.2.7 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 mitkMeanSquaresMetric.

6.68.2.8 void mitkMetric::SetFixedVolume (mitkVolume * *fixedVolume*) [inline]

Set the Fixed Volume.

Parameters:
fixedVolume The pointer to the fixed volume.

6.68.2.9 void mitkMetric::SetInterpolator (mitkInterpolateFilter * *interpolator*) [inline]

Set the Interpolator.

Parameters:
interpolator The pointer to the Interpolator.

6.68.2.10 void mitkMetric::SetMovingVolume (mitkVolume * *movingVolume*) [inline]

Set the Moving Volume.

Parameters:
movingVolume The pointer to the moving volume.

6.68.2.11 void mitkMetric::SetTransform (mitkTransform * *transform*) [inline]

Set the Transform.

Parameters:
transform The pointer to the Transform.

6.68.2.12 void mitkMetric::SetTransformParameters (vector< float > * *parameters*)

Set the Transform parameters.

Parameters:
paraeters The vector pointer to the transform parameters.

6.68.2.13 void mitkMetric::Update ()

Initialize the Metric by making sure that all the components are present and plugged together correctly. Perform before Run() function.

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

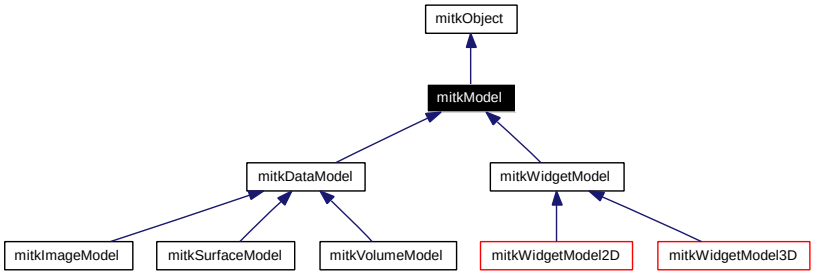
- mitkMetric.h

6.69 mitkModel Class Reference

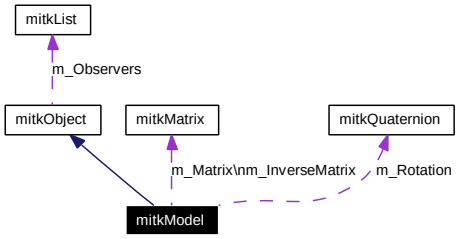
mitkModel - abstract class used to represent an entity in a rendering scene

`#include <mitkModel.h>`

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 (mitkView *view)
- virtual void Select (mitkView *view)
- 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

Protected Member Functions

- void **_computeMatrix** ()
- virtual float * **_getBounds** ()=0

Protected Attributes

- mitkMatrix * **m_Matrix**
- mitkMatrix * **m_InverseMatrix**
- mitkQuaternion * **m_Rotation**
- float **m_Origin** [3]
- float **m_Translation** [3]
- float **m_Scale** [3]

- float **m_Center** [3]
- float **m_Bounds** [6]
- RenderMode **m_RenderMode**
- short **m_Visibility**
- short **m_MatrixModified**
- bool **m_DataChanged**

6.69.1 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.

6.69.2 Member Enumeration Documentation

6.69.2.1 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

6.69.3 Member Function Documentation

6.69.3.1 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.

6.69.3.2 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.

6.69.3.3 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.

6.69.3.4 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

6.69.3.5 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.

6.69.3.6 bool mitkModel::GetDataModifyStatus () const [inline]

Get the status of source data.

Returns:
Return true if the source data is modified. Otherwise return false.

6.69.3.7 mitkMatrix const* mitkModel::GetInverseOfModelMatrix ()

Get the inverse of model matrix.

Returns:
Return a pointer to a mitkMatrix contains the inverse of model matrix.

6.69.3.8 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]
```

6.69.3.9 void mitkModel::GetInverseOfModelMatrix (mitkMatrix * *m*)

Get the inverse of model matrix.

Parameters:
m pointer to a mitkMatrix returns the inverse of model matrix

6.69.3.10 float mitkModel::GetLength ()

Get the length of the diagonal of the bounding box.

Returns:
the length of the diagonal of the bounding box

6.69.3.11 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.

6.69.3.12 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]
```

6.69.3.13 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

6.69.3.14 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

6.69.3.15 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.

6.69.3.16 RenderMode mitkModel::GetRenderMode () const [inline]

Get the render mode of this model.

See also:

RenderMode

6.69.3.17 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

6.69.3.18 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

6.69.3.19 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.

6.69.3.20 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

6.69.3.21 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.

6.69.3.22 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

6.69.3.23 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.

6.69.3.24 int mitkModel::GetVisibility (void) const [inline]

Get the visibility of this model.

Returns:
Return 1 if the model is visible. Otherwise return 0.

6.69.3.25 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, mitkSurfaceModel, mitkVolumeModel, and mitkWidgetModel.

6.69.3.26 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

6.69.3.27 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, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, mitkReslicePlaneWidgetModel, mitkSurfaceModel, mitkVolumeModel, mitkWidgetModel, mitkWidgetModel2D, and mitkWidgetModel3D.

6.69.3.28 virtual int mitkModel::Render (mitkView * *view*) [inline, virtual]

Render this model.

Parameters:
view the pointer of an mitkView 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, mitkPolygonWidgetModel2D, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkRectWidgetModel2D, mitkReslicePlaneWidgetModel, mitkSurfaceModel, and mitkVolumeModel.

6.69.3.29 void mitkModel::Reset ()

Reset all the geometric transformation

6.69.3.30 virtual void mitkModel::Select (mitkView * *view*) [inline, virtual]

Selecting object in interactive mode using widgets.

Parameters:
view the pointer of an mitkView 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.

6.69.3.31 void mitkModel::SetDataModifyStatus (bool *isModify*) [inline]

Set the status of source data.

Parameters:
isModify if the status of source data is modified

6.69.3.32 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

6.69.3.33 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

6.69.3.34 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

6.69.3.35 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

6.69.3.36 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

6.69.3.37 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

6.69.3.38 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

6.69.3.39 void mitkModel::SetRotation (const mitkQuaternion & *q*) [inline]

Set the rotation of the model.

Parameters:

q the quaternion of rotation

6.69.3.40 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

6.69.3.41 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

6.69.3.42 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

6.69.3.43 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

6.69.3.44 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

6.69.3.45 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

6.69.3.46 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

6.69.3.47 void mitkModel::SetVisibility (int *isVisible*) [inline]

Set the visibility of this model.

Parameters:

isVisible if this model is visible (0 - invisible; 1 - visible)

6.69.3.48 void mitkModel::VisibilityOff (void) [inline]

Set the visibility of this model off.

6.69.3.49 void mitkModel::VisibilityOn (void) [inline]

Set the visibility of this model on.

6.69.3.50 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:

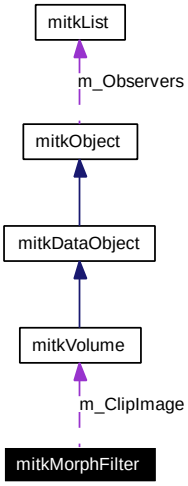
- mitkModel.h

6.70 mitkMorphFilter Class Reference

mitkMorphFilter - a subclass of the mitkLevelSetImageFilter

`#include <mitkMorphFilter.h>`

Collaboration diagram for mitkMorphFilter:



Public Member Functions

- mitkMorphFilter ()
- ~mitkMorphFilter ()
- void SetIterations (int i)
- void SetClipImage (mitkVolume *vol)

Protected Member Functions

- virtual bool **Execute** ()

6.70.1 Detailed Description

mitkMorphFilter - a subclass of the mitkLevelSetImageFilter

mitkMorphFilter is a subclass of the mitkLevelSetImageFilter.

6.70.2 Constructor & Destructor Documentation

6.70.2.1 mitkMorphFilter::mitkMorphFilter () `[inline]`

Constructor of the class

6.70.2.2 mitkMorphFilter::~mitkMorphFilter () `[inline]`

Destructor of the class

6.70.3 Member Function Documentation

6.70.3.1 void mitkMorphFilter::SetClipImage (mitkVolume * *vol*)

Set the clip imge

Parameters:
vol represent the clip image

6.70.3.2 void mitkMorphFilter::SetIterations (int *i*) [inline]

Set the iteration numbers

Parameters:
i represent the iteration number

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

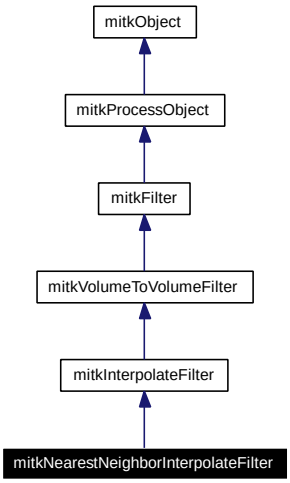
- mitkMorphFilter.h

6.71 mitkNearestNeighborInterpolateFilter Class Reference

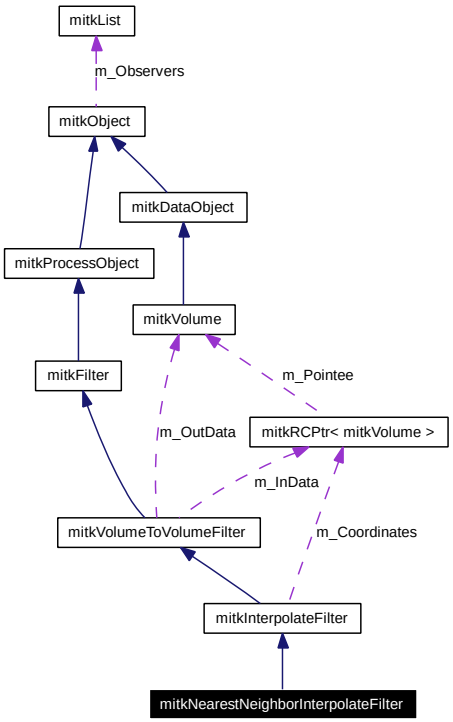
mitkNearestNeighborInterpolateFilter - a concrete interpolator

`#include <mitkNearestNeighborInterpolateFilter.h>`

Inheritance diagram for mitkNearestNeighborInterpolateFilter:



Collaboration diagram for mitkNearestNeighborInterpolateFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual bool **Execute** ()

6.71.1 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.

6.71.2 Member Function Documentation

6.71.2.1 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:

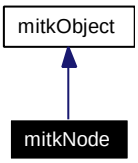
- mitkNearestNeighborInterpolateFilter.h

6.72 mitkNode Class Reference

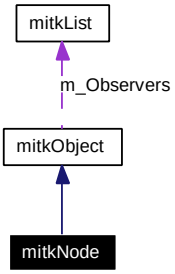
mitkNode - a class that define NodeType used by mitkFastMarchingImageFilter

`#include <mitkNode.h>`

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)

6.72.1 Detailed Description

mitkNode - a class that define NodeType used by mitkFastMarchingImageFilter

mitkNode defines several node types that represent the nodes' positions.

6.72.2 Constructor & Destructor Documentation

6.72.2.1 mitkNode::mitkNode (const mitkNode & *node*) [inline]

Constructor of this class

Parameters:
node The object according which to construct this class

6.72.3 Member Function Documentation

6.72.3.1 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.

6.72.3.2 double mitkNode::GetValue () const [inline]

Get the pix value of the node.

Returns:
Return the pix value of the node.

6.72.3.3 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.

6.72.3.4 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.

6.72.3.5 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.

6.72.3.6 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.

6.72.3.7 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.

6.72.3.8 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:

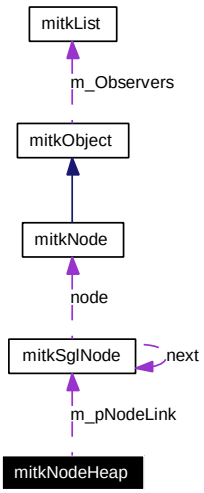
- mitkNode.h

6.73 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 ()

6.73.1 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.

6.73.2 Constructor & Destructor Documentation

6.73.2.1 mitkNodeHeap::mitkNodeHeap ()

Constructor of the class

6.73.2.2 mitkNodeHeap::~mitkNodeHeap ()

Destructor of the class

6.73.3 Member Function Documentation

6.73.3.1 bool mitkNodeHeap::empty ()

Check if the heap is empty

Returns:
Return true if the heap is empty otherwise return false.

6.73.3.2 void mitkNodeHeap::pop ()

remove the top node of the heap.

6.73.3.3 void mitkNodeHeap::push (mitkNode *node*)

Push a node into the heap.

Parameters:
node Represent the node to put into the heap

6.73.3.4 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:

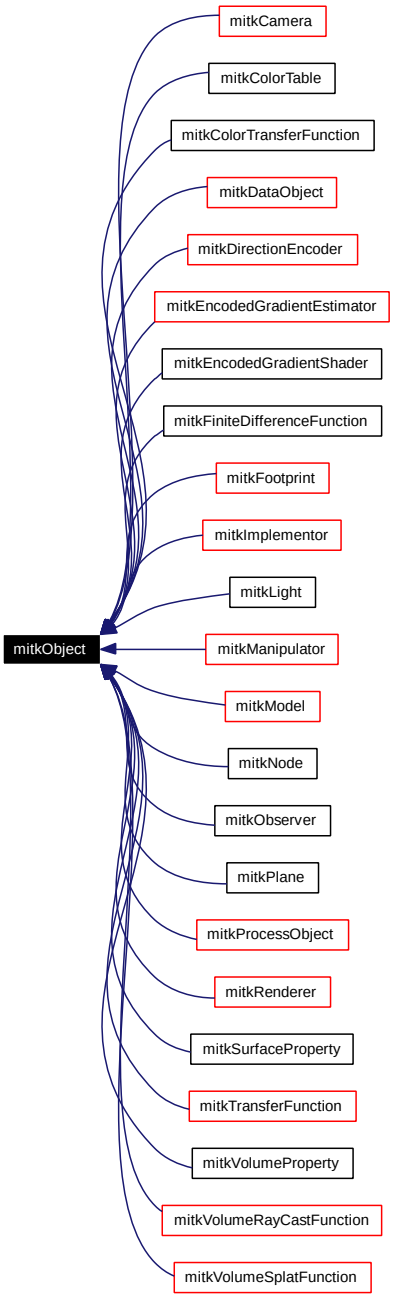
- mitkNodeHeap.h

6.74 mitkObject Class Reference

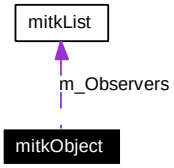
mitkObject - abstract base class for most objects in MITK

`#include <mitkObject.h>`

Inheritance diagram for mitkObject:



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 ()

Protected Member Functions

- void **_updateObservers** ()

Protected Attributes

- int **m_Debug**
- int **m_ReferenceCount**
- mitkList * **m_Observers**

Friends

- ostream & **operator**<< (ostream &os, mitkObject &o)

6.74.1 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.

6.74.2 Member Function Documentation

6.74.2.1 void mitkObject::AddObserver (mitkObserver * *observer*)

Attach an observer to this object.

Parameters:

observer pointer to an mitkObserver to attach

6.74.2.2 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.

6.74.2.3 static void mitkObject::BreakOnError () [static]

This method is called when mitkErrorMessage executes.

6.74.2.4 virtual void mitkObject::DebugOff () [virtual]

Turn debugging output off.

6.74.2.5 virtual void mitkObject::DebugOn () [virtual]

Turn debugging output on.

6.74.2.6 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.

6.74.2.7 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"

6.74.2.8 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.

6.74.2.9 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.

6.74.2.10 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.

6.74.2.11 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.

6.74.2.12 void mitkObject::Print (ostream & *os*)

Print this object to an ostream.

Parameters:
os The specified ostream to output information.

6.74.2.13 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 mitkAffineTransform, mitkAmoebaOptimizer, mitkAngleWidgetModel2D, mitkAngleWidgetModel3D, mitkBinaryFilter, mitkBinMarchingCubes, mitkBMPReader, mitkBMPWriter, mitkCacheVolumeReader, mitkCacheVolumeWriter, mitkCamera, mitkClippingPlaneWidgetModel, mitkColorTable, mitkColorTransferFunction, mitkDataModel, mitkDataObject, mitkDICOMInfoReader, mitkDICOMReader, mitkDICOMWriter, mitkDiffusionFilter, mitkEllipseWidgetModel2D, mitkFilter, mitkFiniteDifferenceFunction, mitkFootprint, mitkFootprint1D, mitkFootprint1DGaussian, mitkFootprint2D, mitkFootprint2DGaussian, mitkHEICTriangleMesh, mitkHEMesh, mitkHEOoCTriangleMesh, mitkHETriangleMesh, mitkICTriangleMesh, mitkICVolume, mitkImageModel, mitkImageView, mitkImageViewManipulatorStandard, mitkImageViewManipulatorWithWidgets, mitkImplementor, mitkInfoReader, mitkInterpolateFilter, mitkJPEGReader, mitkJPEGWriter, mitkLight, mitkLineWidgetModel2D, mitkLineWidgetModel3D, mitkLiveWireImageFilter, mitkManipulator, mitkMarchingCubes, mitkMeanSquaresMetric, mitkMesh, mitkMeshReader, mitkMeshToMeshFilter, mitkMeshViewManipulatorStandard, mitkMeshWriter, mitkMetric, mitkModel, mitkNearestNeighborInterpolateFilter, mitkObserver, mitkOoCSurfaceRendererStandard, mitkOoCSurfaceRendererUseVA, mitkOoCTriangleMesh, mitkOoCVolume, mitkOoCVolumeRayCastCompositeFunction, mitkOoCVolumeRendererRayCasting, mitkOoCVolumeRendererTexture3D, mitkOptimizer, mitkPickManipulator, mitkPlane, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkPLYReader, mitkPolygonWidgetModel2D, mitkProcessObject, mitkPseudocolorWidgetModel, mitkPseudocolorWidgetModelEx, mitkQEMSimplification, mitkRawFilesReader, mitkRawFilesWriter, mitkRawReader, mitkRawWriter, mitkReader, mitkRectWidgetModel2D, mitkRegionGrowImageFilter, mitkRegistrationFilter, mitkRenderer, mitkReslicePlaneWidgetModel, mitkRGBToGrayFilter, mitkRigidTransform, mitkSeedFillFilter, mitkSobelEdgeDetectFilter, mitkSource, mitkSplatCamera, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkSurfaceModel, mitkSurfaceProperty, mitkSurfaceRenderer, mitkSurfaceRendererStandard, mitkSurfaceRendererUseVA, mitkSurfaceRendererUseVBO, mitkTarget, mitkThresholdSegmentationFilter, mitkTIFFReader, mitkTIFFWriter, mitkTransferFunction, mitkTransferFunction1D, mitkTransform, mitkTriangleMesh, mitkTriangleMeshSimplification, mitkView, mitkVolume, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeModel, mitkVolumeProperty, mitkVolumeRayCastCompositeFunction, mitkVolumeRayCastFunction, mitkVolumeReader, mitkVolumeRenderer, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, mitkVolumeRendererTexture3D, mitkVolumeResizeFilter, mitkVolumeResliceFilter, mitkVolumeSplatFunction, mitkVolumeSplatParallel, mitkVolumeSplatPerspective, mitkVolumeToMeshFilter, mitkVolumeToVolumeFilter, mitkVolumeWriter, mitkWidgetModel, mitkWidgetModel2D, mitkWidgetModel3D, mitkWidgetsViewManipulator, mitkWin32Implementor, and mitkWriter.

6.74.2.14 void mitkObject::RemoveAllObservers ()

Detach all observers from this object.

6.74.2.15 void mitkObject::RemoveObserver (mitkObserver * *observer*)

Detach an observer from this object.

Parameters:
observer pointer to an mitkObserver to detach

6.74.2.16 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.

6.74.2.17 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.

6.74.2.18 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:

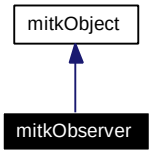
- mitkObject.h

6.75 mitkObserver Class Reference

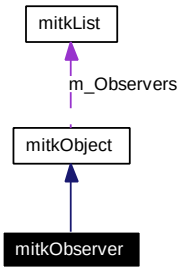
mitkObserver - abstract base class for observers

`#include <mitkObserver.h>`

Inheritance diagram for mitkObserver:



Collaboration diagram for mitkObserver:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void Update ()=0

6.75.1 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.

6.75.2 Member Function Documentation

6.75.2.1 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.

6.75.2.2 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:

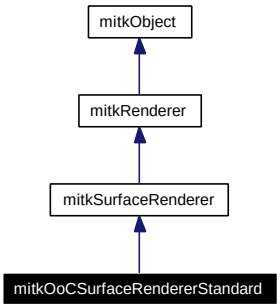
- mitkObserver.h

6.76 mitkOoCSurfaceRendererStandard Class Reference

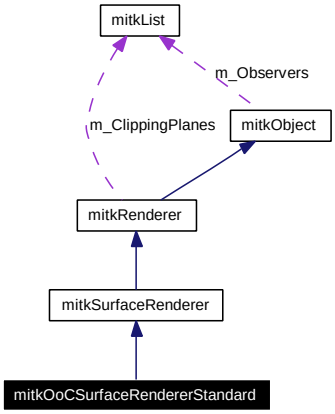
mitkOoCSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

`#include <mitkOoCSurfaceRendererStandard.h>`

Inheritance diagram for mitkOoCSurfaceRendererStandard:



Collaboration diagram for mitkOoCSurfaceRendererStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int **Render** (mitkView *view, mitkSurfaceModel *surf)

Protected Member Functions

- void **_setMaterial** (mitkSurfaceProperty *prop)
- void **_drawPoints** (unsigned int vertexNum, Vertex3f *vertexData)
- void **_drawPoints** (mitkMesh *mesh)
- void **_drawWireFrame** (unsigned int faceNum, Vertex3f *vertexData)
- void **_drawWireFrame** (mitkMesh *mesh)
- void **_drawSurface** (unsigned int faceNum, Vertex3f *vertexData)
- void **_drawSurface** (mitkMesh *mesh)

6.76.1 Detailed Description

mitkOoCSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

mitkOoCSurfaceRendererStandard is a standard renderer object for mitkSurfaceModel using general OpenGL.

6.76.2 Member Function Documentation

6.76.2.1 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:

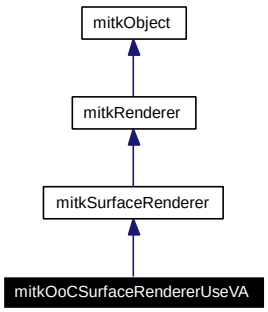
- mitkOoCSurfaceRendererStandard.h

6.77 mitkOoCSurfaceRendererUseVA Class Reference

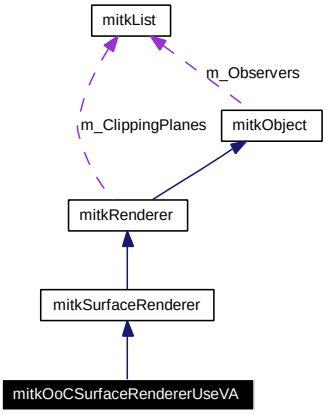
mitkOoCSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

`#include <mitkOoCSurfaceRendererUseVA.h>`

Inheritance diagram for mitkOoCSurfaceRendererUseVA:



Collaboration diagram for mitkOoCSurfaceRendererUseVA:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int **Render** (mitkView *view, mitkSurfaceModel *surf)

Protected Member Functions

- void **_setMaterial** (mitkSurfaceProperty *prop)
- bool **_buildBuffers** (mitkMesh *mesh)
- void **_clearBuffers** ()
- void **_drawPoints** (mitkMesh *mesh)
- void **_drawWireFrame** (mitkMesh *mesh)
- void **_drawSurface** (mitkMesh *mesh)

Protected Attributes

- Vertex3f * **m_VertBuf**
- unsigned int * **m_EdgeBuf**
- unsigned int **m_FaceBufSize**

6.77.1 Detailed Description

mitkOoCSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

mitkOoCSurfaceRendererUseVA is a renderer for mitkSurfaceModel using vertex array

6.77.2 Member Function Documentation

6.77.2.1 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:

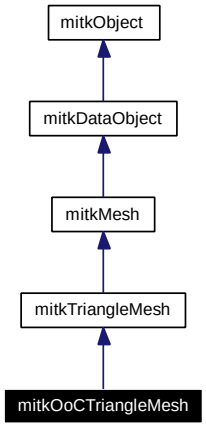
- mitkOoCSurfaceRendererUseVA.h

6.78 mitkOoCTriangleMesh Class Reference

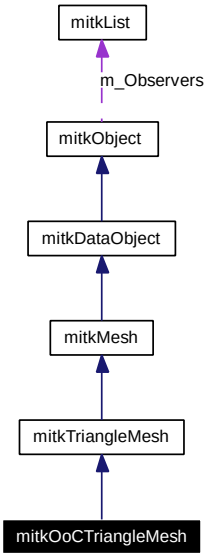
mitkOoCTriangleMesh - a concrete class for out-of-core triangle meshes

`#include <mitkOoCTriangleMesh.h>`

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 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 ()

Protected Member Functions

- virtual index_type **_addVertex** (Vertex3f &vert)
- virtual index_type **_addFace** (unsigned int num, index_type *verts)
- virtual Vertex3f const * **_getVertex** (index_type vertIdx)
- virtual index_type const * **_getFace** (index_type faceIdx)
- virtual Vertex3f * **_getVertexForWrite** (index_type vertIdx)
- virtual index_type * **_getFaceForWrite** (index_type faceIdx)
- virtual size_type **_getVertices** (index_type startIdx, size_type num, Vertex3f *verts)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, TriangleFace *faces)
- virtual size_type **_getTriangleFaces** (index_type startIdx, size_type num, Vertex3f *verts)
- Orientation **_testOrientation** (index_type faceIdx)
- void **_flushVertexTempBuffer** ()
- void **_flushFaceTempBuffer** ()
- void **_flushTempBuffer** ()

Protected Attributes

- VertexStorage * **m_VertStor**
- FaceStorage * **m_FaceStor**
- index_type **m_CurVertBlkIdx**
- index_type **m_CurFaceBlkIdx**
- Vertex3f * **m_CurVertBlk**
- TriangleFace * **m_CurFaceBlk**
- size_type **m_VertBlkSize**
- size_type **m_FaceBlkSize**
- index_type **m_VertIdxMask**
- unsigned int **m_VertIdxLen**
- index_type **m_VertBlkMask**
- index_type **m_FaceIdxMask**
- unsigned int **m_FaceIdxLen**
- index_type **m_FaceBlkMask**

6.78.1 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

6.78.2 Constructor & Destructor Documentation

6.78.2.1 mitkOoCTriangleMesh::mitkOoCTriangleMesh ()

Default constructor.

6.78.3 Member Function Documentation

6.78.3.1 virtual void mitkOoCTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

6.78.3.2 virtual unsigned 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.

6.78.3.3 unsigned int mitkOoCTriangleMesh::GetBufferedFaceBlockNum ()

Get the number of buffered face blocks in memory.

Returns:
Return the number of buffered face blocks.

6.78.3.4 unsigned int mitkOoCTriangleMesh::GetBufferedVertexBlockNum ()

Get the number of buffered vertex blocks in memory.

Returns:
Return the number of buffered vertex blocks.

6.78.3.5 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.

6.78.3.6 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.

6.78.3.7 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.

6.78.3.8 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.

6.78.3.9 virtual void mitkOoCTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkTriangleMesh.

6.78.3.10 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.

6.78.3.11 virtual void mitkOoCTriangleMesh::ReverseNormals () [virtual]

Reverse normals.

Reimplemented from mitkMesh.

6.78.3.12 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.

6.78.3.13 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.

6.78.3.14 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

6.78.3.15 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.

6.78.3.16 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.

6.78.3.17 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

6.78.3.18 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

6.78.3.19 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.

6.78.3.20 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.

6.78.3.21 virtual void mitkOoCTriangleMesh::ShallowCopy (mitkDataObject * *src*) [virtual]

Shallowcopy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkTriangleMesh.

6.78.3.22 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:

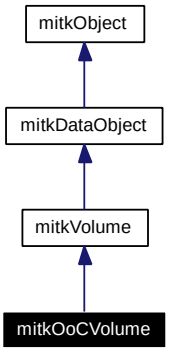
- mitkOoCTriangleMesh.h

6.79 mitkOoCVolume Class Reference

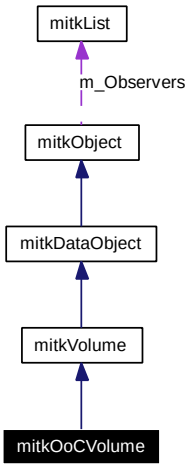
mitkOoCVolume - a concrete data object to represent a out-of-core multi-dimensional medical image dataset

`#include <mitkOoCVolume.h>`

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 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)

Protected Member Functions

- bool **_loadSlice** (int sliceIdx)
- bool **_writeSliceBack** (int sliceIdx)
- void **_clearDiskCache** ()
- void **_flush** ()
- SliceBufferBlockNode * **_getFreeBufferBlock** ()
- SliceBufferBlockNode * **_getSliceBufferBlock** (int sliceIdx)
- bool **_changeBufferSize** (unsigned int bufSliceNum)

Protected Attributes

- mitkString * **m_DiskPath**
- mitkString * **m_NamePrefix**
- size_type **m_MemBufSize**
- unsigned int **m_BufferedSliceNum**
- SliceHeap * **m_BufferedSliceHeap**
- SliceHeapNode * **m_SliceHeapNodes**
- SliceBufferFreeQueue * **m_FreeQueue**
- SliceBufferBlockNode * **m_BufferBlockNodes**
- void * **m_SliceBuffer**
- bool **m_KeepCacheFiles**

6.79.1 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.

6.79.2 Constructor & Destructor Documentation

6.79.2.1 mitkOoCVolume::mitkOoCVolume ()

Default constructor.

6.79.3 Member Function Documentation

6.79.3.1 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.

6.79.3.2 virtual void mitkOoCVolume::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Reimplemented from mitkVolume.

6.79.3.3 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.

6.79.3.4 virtual unsigned 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.

6.79.3.5 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.

6.79.3.6 unsigned int mitkOoCVolume::GetBufferedSliceNum () [inline]

Get the number of buffered slice in memory.

Returns:

Return the number of buffered slice.

6.79.3.7 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.

6.79.3.8 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.

6.79.3.9 virtual int mitkOoCVolume::GetDataObjectType () const [inline, virtual]

Return the data object type.

Returns:
Always return MITK_OUT_OF_CORE_VOLUME

Reimplemented from mitkVolume.

6.79.3.10 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

6.79.3.11 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.

6.79.3.12 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.

6.79.3.13 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.

6.79.3.14 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.

6.79.3.15 virtual void mitkOoCVolume::Initialize () [virtual]

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

Reimplemented from mitkVolume.

6.79.3.16 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.

6.79.3.17 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.

6.79.3.18 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.

6.79.3.19 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.

6.79.3.20 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.

6.79.3.21 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.

6.79.3.22 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

6.79.3.23 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.

6.79.3.24 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

6.79.3.25 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

6.79.3.26 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.

6.79.3.27 virtual void mitkOoCVolume::ShallowCopy (mitkDataObject * *src*) `[virtual]`

Warning:
Don't call this function directly.

Reimplemented from mitkVolume.

6.79.3.28 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.

6.79.3.29 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.

6.79.3.30 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:

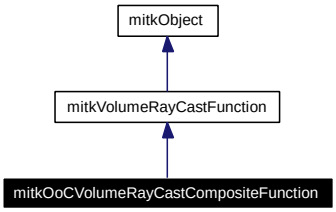
- mitkOoCVolume.h

6.80 mitkOoCVolumeRayCastCompositeFunction Class Reference

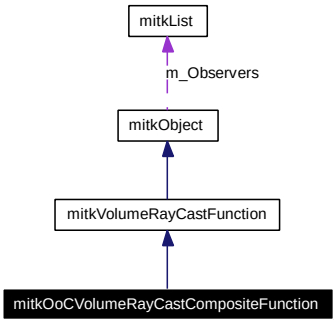
mitkOoCVolumeRayCastCompositeFunction - a concrete ray cast function for compositing

`#include <mitkOoCVolumeRayCastCompositeFunction.h>`

Inheritance diagram for mitkOoCVolumeRayCastCompositeFunction:



Collaboration diagram for mitkOoCVolumeRayCastCompositeFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRayCastCompositeFunction ()
- virtual void CastRay (mitkRay *rayInfo)

6.80.1 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.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 mitkOoCVolumeRayCastCompositeFunction::mitkOoCVolumeRayCastComposite-Function ()

Default constructor.

6.80.3 Member Function Documentation

6.80.3.1 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.

6.80.3.2 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:

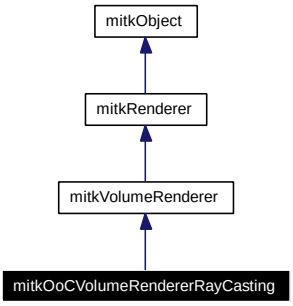
- mitkOoCVolumeRayCastCompositeFunction.h

6.81 mitkOoCVolumeRendererRayCasting Class Reference

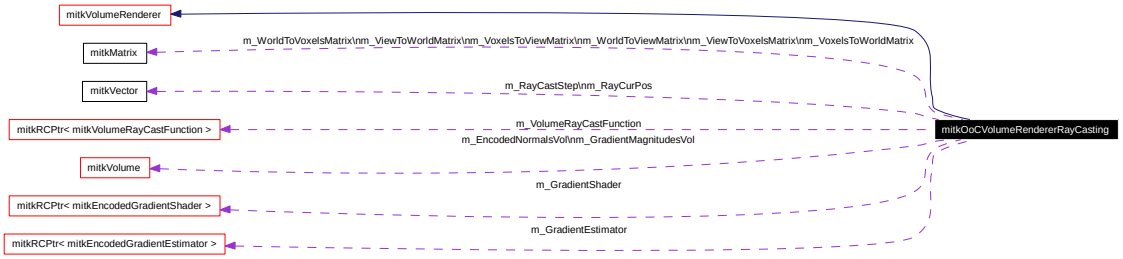
mitkOoCVolumeRendererRayCasting - a concrete volume renderer for rendering an out-of-core volume

`#include <mitkOoCVolumeRendererRayCasting.h>`

Inheritance diagram for mitkOoCVolumeRendererRayCasting:



Collaboration diagram for mitkOoCVolumeRendererRayCasting:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRendererRayCasting ()
- virtual int Render (mitkView *view, mitkVolumeModel *vol)

Protected Member Functions

- void **_deleteCastingRays** ()
- void **_clearCastingRays** ()
- int **_newCastingRays** ()
- int **_initCastingRays** (mitkView *view, mitkVolumeModel *vol)
- int **_computeClippingPlanes** (int planeCount, mitkVector *planeEqus)
- int **_computeRowBounds** (mitkView *view, mitkVolumeModel *vol)
- int **_clipRayAgainstVolume** (mitkVector &rayStart, mitkVector &rayEnd, float bounds[6])
- int **_clipRayAgainstClippingPlanes** (mitkVector &rayStart, mitkVector &rayEnd, int planeCount, mitkVector *planeEqus)
- int **_updateShadingTables** (mitkView *view, mitkVolumeModel *vol)
- void **_renderTexture** (mitkView *view, mitkVolumeModel *vol)
- float **_getZBufferValue** (int x, int y)

Protected Attributes

- mitkRCPtr< mitkVolumeRayCastFunction > **m_VolumeRayCastFunction**
- mitkRCPtr< mitkEncodedGradientEstimator > **m_GradientEstimator**
- mitkRCPtr< mitkEncodedGradientShader > **m_GradientShader**
- mitkRay * **m_Ray**
- mitkVector * **m_RayCurPos**
- mitkVector * **m_RayCastStep**
- int * **m_RayStepsLeft**
- int * **m_RayStatus**
- mitkMatrix * **m_WorldToViewMatrix**
- mitkMatrix * **m_ViewToWorldMatrix**
- mitkMatrix * **m_ViewToVoxelsMatrix**
- mitkMatrix * **m_VoxelsToViewMatrix**
- mitkMatrix * **m_WorldToVoxelsMatrix**
- mitkMatrix * **m_VoxelsToWorldMatrix**
- int **m_ImageViewportSize** [2]
- int **m_ImageMemorySize** [2]
- int **m_ImageInUseSize** [2]
- int **m_ImageOrigin** [2]
- unsigned char * **m_Image**
- float * **m_AccumColors**
- int * **m_RowBounds**
- int * **m_OldRowBounds**
- float **m_MinimumViewDistance**
- float **m_SampleDistance**
- float * **m_ZBuffer**
- int **m_ZBufferSize** [2]
- int **m_ZBufferOrigin** [2]
- float **m_ImageSampleDistance**
- float **m_MinimumImageSampleDistance**
- float **m_MaximumImageSampleDistance**
- int **m_SlicePtrNum**
- void const ** **m_Slices**
- unsigned short const ** **m_EncodedNormalSlices**
- unsigned char const ** **m_GradientMagnitudeSlices**
- mitkVolume * **m_EncodedNormalsVol**
- mitkVolume * **m_GradientMagnitudesVol**
- bool **m_AutoAdjustSampleDistances**
- bool **m_IntermixIntersectingGeometry**
- bool **m_HasForwardRay**
- bool **m_HasReverseRay**

6.81.1 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.

6.81.2 Constructor & Destructor Documentation

6.81.2.1 mitkOoCVolumeRendererRayCasting::mitkOoCVolumeRendererRayCasting ()

Default constructor.

6.81.3 Member Function Documentation

6.81.3.1 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.

6.81.3.2 virtual int mitkOoCVolumeRendererRayCasting::Render (mitkView * *view*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

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

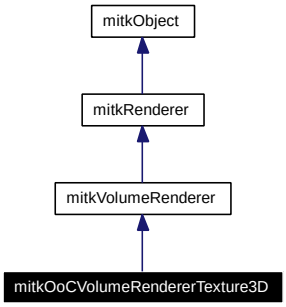
- mitkOoCVolumeRendererRayCasting.h

6.82 mitkOoCVolumeRendererTexture3D Class Reference

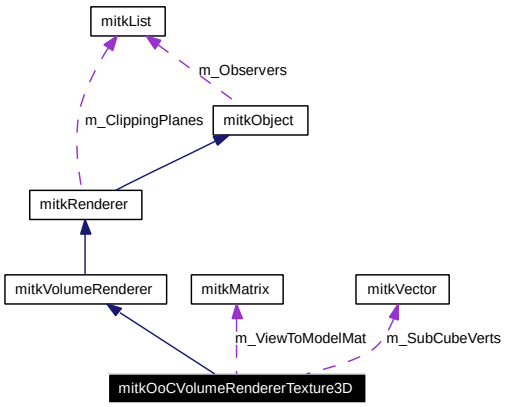
mitkOoCVolumeRendererTexture3D - a concrete volume renderer for rendering an out-of-core volume

`#include <mitkOoCVolumeRendererTexture3D.h>`

Inheritance diagram for mitkOoCVolumeRendererTexture3D:



Collaboration diagram for mitkOoCVolumeRendererTexture3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkOoCVolumeRendererTexture3D ()
- virtual int Render (mitkView *view, mitkVolumeModel *vol)
- void SetSampleDistance (float sd)
- void SetSubVolumeSize (int w, int h, int d)

Protected Member Functions

- bool **_buildDummyTexture** ()
- bool **_applyTFs** (mitkVolumeModel *vol)
- bool **_splitVolume** (mitkVolume *data)
- void **_cutEmptyBlocks** (mitkVolumeProperty *prop)
- void **_generateTFs** (mitkVolumeProperty *prop)
- void **_buildTexture** (int width, int height, int depth, mitkVolumeModel *vol)

- void **_generatePolygons** ()
- void **_renderTexture** ()
- void **_deleteTexture** ()
- void **_clearTexPolygons** ()
- void **_clearTexCache** ()
- void **_renderFromVert0** (mitkVolumeModel *vol)
- void **_renderFromVert1** (mitkVolumeModel *vol)
- void **_renderFromVert2** (mitkVolumeModel *vol)
- void **_renderFromVert3** (mitkVolumeModel *vol)
- void **_renderFromVert4** (mitkVolumeModel *vol)
- void **_renderFromVert5** (mitkVolumeModel *vol)
- void **_renderFromVert6** (mitkVolumeModel *vol)
- void **_renderFromVert7** (mitkVolumeModel *vol)

Protected Attributes

- PFNGLTEXIMAGE3DPROC **glTexImage3D**
- PFNGLTEXSUBIMAGE3DPROC **glTexSubImage3D**
- GLuint **m_TexID**
- int **m_DummyTexWidth**
- int **m_DummyTexHeight**
- int **m_DummyTexDepth**
- int **m_SubVolWidth**
- int **m_SubVolHeight**
- int **m_SubVolDepth**
- float **m_SampleDistance**
- float **m_SubVolBnd** [6]
- float **m_TexBnd** [3]
- float **m_Spacings** [3]
- unsigned int **m_PolyNum**
- unsigned int **m_PolyArraySize**
- TexPolygon * **m_Polygons**
- void * **m_SrcBuf**
- unsigned int **m_SrcBufSize**
- void * **m_TexBuf**
- unsigned int **m_TexBufSize**
- unsigned char * **m_BlockFlags**
- mitkVector * **m_SubCubeVerts**
- mitkMatrix * **m_ViewToModelMat**
- mitkStringList * **m_CacheFileNames**
- int * **m_BlockMinVals**
- int * **m_BlockMaxVals**
- unsigned char * **m_SOTF**
- unsigned char * **m_SCTFR**
- unsigned char * **m_SCTFG**
- unsigned char * **m_SCTFB**
- bool **m_IsTex3DSupported**
- bool **m_FirstRendering**

6.82.1 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. (Currently, it does not support shading.)

Note:
To use this class, your graphics card should support 3D texture.

6.82.2 Constructor & Destructor Documentation

6.82.2.1 mitkOoCVolumeRendererTexture3D::mitkOoCVolumeRendererTexture3D ()

Default constructor.

6.82.3 Member Function Documentation

6.82.3.1 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.

6.82.3.2 virtual int mitkOoCVolumeRendererTexture3D::Render (mitkView * *view*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

6.82.3.3 void mitkOoCVolumeRendererTexture3D::SetSampleDistance (float *sd*) [inline]

Set sample distance.

Parameters:
sd sample distance

6.82.3.4 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:

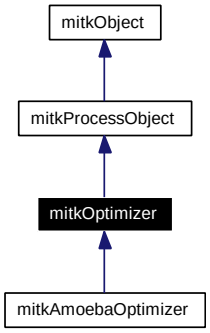
- mitkOoCVolumeRendererTexture3D.h

6.83 mitkOptimizer Class Reference

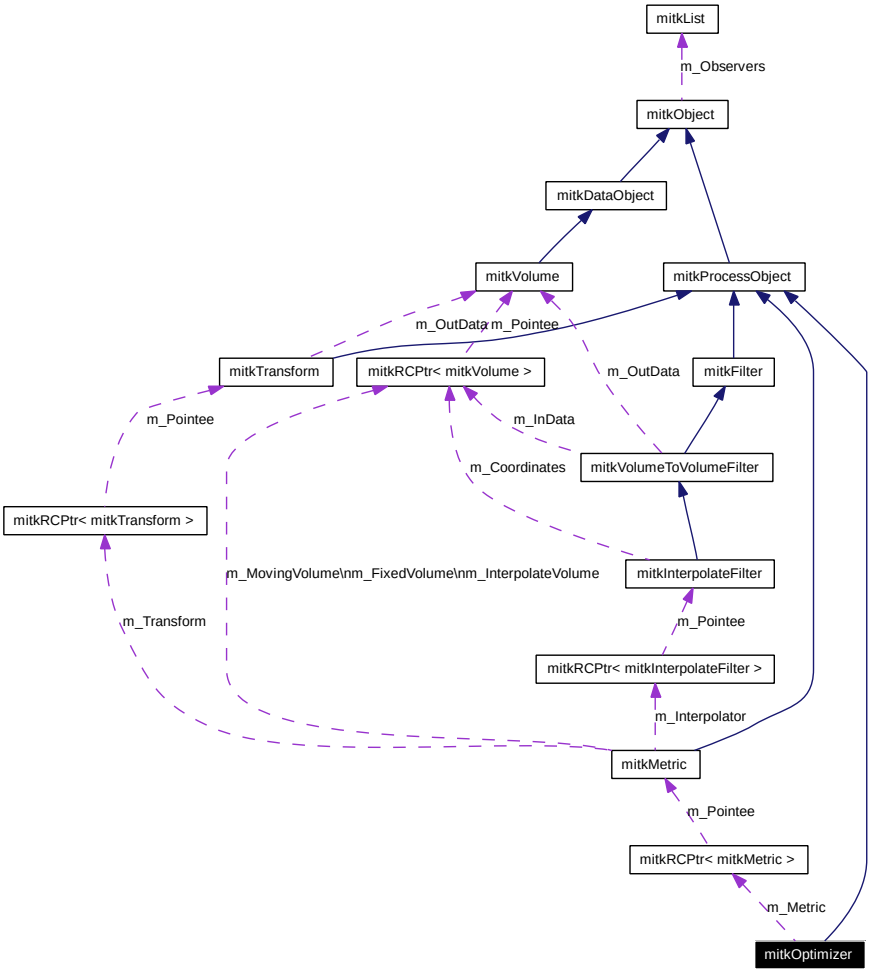
mitkOptimizer - an abstract class specifies interface for an optimization method

`#include <mitkOptimizer.h>`

Inheritance diagram for mitkOptimizer:



Collaboration diagram for mitkOptimizer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInitialParameters (vector< float > *param)
- vector< float > * GetInitialParameters ()
- void SetScales (vector< float > *scales)
- vector< float > * GetScales ()
- vector< float > * GetLastParameters ()
- void SetMetric (mitkMetric *metric)
- mitkMetric * GetMetric ()
- void SetTolerance (float tol)
- void SetMaxIterations (int n)
- void Update ()
- float GetExtremum ()

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- mitkRCPtr< mitkMetric > **m_Metric**
- vector< float > * **m_InitialParameters**
- vector< float > * **m_LastParameters**
- vector< float > * **m_Scales**
- float **m_Tolerance**
- float **m_Extremum**
- int **m_MaxIterations**
- int **m_Iterations**
- int **m_NumberOfParameters**

6.83.1 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.

6.83.2 Member Function Documentation

6.83.2.1 float mitkOptimizer::GetExtremum () [inline]

Get the extremum of optimization.

Returns:
Return the extremum.

6.83.2.2 vector<float>* mitkOptimizer::GetInitialParameters ()

Get the position to initialize the optimization.

Returns:
Return the vector pointer to the initial position for optimization.

6.83.2.3 vector<float>* mitkOptimizer::GetLastParameters ()

Get the last position of optimization.

Returns:
Return the vector pointer to the last position.

6.83.2.4 mitkMetric* mitkOptimizer::GetMetric ()

Get the Metric

Returns:
Return the pointer to the Metric.

6.83.2.5 vector<float>* mitkOptimizer::GetScales ()

Get current parameters scaling.

6.83.2.6 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.

6.83.2.7 void mitkOptimizer::SetInitialParameters (vector< float > * *param*) [inline]

Set the position to initialize the optimization.

Parameters:
param The vector pointer to the initial position for optimization.

6.83.2.8 void mitkOptimizer::SetMaxIterations (int *n*) [inline]

Set the maximum number of iterations.

Parameters:
n The maximum number of iterations.

6.83.2.9 void mitkOptimizer::SetMetric (mitkMetric * *metric*) [inline]

Set the similarity function

Parameters:
metric The pointer to the Metric.

6.83.2.10 void mitkOptimizer::SetScales (vector< float > * *scales*)

Set current parameters scaling.

Parameters:
scales The vector pointer to the scales for every elements of parameters.

6.83.2.11 void mitkOptimizer::SetTolerance (float *tol*) [inline]

Set optimization tolerance.

Parameters:
tol The tolerance as termination criteria.

6.83.2.12 void mitkOptimizer::Update ()

Initialize the Optimizer and make sure the components are present.

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

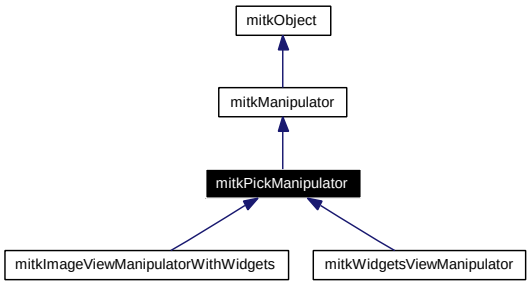
- mitkOptimizer.h

6.84 mitkPickManipulator Class Reference

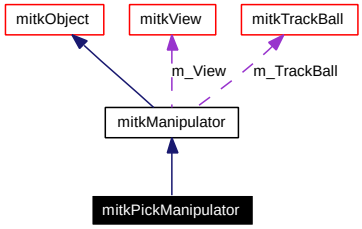
mitkPickManipulator - a manipulator with picking function enabled

`#include <mitkPickManipulator.h>`

Inheritance diagram for mitkPickManipulator:



Collaboration diagram for mitkPickManipulator:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- **mitkPickManipulator** (mitkView *view)
- void ClearPickResult ()
- mitkWidgetModel * GetPickedObject ()

Protected Member Functions

- bool **_pick** (int xPos, int yPos)
- void **_selectRender** ()
- bool **_processHits** (GLint hits)
- void **_init** ()

Protected Attributes

- WidgetNames **m_PickedWidgets**
- GLuint **m_PickBuffer** [MITK_PICK_BUFFER_SIZE]
- GLdouble **m_PickBox**
- bool **m_EnableDepthTest**

6.84.1 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.

6.84.2 Member Function Documentation

6.84.2.1 void mitkPickManipulator::ClearPickResult ()

Clear picked results.

6.84.2.2 mitkWidgetModel* mitkPickManipulator::GetPickedObject () [inline]

Get current picked widget.

Returns:
Return the pointer to a mitkWidgetModel currently picked.

6.84.2.3 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:

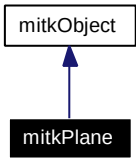
- mitkPickManipulator.h

6.85 mitkPlane Class Reference

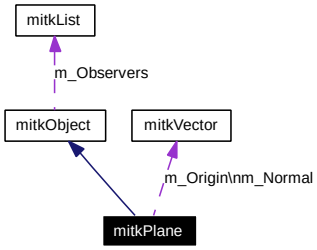
mitkPlane - a class to represent a plane

`#include <mitkPlane.h>`

Inheritance diagram for mitkPlane:



Collaboration diagram for mitkPlane:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- float **Evaluate** (float x[3])
- float **Evaluate** (float x, float y, float z)
- void **SetNormal** (float nx, float ny, float nz)
- void **SetNormal** (float normal[3])
- void **GetNormal** (float &nx, float &ny, float &nz)
- const mitkVector * **GetNormal** (void)
- void **SetOrigin** (float x, float y, float z)
- void **SetOrigin** (float origin[3])
- void **GetOrigin** (float &x, float &y, float &z)
- const mitkVector * **GetOrigin** (void)
- void **TranslateTowardNormal** (float distance)
- void **ProjectPoint** (float x[3], float xproj[3])
- void **ProjectPoint** (const mitkVector &srcPoint, mitkVector &projPoint)
- float **DistanceToPlane** (float x[3])
- float **DistanceToPlane** (const mitkVector &srcPoint)
- int **IntersectWithLine** (float p1[3], float p2[3], float &t, float x[3])
- int **IntersectWithLine** (const mitkVector &point1, const mitkVector &point2, float &t, mitkVector &intersectedPoint)

Protected Attributes

- mitkVector * **m_Normal**
- mitkVector * **m_Origin**

6.85.1 Detailed Description

mitkPlane - a class to represent a plane

mitkPlane is a class to represenet a plane

6.85.2 Member Function Documentation

6.85.2.1 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:

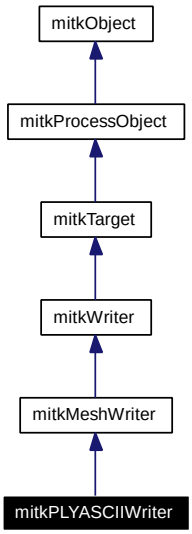
- mitkPlane.h

6.86 mitkPLYASCIIWriter Class Reference

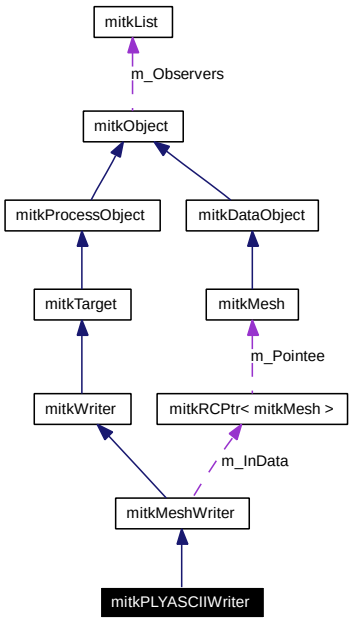
mitkPLYASCIIWriter - a concrete writer for writing a mesh to PLY files with ASCII format

`#include <mitkPLYASCIIWriter.h>`

Inheritance diagram for mitkPLYASCIIWriter:



Collaboration diagram for mitkPLYASCIIWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual bool **Execute** ()

6.86.1 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();
```

6.86.2 Member Function Documentation

6.86.2.1 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:

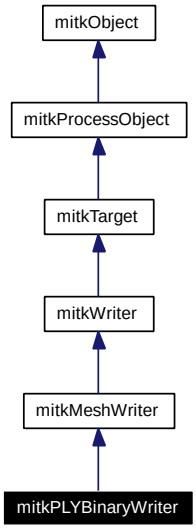
- mitkPLYASCIIWriter.h

6.87 mitkPLYBinaryWriter Class Reference

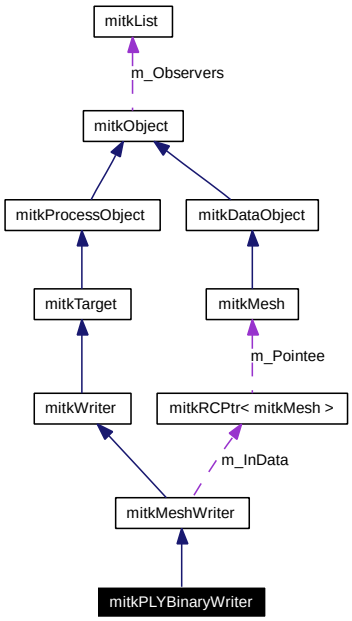
mitkPLYBinaryWriter - a concrete writer for writing a mesh to PLY files with binary format

`#include <mitkPLYBinaryWriter.h>`

Inheritance diagram for mitkPLYBinaryWriter:



Collaboration diagram for mitkPLYBinaryWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void **SetBigEndian** (bool bigEndian=true)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- bool **m_IsBigEndian**

6.87.1 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();
```

6.87.2 Member Function Documentation

6.87.2.1 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:

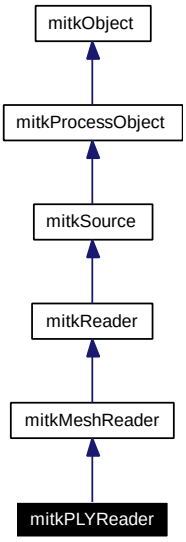
- mitkPLYBinaryWriter.h

6.88 mitkPLYReader Class Reference

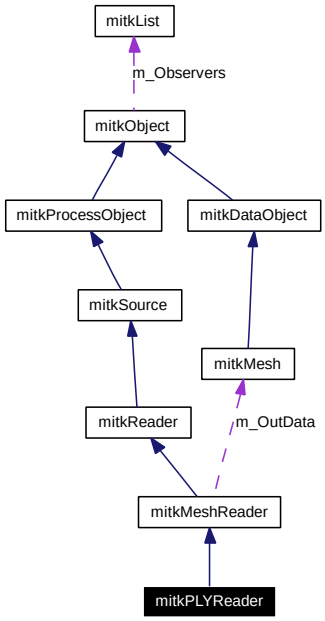
mitkPLYReader - a concrete reader for reading a PLY file

`#include <mitkPLYReader.h>`

Inheritance diagram for mitkPLYReader:



Collaboration diagram for mitkPLYReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)

- void **SetVertexReadBufferSize** (unsigned int n)
- void **SetFaceReadBufferSize** (unsigned int n)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- unsigned int **m_VertReadBufSize**
- unsigned int **m_FaceReadBufSize**

6.88.1 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
}
```

6.88.2 Member Function Documentation

6.88.2.1 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:

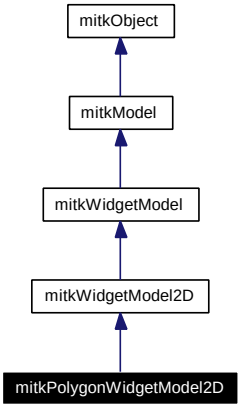
- mitkPLYReader.h

6.89 mitkPolygonWidgetModel2D Class Reference

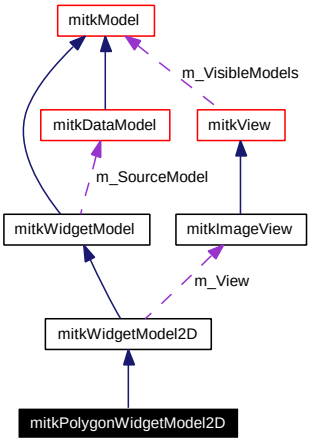
mitkPolygonWidgetModel2D - a 2D widget for displaying an arbitrary polygon in an image view

`#include <mitkPolygonWidgetModel2D.h>`

Inheritance diagram for mitkPolygonWidgetModel2D:



Collaboration diagram for mitkPolygonWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view)
- 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 Types

- enum { **MAXNUM_OF_POINTS** = 1 << 20 }
- enum { **unknown**, **poly**, **startpoint**, **startline** = startpoint + MAXNUM_OF_POINTS }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()

Protected Attributes

- float **m_DotColor** [4]
- float **m_LineColor** [4]
- float **m_PickedDotColor** [4]
- float **m_PickedLineColor** [4]
- float **m_DotSize**
- float **m_Left**
- float **m_Right**
- float **m_Top**
- float **m_Bottom**
- PointList * **m_Points**
- bool **m_Done**
- string **m_UnitName**

6.89.1 Detailed Description

mitkPolygonWidgetModel2D - a 2D widget for displaying an arbitrary polygon in an image view

mitkPolygonWidgetModel2D a 2D widget for displaying an arbitrary polygon in an image view. 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 view (e.g. mitkImageView), and in other conditions the display could be improper.

6.89.2 Member Function Documentation

6.89.2.1 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.

6.89.2.2 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.

6.89.2.3 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.

6.89.2.4 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

6.89.2.5 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

6.89.2.6 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.89.2.7 float mitkPolygonWidgetModel2D::GetArea ()

Get area of this polygon.

Returns:

Return the area of this polygon.

6.89.2.8 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.

6.89.2.9 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.

6.89.2.10 float mitkPolygonWidgetModel2D::GetPerimeter ()

Get perimeter of this polygon.

Returns:

Return the perimeter of this polygon.

6.89.2.11 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.

6.89.2.12 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.

6.89.2.13 int mitkPolygonWidgetModel2D::GetPointsNumber ()

Get the number of points.

Returns:

Return the number of points.

6.89.2.14 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.

6.89.2.15 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

6.89.2.16 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

6.89.2.17 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

6.89.2.18 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.89.2.19 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.

6.89.2.20 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.

6.89.2.21 virtual void mitkPolygonWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.89.2.22 virtual int mitkPolygonWidgetModel2D::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.89.2.23 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:

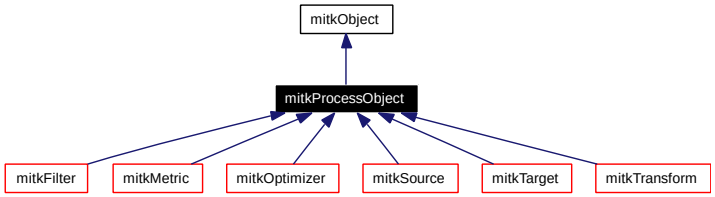
- mitkPolygonWidgetModel2D.h

6.90 mitkProcessObject Class Reference

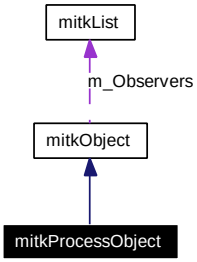
mitkProcessObject - abstract base class for source, filter(algorithm) and mapper

`#include <mitkProcessObject.h>`

Inheritance diagram for mitkProcessObject:



Collaboration diagram for mitkProcessObject:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- bool **Run** ()
- void **SetStartMethod** (void(∗f)(void ∗))
- void **SetEndMethod** (void(∗f)(void ∗))
- void **SetProgressMethod** (void(∗f)(void ∗))
- unsigned long GetProgressRate ()
- void SetProgressRateMax (unsigned long m)
- void **SetProgressRate** (unsigned int rate)
- unsigned int **GetProgressRateMax** ()
- void **UpdateObservers** ()

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- unsigned long **m_ProgressRate**
- unsigned long **m_ProgressRateMax**

6.90.1 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.

6.90.2 Member Function Documentation

6.90.2.1 unsigned long mitkProcessObject::GetProgressRate () [inline]

Get current rate of process of the running process.

Returns:
Return current rate of process.

6.90.2.2 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 mitkAffineTransform, mitkAmoebaOptimizer, mitkBinaryFilter, mitkBinMarchingCubes, mitkBMPReader, mitkBMPWriter, mitkCacheVolumeReader, mitkCacheVolumeWriter, mitkDICOMInfoReader, mitkDICOMReader, mitkDICOMWriter, mitkDiffusionFilter, mitkFilter, mitkImageView, mitkInfoReader, mitkInterpolateFilter, mitkJPEGReader, mitkJPEGWriter, mitkLiveWireImageFilter, mitkMarchingCubes, mitkMeanSquaresMetric, mitkMeshReader, mitkMeshToMeshFilter, mitkMeshWriter, mitkMetric, mitkNearestNeighborInterpolateFilter, mitkOptimizer, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkPLYReader, mitkQEMSimplification, mitkRawFilesReader, mitkRawFilesWriter, mitkRawReader, mitkRawWriter, mitkReader, mitkRegionGrowImageFilter, mitkRegistrationFilter, mitkRGBToGrayFilter, mitkRigidTransform, mitkSeedFillFilter, mitkSobelEdgeDetectFilter, mitkSource, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkTarget, mitkThresholdSegmentationFilter, mitkTIFFReader, mitkTIFFWriter, mitkTransform, mitkTriangleMeshSimplification, mitkView, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeReader, mitkVolumeResizeFilter, mitkVolumeResliceFilter, mitkVolumeToMeshFilter, mitkVolumeToVolumeFilter, mitkVolumeWriter, and mitkWriter.

6.90.2.3 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:

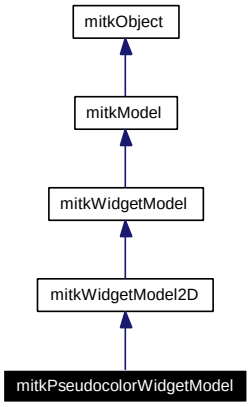
- mitkProcessObject.h

6.91 mitkPseudocolorWidgetModel Class Reference

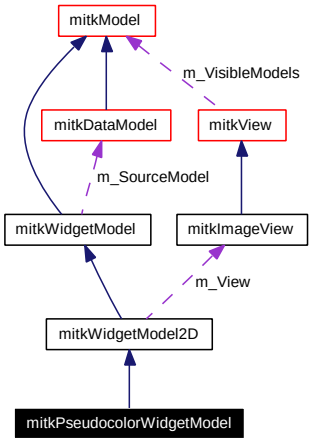
mitkPseudocolorWidgetModel - a 2D widget for displaying pseudocolor image

`#include <mitkPseudocolorWidgetModel.h>`

Inheritance diagram for mitkPseudocolorWidgetModel:



Collaboration diagram for mitkPseudocolorWidgetModel:



Public Types

- enum { **COLOR_TABLE_SIZE** = 256 }

Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view)
- 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 Types

- enum {
  unknown, **image**, **hsline**, **heline**,
  vsline, **veline**, **ssdot**, **esdot**,
  eedot, **sedot** }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()
- void **_drawColorRect** ()
- void **_resetRect** ()

Protected Attributes

- float **m_DotColor** [4]
- float **m_LineColor** [4]
- float **m_PickedDotColor** [4]
- float **m_PickedLineColor** [4]
- float **m_DotSize**
- unsigned char **m_ColorTable** [COLOR_TABLE_SIZE][3]
- unsigned char * **m_PixelBuffer**
- unsigned int **m_BufferSize**
- int **m_StartPoint** [2]
- int **m_EndPoint** [2]
- int **m_Left**

- int **m_Bottom**
- int **m_Right**
- int **m_Top**
- bool **m_StartPointSet**
- bool **m_EndPointSet**

6.91.1 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 view (e.g. mitkImageView), and in other conditions the display could be improper.

6.91.2 Member Function Documentation

6.91.2.1 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.

6.91.2.2 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.

6.91.2.3 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.

6.91.2.4 int mitkPseudocolorWidgetModel::GetBottom () [inline]

Get the bottom of the pseudocolor rectangle.

Returns:

Return the bottom of the pseudocolor rectangle.

6.91.2.5 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.

6.91.2.6 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

6.91.2.7 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

6.91.2.8 int mitkPseudocolorWidgetModel::GetHeight () [inline]

Get the height of the pseudocolor rectangle.

Returns:
Return the height of the pseudocolor rectangle.

6.91.2.9 int mitkPseudocolorWidgetModel::GetLeft () [inline]

Get the left of the pseudocolor rectangle.

Returns:
Return the left of the pseudocolor rectangle.

6.91.2.10 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.

6.91.2.11 int mitkPseudocolorWidgetModel::GetRight () [inline]

Get the right of the pseudocolor rectangle.

Returns:
Return the right of the pseudocolor rectangle.

6.91.2.12 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

6.91.2.13 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

6.91.2.14 int mitkPseudocolorWidgetModel::GetTop () [inline]

Get the top of the pseudocolor rectangle.

Returns:
Return the left of the pseudocolor rectangle.

6.91.2.15 int mitkPseudocolorWidgetModel::GetWidth () [inline]

Get the width of the pseudocolor rectangle.

Returns:
Return the width of the pseudocolor rectangle.

6.91.2.16 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 view).

6.91.2.17 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.

6.91.2.18 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.

6.91.2.19 virtual void mitkPseudocolorWidgetModel::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.91.2.20 virtual int mitkPseudocolorWidgetModel::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:

view the pointer of an mitkView in which this model will be shown

Returns:

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

Reimplemented from mitkModel.

6.91.2.21 void mitkPseudocolorWidgetModel::ResetColorTable ()

Reset the color table to the default values.

6.91.2.22 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).

6.91.2.23 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

6.91.2.24 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

6.91.2.25 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:

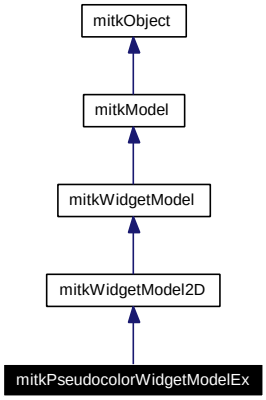
- mitkPseudocolorWidgetModel.h

6.92 mitkPseudocolorWidgetModelEx Class Reference

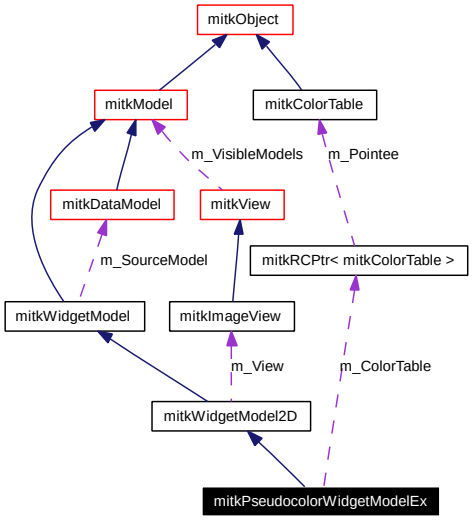
mitkPseudocolorWidgetModelEx - a 2D widget for displaying pseudocolor image

`#include <mitkPseudocolorWidgetModelEx.h>`

Inheritance diagram for mitkPseudocolorWidgetModelEx:



Collaboration diagram for mitkPseudocolorWidgetModelEx:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view)
- 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 Types

- enum {
  unknown, **rect**, **hsline**, **heline**,
  vsline, **veline**, **ssdot**, **esdot**,
  eedot, **sedot** }

Protected Member Functions

- virtual float * **_getBounds** ()
- bool **_mapColor** ()
- 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)
- void **_init** ()
- void **_updateRectParams** ()
- void **_computeImageCoordinates** ()

Protected Attributes

- mitkRCPtr< mitkColorTable > **m_ColorTable**
- float **m_DotColor** [4]
- float **m_LineColor** [4]
- float **m_PickedDotColor** [4]
- float **m_PickedLineColor** [4]
- float **m_DotSize**
- int **m_RectLeft**
- int **m_RectBottom**
- int **m_RectWidth**
- int **m_RectHeight**
- unsigned int **m_RectBufSize**
- unsigned char * **m_RectBuffer**
- int **m_StartPoint** [2]

- int **m_EndPoint** [2]
- float **m_ObjStartPoint** [2]
- float **m_ObjEndPoint** [2]
- mitkImageModel::ViewMode **m_CurViewMode**
- int **m_CurSliceIdx**
- void const * **m_CurSrcSlice**
- void * **m_SrcBuf**
- bool **m_StartPointSet**
- bool **m_EndPointSet**
- bool **m_RectChanged**

6.92.1 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 view (e.g. mitkImageView), 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

6.92.2 Member Function Documentation

6.92.2.1 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.

6.92.2.2 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.

6.92.2.3 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.

6.92.2.4 mitkColorTable* mitkPseudocolorWidgetModelEx::GetColorTable ()

Get color table.

Returns:

Return the pointer to the mitkColorTable object.

6.92.2.5 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

6.92.2.6 int mitkPseudocolorWidgetModelEx::GetRectHeightInImage () [inline]

Get the height (in image space) of the pseudo color window.

Returns:

Return the height value in number of pixels.

6.92.2.7 int mitkPseudocolorWidgetModelEx::GetRectWidthInImage () [inline]

Get the width (in image space) of the pseudo color window.

Returns:

Return the width value in number of pixels.

6.92.2.8 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.

6.92.2.9 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 view).

6.92.2.10 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.

6.92.2.11 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.

6.92.2.12 virtual void mitkPseudocolorWidgetModelEx::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.92.2.13 virtual int mitkPseudocolorWidgetModelEx::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.92.2.14 void mitkPseudocolorWidgetModelEx::SetColorTable (mitkColorTable * *ct*) [inline]

Set color table.

Parameters:
ct the pointer to a mitkColorTable object

6.92.2.15 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.92.2.16 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.)

6.92.2.17 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.92.2.18 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.)

6.92.2.19 void mitkPseudocolorWidgetModelEx::SetRectChanged () `[inline]`

Set that the pseudo-color region is changed.

6.92.2.20 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.

6.92.2.21 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.92.2.22 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.)

6.92.2.23 virtual void mitkPseudocolorWidgetModelEx::Update () `[virtual]`

Update the parameters of the widget.

Reimplemented from mitkWidgetModel2D.

6.92.2.24 bool mitkPseudocolorWidgetModelEx::UpdatePseudocolorRect ()

Update the pseudo-color region.

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

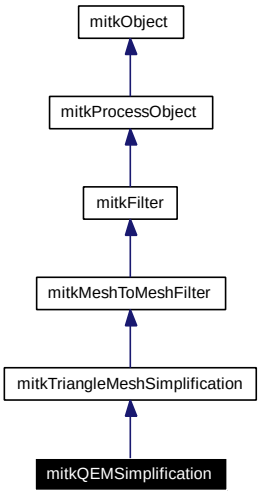
- mitkPseudocolorWidgetModelEx.h

6.93 mitkQEMSimplification Class Reference

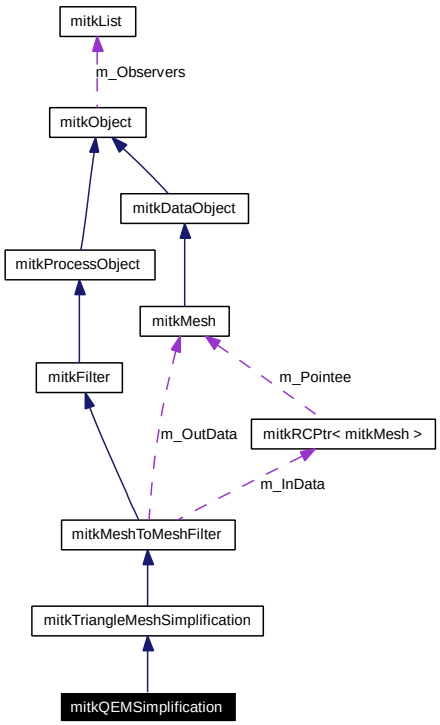
mitkQEMSimplification - an implementation for the simplification algorithm using Quadric Error Metrics (QEM)

`#include <mitkQEMSimplification.h>`

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 ()

Protected Member Functions

- virtual bool **_simplificationProcess** (mitkHETriangleMesh *mesh)

Protected Attributes

- double **m_MeshingPenalty**
- double **m_CompactnessRatio**
- double **m_LocalValidityThreshold**
- size_type **m_VertexDegreeLimit**

6.93.1 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.

6.93.2 Member Function Documentation

6.93.2.1 double mitkQEMSimplification::GetCompactnessRatio () [inline]

Get the compactness ratio for triangle compactness check.

Returns:
Return the compactness ratio.

6.93.2.2 double mitkQEMSimplification::GetLocalValidityThreshold () [inline]

Get the local validity threshold for local validity check.

Returns:
Return the local validity threshold.

6.93.2.3 double mitkQEMSimplification::GetMeshingPenalty () [inline]

Get meshing penalty.

Returns:
Return the meshing penalty.

6.93.2.4 size_type mitkQEMSimplification::GetVertexDegreeLimit () [inline]

Set the limit of the vertex degree.

Returns:
Return the limit of the vertex degree.

6.93.2.5 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.

6.93.2.6 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

6.93.2.7 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

6.93.2.8 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)

6.93.2.9 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

6.94 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)

6.94.1 Detailed Description

mitkQuaternion - it's used to implement 3d rotation

mitkQuaternion is used to implement 3d rotation

6.94.2 Constructor & Destructor Documentation

6.94.2.1 mitkQuaternion::mitkQuaternion ()

Default constructor. Construct a quaternion with zero rotation.

6.94.2.2 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.

6.94.2.3 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

6.94.2.4 mitkQuaternion::mitkQuaternion (const mitkQuaternion & *q*) [inline]

Copy constructor.

6.94.3 Member Function Documentation

6.94.3.1 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

6.94.3.2 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

6.94.3.3 void mitkQuaternion::BuildMatrix (mitkMatrix & *mat*) const

Get the 4x4 rotation matrix representation of this quaternion.

Parameters:

mat return the rotation matrix.

6.94.3.4 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

6.94.3.5 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

6.94.3.6 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

6.94.3.7 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

6.94.3.8 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

6.94.3.9 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

6.94.3.10 void mitkQuaternion::GetValues (float & *x*, float & *y*, float & *z*, float & *w*) const

Get the values of this quaternion.

6.94.3.11 void mitkQuaternion::Identity (void)

Set this quaternion to the identity quaternion.

6.94.3.12 void mitkQuaternion::Normalize (void)

Normalize this quaternion.

6.94.3.13 mitkQuaternion& mitkQuaternion::operator= (const mitkQuaternion & *q*) [inline]

"=" operator.

6.94.4 Friends And Related Function Documentation

6.94.4.1 mitkQuaternion operator * (const mitkQuaternion & *p*, const mitkQuaternion & *q*) [friend]

Multiplication

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

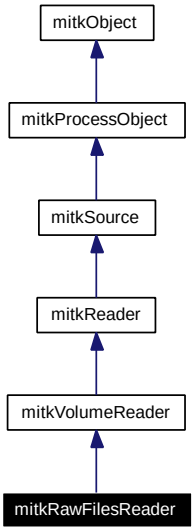
- mitkQuaternion.h

6.95 mitkRawFilesReader Class Reference

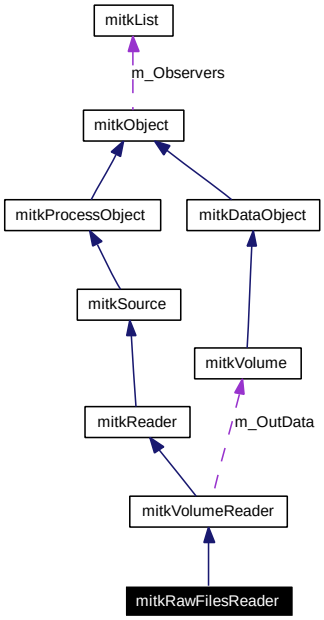
mitkRawFilesReader - a concrete reader for reading a series of files contain raw data (no header information).

`#include <mitkRawFilesReader.h>`

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 ()

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- bool **m_IsBigEndian**
- bool **m_IsColorByPlane**
- int **m_TitleBytes**
- int **m_Width**
- int **m_Height**
- int **m_ChNum**
- int **m_DataType**
- float **m_Spacings** [3]

6.95.1 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
}
```

6.95.2 Constructor & Destructor Documentation

6.95.2.1 mitkRawFilesReader::mitkRawFilesReader ()

Default constructor.

6.95.3 Member Function Documentation

6.95.3.1 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.

6.95.3.2 void mitkRawFilesReader::SetBigEndian (bool *isBig* = true) [inline]

Provided for convenience, just the same as SetEndian().

6.95.3.3 void mitkRawFilesReader::SetChannelNum (int *n*) [inline]

Set number of channels.

Parameters:
n number of channel 1—gray image 3—RGB image 4—RGBA image

6.95.3.4 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)

6.95.3.5 void mitkRawFilesReader::SetDataTypeToChar () [inline]

Set data type to char.

6.95.3.6 void mitkRawFilesReader::SetDataTypeToDouble () [inline]

Set data type to double.

6.95.3.7 void mitkRawFilesReader::SetDataTypeToFloat () [inline]

Set data type to float.

6.95.3.8 void mitkRawFilesReader::SetDataTypeToInt () [inline]

Set data type to int.

6.95.3.9 void mitkRawFilesReader::SetDataTypeToLong () [inline]

Set data type to long.

6.95.3.10 void mitkRawFilesReader::SetDataTypeToShort () [inline]

Set data type to short.

6.95.3.11 void mitkRawFilesReader::SetDataTypeToUnsignedChar () [inline]

Set data type to unsigned char.

6.95.3.12 void mitkRawFilesReader::SetDataTypeToUnsignedInt () [inline]

Set data type to unsigned int.

6.95.3.13 void mitkRawFilesReader::SetDataTypeToUnsignedLong () [inline]

Set data type to unsigned long.

6.95.3.14 void mitkRawFilesReader::SetDataTypeToUnsignedShort () [inline]

Set data type to unsigned short.

6.95.3.15 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)

6.95.3.16 void mitkRawFilesReader::SetHeight (int *h*) [inline]

Set image height.

Parameters:
h image height

6.95.3.17 void mitkRawFilesReader::SetLittleEndian (bool *isBig* = false) [inline]

Provided for convenience, just the same as SetEndian().

6.95.3.18 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 ..."

6.95.3.19 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.

6.95.3.20 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.

6.95.3.21 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.

6.95.3.22 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.

6.95.3.23 void mitkRawFilesReader::SetWidth (int *w*) [inline]

Set image width.

Parameters:
w image width

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

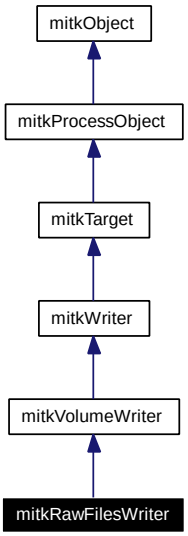
- mitkRawFilesReader.h

6.96 mitkRawFilesWriter Class Reference

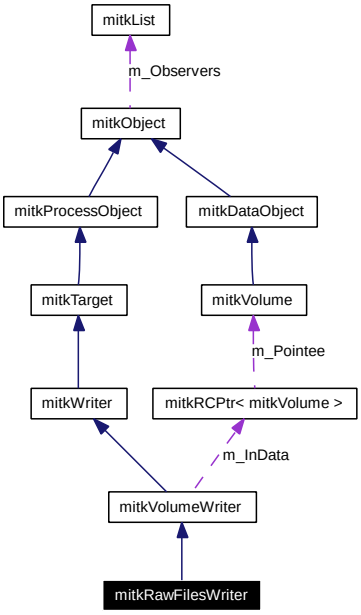
mitkRawFilesWriter - a concrete writer for writing a volume to a a series of raw files.

`#include <mitkRawFilesWriter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- bool **m_IsBigEndian**
- bool **m_IsColorByPlane**
- int **m_TitleBytes**

6.96.1 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();
```

6.96.2 Member Function Documentation

6.96.2.1 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.

6.96.2.2 void mitkRawFilesWriter::SetBigEndian (bool *isBig* = true) [inline]

Provided for convenience, just the same as SetEndian().

6.96.2.3 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)

6.96.2.4 void mitkRawFilesWriter::SetLittleEndian (bool *isBig* = `false`) `[inline]`

Provided for convenience, just the same as SetEndian().

6.96.2.5 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 ..."

6.96.2.6 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:

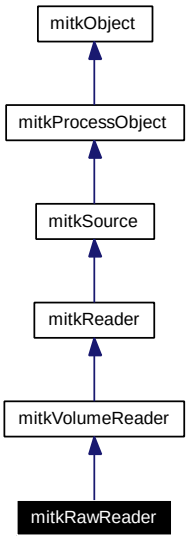
- mitkRawFilesWriter.h

6.97 mitkRawReader Class Reference

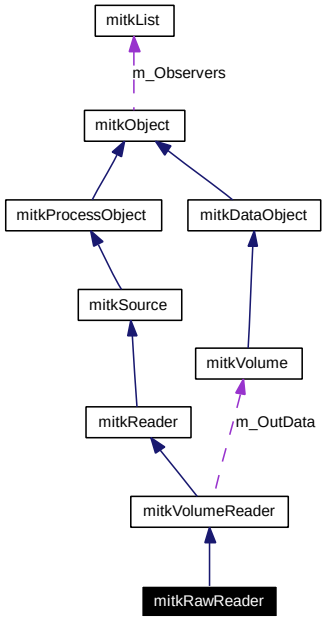
mitkRawReader - a concrete reader for reading raw volume file(no header information).

`#include <mitkRawReader.h>`

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 ()

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- bool **m_IsBigEndian**
- bool **m_IsColorByPlane**
- int **m_TitleBytes**
- int **m_Width**
- int **m_Height**
- int **m_ImageNum**
- int **m_ChNum**
- int **m_DataType**
- float **m_Spacings** [3]

6.97.1 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
}
```

6.97.2 Constructor & Destructor Documentation

6.97.2.1 mitkRawReader::mitkRawReader ()

Default constructor.

6.97.3 Member Function Documentation

6.97.3.1 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.

6.97.3.2 void mitkRawReader::SetBigEndian (bool *isBig* = true) [inline]

Provided for convenience, just the same as SetEndian().

6.97.3.3 void mitkRawReader::SetChannelNum (int *n*) [inline]

Set number of channel.

Parameters:
n number of channel 1—gray image 3—RGB image 4—RGBA image

6.97.3.4 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)

6.97.3.5 void mitkRawReader::SetDataTypeToChar () [inline]

Set data type to char.

6.97.3.6 void mitkRawReader::SetDataTypeToDouble () [inline]

Set data type to double.

6.97.3.7 void mitkRawReader::SetDataTypeToFloat () [inline]

Set data type to float.

6.97.3.8 void mitkRawReader::SetDataTypeToInt () [inline]

Set data type to int.

6.97.3.9 void mitkRawReader::SetDataTypeToLong () [inline]

Set data type to long.

6.97.3.10 void mitkRawReader::SetDataTypeToShort () [inline]

Set data type to short.

6.97.3.11 void mitkRawReader::SetDataTypeToUnsignedChar () [inline]

Set data type to unsigned char.

6.97.3.12 void mitkRawReader::SetDataTypeToUnsignedInt () [inline]

Set data type to unsigned int.

6.97.3.13 void mitkRawReader::SetDataTypeToUnsignedLong () [inline]

Set data type to unsigned long.

6.97.3.14 void mitkRawReader::SetDataTypeToUnsignedShort () [inline]

Set data type to unsigned short.

6.97.3.15 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)

6.97.3.16 void mitkRawReader::SetHeight (int *h*) [inline]

Set image height.

Parameters:
h image height

6.97.3.17 void mitkRawReader::SetImageNum (int *n*) [inline]

Set number of images.

Parameters:
n image number

6.97.3.18 void mitkRawReader::SetLittleEndian (bool *isBig* = false) [inline]

Provided for convenience, just the same as SetEndian().

6.97.3.19 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 ..."

6.97.3.20 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.

6.97.3.21 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.

6.97.3.22 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.

6.97.3.23 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.

6.97.3.24 void mitkRawReader::SetWidth (int *w*) [inline]

Set image width.

Parameters:
w image width

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

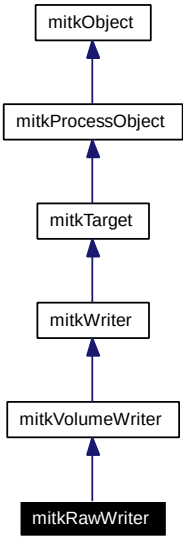
- mitkRawReader.h

6.98 mitkRawWriter Class Reference

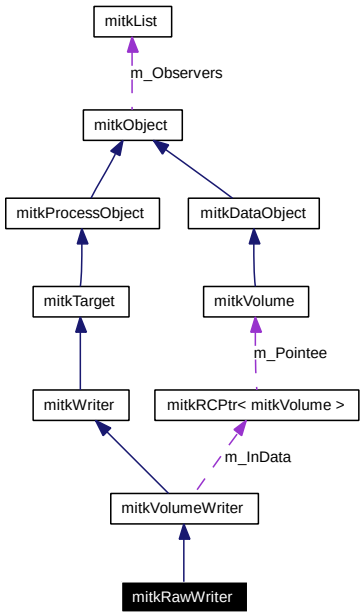
mitkRawWriter - a concrete writer for writing a volume to a single raw file(no any header information).

`#include <mitkRawWriter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- bool **m_IsBigEndian**
- bool **m_IsColorByPlane**
- int **m_TitleBytes**

6.98.1 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();
```

6.98.2 Member Function Documentation

6.98.2.1 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.

6.98.2.2 void mitkRawWriter::SetBigEndian (bool *isBig* = true) [inline]

Provided for convenience, just the same as SetEndian().

6.98.2.3 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)

6.98.2.4 void mitkRawWriter::SetLittleEndian (bool *isBig* = `false`) `[inline]`

Provided for convenience, just the same as SetEndian().

6.98.2.5 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 ..."

6.98.2.6 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

6.99 mitkRCPtr< T > Class Template Reference

mitkRCPtr - a smart pointer class

`#include <mitkRCPtr.h>`

Public Member Functions

- **mitkRCPtr** (T *realPtr=0)
- mitkRCPtr & **operator=** (T *realPtr)
- T * **operator** → () const
- T & **operator** * () const
- **operator T** * ()
- **operator T const** * () const

6.99.1 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:

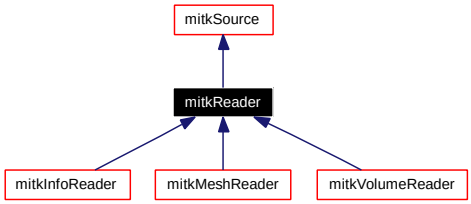
- mitkRCPtr.h

6.100 mitkReader Class Reference

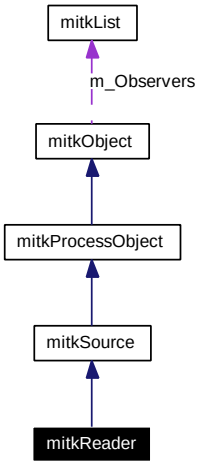
mitkReader - an abstract class represents a reader

`#include <mitkReader.h>`

Inheritance diagram for mitkReader:



Collaboration diagram for mitkReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddFileName (const char *inFileName)
- void SortFileNames ()

Protected Member Functions

- int **_getFileCount** (void)
- const char * **_getFileName** (int nIndex)

Protected Attributes

- mitkStringList * **m_FileNames**

6.100.1 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
```

6.100.2 Member Function Documentation

6.100.2.1 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.

6.100.2.2 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 mitkBMPReader, mitkCacheVolumeReader, mitkDICOMInfoReader, mitkDICOMReader, mitkInfoReader, mitkJPEGReader, mitkMeshReader, mitkPLYReader, mitkRawFilesReader, mitkRawReader, mitkTIFFReader, and mitkVolumeReader.

6.100.2.3 void mitkReader::SortFileNames ()

Sort all the file names alphabetically.

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

- mitkReader.h

6.101 mitkRectPlane Class Reference

mitkRectPlane - an encapsulation of a rectangle in 3D space

`#include <mitkVolumeResliceFilter.h>`

Public Types

- typedef double **coord_type**

Public Member Functions

- **mitkRectPlane** (coord_type pv0[4], coord_type pv1[4], coord_type pv2[4])
- **mitkRectPlane** (coord_type v0x, coord_type v0y, coord_type v0z, coord_type v1x, coord_type v1y, coord_type v1z, coord_type v2x, coord_type v2y, coord_type v2z)
- mitkRectPlane & **operator=** (mitkRectPlane const &p)
- **mitkRectPlane** (mitkRectPlane const &p)
- bool CheckValidity ()

Public Attributes

- coord_type **v0** [4]
- coord_type **v1** [4]
- coord_type **v2** [4]

6.101.1 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.

6.101.2 Member Function Documentation

6.101.2.1 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:

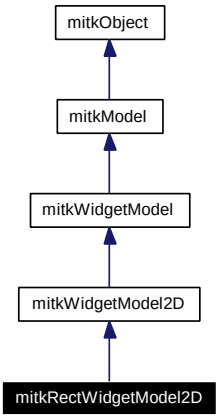
- mitkVolumeResliceFilter.h

6.102 mitkRectWidgetModel2D Class Reference

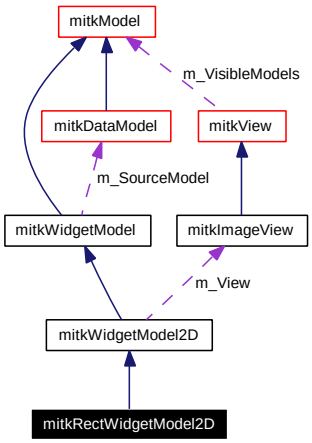
mitkRectWidgetModel2D - a 2D widget for displaying a rectangle in an image view

`#include <mitkRectWidgetModel2D.h>`

Inheritance diagram for mitkRectWidgetModel2D:



Collaboration diagram for mitkRectWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view)
- 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 Types

- enum {

  unknown, **rect**, **hsline**, **heline**,

  vsline, **veline**, **ssdot**, **esdot**,

  eedot, **sedot** }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()

Protected Attributes

- float **m_DotColor** [4]
- float **m_LineColor** [4]
- float **m_PickedDotColor** [4]
- float **m_PickedLineColor** [4]
- float **m_DotSize**
- int **m_StartPoint** [2]
- int **m_EndPoint** [2]
- float **m_TrueStartPoint** [2]
- float **m_TrueEndPoint** [2]
- float **m_ObjStartPoint** [2]
- float **m_ObjEndPoint** [2]
- bool **m_StartPointSet**
- bool **m_EndPointSet**
- string **m_UnitName**

6.102.1 Detailed Description

mitkRectWidgetModel2D - a 2D widget for displaying a rectangle in an image view

mitkRectWidgetModel2D is a 2D widget for displaying a rectangle in an image view. 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 view (e.g. mitkImageView), and in other conditions the display could be improper.

6.102.2 Member Function Documentation

6.102.2.1 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.

6.102.2.2 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.

6.102.2.3 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.

6.102.2.4 int mitkRectWidgetModel2D::GetBottom ()

Get the bottom of the rectangle in screen space.

Returns:

Return the bottom of the rectangle.

6.102.2.5 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

6.102.2.6 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

6.102.2.7 int mitkRectWidgetModel2D::GetHeight ()

Get the height of the rectangle in screen space.

Returns:
Return the height of the pseudocolor rectangle.

6.102.2.8 int mitkRectWidgetModel2D::GetLeft ()

Get the left of the rectangle in screen space.

Returns:
Return the left of the rectangle.

6.102.2.9 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.

6.102.2.10 int mitkRectWidgetModel2D::GetRight ()

Get the right of the rectangle in screen space.

Returns:
Return the right of the rectangle.

6.102.2.11 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

6.102.2.12 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

6.102.2.13 int mitkRectWidgetModel2D::GetTop ()

Get the top of the rectangle in screen space.

Returns:
Return the left of the rectangle.

6.102.2.14 float mitkRectWidgetModel2D::GetTrueBottom ()

Get the true bottom of the rectangle in source model space.

Returns:
Return the bottom of the rectangle.

6.102.2.15 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

6.102.2.16 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

6.102.2.17 float mitkRectWidgetModel2D::GetTrueHeight ()

Get the true height of the rectangle in source model space.

Returns:
Return the height of the pseudocolor rectangle.

6.102.2.18 float mitkRectWidgetModel2D::GetTrueLeft ()

Get the true left of the rectangle in source model space.

Returns:
Return the left of the rectangle.

6.102.2.19 float mitkRectWidgetModel2D::GetTrueRight ()

Get the true right of the rectangle in source model space.

Returns:
Return the right of the rectangle.

6.102.2.20 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

6.102.2.21 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

6.102.2.22 float mitkRectWidgetModel2D::GetTrueTop ()

Get the true top of the rectangle in source model space.

Returns:
Return the left of the rectangle.

6.102.2.23 float mitkRectWidgetModel2D::GetTrueWidth ()

Get the true width of the rectangle in source model space.

Returns:
Return the width of the pseudocolor rectangle.

6.102.2.24 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

6.102.2.25 int mitkRectWidgetModel2D::GetWidth ()

Get the width of the rectangle in screen space.

Returns:
Return the width of the pseudocolor rectangle.

6.102.2.26 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 view).

6.102.2.27 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.

6.102.2.28 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.

6.102.2.29 virtual void mitkRectWidgetModel2D::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.102.2.30 virtual int mitkRectWidgetModel2D::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.102.2.31 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.102.2.32 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.)

6.102.2.33 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.)

6.102.2.34 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.102.2.35 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.)

6.102.2.36 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.)

6.102.2.37 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 view 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 SetView() of this widget first) before calling this function, because this function needs the transform matrix of the source model and the view to calculate the original coordinates.

6.102.2.38 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.)

6.102.2.39 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.)

6.102.2.40 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:

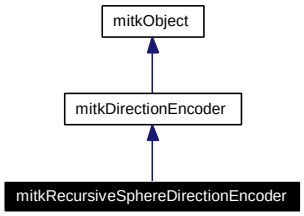
- mitkRectWidgetModel2D.h

6.103 mitkRecursiveSphereDirectionEncoder Class Reference

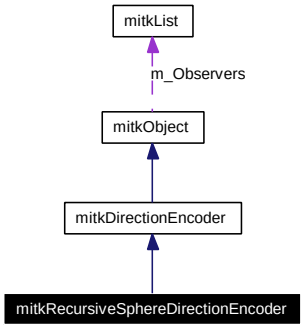
mitkRecursiveSphereDirectionEncoder - A direction encoder based on the recursive subdivision of an octahedron.

`#include <mitkRecursiveSphereDirectionEncoder.h>`

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 ()

Protected Member Functions

- void **_initializeIndexTable** ()

Protected Attributes

- int **m_RecursionDepth**
- int * **m_IndexTable**

- float * **m_DecodedNormal**
- int **m_IndexTableRecursionDepth**
- int **m_OuterSize**
- int **m_InnerSize**
- int **m_GridSize**

6.103.1 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.

6.103.2 Member Function Documentation

6.103.2.1 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.

6.103.2.2 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.

6.103.2.3 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.

6.103.2.4 int mitkRecursiveSphereDirectionEncoder::GetNumberOfEncodedDirections () [virtual]

Return the number of encoded directions

Returns:
Return the number of encoded directions

Implements mitkDirectionEncoder.

6.103.2.5 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

6.103.2.6 virtual int mitkRecursiveSphereDirectionEncoder::GetRecursionDepthMaxValue () [inline, virtual]

Get the max recursion depth value

Returns:
fnDepth Return the max recursion depth value

6.103.2.7 virtual int mitkRecursiveSphereDirectionEncoder::GetRecursionDepthMinValue () [inline, virtual]

Get the min recursion depth value

Returns:
fnDepth Return the min recursion depth value

6.103.2.8 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:

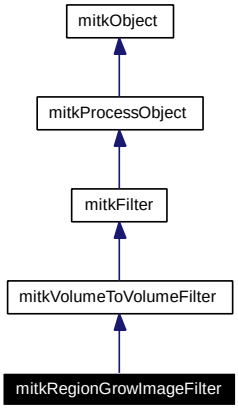
- mitkRecursiveSphereDirectionEncoder.h

6.104 mitkRegionGrowImageFilter Class Reference

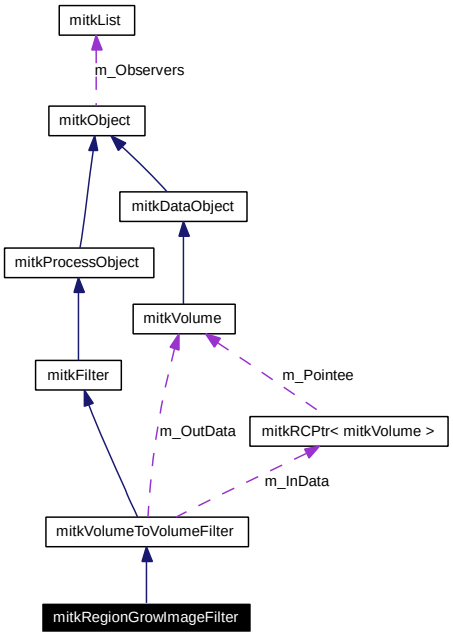
mitkRegionGrowImageFilter - A class that implement region grow segmentation arithmetic

`#include <mitkRegionGrowImageFilter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

6.104.1 Detailed Description

mitkRegionGrowImageFilter - A class that implement region grow segmentation arithmetic

mitkRegionGrowImageFilter - A class that implement region grow segmentation arithmetic

6.104.2 Constructor & Destructor Documentation

6.104.2.1 mitkRegionGrowImageFilter::mitkRegionGrowImageFilter ()

Constructor of the class.

6.104.2.2 virtual mitkRegionGrowImageFilter::~mitkRegionGrowImageFilter () [virtual]

Constructor of the class.

6.104.3 Member Function Documentation

6.104.3.1 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.

6.104.3.2 void mitkRegionGrowImageFilter::SetMaxChangeValue (double *v*) [inline]

Set the max change value

Parameters:
v Represent the max change value

6.104.3.3 void mitkRegionGrowImageFilter::SetMaxDifferentValue (double *v*) [inline]

Set the max different value

Parameters:
v Represent the max different value

6.104.3.4 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:

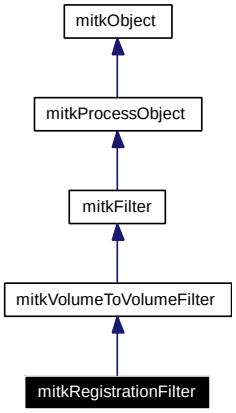
- mitkRegionGrowImageFilter.h

6.105 mitkRegistrationFilter Class Reference

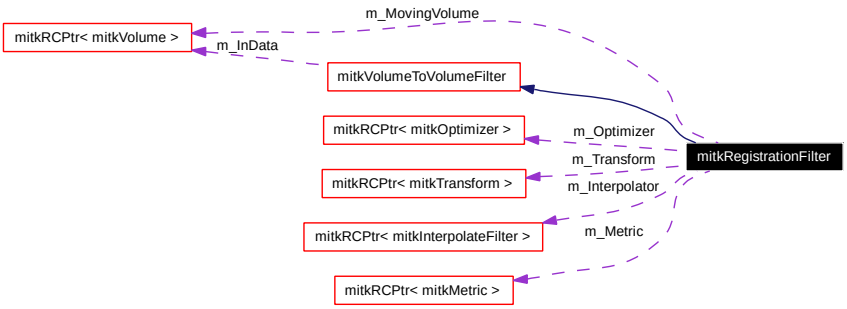
mitkRegistrationFilter - a class for registration filter

`#include <mitkRegistrationFilter.h>`

Inheritance diagram for mitkRegistrationFilter:



Collaboration diagram for mitkRegistrationFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetMovingVolume (mitkVolume *movingVolume)
- mitkVolume * GetMovingVolume ()
- void SetOptimizer (mitkOptimizer *optimizer)
- mitkOptimizer * GetOptimizer ()
- void SetMetric (mitkMetric *metric)
- mitkMetric * GetMetric ()
- void SetInterpolator (mitkInterpolateFilter *interpolator)
- mitkInterpolateFilter * GetInterpolator ()
- void SetTransform (mitkTransform *transform)
- mitkTransform * GetTransform ()
- void SetInitialTransformParameters (vector< float > *initialTransformParameters)
- vector< float > * GetInitialParameters ()
- vector< float > * GetLastTransformParameters ()

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_MovingVolume**
- mitkRCPtr< mitkOptimizer > **m_Optimizer**
- mitkRCPtr< mitkMetric > **m_Metric**
- mitkRCPtr< mitkInterpolateFilter > **m_Interpolator**
- mitkRCPtr< mitkTransform > **m_Transform**
- vector< float > * **m_InitialTransformParameters**
- vector< float > * **m_LastTransformParameters**

6.105.1 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.

6.105.2 Member Function Documentation

6.105.2.1 vector<float>* mitkRegistrationFilter::GetInitialParameters ()

Get the initial transform parameters

Returns:
Return the intial transform parameters

6.105.2.2 mitkInterpolateFilter* mitkRegistrationFilter::GetInterpolator ()

Get the interpolate method

Returns:
Return interpolate method

6.105.2.3 vector<float>* mitkRegistrationFilter::GetLastTransformParameters ()

Get the last transform parameters

Returns:
Return the last transform parameters

6.105.2.4 mitkMetric* mitkRegistrationFilter::GetMetric ()

Get the similarity metric method

Returns:
Return similarity metric method

6.105.2.5 mitkVolume* mitkRegistrationFilter::GetMovingVolume () [inline]

Get the Fixed volume

Returns:
Return the Fixed volume

6.105.2.6 mitkOptimizer* mitkRegistrationFilter::GetOptimizer ()

Get the optimize method

Returns:
Return the optimize method

6.105.2.7 mitkTransform* mitkRegistrationFilter::GetTransform ()

Get the transform method

Returns:
Return transform method

6.105.2.8 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.

6.105.2.9 void mitkRegistrationFilter::SetInitialTransformParameters (vector< float > * *initialTransformParameters*)

Set the initial transform parameters

Parameters:
initialTransformParameters Specify the initial transform parameters

6.105.2.10 void mitkRegistrationFilter::SetInterpolator (mitkInterpolateFilter * *interpolator*) [inline]

Set the interpolate method

Parameters:
interpolator Specify interpolate method

6.105.2.11 void mitkRegistrationFilter::SetMetric (mitkMetric * *metric*) [inline]

Set the similarity metric method

Parameters:
metric Specify similarity metric method

6.105.2.12 void mitkRegistrationFilter::SetMovingVolume (mitkVolume * *movingVolume*) [inline]

Set the Moving volume

Parameters:
movingVolume Specify the Moving volume

6.105.2.13 void mitkRegistrationFilter::SetOptimizer (mitkOptimizer * *optimizer*) [inline]

Set the optimize method

Parameters:
optimizer Specify the optimize method

6.105.2.14 void mitkRegistrationFilter::SetTransform (mitkTransform * *transform*) [inline]

Set the transform method

Parameters:
transform Specify transform method

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

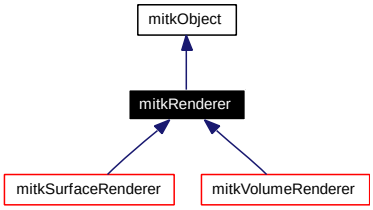
- mitkRegistrationFilter.h

6.106 mitkRenderer Class Reference

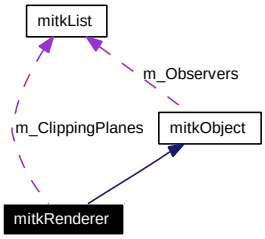
mitkRenderer - an abstract class to define the common interface of a renderer

`#include <mitkRenderer.h>`

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 ()

Protected Attributes

- mitkList * **m_ClippingPlanes**
- bool **m_EnableClipping**

6.106.1 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.

6.106.2 Member Function Documentation

6.106.2.1 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

6.106.2.2 void mitkRenderer::ClippingOff () [inline]

Disable the clipping

6.106.2.3 void mitkRenderer::ClippingOn () [inline]

Enable the clipping

6.106.2.4 bool mitkRenderer::GetClipping () [inline]

Get the status of clipping

Returns:
Return true163172the clipping is enabled Return false, the clipping is disabled

6.106.2.5 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

6.106.2.6 int mitkRenderer::GetClippingPlaneCount (void) [inline]

Get the count of clipping plane in this renderer.

Returns:
Return the count of clipping plane in this renderer.

6.106.2.7 mitkList* mitkRenderer::GetClippingPlanes (void) [inline]

Get the plane list of this renderer.

Returns:
Return the internal plane list

6.106.2.8 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, mitkOoCVolumeRendererTexture3D, mitkSurfaceRenderer, mitkSurfaceRendererStandard, mitkSurfaceRendererUseVA, mitkSurfaceRendererUseVBO, mitkVolumeRenderer, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, and mitkVolumeRendererTexture3D.

6.106.2.9 void mitkRenderer::RemoveAllClippingPlanes ()

Remove all clipping planes from this renderer.

6.106.2.10 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.

6.106.2.11 void mitkRenderer::SetClipping (bool *enableClipping*) [inline]

Set the status of clipping

Parameters:
enableClipping enableClipping = true163172enable the clipping enableClipping = false163172disable the clipping

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

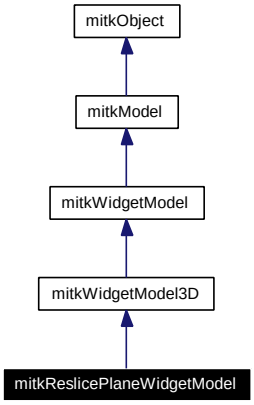
- mitkRenderer.h

6.107 mitkReslicePlaneWidgetModel Class Reference

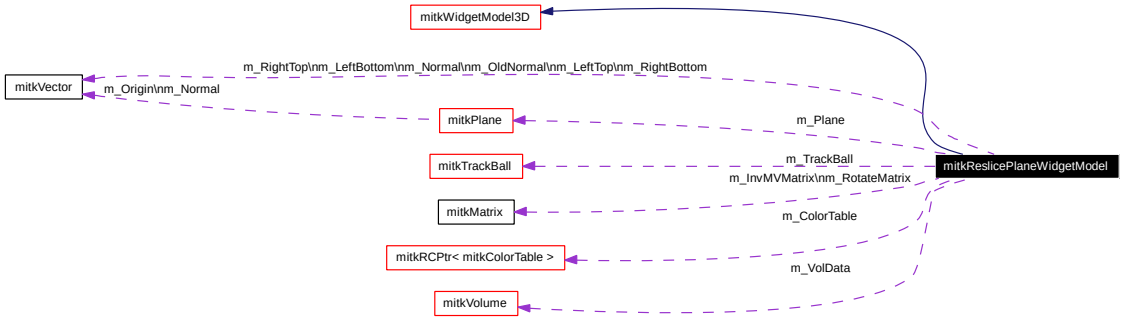
mitkReslicePlaneWidgetModel - a 3D widget for re-slice plane

`#include <mitkReslicePlaneWidgetModel.h>`

Inheritance diagram for mitkReslicePlaneWidgetModel:



Collaboration diagram for mitkReslicePlaneWidgetModel:



Public Types

- typedef float **coord_type**

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 (mitkView *view)
- 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)
- 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 ()
- 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 Types

- enum {
  unknown, **lbball**, **ltball**, **rbball**,
  rtball, **lline**, **tline**, **rline**,
  bline, **plane** }

Protected Member Functions

- virtual float * **_getBounds** ()
- 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)
- void **_init** ()
- bool **_reslice** ()
- bool **_mapColor** ()
- bool **_setDummyTexture** ()
- bool **_updateTexture** ()
- void **_deleteTexture** ()
- void **_rotate** ()

Protected Attributes

- mitkRCPtr< mitkColorTable > **m_ColorTable**
- mitkVolume * **m_VolData**
- mitkPlane * **m_Plane**
- mitkVector * **m_LeftBottom**
- mitkVector * **m_RightBottom**
- mitkVector * **m_RightTop**
- mitkVector * **m_LeftTop**
- mitkVector * **m_Normal**
- mitkVector * **m_OldNormal**
- mitkMatrix * **m_InvMVMatrix**
- mitkMatrix * **m_RotateMatrix**
- mitkTrackBall **m_TrackBall**
- float **m_ArrowLength**
- float **m_ArrowColor** [4]
- float **m_BallColor** [4]
- float **m_LineColor** [4]
- float **m_PickedBallColor** [4]
- float **m_PickedLineColor** [4]
- float **m_OldOrigin** [3]
- float **m_StartOrigin** [3]
- float **m_PlaneOpacity**
- float **m_SliceOpacity**
- float **m_SpacingX**
- float **m_SpacingY**
- float **m_MoveUnit**

- float **m_ChangingValue**
- GLuint **m_TexID**
- int **m_DummyTexWidth**
- int **m_DummyTexHeight**
- int **m_SliceImgWidth**
- int **m_SliceImgHeight**
- unsigned int **m_BufLen**
- unsigned int **m_ClrBufLen**
- unsigned char * **m_TexBuf**
- unsigned char * **m_ClrTexBuf**
- bool **m_NeedSetDummyTexture**
- bool **m_NeedUpdateTexture**
- bool **m_BufSizeChanged**
- bool **m_TexNotUpdate**
- bool **m_EnableReslice**
- bool **m_EnableColor**
- string **m_UnitName**

6.107.1 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.

6.107.2 Constructor & Destructor Documentation

6.107.2.1 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

6.107.3 Member Function Documentation

6.107.3.1 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.

6.107.3.2 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.

6.107.3.3 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.

6.107.3.4 void mitkReslicePlaneWidgetModel::DisablePseudocolor () [inline]

Display pseudo-color display for re-slice.

6.107.3.5 void mitkReslicePlaneWidgetModel::DisableReslice () [inline]

Disable re-slice (do not reconstruct the slice of the transection).

6.107.3.6 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

6.107.3.7 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

6.107.3.8 mitkColorTable* mitkReslicePlaneWidgetModel::GetColorTable ()

Get color table.

Returns:

Return the pointer to the mitkColorTable object.

6.107.3.9 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.

6.107.3.10 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

6.107.3.11 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.

6.107.3.12 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

6.107.3.13 float mitkReslicePlaneWidgetModel::GetMovedDistance ()

Get moved distance since mouse button is pressed.

Returns:

Return the value of moved distance.

6.107.3.14 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

6.107.3.15 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

6.107.3.16 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.

6.107.3.17 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.

6.107.3.18 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

6.107.3.19 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.

6.107.3.20 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

6.107.3.21 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.

6.107.3.22 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.

6.107.3.23 int mitkReslicePlaneWidgetModel::GetSliceImageHeight () [inline]

Get the image height of the reconstructed slice.

Returns:
Return the image height (measured by pixels) of the reconstructed slice.

6.107.3.24 int mitkReslicePlaneWidgetModel::GetSliceImageWidth () [inline]

Get the image width of the reconstructed slice.

Returns:
Return the image width (measured by pixels) of the reconstructed slice.

6.107.3.25 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

6.107.3.26 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.

6.107.3.27 const string& mitkReslicePlaneWidgetModel::GetUnitName () [inline]

Get name string of unit.

Returns:
Return a constant reference to a string contains the name

6.107.3.28 float mitkReslicePlaneWidgetModel::GetVertMoveFromStart ()

Get vertical movement from the start position (plane center).

Returns:
Return the vertical movement from the start position.

6.107.3.29 mitkVolume* mitkReslicePlaneWidgetModel::GetVolumeData () [inline]

Get the source volume data for reconstructing slice.

Returns:
Return the pointer to the source volume object.

6.107.3.30 bool mitkReslicePlaneWidgetModel::IsBallSelected ()

Whether any one of the balls is selected.

Returns:
Return true if one of the four balls is selected.

6.107.3.31 bool mitkReslicePlaneWidgetModel::IsLineSelected ()

Whether any one of the lines is selected.

Returns:
Return true if one of the four lines is selected.

6.107.3.32 bool mitkReslicePlaneWidgetModel::IsPseudocolorEnabled () [inline]

Test if the pseudo-color function is enabled.

6.107.3.33 bool mitkReslicePlaneWidgetModel::IsResliceEnabled () [inline]

Test if the re-slice function is enabled.

6.107.3.34 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.

6.107.3.35 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.

6.107.3.36 virtual void mitkReslicePlaneWidgetModel::Release () [virtual]

Maintain the selection status when this widget is released.

Implements mitkWidgetModel.

6.107.3.37 virtual int mitkReslicePlaneWidgetModel::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView in which this model will be shown

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

Reimplemented from mitkModel.

6.107.3.38 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

6.107.3.39 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

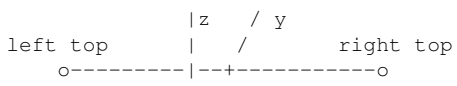
6.107.3.40 void mitkReslicePlaneWidgetModel::RotateDegAroundZAxisOfPlane (float *angle*)

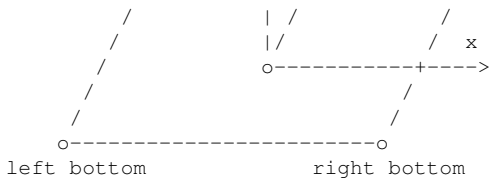
Rotate the plane around the z axis of the plane. This function is provided for convenience. It behaves essentially like RotateRadAroundZAxisOfPlane(), but the parameter angle is in degree.

Parameters:
angle rotation angle in degree

6.107.3.41 void mitkReslicePlaneWidgetModel::RotateRadAroundXAxisOfPlane (float *angle*)

Rotate the plane around the x axis of the plane.



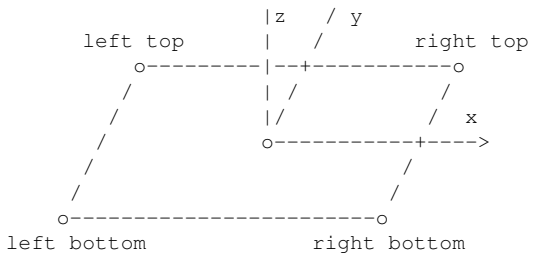


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

6.107.3.42 void mitkReslicePlaneWidgetModel::RotateRadAroundYAxisOfPlane (float *angle*)

Rotate the plane around the y axis of the plane.

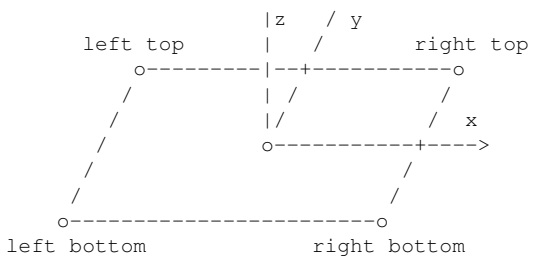


Note:
The rotation is performed in the local coordinates of the plane.

Parameters:
angle rotation angle in radian

6.107.3.43 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

6.107.3.44 void mitkReslicePlaneWidgetModel::SetColorTable (mitkColorTable * *ct*) [inline]

Set color table.

Parameters:
ct the pointer to a mitkColorTable object

6.107.3.45 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

6.107.3.46 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)

6.107.3.47 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

6.107.3.48 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

6.107.3.49 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

6.107.3.50 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)

6.107.3.51 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.

6.107.3.52 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

6.107.3.53 void mitkReslicePlaneWidgetModel::SetVolumeData (mitkVolume * *vol*)

Set the source volume data for reconstructing slice.

Parameters:
vol the pointer to the source volume object

6.107.3.54 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

6.107.3.55 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:

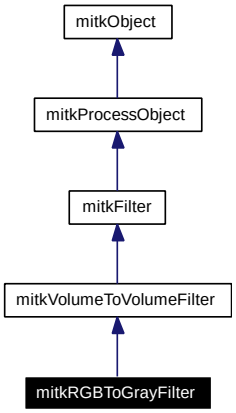
- mitkReslicePlaneWidgetModel.h

6.108 mitkRGBToGrayFilter Class Reference

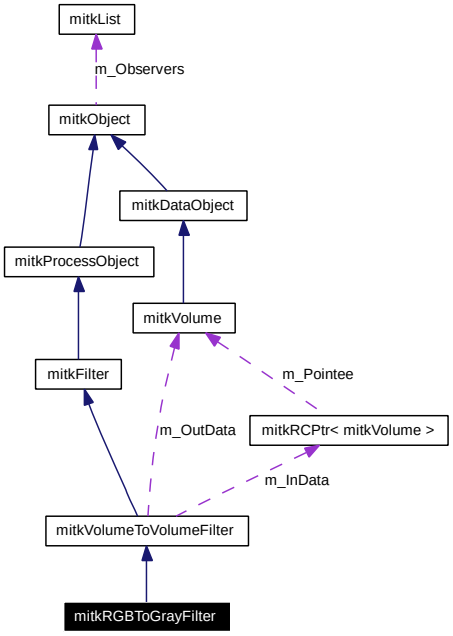
mitkRGBToGrayFilter - a filter to transfer RGB volume to gray volume

`#include <mitkRGBToGrayFilter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- float **m_CoefRed**
- float **m_CoefGreen**
- float **m_CoefBlue**

6.108.1 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().

6.108.2 Member Function Documentation

6.108.2.1 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.

6.108.2.2 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

6.108.2.3 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:

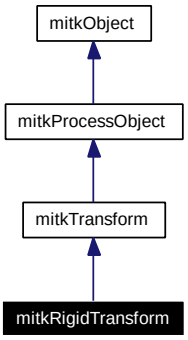
- mitkRGBToGrayFilter.h

6.109 mitkRigidTransform Class Reference

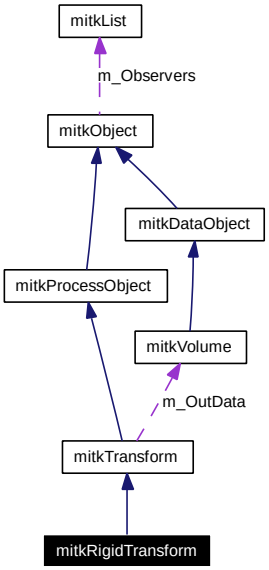
mitkRigidTransform - a concrete transform to perform rigid transformation

`#include <mitkRigidTransform.h>`

Inheritance diagram for mitkRigidTransform:



Collaboration diagram for mitkRigidTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- void **Initialize** ()
- virtual bool **Execute** ()
- virtual bool **_transform** (float out[3], float x, float y, float z)

Protected Attributes

- float **m_Matrix** [3][4]

6.109.1 Detailed Description

mitkRigidTransform - a concrete transform to perform rigid transformation

mitkRigidTransform is a concrete transform to perform rigid transformation. This transform applies a rotation and translation to the space.

6.109.2 Member Function Documentation

6.109.2.1 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:

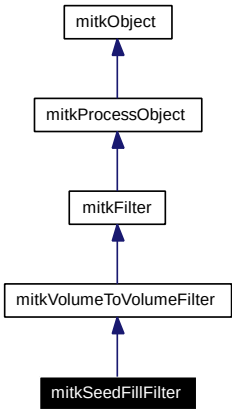
- mitkRigidTransform.h

6.110 mitkSeedFillFilter Class Reference

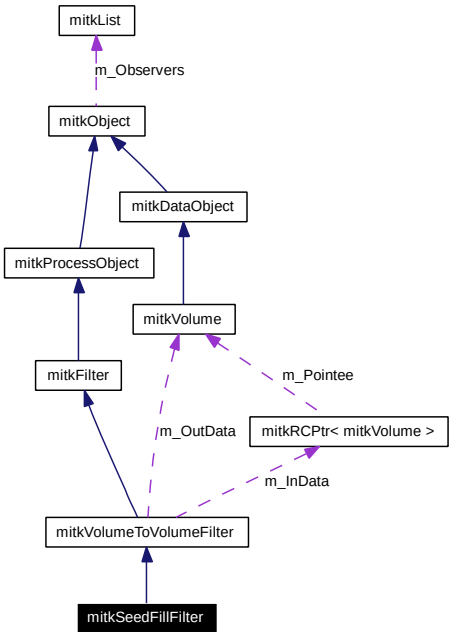
mitkSeedFillFilter - a concrete filter for seed fill algorithm

`#include <mitkSeedFillFilter.h>`

Inheritance diagram for mitkSeedFillFilter:



Collaboration diagram for mitkSeedFillFilter:



Public Member Functions

- mitkSeedFillFilter ()
- virtual void PrintSelf (ostream &os)
- virtual ~mitkSeedFillFilter ()
- void **SetSeedPoint** (int x, int y)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- int **m_SeedX**
- int **m_SeedY**

6.110.1 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.

6.110.2 Constructor & Destructor Documentation

6.110.2.1 mitkSeedFillFilter::mitkSeedFillFilter () [inline]

Constructor of the class.

6.110.2.2 virtual mitkSeedFillFilter::~mitkSeedFillFilter () [inline, virtual]

Constructor of the class.

6.110.3 Member Function Documentation

6.110.3.1 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:

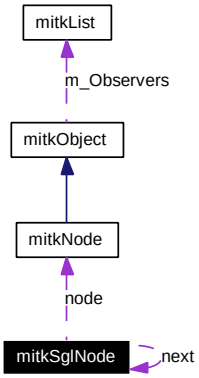
- mitkSeedFillFilter.h

6.111 mitkSglNode Class Reference

mitkSglNode - a class that define mitkSglNode used by mitkNodeHeap

`#include <mitkNodeHeap.h>`

Collaboration diagram for mitkSglNode:



Public Member Functions

- mitkSglNode ()
- ~mitkSglNode ()

Public Attributes

- mitkNode **node**
- mitkSglNode * **next**

6.111.1 Detailed Description

mitkSglNode - a class that define mitkSglNode used by mitkNodeHeap

mitkSglNode define a single node of a SgnNode link

6.111.2 Constructor & Destructor Documentation

6.111.2.1 mitkSglNode::mitkSglNode () `[inline]`

Constructor of the class

6.111.2.2 mitkSglNode::~mitkSglNode () `[inline]`

Destructor of the class

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

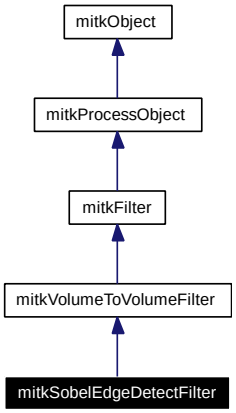
- mitkNodeHeap.h

6.112 mitkSobelEdgeDetectFilter Class Reference

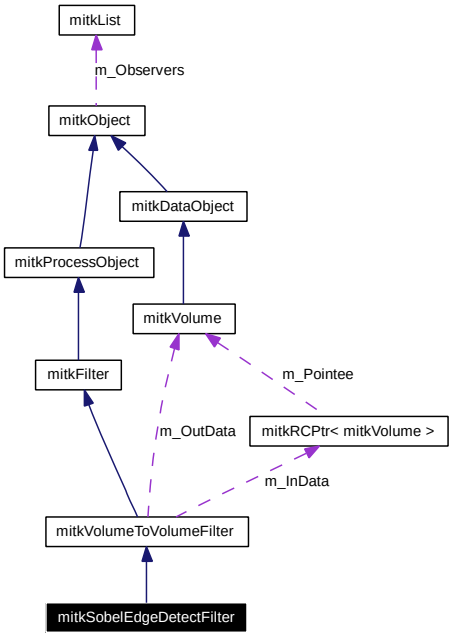
mitkSobelEdgeDetectFilter - a class used to detect the edge in an image

`#include <mitkSobelEdgeDetectFilter.h>`

Inheritance diagram for mitkSobelEdgeDetectFilter:



Collaboration diagram for mitkSobelEdgeDetectFilter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetThreshold (float Threshold)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- float **m_Threshold**

6.112.1 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.

6.112.2 Member Function Documentation

6.112.2.1 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.

6.112.2.2 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:

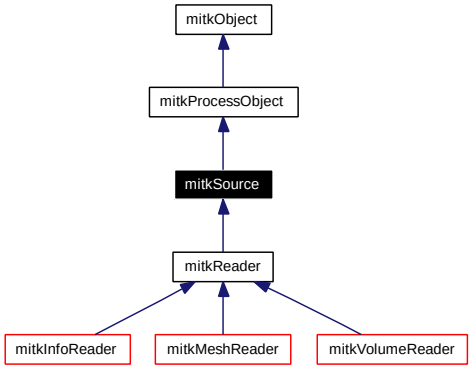
- mitkSobelEdgeDetectFilter.h

6.113 mitkSource Class Reference

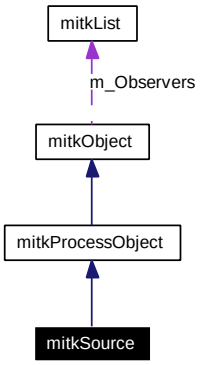
mitkSource - abstract class specifies interface for source object

`#include <mitkSource.h>`

Inheritance diagram for mitkSource:



Collaboration diagram for mitkSource:



Public Member Functions

- virtual void PrintSelf (ostream &os)

6.113.1 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.

6.113.2 Member Function Documentation

6.113.2.1 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 mitkBMPReader, mitkCacheVolumeReader, mitkDICOMInfoReader, mitkDICOMReader, mitkInfoReader, mitkJPEGReader, mitkMeshReader, mitkPLYReader, mitkRawFilesReader, mitkRawReader, mitkReader, mitkTIFFReader, and mitkVolumeReader.

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

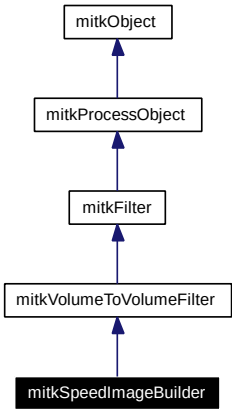
- mitkSource.h

6.114 mitkSpeedImageBuilder Class Reference

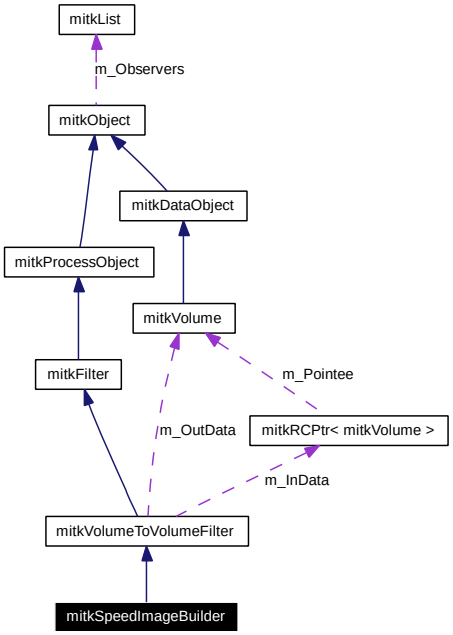
mitkSpeedImageBuilder - a class that build SpeedImage for mitkFastMarchingImageFilter

`#include <mitkSpeedImageBuilder.h>`

Inheritance diagram for mitkSpeedImageBuilder:



Collaboration diagram for mitkSpeedImageBuilder:



Protected Member Functions

- virtual bool **Execute** ()

6.114.1 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:

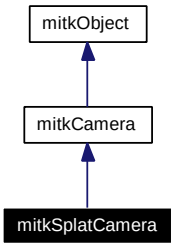
- mitkSpeedImageBuilder.h

6.115 mitkSplatCamera Class Reference

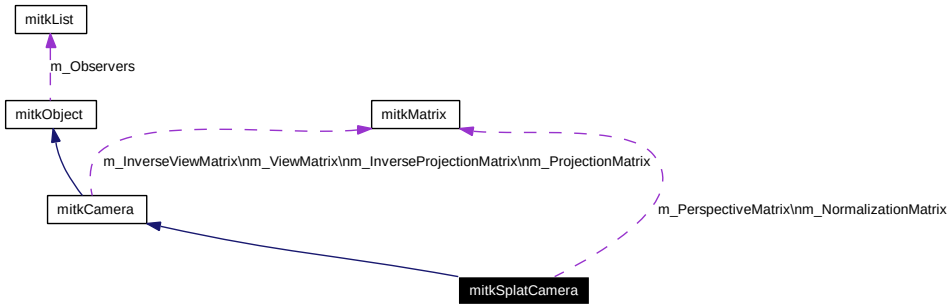
mitkSplatCamera - a camera in 3D view

`#include <mitkSplatCamera.h>`

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)

Protected Attributes

- mitkMatrix * **m_PerspectiveMatrix**
- mitkMatrix * **m_NormalizationMatrix**

6.115.1 Detailed Description

mitkSplatCamera - a camera in 3D view

mitkSplatCamera is a camera in 3D view. The only way you can access it is to get a pointer to it by using the member function GetCamera() of mitkView. Then you can control it through its interface, and the change will affect the image rendered in the view. 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 view's camera to be mitkSplatCamera object.

See also:
Please see more information from mitkCamera class.

6.115.2 Member Function Documentation

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

Perspective Projection

Reimplemented from mitkCamera.

6.115.2.2 const mitkMatrix* mitkSplatCamera::GetNormalizationMatrix (void) [inline]

Get the Normalization transform matrix.

6.115.2.3 void mitkSplatCamera::GetNormalizationMatrix (float *m*[16])

Get the Normalization transform matrix.

6.115.2.4 void mitkSplatCamera::GetNormalizationMatrix (mitkMatrix * *m*)

Get the Normalization transform matrix.

6.115.2.5 const mitkMatrix* mitkSplatCamera::GetPerspectiveMatrix (void) [inline]

Get the perspective transform matrix.

6.115.2.6 void mitkSplatCamera::GetPerspectiveMatrix (float *m*[16])

Get the perspective transform matrix.

6.115.2.7 void mitkSplatCamera::GetPerspectiveMatrix (mitkMatrix * *m*)

Get the perspective transform matrix.

6.115.2.8 virtual void mitkSplatCamera::Perspective (float *fovy*, float *aspect*, float *zNear*, float *zFar*) [virtual]

Perspective Projection

Reimplemented from mitkCamera.

6.115.2.9 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:

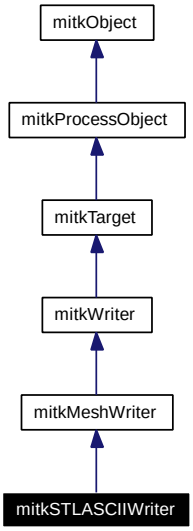
- mitkSplatCamera.h

6.116 mitkSTLASCIIWriter Class Reference

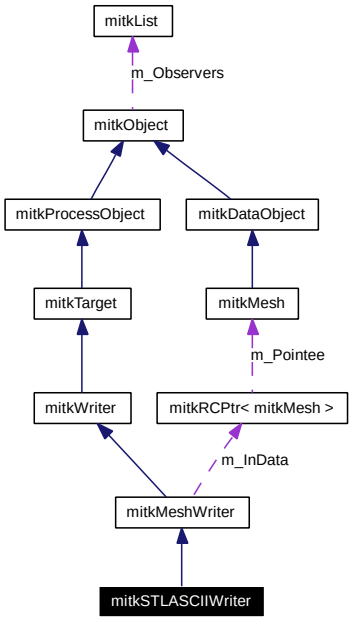
mitkSTLASCIIWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using ASCII format

`#include <mitkSTLASCIIWriter.h>`

Inheritance diagram for mitkSTLASCIIWriter:



Collaboration diagram for mitkSTLASCIIWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual bool **Execute** ()

6.116.1 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();
```

6.116.2 Member Function Documentation

6.116.2.1 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:

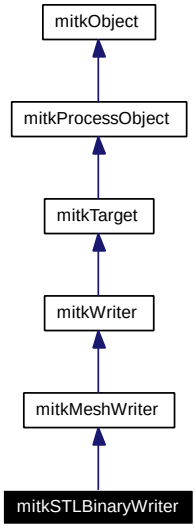
- mitkSTLASCIIWriter.h

6.117 mitkSTLBinaryWriter Class Reference

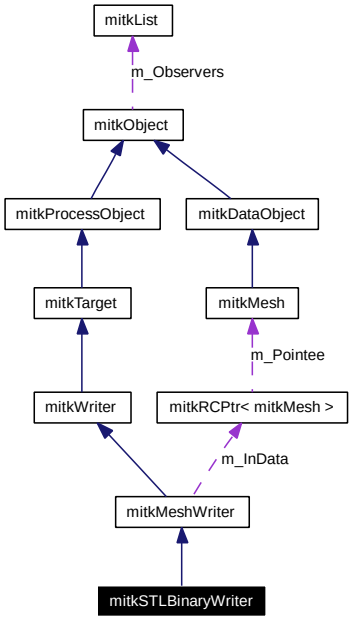
mitkSTLBinaryWriter - a concrete writer for writing a mesh to STL (STereo Lithography) file using binary format

`#include <mitkSTLBinaryWriter.h>`

Inheritance diagram for mitkSTLBinaryWriter:



Collaboration diagram for mitkSTLBinaryWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual bool **Execute** ()

6.117.1 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:

```
mitkIM0Writer *aWriter = new mitkSTLBinaryWriter;
aWriter->SetInput(aMesh);
aWriter->AddFileName(filename);
aWriter->Run();
```

6.117.2 Member Function Documentation

6.117.2.1 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:

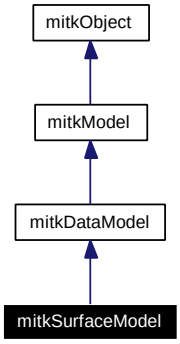
- mitkSTLBinaryWriter.h

6.118 mitkSurfaceModel Class Reference

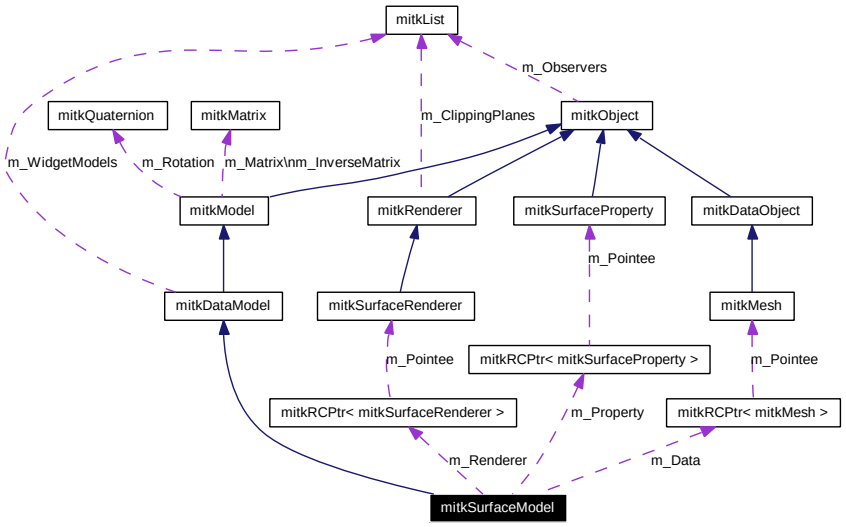
mitkSurfaceModel - an 3D entity in a rendering scene represented in surface

`#include <mitkSurfaceModel.h>`

Inheritance diagram for mitkSurfaceModel:



Collaboration diagram for mitkSurfaceModel:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- mitkSurfaceModel ()
- virtual int Render (mitkView *view)
- virtual mitkRenderer * GetBasicRenderer ()
- void SetRenderer (mitkSurfaceRenderer *renderer)
- mitkSurfaceRenderer * GetRenderer ()
- void SetProperty (mitkSurfaceProperty *prop)
- mitkSurfaceProperty * GetProperty ()
- void SetData (mitkMesh *data)
- mitkMesh * GetData ()

- virtual bool IsOpaque ()

Protected Member Functions

- virtual float * **_getBounds** ()

Protected Attributes

- mitkRCPtr< mitkMesh > **m_Data**
- mitkRCPtr< mitkSurfaceProperty > **m_Property**
- mitkRCPtr< mitkSurfaceRenderer > **m_Renderer**

6.118.1 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.

6.118.2 Constructor & Destructor Documentation

6.118.2.1 mitkSurfaceModel::mitkSurfaceModel ()

Default constructor.

6.118.3 Member Function Documentation

6.118.3.1 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.

6.118.3.2 mitkMesh* mitkSurfaceModel::GetData () [inline]

Get the surface data.

Returns:
Return pointer to the surface model's mesh data.

6.118.3.3 mitkSurfaceProperty* mitkSurfaceModel::GetProperty ()

Get the surface property.

Returns:
Return pointer to this surface model's property.

6.118.3.4 mitkSurfaceRenderer* mitkSurfaceModel::GetRenderer ()

Get the surface renderer.

Returns:
Return a pointer to this surface model's render.

6.118.3.5 virtual bool mitkSurfaceModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Implements mitkModel.

6.118.3.6 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.

6.118.3.7 virtual int mitkSurfaceModel::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView 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.

6.118.3.8 void mitkSurfaceModel::SetData (mitkMesh * *data*)

Set the surface data.

Parameters:
data pointer to an mitkMesh

6.118.3.9 void mitkSurfaceModel::SetProperty (mitkSurfaceProperty * *prop*) [inline]

Set the surface property.

Parameters:
prop pointer to an mitkSurfaceProperty

6.118.3.10 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:

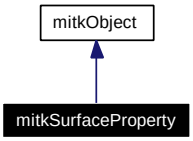
- mitkSurfaceModel.h

6.119 mitkSurfaceProperty Class Reference

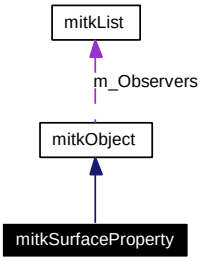
mitkSurfaceProperty - properties of an mitkSurfaceModel

`#include <mitkSurfaceProperty.h>`

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)

Protected Attributes

- float **m_Color** [4]
- float **m_AmbientColor** [4]
- float **m_DiffuseColor** [4]
- float **m_SpecularColor** [4]
- float **m_EmissionColor** [4]
- float **m_EdgeColor** [4]
- float **m_Ambient**
- float **m_Diffuse**
- float **m_Specular**
- float **m_Emission**
- float **m_SpecularPower**
- float **m_Opacity**
- float **m_PointSize**
- float **m_LineWidth**
- int **m_RepresentationType**
- int **m_InterpolationType**
- int **m_LineStippleRepeatFactor**
- unsigned short **m_LineStipplePattern**
- bool **m_Modified**

6.119.1 Detailed Description

mitkSurfaceProperty - properties of an mitkSurfaceModel

mitkSurfaceProperty is a class contains properties of an mitkSurfaceModel including geometry representation parameters and material parameters.

6.119.2 Constructor & Destructor Documentation

6.119.2.1 mitkSurfaceProperty::mitkSurfaceProperty ()

Default constructor.

6.119.3 Member Function Documentation

6.119.3.1 float mitkSurfaceProperty::GetAmbient () const [inline]

Get the ambient coefficient.

Returns:
Return the ambient coefficient.

6.119.3.2 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.

6.119.3.3 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.

6.119.3.4 float mitkSurfaceProperty::GetDiffuse () const [inline]

Get the diffuse coefficient.

Returns:
Return the diffuse coefficient.

6.119.3.5 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).

6.119.3.6 const char * mitkSurfaceProperty::GetInterpolationTypeAsString () const [inline]

Get the interpolation type for sampling a surface as a string.

Returns:
Return the string of interpolation type.

6.119.3.7 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).

6.119.3.8 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

6.119.3.9 float mitkSurfaceProperty::GetLineWidth () const [inline]

Get the line width. The width is expressed in screen units.

Returns:
Return the line width.

6.119.3.10 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.

6.119.3.11 float mitkSurfaceProperty::GetPointSize () const [inline]

Get the point size. The size is expressed in screen units.

Returns:
Return the point size.

6.119.3.12 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).

6.119.3.13 const char * mitkSurfaceProperty::GetRepresentationTypeAsString () const [inline]

Get the geometry representation of the surface as a string.

Returns:
Return the string of geometry representation.

6.119.3.14 float mitkSurfaceProperty::GetSpecular () const [inline]

Get the specular coefficient.

Returns:
Return the specular coefficient.

6.119.3.15 float mitkSurfaceProperty::GetSpecularPower () const [inline]

Get the specular power.

Returns:
Return the specular power (between 0.0 and 128.0).

6.119.3.16 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.

6.119.3.17 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]);

6.119.3.18 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]);

6.119.3.19 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]);

6.119.3.20 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]);

6.119.3.21 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]);

6.119.3.22 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]);

6.119.3.23 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]);

6.119.3.24 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]);

6.119.3.25 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]);

6.119.3.26 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]);

6.119.3.27 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.

6.119.3.28 void mitkSurfaceProperty::SetAmbient (float *value*) [inline]

Set the ambient coefficient.

Parameters:
value the value of ambient coefficient

6.119.3.29 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)

6.119.3.30 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)

6.119.3.31 void mitkSurfaceProperty::SetDiffuse (float *value*) [inline]

Set the diffuse coefficient.

Parameters:
value the diffuse coefficient

6.119.3.32 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)

6.119.3.33 void mitkSurfaceProperty::SetInterpolationTypeToFlat () [inline]

Set the interpolation type for sampling a surface to MITK_SURFACE_FLAT.

6.119.3.34 void mitkSurfaceProperty::SetInterpolationTypeToGouraud () [inline]

Set the interpolation type for sampling a surface to MITK_SURFACE_GOURAUD.

6.119.3.35 void mitkSurfaceProperty::SetInterpolationTypeToPhong () [inline]

Set the interpolation type for sampling a surface to MITK_SURFACE_PHONG.

6.119.3.36 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).

6.119.3.37 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

6.119.3.38 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)

6.119.3.39 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

6.119.3.40 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)

6.119.3.41 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)

6.119.3.42 void mitkSurfaceProperty::SetRepresentationTypeToPoints () [inline]

Set the geometry representation of the surface to MITK_MESH_POINTS.

6.119.3.43 void mitkSurfaceProperty::SetRepresentationTypeToSurface () [inline]

Set the geometry representation of the surface to MITK_MESH_SURFACE.

6.119.3.44 void mitkSurfaceProperty::SetRepresentationTypeToWireframe () [inline]

Set the geometry representation of the surface to MITK_MESH_WIREFRAME.

6.119.3.45 void mitkSurfaceProperty::SetSpecular (float *value*) [inline]

Set the specular coefficient.

Parameters:
value the specular coefficient

6.119.3.46 void mitkSurfaceProperty::SetSpecularPower (float *value*) [inline]

Set the specular power.

Parameters:
value the specular power (between 0.0 and 128.0)

6.119.3.47 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:

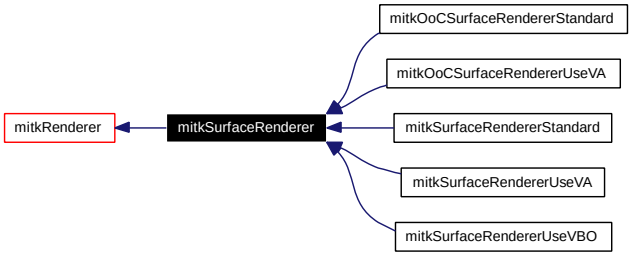
- mitkSurfaceProperty.h

6.120 mitkSurfaceRenderer Class Reference

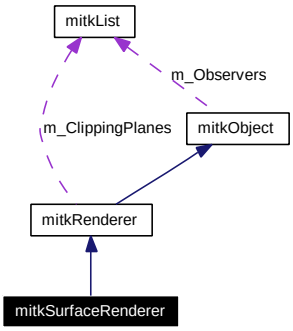
mitkSurfaceRenderer - an abstract class for surface renderer object

`#include <mitkSurfaceRenderer.h>`

Inheritance diagram for mitkSurfaceRenderer:



Collaboration diagram for mitkSurfaceRenderer:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int **Render** (mitkView *view, mitkSurfaceModel *surf)=0

6.120.1 Detailed Description

mitkSurfaceRenderer - an abstract class for surface renderer object

mitkSurfaceRenderer is an abstract class for surface renderer object.

6.120.2 Member Function Documentation

6.120.2.1 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:

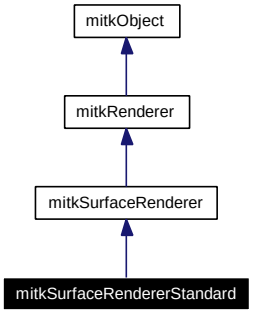
- mitkSurfaceRenderer.h

6.121 mitkSurfaceRendererStandard Class Reference

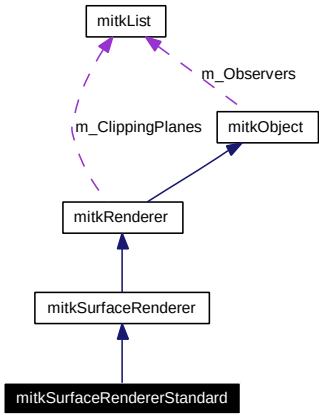
mitkSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

`#include <mitkSurfaceRendererStandard.h>`

Inheritance diagram for mitkSurfaceRendererStandard:



Collaboration diagram for mitkSurfaceRendererStandard:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int **Render** (mitkView *view, mitkSurfaceModel *surf)

Protected Member Functions

- void **_setMaterial** (mitkSurfaceProperty *prop)
- void **_drawPoints** (unsigned int vertexNum, float *vertexData)
- void **_drawWireFrame** (unsigned int faceNum, float *vertexData, unsigned int *faceData)
- void **_drawSurface** (unsigned int faceNum, float *vertexData, unsigned int *faceData)

6.121.1 Detailed Description

mitkSurfaceRendererStandard - a standard renderer object for mitkSurfaceModel

mitkSurfaceRendererStandard is a standard renderer object for mitkSurfaceModel using general OpenGL.

6.121.2 Member Function Documentation

6.121.2.1 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:

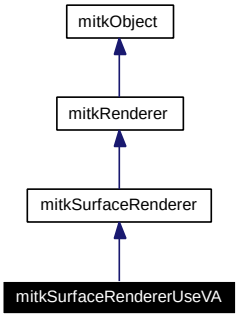
- mitkSurfaceRendererStandard.h

6.122 mitkSurfaceRendererUseVA Class Reference

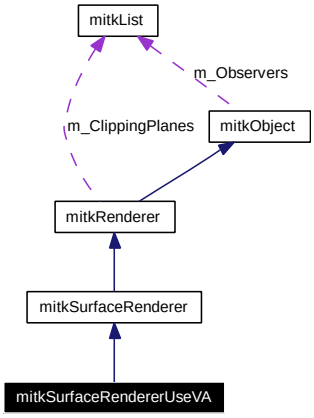
mitkSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

`#include <mitkSurfaceRendererUseVA.h>`

Inheritance diagram for mitkSurfaceRendererUseVA:



Collaboration diagram for mitkSurfaceRendererUseVA:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int **Render** (mitkView *view, mitkSurfaceModel *surf)

Protected Member Functions

- void **_setMaterial** (mitkSurfaceProperty *prop)
- bool **_buildArrays** (mitkMesh *mesh)
- void **_clearArrays** ()

Protected Attributes

- float * **m_Vertices**
- unsigned int * **m_Faces**

- unsigned int * **m_Edges**
- unsigned long **m_VertNum**
- unsigned long **m_FaceNum**
- unsigned long **m_EdgeNum**

6.122.1 Detailed Description

mitkSurfaceRendererUseVA - a renderer for mitkSurfaceModel using vertex array

mitkSurfaceRendererUseVA is a renderer for mitkSurfaceModel using vertex array

6.122.2 Member Function Documentation

6.122.2.1 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:

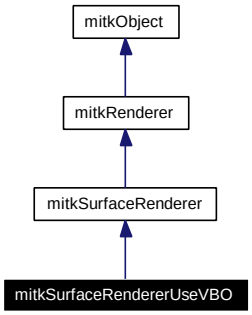
- mitkSurfaceRendererUseVA.h

6.123 mitkSurfaceRendererUseVBO Class Reference

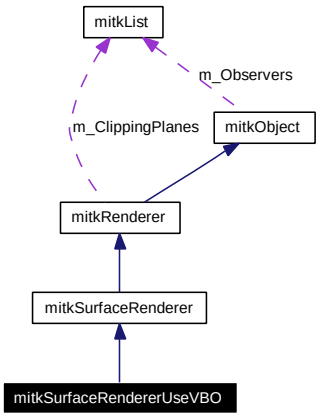
mitkSurfaceRendererUseVBO - a renderer for mitkSurfaceModel using vertex buffer object

`#include <mitkSurfaceRendererUseVBO.h>`

Inheritance diagram for mitkSurfaceRendererUseVBO:



Collaboration diagram for mitkSurfaceRendererUseVBO:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int **Render** (mitkView *view, mitkSurfaceModel *surf)

Protected Member Functions

- void **_setMaterial** (mitkSurfaceProperty *prop)
- bool **_buildVBOs** (mitkMesh *mesh)
- bool **_buildArrays** (mitkMesh *mesh)
- void **_clearArrays** ()

Protected Attributes

- PFNGLGENBUFFERSARBPROC **glGenBuffersARB**

- PFNGLBINDBUFFERARBPROC **glBindBufferARB**
- PFNGLBUFFERDATAARBPROC **glBufferDataARB**
- PFNGLDELETEBUFFERSARBPROC **glDeleteBuffersARB**
- GLuint **m_VertVBO**
- GLuint **m_FaceVBO**
- GLuint **m_EdgeVBO**
- float * **m_Vertices**
- unsigned int * **m_Faces**
- unsigned int * **m_Edges**
- unsigned long **m_VertNum**
- unsigned long **m_FaceNum**
- unsigned long **m_EdgeNum**
- bool **m_IsVBOSupported**
- bool **m_VBOBuilt**
- bool **m_NormalReversed**

6.123.1 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).

6.123.2 Member Function Documentation

6.123.2.1 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:

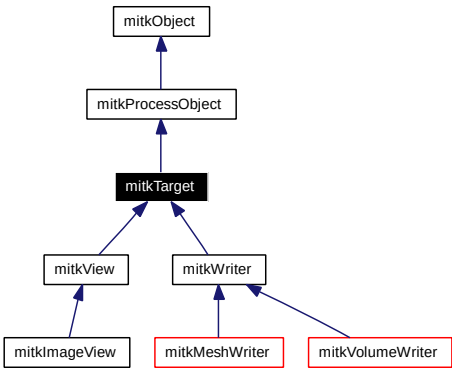
- mitkSurfaceRendererUseVBO.h

6.124 mitkTarget Class Reference

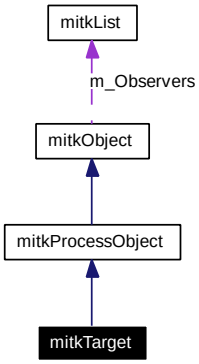
mitkTarget - abstract class specifies interface for Target object

`#include <mitkTarget.h>`

Inheritance diagram for mitkTarget:



Collaboration diagram for mitkTarget:



Public Member Functions

- virtual void PrintSelf (ostream &os)

6.124.1 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.

6.124.2 Member Function Documentation

6.124.2.1 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 mitkBMPWriter, mitkCacheVolumeWriter, mitkDICOMWriter, mitkImageView, mitkJPEGWriter, mitkMeshWriter, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkRawFilesWriter, mitkRawWriter, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkTIFFWriter, mitkView, mitkVolumeWriter, and mitkWriter.

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

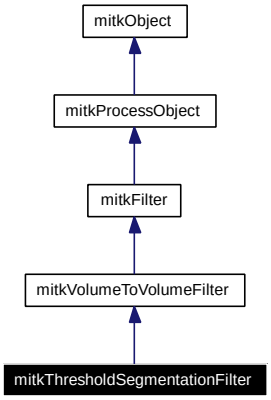
- mitkTarget.h

6.125 mitkThresholdSegmentationFilter Class Reference

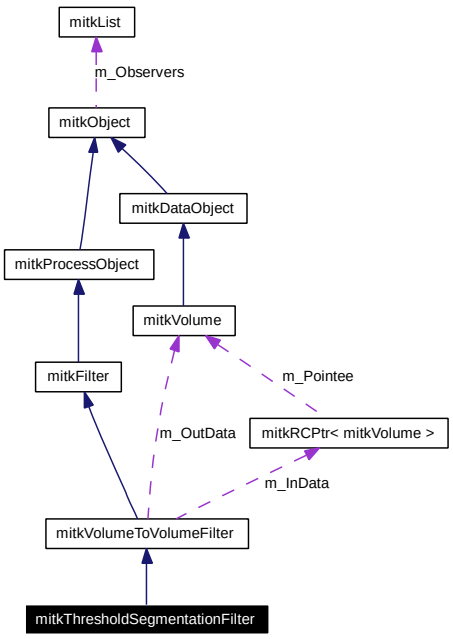
mitkThresholdSegmentationFilter - a class implement threshold segmentation arithmetic

`#include <mitkThresholdSegmentationFilter.h>`

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 ()

Protected Member Functions

- virtual bool **Execute** ()

6.125.1 Detailed Description

mitkThresholdSegmentationFilter - a class implement threshold segmentation arithmetic

mitkThresholdSegmentationFilter implement threshold segmentation arithmetic. The output is of the same datatype of the input.

6.125.2 Constructor & Destructor Documentation

6.125.2.1 mitkThresholdSegmentationFilter::mitkThresholdSegmentationFilter ()

Constructor of the class

6.125.3 Member Function Documentation

6.125.3.1 int mitkThresholdSegmentationFilter::GetHighThreshValue () [inline]

Get the high thresh value used by this class

Returns:
Return the high thresh value

6.125.3.2 int mitkThresholdSegmentationFilter::GetLowThreshValue () [inline]

Get the low thresh value used by this class

Returns:
Return the low thresh value

6.125.3.3 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.

6.125.3.4 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

6.125.3.5 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:

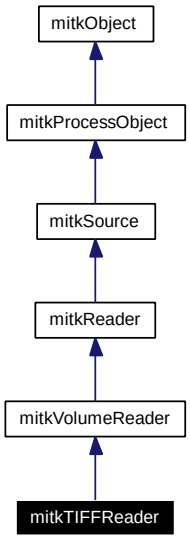
- mitkThresholdSegmentationFilter.h

6.126 mitkTIFFReader Class Reference

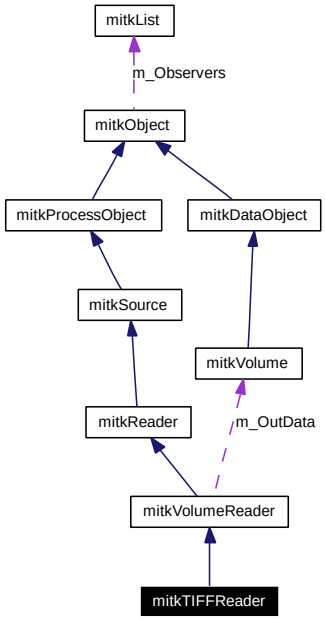
mitkTIFFReader - a concrete reader for reading TIFF image files

`#include <mitkTIFFReader.h>`

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])

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- float **m_Spacings** [3]

6.126.1 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.

6.126.2 Member Function Documentation

6.126.2.1 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.

6.126.2.2 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.

6.126.2.3 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.

6.126.2.4 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.

6.126.2.5 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:

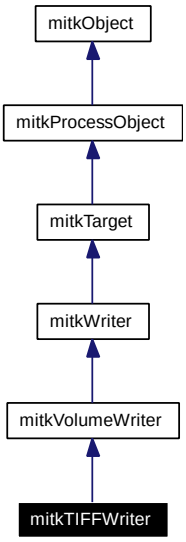
- mitkTIFFReader.h

6.127 mitkTIFFWriter Class Reference

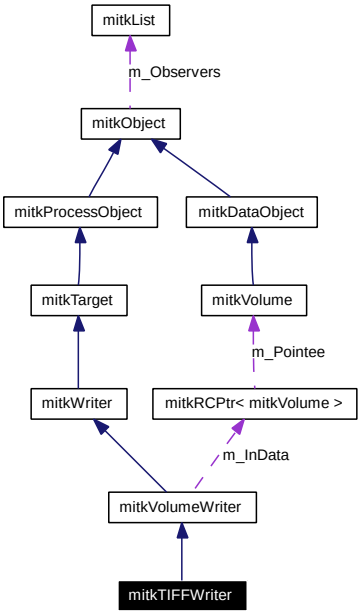
mitkTIFFWriter - a concrete writer for writing a volume to TIFF image files

`#include <mitkTIFFWriter.h>`

Inheritance diagram for mitkTIFFWriter:



Collaboration diagram for mitkTIFFWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)

Protected Member Functions

- virtual bool **Execute** ()

6.127.1 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();
```

6.127.2 Member Function Documentation

6.127.2.1 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:

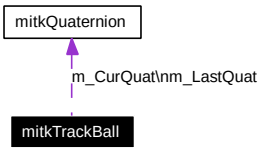
- mitkTIFFWriter.h

6.128 mitkTrackBall Class Reference

mitkTrackBall - a virtual trackball for tracking the movement of mouse

`#include <mitkTrackBall.h>`

Collaboration diagram for mitkTrackBall:



Public Member Functions

- void **Start** (float x, float y)
- void **Update** (float x, float y)
- void **BuildRotMatrix** (mitkMatrix &mat)
- const mitkQuaternion & **GetCurrentQuat** ()
- const mitkQuaternion & **GetUpdateQuat** ()

6.128.1 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:

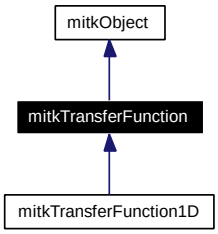
- mitkTrackBall.h

6.129 mitkTransferFunction Class Reference

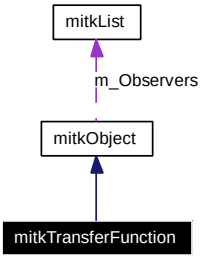
mitkTransferFunction - an abstract class to map the data property to opacity

`#include <mitkTransferFunction.h>`

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 ()

Protected Attributes

- float * **m_Data**
- bool **m_Modified**

6.129.1 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

6.129.2 Member Function Documentation

6.129.2.1 float* mitkTransferFunction::GetData () [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.

6.129.2.2 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.

6.129.2.3 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.

6.129.2.4 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.

6.129.2.5 void mitkTransferFunction::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:

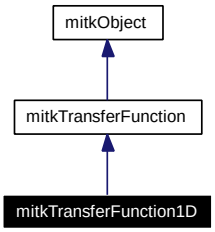
- mitkTransferFunction.h

6.130 mitkTransferFunction1D Class Reference

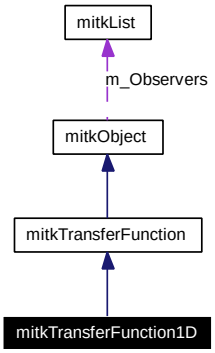
mitkTransferFunction1D - a concrete 1D transfer function to map the data property to opacity

`#include <mitkTransferFunction1D.h>`

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 ()

Protected Member Functions

- void **_buildFunction** (int x1, float val1, int x2, float val2)

Protected Attributes

- int **m_MaxX**
- mitkPointArray * **m_Points**

6.130.1 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.

6.130.2 Member Function Documentation

6.130.2.1 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

6.130.2.2 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

6.130.2.3 virtual int mitkTransferFunction1D::GetDimension () [inline, virtual]

Get the dimensionality of this transfer function.

Returns:

Always return 1

Implements mitkTransferFunction.

6.130.2.4 int mitkTransferFunction1D::GetMaxX () [inline]

Get the maximum data value of this transfer function.

Returns:

Return the maximum data value of this transfer function.

6.130.2.5 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

6.130.2.6 int mitkTransferFunction1D::GetPointsCount ()

Get the number of points in this transfer function

Returns:
Return the points number

6.130.2.7 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.

6.130.2.8 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.

6.130.2.9 void mitkTransferFunction1D::RemoveAllPoints ()

Remove all points from this transfer function

6.130.2.10 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.

6.130.2.11 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:

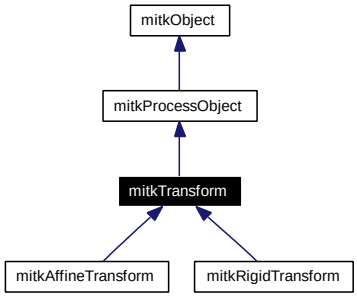
- mitkTransferFunction1D.h

6.131 mitkTransform Class Reference

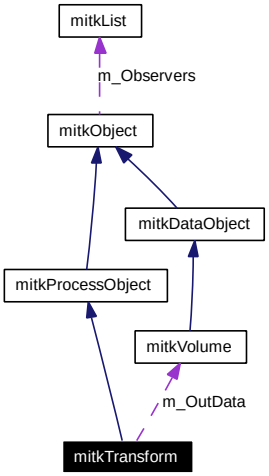
mitkTransform - an abstract class to perform coordinates transformation

`#include <mitkTransform.h>`

Inheritance diagram for mitkTransform:



Collaboration diagram for mitkTransform:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetParameters (vector< float > *parameters)
- void SetDimensions (int d[3])
- void SetWidth (int w)
- void SetDepth (int d)
- void SetImageNum (int s)
- void GetDimensions (int d[3])
- int GetWidth ()
- int GetDepth ()
- int GetImageNum ()
- void SetSpacings (float s[3])
- void SetSpacingX (float x)

- void SetSpacingY (float y)
- void SetSpacingZ (float z)
- void GetSpacings (float s[3])
- float GetSpacingX ()
- float GetSpacingY ()
- float GetSpacingZ ()
- mitkVolume * GetOutput ()
- vector< float > * GetParameters ()
- int GetNumberOfParameters ()
- void SetCentreTransformFlag (bool flag)
- void SetRegion (int r[6])
- void GetRegion (int r[6])
- void Update ()

Protected Member Functions

- virtual bool **Execute** ()
- virtual bool **_transform** (float out[3], float x, float y, float z)

Protected Attributes

- int **m_Dimensions** [3]
- int **m_Region** [3][2]
- float **m_Spacings** [3]
- bool **m_IsCentreTransform**
- int **m_NumberOfParameters**
- vector< float > * **m_Parameters**
- mitkVolume * **m_OutData**

6.131.1 Detailed Description

mitkTransform - an abstract class to perform coordinates transformation

mitkTransform is an abstract class to perform coordinates transformation, which will conver one point from the Reference object space to the Target object space.

6.131.2 Member Function Documentation

6.131.2.1 int mitkTransform::GetDepth () `[inline]`

Get the depth of fixed volume.

Returns:
Return the depth of fixed volume.

6.131.2.2 void mitkTransform::GetDimensions (int *d*[3]) [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.

6.131.2.3 int mitkTransform::GetImageNum () [inline]

Get the image number of fixed volume.

Returns:

Return the image number of fixed volume.

6.131.2.4 int mitkTransform::GetNumberOfParameters () [inline]

Get number of transform parameters.

Returns:

Return the number of transform parameters.

6.131.2.5 mitkVolume* mitkTransform::GetOutput ()

Get the output volume.

Returns:

Return the output volume.

6.131.2.6 vector<float>* mitkTransform::GetParameters ()

Get Transorm Parameters.

Returns:

Return the transform parameters.

6.131.2.7 void mitkTransform::GetRegion (int *r*[6]) [inline]

Get the region of transform

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.

6.131.2.8 void mitkTransform::GetSpacings (float *s*[3]) [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()

6.131.2.9 float mitkTransform::GetSpacingX () [inline]

Get the spacing (mm) in two adjacent voxels in x axis.

Returns:

Return the spacing (mm) in two adjacent voxels in x axis.

6.131.2.10 float mitkTransform::GetSpacingY () [inline]

Get the spacing (mm) in two adjacent voxels in y axis.

Returns:

Return the spacing (mm) in two adjacent voxels in y axis.

6.131.2.11 float mitkTransform::GetSpacingZ () [inline]

Get the spacing (mm) in two adjacent voxels in z axis.

Returns:

Return the spacing (mm) in two adjacent voxels in z axis.

6.131.2.12 int mitkTransform::GetWidth () [inline]

Get the width of fixed volume.

Returns:

Return the width of fixed volume.

6.131.2.13 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, and mitkRigidTransform.

6.131.2.14 void mitkTransform::SetCentreTransformFlag (bool *flag*) [inline]

Set the flag of centre transform.

Parameters:
flag The flag of centre transform.

6.131.2.15 void mitkTransform::SetDepth (int *d*) [inline]

Set the depth of fixed volume (dimension in y).

Parameters:
d The depth of fixed volume, the unit is pixel.

6.131.2.16 void mitkTransform::SetDimensions (int *d*[3]) [inline]

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.

6.131.2.17 void mitkTransform::SetImageNum (int *s*) [inline]

Set the image number of fixed volume (dimension in z).

Parameters:
d The image number of fixed volume, the unit is pixel.

6.131.2.18 void mitkTransform::SetParameters (vector< float > * *parameters*)

Set the transform parameters.

Parameters:
parameters The transform parameters.

6.131.2.19 void mitkTransform::SetRegion (int *r*[6]) [inline]

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.

6.131.2.20 void mitkTransform::SetSpacings (float *s*[3]) [inline]

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.

6.131.2.21 void mitkTransform::SetSpacingX (float *x*) [inline]

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

Parameters:

x The spacing (mm) in two adjacent voxels in x axis.

6.131.2.22 void mitkTransform::SetSpacingY (float *y*) [inline]

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

Parameters:

y The spacing (mm) in two adjacent voxels in y axis.

6.131.2.23 void mitkTransform::SetSpacingZ (float *z*) [inline]

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

Parameters:

z The spacing (mm) in two adjacent voxels in z axis.

6.131.2.24 void mitkTransform::SetWidth (int *w*) [inline]

Set the width of fixed volume (dimension in x).

Parameters:

w The width of fixed volume, the unit is pixel.

6.131.2.25 void mitkTransform::Update ()

Initialization, perform before Run() function

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

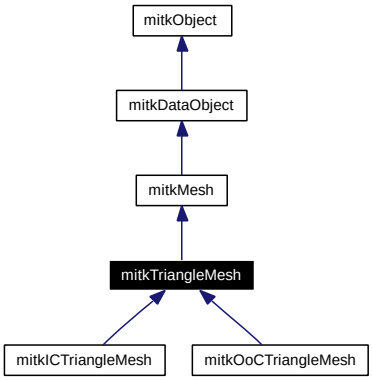
- mitkTransform.h

6.132 mitkTriangleMesh Class Reference

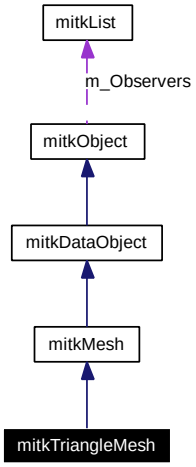
mitkTriangleMesh - an abstract class for triangle meshes

`#include <mitkTriangleMesh.h>`

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)

Protected Attributes

- size_type **m_VertNum**
- size_type **m_FaceNum**
- size_type **m_MaxVertNum**
- size_type **m_MaxFaceNum**

6.132.1 Detailed Description

mitkTriangleMesh - an abstract class for triangle meshes

mitkTriangleMesh is an abstract class for triangle meshes.

See also:
mitkICTriangleMesh and mitkOoCTriangleMesh

6.132.2 Constructor & Destructor Documentation

6.132.2.1 mitkTriangleMesh::mitkTriangleMesh ()

Default constructor of this class.

6.132.3 Member Function Documentation

6.132.3.1 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.

6.132.3.2 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.

6.132.3.3 virtual void mitkTriangleMesh::DeepCopy (mitkDataObject * *src*) [virtual]

Deep copy.

Parameters:
src pointer to the source mitkDataObject

Reimplemented from mitkMesh.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

6.132.3.4 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.

6.132.3.5 virtual size_type mitkTriangleMesh::GetFaceNumber () const [inline, virtual]

Get the mesh's face number.

Returns:
Return the number of faces.

Implements mitkMesh.

6.132.3.6 virtual size_type mitkTriangleMesh::GetVertexNumber () const [inline, virtual]

Get the mesh's vertex number.

Returns:
Return the number of vertices.

Implements mitkMesh.

6.132.3.7 virtual void mitkTriangleMesh::Initialize () [virtual]

Make the output data ready for new data to be inserted.

Reimplemented from mitkMesh.

Reimplemented in mitkICTriangleMesh, and mitkOoCTriangleMesh.

6.132.3.8 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.

6.132.3.9 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:

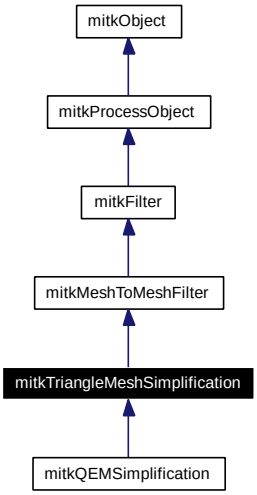
- mitkTriangleMesh.h

6.133 mitkTriangleMeshSimplification Class Reference

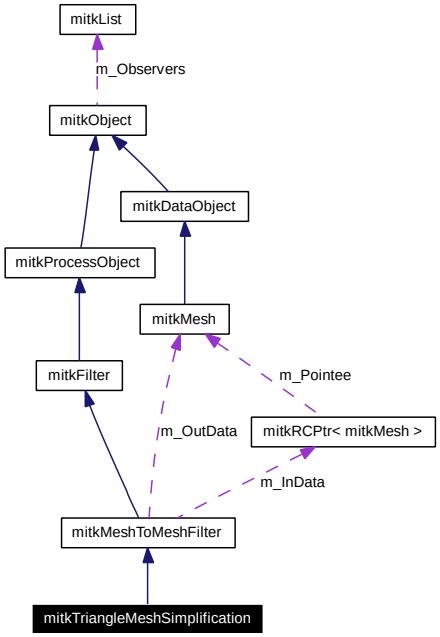
mitkTriangleMeshSimplification - abstract class for triangle mesh simplification algorithms

`#include <mitkTriangleMeshSimplification.h>`

Inheritance diagram for mitkTriangleMeshSimplification:



Collaboration diagram for mitkTriangleMeshSimplification:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetTargetFaceNumber (size_type num)

Protected Member Functions

- virtual mitkMesh * **_createOutMesh** ()
- virtual bool **_simplificationProcess** (mitkHETriangleMesh *mesh)=0
- virtual bool **Execute** ()

Protected Attributes

- size_type **m_TargetFaceNumber**

6.133.1 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.

6.133.2 Member Function Documentation

6.133.2.1 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.

6.133.2.2 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

6.134 mitkVector Class Reference

mitkVector - an encapsulation of a 4-element vector

`#include <mitkMatrix.h>`

Public Member Functions

- **mitkVector** (const mitkVector &v)
- **mitkVector** (const float x, const float y, const float z, const float w=1.0f)
- **mitkVector** (const float srcVector[4])
- **operator float** * ()
- **operator const float** * () const
- mitkVector & **operator=** (const mitkVector &a)
- mitkVector & **operator** *= (const mitkMatrix &)
- mitkVector & **operator** *= (const float)
- mitkVector & **operator+=** (const mitkVector &)
- mitkVector & **operator-=** (const mitkVector &)
- float **Length** ()
- float **Length2** ()
- void **Normalize** ()

Public Attributes

- float **ele** [4]

Friends

- mitkVector **operator** * (const mitkVector &, const mitkMatrix &)
- float **operator** * (const mitkVector &, const mitkVector &)
- mitkVector **operator%** (const mitkVector &, const mitkVector &)
- mitkVector **operator** * (const mitkVector &, const float)
- mitkVector **operator** * (const float, const mitkVector &)
- mitkVector **operator+** (const mitkVector &)
- mitkVector **operator+** (const mitkVector &, const mitkVector &)
- mitkVector **operator-** (const mitkVector &)
- mitkVector **operator-** (const mitkVector &, const mitkVector &)
- mitkVector **operator**~ (const mitkVector &)

6.134.1 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:

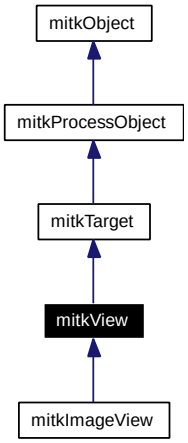
- mitkMatrix.h

6.135 mitkView Class Reference

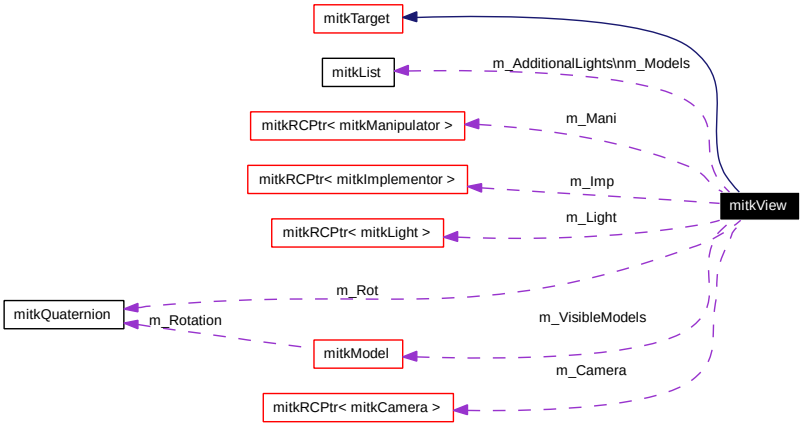
mitkView - a view to display 3D surface rendered or volume rendered image

`#include <mitkView.h>`

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)
- 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 ()
- 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)
- int const * GetViewPort ()
- void GetViewPort (int vp[4])
- const char * GetWindowName ()
- void SetWindowName (const char *winName)
- void Show ()
- void Hide ()
- void Update ()
- void MakeCurrent ()
- void SwapBuffers ()
- virtual void OnCreate ()
- virtual void OnDestroy ()
- virtual void OnSize (int newX, int newY)
- virtual void OnDraw ()
- 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 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])
- float GetRotationX ()
- float GetRotationY ()
- float GetRotationZ ()
- const mitkQuaternion & GetRotation ()
- void GetRotation (float rots[3])
- float GetScale ()
- bool HasUnprocessedMouseMessage ()
- void EnableLoD ()
- void DisableLoD ()
- bool IsLoDEnabled ()
- double GetFrameSpeed ()
- double GetRenderTime ()
- void EnableCalculateFrameSpeed ()
- void DisableCalculateFrameSpeed ()

Protected Member Functions

- void **_drawScene** ()
- void **_drawBorder** ()
- void **_drawLights** ()
- bool **_isAdditionalLightsModified** ()
- void **_rotateModels** ()

Protected Attributes

- mitkString * **m_WindowName**
- int **m_Size** [2]
- int **m_Position** [2]
- int **m_HaveTitleBar**
- float **m_RotX**
- float **m_RotY**
- float **m_RotZ**
- float **m_RotateSpeed**
- float **m_RotDeltX**
- float **m_RotDeltY**
- float **m_RotDeltZ**
- mitkQuaternion **m_Rot**
- float **m_TransX**
- float **m_TransY**
- float **m_TransZ**
- float **m_TranslateSpeed**
- float **m_Scale**
- float **m_ScaleSpeed**
- float **m_BackColor** [4]
- int **m_ViewPort** [4]
- mitkRCPtr< mitkManipulator > **m_Mani**
- mitkRCPtr< mitkImplementor > **m_Imp**
- mitkRCPtr< mitkCamera > **m_Camera**
- mitkRCPtr< mitkLight > **m_Light**
- mitkList * **m_AdditionalLights**
- mitkList * **m_Models**
- mitkModel ** **m_VisibleModels**
- int **m_NumberOfPropsRendered**
- int **m_RotAccuCount**
- unsigned char **m_VisibleModelCount**
- unsigned char **m_CameraModified**
- unsigned char **m_AdditionalLightModified**
- unsigned char **m_Selected**
- unsigned char **m_ModelsModified**

Friends

- class mitkWin32Implementor

6.135.1 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.

6.135.2 Member Function Documentation

6.135.2.1 void mitkView::AddAdditionalLight (mitkLight * *lit*)

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.

6.135.2.2 void mitkView::AddModel (mitkModel * *model*)

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.

6.135.2.3 void mitkView::AlignScene ()

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

6.135.2.4 virtual mitkManipulator* mitkView::CreateManipulator () [virtual]

Create a default manipulator for the mouse events processing. Subclass can override this function to create a proper manipulator.

Reimplemented in mitkImageView.

6.135.2.5 void mitkView::DisableCalculateFrameSpeed ()

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.

6.135.2.6 void mitkView::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.

6.135.2.7 void mitkView::EnableCalculateFrameSpeed ()

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.

6.135.2.8 void mitkView::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.

6.135.2.9 mitkLight* mitkView::GetAdditionalLight (int *idx*)

Get the idx'th additional light.

Parameters:
idx the index of the light to get

6.135.2.10 int mitkView::GetAdditionalLightCount ()

Get the count of additional lights.

Returns:
Return the count of additional lights.

6.135.2.11 void* mitkView::GetApplicationId (void) [inline]

OS dependent function, don't call it.

6.135.2.12 void mitkView::GetBackColor (int & *rColor*, int & *gColor*, int & *bColor*)

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.

6.135.2.13 void mitkView::GetBackColor (unsigned char & *rColor*, unsigned char & *gColor*, unsigned char & *bColor*)

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.

6.135.2.14 void mitkView::GetBackColor (float & *rColor*, float & *gColor*, float & *bColor*)

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.

6.135.2.15 mitkCamera* mitkView::GetCamera ()

Get the camera of this view.

Returns:
Return the camera of this view.

6.135.2.16 mitkLight* mitkView::GetDefaultLight ()

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.

6.135.2.17 void* mitkView::GetDeviceContext (void) [inline]

OS dependent function, don't call it.

6.135.2.18 double mitkView::GetFrameSpeed ()

Get frame speed.

Returns:
Return the frame speed.

6.135.2.19 int mitkView::GetHeight () [inline]

Get the Height of this view

Returns:
Return the height of this view

6.135.2.20 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

6.135.2.21 mitkLight* mitkView::GetLight () [inline]

Provided for back compatibility, just the same as SetSliceNum().

6.135.2.22 mitkModel* mitkView::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 view.

6.135.2.23 int mitkView::GetModelCount ()

Get the model count in this view.

Returns:
Return the model count in this view.

6.135.2.24 mitkList* mitkView::GetModels ()

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.

6.135.2.25 int mitkView::GetNumberOfPropsRendered () [inline]

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

6.135.2.26 void* mitkView::GetParent (void) [inline]

OS dependent function, don't call it.

6.135.2.27 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.

6.135.2.28 unsigned char* mitkView::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.

6.135.2.29 void mitkView::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.

6.135.2.30 void mitkView::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.

6.135.2.31 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.

6.135.2.32 void∗ mitkView::GetRenderContext (void) [inline]

OS dependent function, don't call it.

6.135.2.33 double mitkView::GetRenderTime ()

Get the elapsed time of render process in seconds.

Returns:
the elapsed time in seconds

6.135.2.34 void mitkView::GetRotation (float *rots*[3]) [inline]

Get the Rotation of the view. (obsolete, do not use it any more)

Parameters:
rots[0] return rotation around x-axis

rots[1] return rotation around y-axis

rots[2] return rotation around z-axis

Warning:
This function is provided for convenience. Don't call it directly.

6.135.2.35 const mitkQuaternion& mitkView::GetRotation (void) [inline]

Get the Rotation of the view.

Returns:
Return the quaternion of the rotation.

6.135.2.36 float mitkView::GetRotationX () [inline]

Get the rotation around x-axis of the view. (obsolete, do not use it any more)

Returns:
Return the rotation around x-axis of the view.

Warning:
This function is provided for convenience. Don't call it directly.

6.135.2.37 float mitkView::GetRotationY () [inline]

Get the rotation around y-axis of the view. (obsolete, do not use it any more)

Returns:
Return the rotation around y-axis of the view.

Warning:
This function is provided for convenience. Don't call it directly.

6.135.2.38 float mitkView::GetRotationZ () [inline]

Get the rotation around z-axis of the view. (obsolete, do not use it any more)

Returns:
Return the rotation around z-axis of the view.

Warning:
This function is provided for convenience. Don't call it directly.

6.135.2.39 float mitkView::GetScale (void) [inline]

Get the scale of the view.

Returns:
Return the scale of the view.

6.135.2.40 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.

6.135.2.41 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.

6.135.2.42 int mitkView::GetTitleBar () [inline]

OS dependent function, don't call it.

6.135.2.43 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

6.135.2.44 float mitkView::GetTranslateSpeed () [inline]

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

6.135.2.45 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.

6.135.2.46 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.

6.135.2.47 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.

6.135.2.48 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.

6.135.2.49 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

6.135.2.50 int const* 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.

6.135.2.51 int mitkView::GetWidth () [inline]

Get the Width of this view

Returns:
Return the width of this view

6.135.2.52 void* mitkView::GetWindowId (void) [inline]

OS dependent function, don't call it.

6.135.2.53 const char* mitkView::GetWindowName ()

OS dependent function, don't call it.

6.135.2.54 void mitkView::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.

6.135.2.55 float* mitkView::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.

6.135.2.56 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.

6.135.2.57 void mitkView::Hide ()

Hide this view.

6.135.2.58 bool mitkView::IsLoDEnabled ()

Test if the LoD mode is enabled.

Returns:

Return true if the LoD mode is enabled, otherwise return false.

6.135.2.59 void mitkView::MakeCurrent ()

OS dependent function, don't call it.

6.135.2.60 virtual void mitkView::OnCreate () [virtual]

OS dependent function, don't call it.

Reimplemented in mitkImageView.

6.135.2.61 virtual void mitkView::OnDestroy () [virtual]

OS dependent function, don't call it.

6.135.2.62 virtual void mitkView::OnDraw () [virtual]

OS dependent function, don't call it.

Reimplemented in mitkImageView.

6.135.2.63 void mitkView::OnMouseDown (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

OS dependent function, don't call it.

6.135.2.64 void mitkView::OnMouseMove (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

OS dependent function, don't call it.

6.135.2.65 void mitkView::OnMouseUp (int *mouseButton*, bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*)

OS dependent function, don't call it.

6.135.2.66 void mitkView::OnMouseWheel (bool *ctrlDown*, bool *shiftDown*, int *xPos*, int *yPos*, int *delta*)

OS dependent function, don't call it.

6.135.2.67 virtual void mitkView::OnSize (int *newX*, int *newY*) [virtual]

OS dependent function, don't call it.

Reimplemented in mitkImageView.

6.135.2.68 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 mitkTarget.

Reimplemented in mitkImageView.

6.135.2.69 void mitkView::RemoveAdditionalLight (mitkLight * *lit*)

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.

6.135.2.70 void mitkView::RemoveAdditionalLight (int *idx*)

Remove the additional light in specified position.

Parameters:
idx zero based index to specify the position of the additional light

6.135.2.71 void mitkView::RemoveAllAdditionalLights ()

Remove all additional lights in this view.

6.135.2.72 void mitkView::RemoveAllModel () [inline]

Provided for back compatibility, just the same as SetSliceNum().

6.135.2.73 void mitkView::RemoveAllModels ()

Remove all models in this view.

6.135.2.74 void mitkView::RemoveModel (int *i*)

Remove the model in specified position.

Parameters:
i Zero based index to specify the position of the model.

6.135.2.75 void mitkView::RemoveModel (mitkModel * *model*)

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.

6.135.2.76 virtual void mitkView::ResetScene () [virtual]

Reset the geometric position of all of the models in this view.

Reimplemented in mitkImageView.

6.135.2.77 virtual void mitkView::Rotate (const mitkQuaternion & *q*) [virtual]

Rotate all of the models in this view.

Parameters:

q the quaternion of the rotation

Note:

The rotation is incremental.

Reimplemented in mitkImageView.

6.135.2.78 virtual void mitkView::Rotate (float *ax*, float *ay*, float *az*, float *angle*) [virtual]

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.

Reimplemented in mitkImageView.

6.135.2.79 virtual void mitkView::Rotate (float *deltX*, float *deltY*, float *deltZ*) [virtual]

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.

Reimplemented in mitkImageView.

6.135.2.80 virtual void mitkView::Scale (float *delt*) [virtual]

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.

Reimplemented in mitkImageView.

6.135.2.81 void mitkView::SetBackColor (int *rColor*, int *gColor*, int *bColor*)

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.

6.135.2.82 void mitkView::SetBackColor (float *rColor*, float *gColor*, float *bColor*)

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.

6.135.2.83 void mitkView::SetBackColor (unsigned char *rColor*, unsigned char *gColor*, unsigned char *bColor*)

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.

6.135.2.84 void mitkView::SetCamera (mitkCamera * *cam*)

Set the camera of this view.

Parameters:

cam the new camera of this view

6.135.2.85 void mitkView::SetDefaultLight (mitkLight * *lit*)

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.

6.135.2.86 void mitkView::SetHeight (int *heightSize*)

Set the Height of this view

Parameters:
heightSize Specify the height of this view.

6.135.2.87 void mitkView::SetLeft (int *leftPos*)

Set the Left position in local coordinate of its parent window

Parameters:
leftPos Specify the Left coordinate.

6.135.2.88 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

6.135.2.89 void mitkView::SetParent (void * *parentId*)

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.

6.135.2.90 void mitkView::SetPosition (int *a*[2])

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.

6.135.2.91 void mitkView::SetPosition (int *leftPos*, int *topPos*)

Set the position (Left and Top) in local coordinate of its parent window

Parameters:

leftPos Specify the Left coordinate.

topPos Specify the Top coordinate.

6.135.2.92 virtual void mitkView::SetRotation (const mitkQuaternion & *q*) [virtual]

Set rotations of all models in this view to the quaternion "q".

Parameters:

q the quaternion of the rotation

Reimplemented in mitkImageView.

6.135.2.93 virtual void mitkView::SetRotation (float *ax*, float *ay*, float *az*, float *angle*) [virtual]

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

Reimplemented in mitkImageView.

6.135.2.94 virtual void mitkView::SetRotation (float *x*, float *y*, float *z*) [virtual]

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.

Reimplemented in mitkImageView.

6.135.2.95 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.

6.135.2.96 void mitkView::SetSize (int *a*[2])

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.

6.135.2.97 void mitkView::SetSize (int *widthSize*, int *heightSize*)

Set the size (Width and Height) of this view

Parameters:
widthSize Specify the width of this view.
heightSize Specify the height of this view.

6.135.2.98 void mitkView::SetTitleBar (int *titleBar*)

OS dependent function, don't call it.

6.135.2.99 void mitkView::SetTitleBarOff ()

OS dependent function, don't call it.

6.135.2.100 void mitkView::SetTitleBarOn ()

OS dependent function, don't call it.

6.135.2.101 void mitkView::SetTop (int *topPos*)

Get the Top position in local coordinate of its parent window

Parameters:
topPos Specify the Top coordinate.

6.135.2.102 void mitkView::SetWidth (int *widthSize*)

Set the Width of this view

Parameters:
widthSize Specify the width of this view.

6.135.2.103 void mitkView::SetWindowName (const char * *winName*)

OS dependent function, don't call it.

6.135.2.104 void mitkView::Show ()

Show this view. Usually call it when a view is created. And the later updating of this view should call Update().

6.135.2.105 void mitkView::SwapBuffers ()

OS dependent function, don't call it.

6.135.2.106 virtual void mitkView::Translate (float *deltX*, float *deltY*, float *deltZ*) [virtual]

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

Reimplemented in mitkImageView.

6.135.2.107 void mitkView::Update ()

Update the content of this view.

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

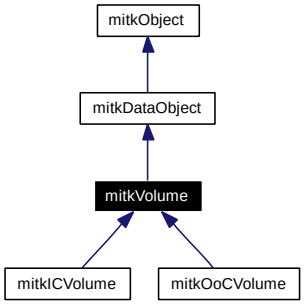
- mitkView.h

6.136 mitkVolume Class Reference

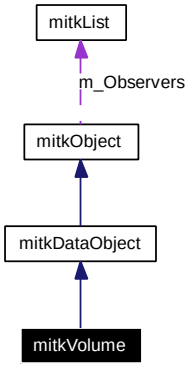
mitkVolume - an abstract class to represent a multi-dimensional medical image dataset

`#include <mitkVolume.h>`

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
- 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

Protected Member Functions

- void **_computeIncrements** ()
- void **_clearMinMaxCache** ()

Protected Attributes

- int **m_Dimensions** [3]
- int **m_Increments** [3]
- float **m_Spacings** [3]
- int **m_DataType**
- int **m_NumberOfChannel**
- float **m_WindowWidth**
- float **m_WindowCenter**
- bool **m_HasWidthCenter**
- bool **m_IsFirstWidthCenter**
- double * **m_MinArray**
- double * **m_MaxArray**
- unsigned char * **m_MinMaxFlags**
- int **m_DataTypeSize**
- double **m_DataTypeMin**
- double **m_DataTypeMax**

6.136.1 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;
}
```

6.136.2 Member Function Documentation

6.136.2.1 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.

6.136.2.2 virtual void mitkVolume::DeepCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Implements mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

6.136.2.3 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.

6.136.2.4 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.

6.136.2.5 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

6.136.2.6 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.

6.136.2.7 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.

6.136.2.8 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.

6.136.2.9 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.

6.136.2.10 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

6.136.2.11 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.

6.136.2.12 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.

6.136.2.13 int mitkVolume::GetDataTypeSize () const [inline]

Get the size of the data type in bytes.

Returns:
Return the size of the data type in bytes.

6.136.2.14 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()

6.136.2.15 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.

6.136.2.16 int mitkVolume::GetImageNum () const [inline]

Provided for convenience, just the same as GetSliceNum().

6.136.2.17 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()

6.136.2.18 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).

6.136.2.19 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).

6.136.2.20 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).

6.136.2.21 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.

6.136.2.22 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.

6.136.2.23 int mitkVolume::GetNumberOfChannel () const [inline]

Just the same as GetChannelNum().

6.136.2.24 void mitkVolume::GetPointGradient (int *i*, int *j*, int *k*, float *g*[3])

Useless. Obsolete.

6.136.2.25 void* mitkVolume::GetSliceData (int *sliceIdx*) [inline]

Provided for back compatibility, just the same as GetSliceForReadWrite().

6.136.2.26 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.

6.136.2.27 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.

6.136.2.28 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.

6.136.2.29 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.

6.136.2.30 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()

6.136.2.31 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.

6.136.2.32 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.

6.136.2.33 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.

6.136.2.34 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.

6.136.2.35 void mitkVolume::GetWidthCenter (float & *wWidth*, float & *wCenter*)

Get the window width and center for display in mitkImageView.

Parameters:
wWidth Return the window width of this volume.

wCenter Return the window center of this volume.

6.136.2.36 float mitkVolume::GetWindowCenter ()

Get the window center for display in mitkImageView.

Returns:
Return the window center of this volume.

6.136.2.37 float mitkVolume::GetWindowWidth ()

Get the window width for display in mitkImageView.

Returns:
Return the window width of this volume.

6.136.2.38 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

6.136.2.39 virtual void mitkVolume::Initialize () [virtual]

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

Implements mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

6.136.2.40 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.

6.136.2.41 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.

6.136.2.42 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.

6.136.2.43 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.

6.136.2.44 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.

6.136.2.45 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

6.136.2.46 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

6.136.2.47 void mitkVolume::SetDataTypeToChar () [inline]

Set data type of this volume to char.

6.136.2.48 void mitkVolume::SetDataTypeToDouble () [inline]

Set data type of this volume to double.

6.136.2.49 void mitkVolume::SetDataTypeToFloat () [inline]

Set data type of this volume to float.

6.136.2.50 void mitkVolume::SetDataTypeToInt () [inline]

Set data type of this volume to int.

6.136.2.51 void mitkVolume::SetDataTypeToLong () [inline]

Set data type of this volume to long.

6.136.2.52 void mitkVolume::SetDataTypeToShort () [inline]

Set data type of this volume to short.

6.136.2.53 void mitkVolume::SetDataTypeToUnsignedChar () [inline]

Set data type of this volume to unsigned char.

6.136.2.54 void mitkVolume::SetDataTypeToUnsignedInt () [inline]

Set data type of this volume to unsigned int.

6.136.2.55 void mitkVolume::SetDataTypeToUnsignedLong () [inline]

Set data type of this volume to unsigned long.

6.136.2.56 void mitkVolume::SetDataTypeToUnsignedShort () [inline]

Set data type of this volume to unsigned short.

6.136.2.57 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

6.136.2.58 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.

6.136.2.59 void mitkVolume::SetImageNum (int *s*) [inline]

Provided for convenience, just the same as SetSliceNum().

6.136.2.60 void mitkVolume::SetNumberOfChannel (int *n*) [inline]

Just the same as SetChannelNum().

6.136.2.61 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.

6.136.2.62 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.

6.136.2.63 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.

6.136.2.64 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.

6.136.2.65 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.

6.136.2.66 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.

6.136.2.67 void mitkVolume::SetWindowCenter (float *wCenter*) [inline]

Set the window center for display in mitkImageView.

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.

6.136.2.68 void mitkVolume::SetWindowWidth (float *wWidth*) [inline]

Set the window width for display in mitkImageView.

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.

6.136.2.69 virtual void mitkVolume::ShallowCopy (mitkDataObject * *src*) [virtual]

Warning:
Don't call this function directly.

Implements mitkDataObject.

Reimplemented in mitkICVolume, and mitkOoCVolume.

6.136.2.70 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.

6.136.2.71 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.

6.136.2.72 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:

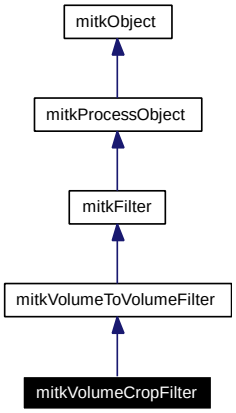
- mitkVolume.h

6.137 mitkVolumeCropFilter Class Reference

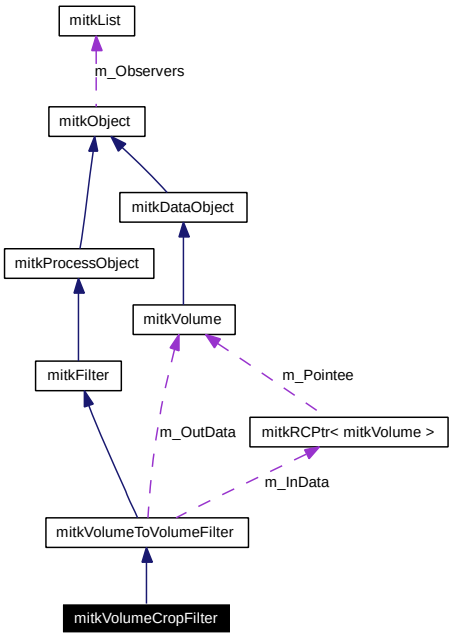
mitkVolumeCropFilter - A concrete Filter class to crop a volume

`#include <mitkVolumeCropFilter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- int **m_XBeginPoint**
- int **m_YBeginPoint**
- int **m_ZBeginPoint**
- int **m_XLength**
- int **m_YLength**
- int **m_ZLength**
- bool **flag**

6.137.1 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.

6.137.2 Member Function Documentation

6.137.2.1 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.

6.137.2.2 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)

6.137.2.3 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:

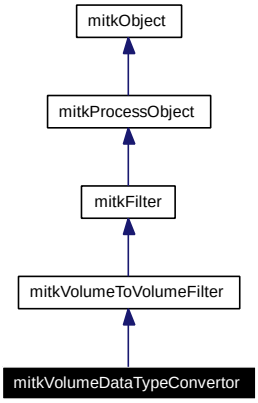
- mitkVolumeCropFilter.h

6.138 mitkVolumeDataTypeConvertor Class Reference

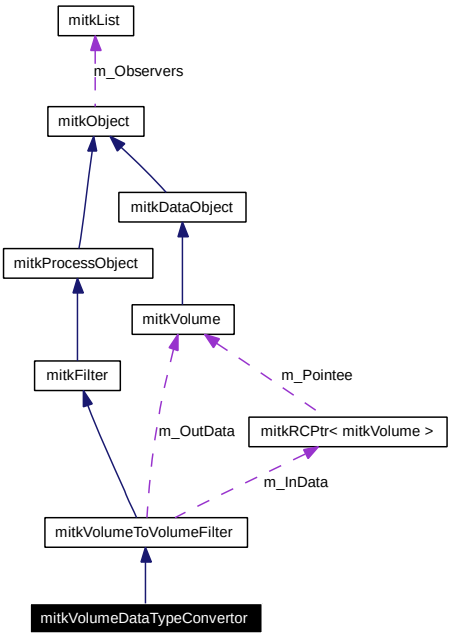
mitkVolumeDataTypeConvertor - a filter to convert the type of volume data

`#include <mitkVolumeDataTypeConvertor.h>`

Inheritance diagram for mitkVolumeDataTypeConvertor:



Collaboration diagram for mitkVolumeDataTypeConvertor:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetOutputDataType (int dataType)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- int **m_OutDataType**

6.138.1 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.

6.138.2 Member Function Documentation

6.138.2.1 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.

6.138.2.2 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:

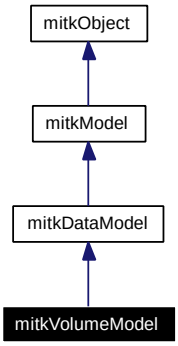
- mitkVolumeDataTypeConvertor.h

6.139 mitkVolumeModel Class Reference

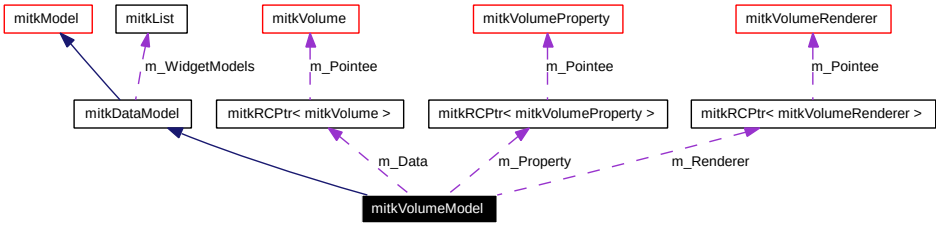
mitkVolumeModel - an 3D entity in a rendering scene represented in volume

`#include <mitkVolumeModel.h>`

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 (mitkView *view)
- virtual bool IsOpaque ()

Protected Member Functions

- virtual float * **_getBounds** ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_Data**
- mitkRCPtr< mitkVolumeProperty > **m_Property**
- mitkRCPtr< mitkVolumeRenderer > **m_Renderer**

6.139.1 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.

6.139.2 Member Function Documentation

6.139.2.1 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.

6.139.2.2 mitkVolume* mitkVolumeModel::GetData (void)

Get the volume data.

Returns:
Return pointer to the volume data.

6.139.2.3 mitkVolumeProperty* mitkVolumeModel::GetProperty (void)

Get the volume property.

Returns:
Return pointer to this volume model's property.

6.139.2.4 mitkVolumeRenderer* mitkVolumeModel::GetRenderer (void)

Get the volume renderer.

Returns:
Return a pointer to this volume model's renderer.

6.139.2.5 virtual bool mitkVolumeModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Implements mitkModel.

6.139.2.6 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.

6.139.2.7 virtual int mitkVolumeModel::Render (mitkView * *view*) [virtual]

Render this model.

Parameters:
view the pointer of an mitkView 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.

6.139.2.8 void mitkVolumeModel::SetData (mitkVolume * *data*)

Set the volume data.

Parameters:
data pointer to an mitkVolume

6.139.2.9 void mitkVolumeModel::SetProperty (mitkVolumeProperty * *prop*)

Set the volume property.

Parameters:
prop pointer to an mitkVolumeProperty

6.139.2.10 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:

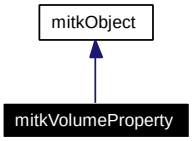
- mitkVolumeModel.h

6.140 mitkVolumeProperty Class Reference

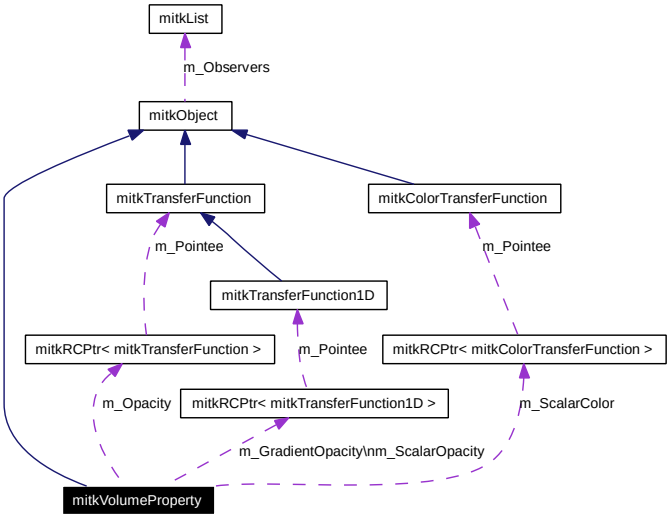
mitkVolumeProperty - properties of an mitkVolumeModel

`#include <mitkVolumeProperty.h>`

Inheritance diagram for mitkVolumeProperty:



Collaboration diagram for mitkVolumeProperty:



Public Types

- enum **TransferFunctionStyle** { **Classical**, **MultiDimensional** }

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)
- mitkColorTransferFunction * GetScalarColor ()
- mitkTransferFunction1D * GetScalarOpacity ()

- mitkTransferFunction1D * GetGradientOpacity ()
- 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 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 ()

Protected Attributes

- mitkRCPtr< mitkColorTransferFunction > **m_ScalarColor**
- mitkRCPtr< mitkTransferFunction1D > **m_ScalarOpacity**
- mitkRCPtr< mitkTransferFunction1D > **m_GradientOpacity**
- mitkRCPtr< mitkTransferFunction > **m_Opacity**
- float * **m_CorrectedScalarOpacity**
- int **m_OldArraySize**
- float **m_OldSampleDistance**
- int **m_InterpolationType**
- float **m_Ambient**
- float **m_Diffuse**
- float **m_Specular**
- float **m_SpecularPower**
- TransferFunctionStyle **m_TransferFunctionStyle**
- bool **m_Shade**
- bool **m_GOEnable**
- bool **m_Modified**

6.140.1 Detailed Description

mitkVolumeProperty - properties of an mitkVolumeModel

mitkVolumeProperty is a class contains properties of an mitkVolumeModel including shading parameters and transfer functions.

6.140.2 Member Function Documentation

6.140.2.1 void mitkVolumeProperty::CorrectScalarOpacity (float *sampleDistance*)

Correct the scalar-opacity transfer function when the sample distance changed.

Parameters:
sampleDistance Specify the new sample distance

6.140.2.2 float mitkVolumeProperty::GetAmbient () [inline]

Get the ambient lighting coefficient.

Returns:
Return the ambient lighting coefficient.

6.140.2.3 float mitkVolumeProperty::GetDiffuse () [inline]

Get the diffuse lighting coefficient.

Returns:
Return the diffuse lighting coefficient.

6.140.2.4 mitkTransferFunction1D* mitkVolumeProperty::GetGradientOpacity ()

Get the gradient value to opacity transfer function.

Returns:
Return the gradient-opacity transfer function

6.140.2.5 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

6.140.2.6 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

6.140.2.7 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"

6.140.2.8 mitkTransferFunction* mitkVolumeProperty::GetOpacityTransferFunction ()

Get the general transfer function (may be multi-dimensional).

Returns:
Return the general transfer function

6.140.2.9 mitkColorTransferFunction* mitkVolumeProperty::GetScalarColor ()

Get the scalar value to color transfer function.

Returns:
Return the scalar-color transfer function

6.140.2.10 mitkTransferFunction1D* mitkVolumeProperty::GetScalarOpacity ()

Get the scalar value to opacity transfer function.

Returns:
Return the scalar-opacity transfer function

6.140.2.11 int mitkVolumeProperty::GetShade () [inline]

Get the status of shading.

Returns:
Return true, the shading is on Return false, the shading is off

6.140.2.12 float mitkVolumeProperty::GetSpecular () [inline]

Get the specular lighting coefficient.

Returns:
Return the specular lighting coefficient.

6.140.2.13 float mitkVolumeProperty::GetSpecularPower () [inline]

Get the specular power.

Returns:
Return the specular power.

6.140.2.14 TransferFunctionStyle mitkVolumeProperty::GetTransferFunctionStyle () [inline]

Get the style of transfer function.

Returns:
Return the style of transfer function.

See also:
TransferFunctionStyle

6.140.2.15 void mitkVolumeProperty::GradientOpacityCalculationOff () [inline]

Disable the calculation of Gradient-Opacity transfer function.

6.140.2.16 void mitkVolumeProperty::GradientOpacityCalculationOn () [inline]

Enable the calculation of Gradient-Opacity transfer function.

6.140.2.17 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.

6.140.2.18 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.

6.140.2.19 void mitkVolumeProperty::SetAmbient (float *value*) [inline]

Set the ambient lighting coefficient.

Parameters:
value Specify the ambient lighting coefficient.

6.140.2.20 void mitkVolumeProperty::SetDiffuse (float *value*) [inline]

Set the diffuse lighting coefficient.

Parameters:
value Specify the diffuse lighting coefficient.

6.140.2.21 void mitkVolumeProperty::SetGradientOpacity (mitkTransferFunction1D * *goFunction*)

Set the gradient value to opacity transfer function.

Parameters:

goFunction Specify the gradient-opacity transfer function

6.140.2.22 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

6.140.2.23 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

6.140.2.24 void mitkVolumeProperty::SetInterpolationTypeToLinear () [inline]

Set the interpolation type to linear interpolation.

6.140.2.25 void mitkVolumeProperty::SetInterpolationTypeToNearest () [inline]

Set the interpolation type to nearest interpolation.

6.140.2.26 void mitkVolumeProperty::SetOpacityTransferFunction (mitkTransferFunction * *moFunction*)

Set the general transfer function (may be multi-dimensional).

Parameters:

moFunction Specify the general transfer function

6.140.2.27 void mitkVolumeProperty::SetScalarColor (mitkColorTransferFunction * *scFunction*)

Set the scalar value to color transfer function.

Parameters:

scFunction Specify the scalar-color transfer function

6.140.2.28 void mitkVolumeProperty::SetScalarOpacity (mitkTransferFunction1D * *soFunction*)

Set the scalar value to opacity transfer function.

Parameters:
soFunction Specify the scalar-opacity transfer function

6.140.2.29 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

6.140.2.30 void mitkVolumeProperty::SetSpecular (float *value*) [inline]

Set the specular lighting coefficient.

Parameters:
value Specify the specular lighting coefficient.

6.140.2.31 void mitkVolumeProperty::SetSpecularPower (float *value*) [inline]

Set the specular power.

Parameters:
value Specify the specular power.

6.140.2.32 void mitkVolumeProperty::SetTransferFunctionStyle (TransferFunctionStyle *tfStyle*) [inline]

Set the style of transfer function.

Parameters:
tfStyle Specify the style of transfer function.

See also:
TransferFunctionStyle

6.140.2.33 void mitkVolumeProperty::SetUnmodified ()

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

6.140.2.34 void mitkVolumeProperty::ShadeOff () [inline]

Turn the shading off.

6.140.2.35 void mitkVolumeProperty::ShadeOn () [inline]

Turn the shading on.

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

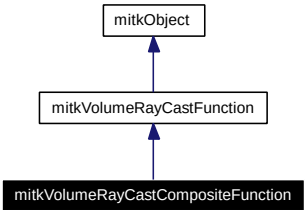
- mitkVolumeProperty.h

6.141 mitkVolumeRayCastCompositeFunction Class Reference

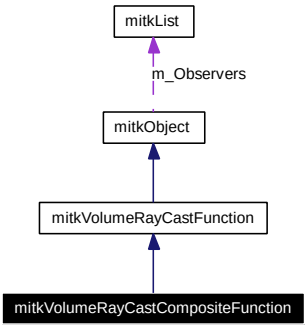
mitkVolumeRayCastCompositeFunction - a concrete ray cast function for compositing

`#include <mitkVolumeRayCastCompositeFunction.h>`

Inheritance diagram for mitkVolumeRayCastCompositeFunction:



Collaboration diagram for mitkVolumeRayCastCompositeFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void CastRay (mitkRay *rayInfo)

6.141.1 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.

6.141.2 Member Function Documentation

6.141.2.1 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.

6.141.2.2 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:

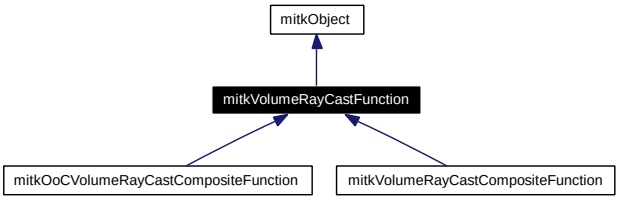
- mitkVolumeRayCastCompositeFunction.h

6.142 mitkVolumeRayCastFunction Class Reference

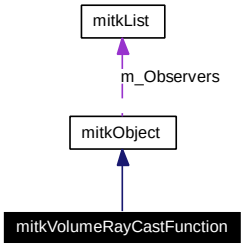
mitkVolumeRayCastFunction - an abstract class for calculating the ray casting function.

`#include <mitkVolumeRayCastFunction.h>`

Inheritance diagram for mitkVolumeRayCastFunction:



Collaboration diagram for mitkVolumeRayCastFunction:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void CastRay (mitkRay *rayInfo)=0

6.142.1 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.

6.142.2 Member Function Documentation

6.142.2.1 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, and mitkVolumeRayCastCompositeFunction.

6.142.2.2 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, and mitkVolumeRayCastCompositeFunction.

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

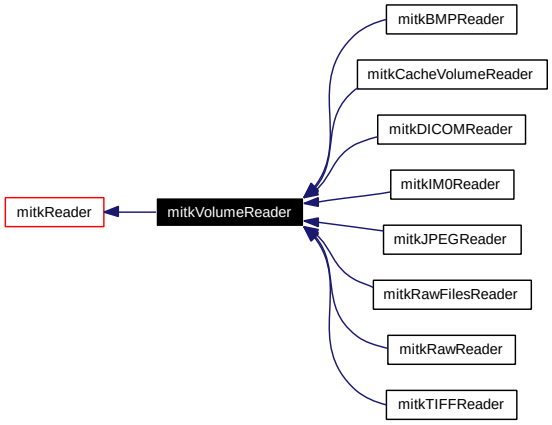
- mitkVolumeRayCastFunction.h

6.143 mitkVolumeReader Class Reference

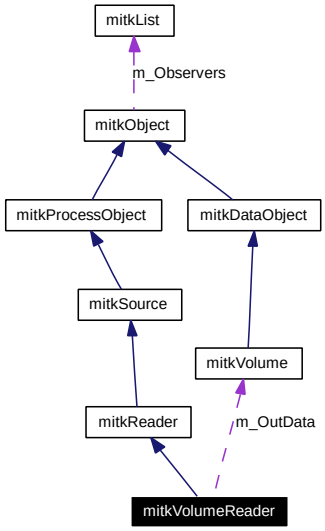
mitkVolumeReader - an abstract class represents a volume reader to read image/volume files

`#include <mitkVolumeReader.h>`

Inheritance diagram for mitkVolumeReader:



Collaboration diagram for mitkVolumeReader:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetOoCSupport (char const *diskPath, unsigned int bufSliceNum=32, bool supportOoC=true)
- mitkVolume * GetOutput ()

Protected Attributes

- mitkVolume * **m_OutData**

- mitkString * **m_DiskPath**
- unsigned int **m_BufferedSliceNum**
- bool **m_NeedOoC**

6.143.1 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
}
```

6.143.2 Member Function Documentation

6.143.2.1 mitkVolume* mitkVolumeReader::GetOutput ()

Get the output volume the reader has read.

Returns:
the output volume.

6.143.2.2 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 mitkBMPReader, mitkCacheVolumeReader, mitkDICOMReader, mitkJPEGReader, mitkRawFilesReader, mitkRawReader, and mitkTIFFReader.

6.143.2.3 void mitkVolumeReader::SetOoCSupport (char const * *diskPath*, unsigned int *bufSliceNum* = 32, 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:

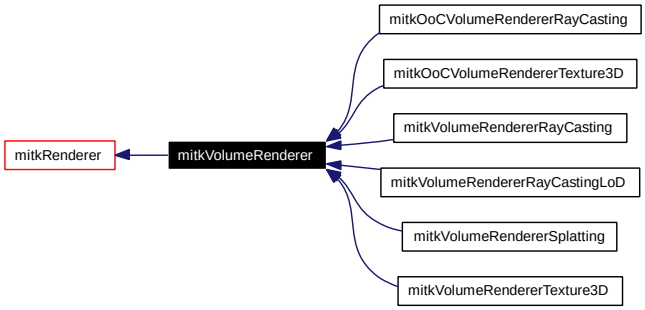
- mitkVolumeReader.h

6.144 mitkVolumeRenderer Class Reference

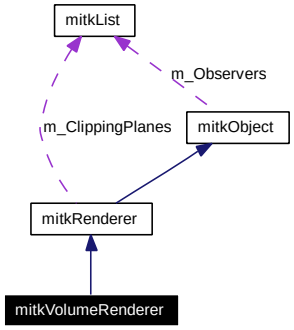
mitkVolumeRenderer - an abstract class for volume rendering

`#include <mitkVolumeRenderer.h>`

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 ()
- virtual float **GetGradientMagnitudeScale** ()
- virtual float **GetGradientMagnitudeBias** ()
- void SetClassifyMethod (int classifyMethod)
- int GetClassifyMethod ()
- void SetClassifyMethodToPreClassification ()
- void SetClassifyMethodToPostClassification ()
- void SetClassifyMethodToPreIntegration ()
- virtual int Render (mitkView *view, mitkVolumeModel *vol)=0
- void **SetAdjustClippingPlane** (bool enable=true)

Protected Attributes

- float **m_CroppingRegionPlanes** [6]
- int **m_CroppingRegionFlags**
- int **m_ClassifyMethod**
- bool **m_EnableCropping**
- bool **m_NeedAdjustClippingPlane**

6.144.1 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.

6.144.2 Member Function Documentation

6.144.2.1 void mitkVolumeRenderer::CroppingOff () [inline]

Turn Off cropping

6.144.2.2 void mitkVolumeRenderer::CroppingOn () [inline]

Turn On cropping

6.144.2.3 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.

6.144.2.4 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.

6.144.2.5 int mitkVolumeRenderer::GetCroppingRegionFlags () [inline]

Get the flags for the cropping regions.

Returns:
Return the cropping flag

6.144.2.6 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

6.144.2.7 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.

6.144.2.8 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, mitkOoCVolumeRendererTexture3D, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, and mitkVolumeRendererTexture3D.

6.144.2.9 virtual int mitkVolumeRenderer::Render (mitkView * *view*, mitkVolumeModel * *vol*) [pure virtual]

Internal function. Don't call it directly.

Implemented in mitkOoCVolumeRendererRayCasting, mitkOoCVolumeRendererTexture3D, mitkVolumeRendererRayCasting, mitkVolumeRendererRayCastingLoD, mitkVolumeRendererSplatting, and mitkVolumeRendererTexture3D.

6.144.2.10 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.

6.144.2.11 void mitkVolumeRenderer::SetClassifyMethodToPostClassification () [inline]

Set the classification method to PostClassification (interpolation first).

6.144.2.12 void mitkVolumeRenderer::SetClassifyMethodToPreClassification () [inline]

Set the classification method to PreClassification (classify first).

6.144.2.13 void mitkVolumeRenderer::SetClassifyMethodToPreIntegration () [inline]

Set the classification method to Pre-integration.

6.144.2.14 void mitkVolumeRenderer::SetCropping (bool *isCropping*) [inline]

Turn On / Off cropping

Parameters:

isCropping isCropping = true, turn on cropping isCropping = false, turn off cropping

6.144.2.15 void mitkVolumeRenderer::SetCroppingRegionFlags (int *flags*) [inline]

Set the flags for the cropping regions.

Parameters:

flags Specify the cropping flag

6.144.2.16 void mitkVolumeRenderer::SetCroppingRegionFlagsToCross () [inline]

Set the flags for the cropping regions to Cross

6.144.2.17 void mitkVolumeRenderer::SetCroppingRegionFlagsToFence () [inline]

Set the flags for the cropping regions to Fence

6.144.2.18 void mitkVolumeRenderer::SetCroppingRegionFlagsToInvertedCross () [inline]

Set the flags for the cropping regions to InvertedCross

6.144.2.19 void mitkVolumeRenderer::SetCroppingRegionFlagsToInvertedFence () [inline]

Set the flags for the cropping regions to InvertedFence

6.144.2.20 void mitkVolumeRenderer::SetCroppingRegionFlagsToSubVolume () [inline]

Set the flags for the cropping regions to SubVolume

6.144.2.21 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

6.144.2.22 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:

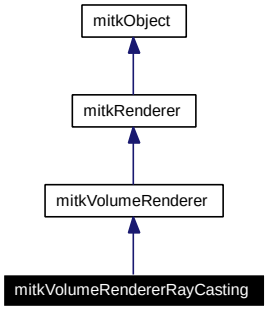
- mitkVolumeRenderer.h

6.145 mitkVolumeRendererRayCasting Class Reference

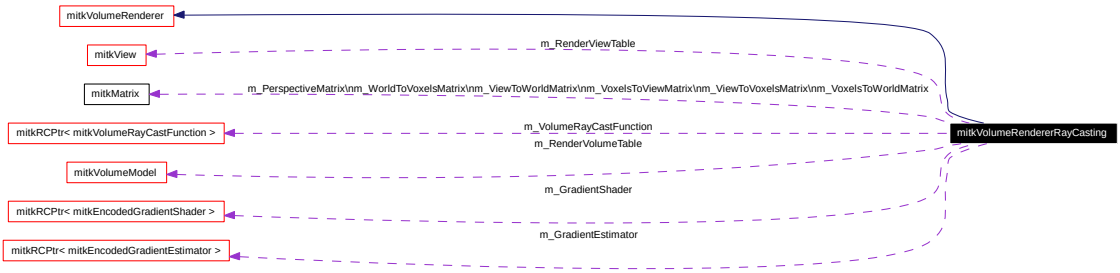
mitkVolumeRendererRayCasting - a concrete volume renderer for rendering a volume

`#include <mitkVolumeRendererRayCasting.h>`

Inheritance diagram for mitkVolumeRendererRayCasting:



Collaboration diagram for mitkVolumeRendererRayCasting:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view, 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 ()

Protected Member Functions

- void **_updateShadingTables** (mitkView *view, mitkVolumeModel *vol)
- void **_renderTexture** (mitkVolumeModel *vol, mitkView *view)
- int **_computeRowBounds** (mitkVolumeModel *vol, mitkView *view)
- int **_computeClippingPlanes** (float *planes)
- int **_clipRayAgainstVolume** (mitkRay *rayInfo, float bounds[6])
- int **_clipRayAgainstClippingPlanes** (int planeCount, float *planeEqus, mitkRay *rayInfo)
- float **_getZBufferValue** (int x, int y)
- void **_castRaysClipOffCropOff** (mitkView *view, mitkVolumeModel *vol)
- void **_castRaysClipOnCropOff** (mitkView *view, mitkVolumeModel *vol)
- void **_castRaysClipOffCropSub** (mitkView *view, mitkVolumeModel *vol)
- void **_castRaysClipOffCropNonSub** (mitkView *view, mitkVolumeModel *vol)
- void **_castRaysClipOnCropSub** (mitkView *view, mitkVolumeModel *vol)
- void **_castRaysClipOnCropNonSub** (mitkView *view, mitkVolumeModel *vol)

Protected Attributes

- mitkRCPtr< mitkVolumeRayCastFunction > **m_VolumeRayCastFunction**
- mitkRCPtr< mitkEncodedGradientEstimator > **m_GradientEstimator**
- mitkRCPtr< mitkEncodedGradientShader > **m_GradientShader**
- mitkVolumeModel ** **m_RenderVolumeTable**
- mitkView ** **m_RenderViewTable**
- mitkMatrix * **m_PerspectiveMatrix**
- mitkMatrix * **m_ViewToWorldMatrix**
- mitkMatrix * **m_ViewToVoxelsMatrix**
- mitkMatrix * **m_VoxelsToViewMatrix**
- mitkMatrix * **m_WorldToVoxelsMatrix**
- mitkMatrix * **m_VoxelsToWorldMatrix**
- int **m_ImageViewportSize** [2]
- int **m_ImageMemorySize** [2]
- int **m_ImageInUseSize** [2]
- int **m_ImageOrigin** [2]
- unsigned char * **m_Image**
- int * **m_RowBounds**

- int * **m_OldRowBounds**
- int **m_RenderTableSize**
- int **m_RenderTableEntries**
- int **m_IntermixIntersectingGeometry**
- float * **m_ZBuffer**
- int **m_ZBufferSize** [2]
- int **m_ZBufferOrigin** [2]
- float **m_MinimumViewDistance**
- float **m_SampleDistance**
- float **m_ImageSampleDistance**
- float **m_MinimumImageSampleDistance**
- float **m_MaximumImageSampleDistance**
- int **m_AutoAdjustSampleDistances**
- float **m_WorldSampleDistance**
- int **m_ScalarDataType**
- void * **m_ScalarDataPointer**
- mitkRay * **m_Ray**

6.145.1 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.

6.145.2 Member Function Documentation

6.145.2.1 void mitkVolumeRendererRayCasting::AutoAdjustSampleDistancesOff () [inline]

Set sample distances auto-adjusting off.

6.145.2.2 void mitkVolumeRendererRayCasting::AutoAdjustSampleDistancesOn () [inline]

Set sample distances auto-adjusting on.

6.145.2.3 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).

6.145.2.4 mitkEncodedGradientShader* mitkVolumeRendererRayCasting::GetEncodedGradientShader () [inline]

Get the gradient shader

Returns:
Return the gradient shader

6.145.2.5 mitkEncodedGradientEstimator* mitkVolumeRendererRayCasting::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals

Returns:
Return the gradient estimator used to estimate normals.

6.145.2.6 virtual float mitkVolumeRendererRayCasting::GetGradientMagnitudeBias () [virtual]

Get gradient magnitude bias.

Returns:
Return the gradient magnitude bias.

Reimplemented from mitkVolumeRenderer.

6.145.2.7 virtual float mitkVolumeRendererRayCasting::GetGradientMagnitudeScale () [virtual]

Get gradient magnitude scale.

Returns:
Return the gradient magnitude scale.

Reimplemented from mitkVolumeRenderer.

6.145.2.8 float mitkVolumeRendererRayCasting::GetImageSampleDistance () [inline]

Get the sample distance in the image plane.

Returns:
Return the sample distance in the image plane.

6.145.2.9 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.

6.145.2.10 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.

6.145.2.11 int mitkVolumeRendererRayCasting::GetIntermixIntersectingGeometry ()
[inline]

Get the intermix intersecting geometry value

Returns:
Return the intermix intersecting geometry value

6.145.2.12 float mitkVolumeRendererRayCasting::GetMaximumImageSampleDistance ()
[inline]

Get the maximum sample distance in the image plane.

Returns:
Return the maximum sample distance in the image plane.

6.145.2.13 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.

6.145.2.14 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.

6.145.2.15 float mitkVolumeRendererRayCasting::GetMinimumImageSampleDistance ()
[inline]

Get the minimum sample distance in the image plane.

Returns:
Return the minimum sample distance in the image plane.

6.145.2.16 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.

6.145.2.17 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.

6.145.2.18 float mitkVolumeRendererRayCasting::GetSampleDistance () [inline]

Get the sample distance in the ray direction.

Returns:
Return the sample distance.

6.145.2.19 mitkVolumeRayCastFunction* mitkVolumeRendererRayCasting::GetVolumeRayCastFunction () [inline]

Get the volume ray cast function.

Returns:
Return the volume ray cast function.

6.145.2.20 void mitkVolumeRendererRayCasting::IntermixIntersectingGeometryOff () [inline]

Set IntermixIntersectingGeometry to off

6.145.2.21 void mitkVolumeRendererRayCasting::IntermixIntersectingGeometryOn () [inline]

Set IntermixIntersectingGeometry to on

6.145.2.22 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.

6.145.2.23 virtual int mitkVolumeRendererRayCasting::Render (mitkView * *view*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

6.145.2.24 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)

6.145.2.25 void mitkVolumeRendererRayCasting::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals

Parameters:
gradest Represent the gradient estimator used to estimate normals

6.145.2.26 void mitkVolumeRendererRayCasting::SetImageSampleDistance (float *fVal*) [inline]

Set the sample distance in the image plane.

Parameters:
fVal Specify the sample distance in the image plane.

6.145.2.27 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

6.145.2.28 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.

6.145.2.29 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

6.145.2.30 void mitkVolumeRendererRayCasting::SetSampleDistance (float *fVal*) [inline]

Set the sample distance in the ray direction.

Parameters:
fVal Specify the sample distance.

6.145.2.31 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:

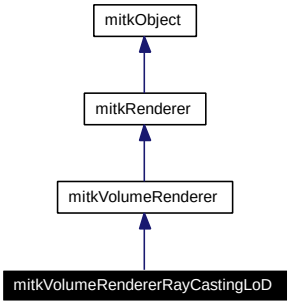
- mitkVolumeRendererRayCasting.h

6.146 mitkVolumeRendererRayCastingLoD Class Reference

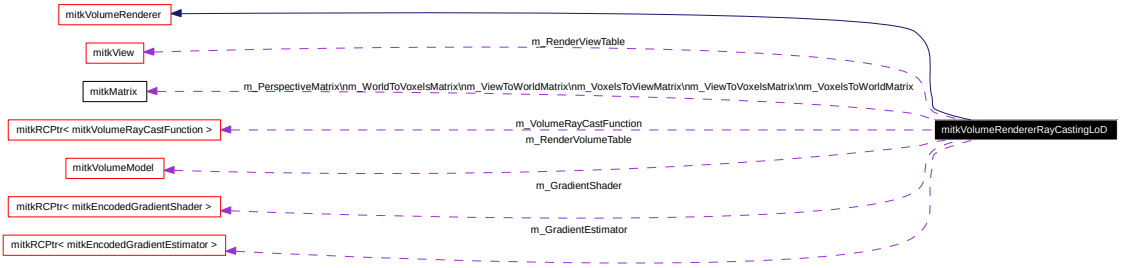
mitkVolumeRendererRayCastingLoD - a concrete volume renderer with LoD function for rendering a volume

`#include <mitkVolumeRendererRayCastingLoD.h>`

Inheritance diagram for mitkVolumeRendererRayCastingLoD:



Collaboration diagram for mitkVolumeRendererRayCastingLoD:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view, 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 ()

Protected Member Functions

- void **_updateShadingTables** (mitkView *view, mitkVolumeModel *vol)
- void **_renderTexture** (mitkVolumeModel *vol, mitkView *view, unsigned char *img)
- int **_computeClippingPlanes** (float *planes)
- int **_computeRowBounds** (mitkVolumeModel *vol, mitkView *view)
- int **_clipRayAgainstVolume** (mitkRay *rayInfo, float bounds[6])
- int **_clipRayAgainstClippingPlanes** (int planeCount, float *planeEqus, mitkRay *rayInfo)
- float **_getZBufferValue** (int x, int y)
- void **_castRaysClipOffCropOff** (mitkView *view, mitkVolumeModel *vol, unsigned char *img)
- void **_castRaysClipOnCropOff** (mitkView *view, mitkVolumeModel *vol, unsigned char *img)
- void **_castRaysClipOffCropSub** (mitkView *view, mitkVolumeModel *vol, unsigned char *img)
- void **_castRaysClipOffCropNonSub** (mitkView *view, mitkVolumeModel *vol, unsigned char *img)
- void **_castRaysClipOnCropSub** (mitkView *view, mitkVolumeModel *vol, unsigned char *img)
- void **_castRaysClipOnCropNonSub** (mitkView *view, mitkVolumeModel *vol, unsigned char *img)

Protected Attributes

- mitkRCPtr< mitkVolumeRayCastFunction > **m_VolumeRayCastFunction**
- mitkRCPtr< mitkEncodedGradientEstimator > **m_GradientEstimator**
- mitkRCPtr< mitkEncodedGradientShader > **m_GradientShader**
- mitkVolumeModel ** **m_RenderVolumeTable**
- mitkView ** **m_RenderViewTable**
- mitkMatrix * **m_PerspectiveMatrix**
- mitkMatrix * **m_ViewToWorldMatrix**
- mitkMatrix * **m_ViewToVoxelsMatrix**
- mitkMatrix * **m_VoxelsToViewMatrix**
- mitkMatrix * **m_WorldToVoxelsMatrix**
- mitkMatrix * **m_VoxelsToWorldMatrix**
- int **m_ImageViewportSize** [2]
- int **m_ImageMemorySize** [2]
- int **m_ImageInUseSize** [2]

- int **m_ImageOrigin** [2]
- unsigned char * **m_Image**
- int * **m_RowBounds**
- int * **m_OldRowBounds**
- int **m_RenderTableSize**
- int **m_RenderTableEntries**
- int **m_IntermixIntersectingGeometry**
- float * **m_ZBuffer**
- int **m_ZBufferSize** [2]
- int **m_ZBufferOrigin** [2]
- float **m_MinimumViewDistance**
- float **m_SampleDistance**
- float **m_ImageSampleDistance**
- float **m_MinimumImageSampleDistance**
- float **m_MaximumImageSampleDistance**
- int **m_AutoAdjustSampleDistances**
- float **m_WorldSampleDistance**
- int **m_ScalarDataType**
- void * **m_ScalarDataPointer**
- mitkRay * **m_Ray**

6.146.1 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.

6.146.2 Member Function Documentation

6.146.2.1 void mitkVolumeRendererRayCastingLoD::AutoAdjustSampleDistancesOff ()

`[inline]`

Set sample distances auto-adjusting off.

6.146.2.2 void mitkVolumeRendererRayCastingLoD::AutoAdjustSampleDistancesOn ()

`[inline]`

Set sample distances auto-adjusting on.

6.146.2.3 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).

6.146.2.4 mitkEncodedGradientShader* mitkVolumeRendererRayCastingLoD::GetEncodedGradientShader () [inline]

Get the gradient shader

Returns:
Return the gradient shader

6.146.2.5 mitkEncodedGradientEstimator* mitkVolumeRendererRayCastingLoD::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals

Returns:
Return the gradient estimator used to estimate normals

6.146.2.6 virtual float mitkVolumeRendererRayCastingLoD::GetGradientMagnitudeBias () [virtual]

Get gradient magnitude bias.

Returns:
Return the gradient magnitude bias.

Reimplemented from mitkVolumeRenderer.

6.146.2.7 virtual float mitkVolumeRendererRayCastingLoD::GetGradientMagnitudeScale () [virtual]

Get gradient magnitude scale.

Returns:
Return the gradient magnitude scale.

Reimplemented from mitkVolumeRenderer.

6.146.2.8 float mitkVolumeRendererRayCastingLoD::GetImageSampleDistance () [inline]

Get the sample distance in the image plane.

Returns:
Return the sample distance in the image plane.

6.146.2.9 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.

6.146.2.10 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.

6.146.2.11 int mitkVolumeRendererRayCastingLoD::GetIntermixIntersectingGeometry () [inline]

Get the intermix intersecting geometry value

Returns:
Return the intermix intersecting geometry value

6.146.2.12 float mitkVolumeRendererRayCastingLoD::GetMaximumImageSampleDistance () [inline]

Get the maximum sample distance in the image plane.

Returns:
Return the maximum sample distance in the image plane.

6.146.2.13 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.

6.146.2.14 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.

6.146.2.15 float mitkVolumeRendererRayCastingLoD::GetMinimumImageSampleDistance () [inline]

Get the minimum sample distance in the image plane.

Returns:
Return the minimum sample distance in the image plane.

6.146.2.16 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.

6.146.2.17 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.

6.146.2.18 float mitkVolumeRendererRayCastingLoD::GetSampleDistance () [inline]

Get the sample distance in the ray direction.

Returns:
Return the sample distance

6.146.2.19 mitkVolumeRayCastFunction* mitkVolumeRendererRayCastingLoD::GetVolumeRayCastFunction () [inline]

Get the volume ray cast function.

Returns:
Return the volume ray cast function

6.146.2.20 void mitkVolumeRendererRayCastingLoD::IntermixIntersectingGeometryOff () [inline]

Set IntermixIntersectingGeometry to off

6.146.2.21 void mitkVolumeRendererRayCastingLoD::IntermixIntersectingGeometryOn () [inline]

Set IntermixIntersectingGeometry to on

6.146.2.22 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.

6.146.2.23 virtual int mitkVolumeRendererRayCastingLoD::Render (mitkView * *view*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

6.146.2.24 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)

6.146.2.25 void mitkVolumeRendererRayCastingLoD::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals

Parameters:
gradest Represent the gradient estimator used to estimate normals

6.146.2.26 void mitkVolumeRendererRayCastingLoD::SetImageSampleDistance (float *fVal*) [inline]

Set the sample distance in the image plane.

Parameters:
fVal Specify the sample distance in the image plane.

6.146.2.27 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

6.146.2.28 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.

6.146.2.29 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

6.146.2.30 void mitkVolumeRendererRayCastingLoD::SetSampleDistance (float *fVal*) [inline]

Set the sample distance in the ray direction.

Parameters:
fVal Specify the sample distance

6.146.2.31 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:

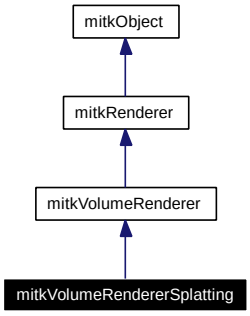
- mitkVolumeRendererRayCastingLoD.h

6.147 mitkVolumeRendererSplatting Class Reference

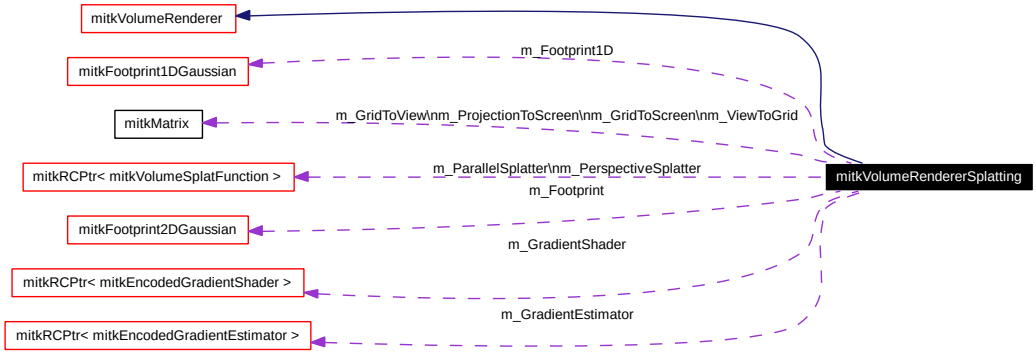
mitkVolumeRendererSplatting - a concrete volume renderer for rendering a volume

`#include <mitkVolumeRendererSplatting.h>`

Inheritance diagram for mitkVolumeRendererSplatting:



Collaboration diagram for mitkVolumeRendererSplatting:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view, 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 ()

Protected Types

- enum { **INTEGRAL**, **MOP** }

Protected Member Functions

- void **_updateShadingTables** (mitkView *view, mitkVolumeModel *vol)
- void **_renderTexture** (mitkVolumeModel *vol, mitkView *view, unsigned char *img)
- void **_getTransformMatrix** (mitkView *view, mitkVolumeModel *vol)
- int **_computeRowBounds** (mitkVolumeModel *vol, mitkView *view)
- void **_splatting** (mitkView *view, mitkVolumeModel *vol)
- void **_getSplatOrder** ()
- void **_computeFootprint** (float x, float y, float z, int offset)
- bool **_cullVoxel** (float x, float y, float z, float *clipPlane, int pCount)

Protected Attributes

- mitkRCPtr< mitkVolumeSplatFunction > **m_ParallelSplatter**
- mitkRCPtr< mitkVolumeSplatFunction > **m_PerspectiveSplatter**
- mitkRCPtr< mitkEncodedGradientEstimator > **m_GradientEstimator**
- mitkRCPtr< mitkEncodedGradientShader > **m_GradientShader**
- mitkMatrix * **m_GridToView**
- mitkMatrix * **m_ViewToGrid**
- mitkMatrix * **m_GridToScreen**
- mitkMatrix * **m_ProjectionToScreen**
- int **m_ImageViewportSize** [2]
- int **m_ImageMemorySize** [2]
- int **m_ImageInUseSize** [2]
- int **m_ImageOrigin** [2]
- unsigned char * **m_Image**
- int * **m_RowBounds**
- float * **m_SheetBuffer**
- float * **m_CompositeBuffer**
- int **m_GridSampleDistance**
- int **m_MinimumGridSampleDistance**
- int **m_MaximumGridSampleDistance**
- bool **m_AutoAdjustGridSampleDistances**
- float **m_ImageSampleDistance**
- float **m_MinimumImageSampleDistance**
- float **m_MaximumImageSampleDistance**
- bool **m_AutoAdjustImageSampleDistances**
- float **m_KernelRadii**
- float **m_Coeff**
- float **m_AdjustRadii**
- mitkSplat * **m_Splat**
- mitkFootprint2DGaussian * **m_Footprint**
- mitkFootprint1DGaussian * **m_Footprint1D**
- int **m_Mode**

6.147.1 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 mitkView's camera must be set to be mitkSplatCamera.

6.147.2 Member Function Documentation

6.147.2.1 void mitkVolumeRendererSplatting::AutoAdjustGridSampleDistancesOff ()
`[inline]`

Set grid sample distances auto-adjusting off.

6.147.2.2 void mitkVolumeRendererSplatting::AutoAdjustGridSampleDistancesOn ()
`[inline]`

Set grid sample distances auto-adjusting on.

6.147.2.3 void mitkVolumeRendererSplatting::AutoAdjustImageSampleDistancesOff ()
`[inline]`

Set image sample distances auto-adjusting off.

6.147.2.4 void mitkVolumeRendererSplatting::AutoAdjustImageSampleDistancesOn ()
`[inline]`

Set image sample distances auto-adjusting on.

6.147.2.5 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].

6.147.2.6 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).

6.147.2.7 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).

6.147.2.8 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].

6.147.2.9 mitkEncodedGradientShader* mitkVolumeRendererSplatting::GetEncodedGradientShader () [inline]

Get the gradient shader.

Returns:
Return the gradient shader.

6.147.2.10 mitkEncodedGradientEstimator* mitkVolumeRendererSplatting::GetGradientEstimator () [inline]

Get the gradient estimator used to estimate normals.

Returns:
Return the gradient estimator used to estimate normals.

6.147.2.11 virtual float mitkVolumeRendererSplatting::GetGradientMagnitudeBias () [virtual]

Get gradient magnitude bias.

Returns:
Return the gradient magnitude bias.

Reimplemented from mitkVolumeRenderer.

6.147.2.12 virtual float mitkVolumeRendererSplatting::GetGradientMagnitudeScale () [virtual]

Get gradient magnitude scale.

Returns:
Return the gradient magnitude scale.

Reimplemented from mitkVolumeRenderer.

6.147.2.13 int mitkVolumeRendererSplatting::GetGridSampleDistance () [inline]

Get the sample distance in the grid space.

Returns:
Return the grid sample distance in the grid space.

6.147.2.14 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.

6.147.2.15 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.

6.147.2.16 float mitkVolumeRendererSplatting::GetImageSampleDistance () [inline]

Get the sample distance in the image plane.

Returns:
Return the image sample distance in the image plane.

6.147.2.17 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.

6.147.2.18 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.

6.147.2.19 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].

6.147.2.20 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.

6.147.2.21 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.

6.147.2.22 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.

6.147.2.23 float mitkVolumeRendererSplatting::GetMaximumImageSampleDistance () [inline]

Get the maximum image sample distance in the image plane.

Returns:
Return the maximum sample distance in the image plane.

6.147.2.24 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.

6.147.2.25 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.

6.147.2.26 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.

6.147.2.27 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.

6.147.2.28 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.

6.147.2.29 float mitkVolumeRendererSplatting::GetMinimumImageSampleDistance () [inline]

Get the minimum image sample distance in the image plane.

Returns:
Return the minimum sample distance in the image plane.

6.147.2.30 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.

6.147.2.31 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.

6.147.2.32 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).

6.147.2.33 mitkVolumeSplatFunction* mitkVolumeRendererSplatting::GetParallelSplatter ()

Get the Parallel splatter.

Returns:
Return the current Parallel splatter.

6.147.2.34 mitkVolumeSplatFunction* mitkVolumeRendererSplatting::GetPerspectiveSplatter ()

Get the perspective splatter.

Returns:
Return the current perspective splatter.

6.147.2.35 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.

6.147.2.36 virtual int mitkVolumeRendererSplatting::Render (mitkView * *view*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

6.147.2.37 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.

6.147.2.38 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.

6.147.2.39 void mitkVolumeRendererSplatting::SetGradientEstimator (mitkEncodedGradientEstimator * *gradest*)

Set the gradient estimator used to estimate normals.

Parameters:
gradest Represent the gradient estimator used to estimate normals.

6.147.2.40 void mitkVolumeRendererSplatting::SetGridSampleDistance (int *fVal*) [inline]

Set the sample distance in the grid space.

Parameters:
fVal Specify the sample distance in the grid space.

6.147.2.41 void mitkVolumeRendererSplatting::SetImageSampleDistance (float *fVal*) [inline]

Set the sample distance in the image plane.

Parameters:
fVal Specify the sample distance in the image plane.

6.147.2.42 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.

6.147.2.43 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.

6.147.2.44 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.

6.147.2.45 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.

6.147.2.46 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

6.147.2.47 void mitkVolumeRendererSplatting::SetModeIntegral () [inline]

Set volume rendering mode to be integral projection.

6.147.2.48 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).

6.147.2.49 void mitkVolumeRendererSplatting::SetParallelSplatter (mitkVolumeSplatFunction * *splatter*)

Set Parallel splatter to splat the volume.

Parameters:
splatter Represent the Parallel splatter.

6.147.2.50 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:

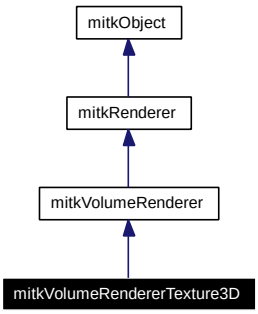
- mitkVolumeRendererSplatting.h

6.148 mitkVolumeRendererTexture3D Class Reference

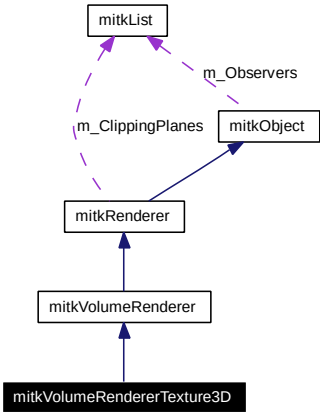
mitkVolumeRendererTexture3D - a concrete volume renderer for rendering a volume

`#include <mitkVolumeRendererTexture3D.h>`

Inheritance diagram for mitkVolumeRendererTexture3D:



Collaboration diagram for mitkVolumeRendererTexture3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual int Render (mitkView *view, mitkVolumeModel *vol)
- void SetSampleDistance (float sd)
- void EnableAutoSampleDistance ()

Protected Member Functions

- bool **_buildDummyTexture** (int dims[3])
- bool **_buildTexture** (mitkVolumeModel *vol)
- void **_deleteTexture** ()
- void **_clearTexPolygons** ()

Protected Attributes

- PFNGLTEXIMAGE3DPROC **glTexImage3D**
- PFNGLTEXSUBIMAGE3DPROC **glTexSubImage3D**
- GLuint **m_TexID**
- int **m_DummyTexWidth**
- int **m_DummyTexHeight**
- int **m_DummyTexDepth**
- float **m_SampleDistance**
- unsigned int **m_PolyNum**
- unsigned int **m_PolyArraySize**
- TexPolygon * **m_Polygons**
- bool **m_IsTex3DSupported**
- bool **m_AutoAdjustSampleDistance**
- bool **m_FirstRendering**

6.148.1 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.

6.148.2 Member Function Documentation

6.148.2.1 void mitkVolumeRendererTexture3D::EnableAutoSampleDistance () [inline]

Enable automatic calculation of the sample distance.

6.148.2.2 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.

6.148.2.3 virtual int mitkVolumeRendererTexture3D::Render (mitkView * *view*, mitkVolumeModel * *vol*) [virtual]

Internal method. Don't call it directly.

Implements mitkVolumeRenderer.

6.148.2.4 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:

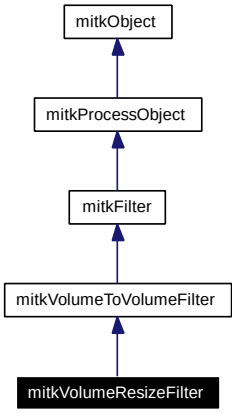
- mitkVolumeRendererTexture3D.h

6.149 mitkVolumeResizeFilter Class Reference

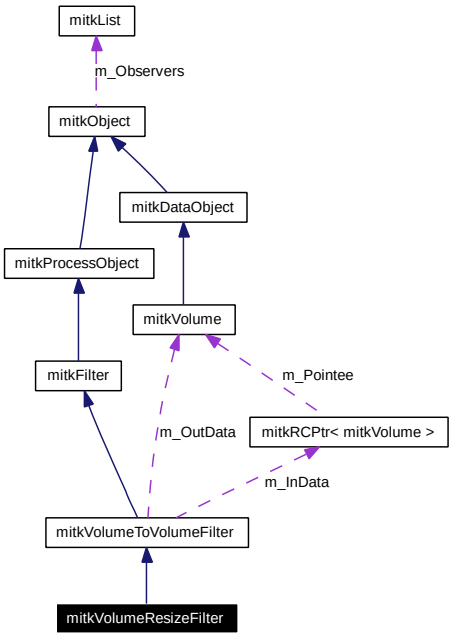
mitkVolumeResizeFilter - a concrete filter class to zoom the specified volume

`#include <mitkVolumeResizeFilter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- int **m_OutDatacoordinates** [3]

6.149.1 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!

6.149.2 Member Function Documentation

6.149.2.1 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.

6.149.2.2 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.

6.149.2.3 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:

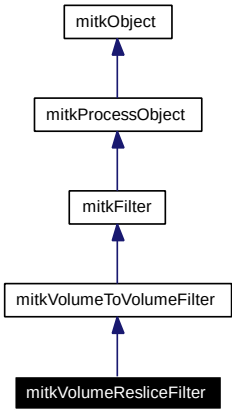
- mitkVolumeResizeFilter.h

6.150 mitkVolumeResliceFilter Class Reference

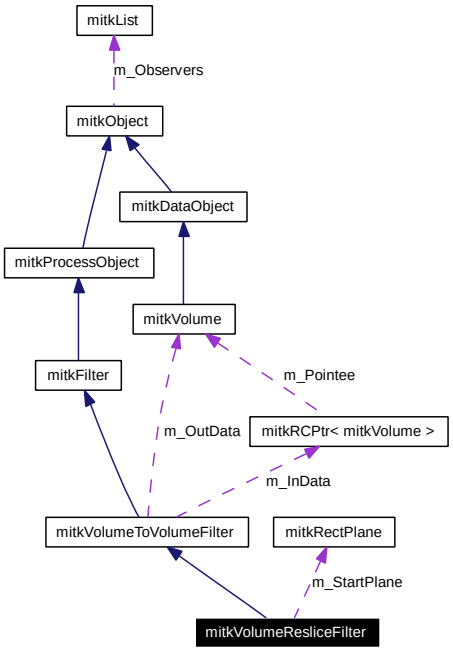
mitkVolumeResliceFilter - a volume re-slice filter

`#include <mitkVolumeResliceFilter.h>`

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)

Protected Member Functions

- virtual bool **Execute** ()

Protected Attributes

- mitkRectPlane **m_StartPlane**
- double **m_StopPoint** [4]
- double **m_SpacingX**
- double **m_SpacingY**
- double **m_SpacingZ**
- int **m_SliceWidth**
- int **m_SliceHeight**
- int **m_SliceNum**
- bool **m_SpacingXFirst**
- bool **m_SpacingYFirst**
- bool **m_SpacingZFirst**

6.150.1 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.

6.150.2 Constructor & Destructor Documentation

6.150.2.1 mitkVolumeResliceFilter::mitkVolumeResliceFilter ()

The default constructor.

6.150.3 Member Function Documentation

6.150.3.1 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.

6.150.3.2 void mitkVolumeResliceFilter::SetSliceHeight (int *h*) [inline]

Set the image height of the reconstructed slices.

Parameters:
h the image height of the reconstructed slices

6.150.3.3 void mitkVolumeResliceFilter::SetSliceNumber (int *num*) [inline]

Set the number of the reconstructed slices.

Parameters:
num the number of the reconstructed slices

6.150.3.4 void mitkVolumeResliceFilter::SetSliceWidth (int *w*) [inline]

Set the image width of the reconstructed slices.

Parameters:
w the image width of the reconstructed slices

6.150.3.5 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.

6.150.3.6 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.

6.150.3.7 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.

6.150.3.8 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.

6.150.3.9 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:

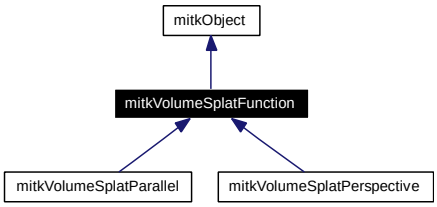
- mitkVolumeResliceFilter.h

6.151 mitkVolumeSplatFunction Class Reference

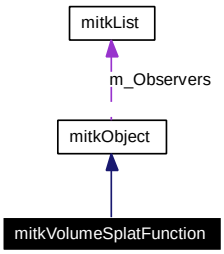
mitkVolumeSplatFunction - abstract class defines interface for volume splatting

`#include <mitkVolumeSplatFunction.h>`

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

6.151.1 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.

6.151.2 Constructor & Destructor Documentation

6.151.2.1 mitkVolumeSplatFunction::mitkVolumeSplatFunction ()

The default constructor.

6.151.3 Member Function Documentation

6.151.3.1 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.

6.151.3.2 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.

6.151.3.3 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:

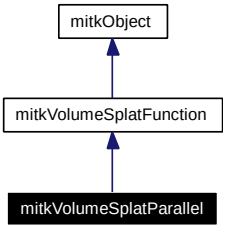
- mitkVolumeSplatFunction.h

6.152 mitkVolumeSplatParallel Class Reference

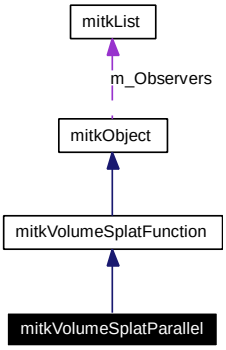
mitkVolumeSplatParallel - concrete class for parallel volume splatting

`#include <mitkVolumeSplatParallel.h>`

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)

6.152.1 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.

6.152.2 Constructor & Destructor Documentation

6.152.2.1 mitkVolumeSplatParallel::mitkVolumeSplatParallel ()

The default constructor.

6.152.3 Member Function Documentation

6.152.3.1 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.

6.152.3.2 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.

6.152.3.3 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:

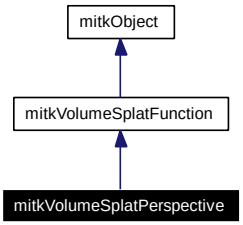
- mitkVolumeSplatParallel.h

6.153 mitkVolumeSplatPerspective Class Reference

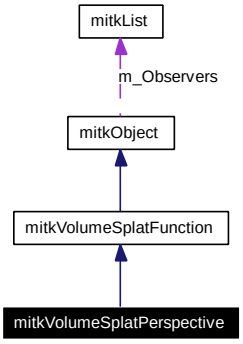
mitkVolumeSplatCompositeFunction - concrete class for perspective volume splatting

`#include <mitkVolumeSplatPerspective.h>`

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)

6.153.1 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.

6.153.2 Constructor & Destructor Documentation

6.153.2.1 mitkVolumeSplatPerspective::mitkVolumeSplatPerspective ()

The default constructor.

6.153.3 Member Function Documentation

6.153.3.1 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.

6.153.3.2 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.

6.153.3.3 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:

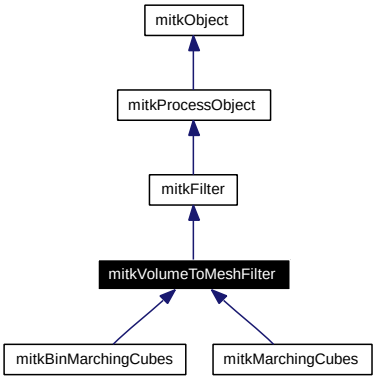
- mitkVolumeSplatPerspective.h

6.154 mitkVolumeToMeshFilter Class Reference

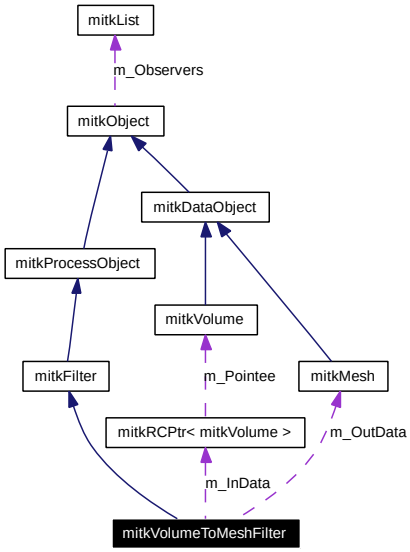
mitkVolumeToMeshFilter - abstract class specifies interface for volume to mesh filter

`#include <mitkVolumeToMeshFilter.h>`

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, unsigned int bufBlockNum=32, bool supportOoC=true)
- mitkMesh * GetOutput ()
- void SetOutputMesh (mitkMesh *mesh)

Protected Attributes

- mitkRCPtr< mitkVolume > **m_InData**
- mitkMesh * **m_OutData**
- mitkString * **m_DiskPath**
- unsigned int **m_BufferedBlockNum**
- bool **m_NeedOoC**

6.154.1 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.

6.154.2 Member Function Documentation

6.154.2.1 mitkVolume* mitkVolumeToMeshFilter::GetInput () [inline]

Get the input volume

Returns:
Return the input volume

6.154.2.2 mitkMesh* mitkVolumeToMeshFilter::GetOutput ()

Get the output mesh

Returns:
Return the output mesh

6.154.2.3 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.

6.154.2.4 void mitkVolumeToMeshFilter::SetInput (mitkVolume * *inData*) [inline]

Set the input volume

Parameters:
inData Specify the input volume

6.154.2.5 void mitkVolumeToMeshFilter::SetOoCSupport (char const * *diskPath*, unsigned int *bufBlockNum* = 32, 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.

6.154.2.6 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:

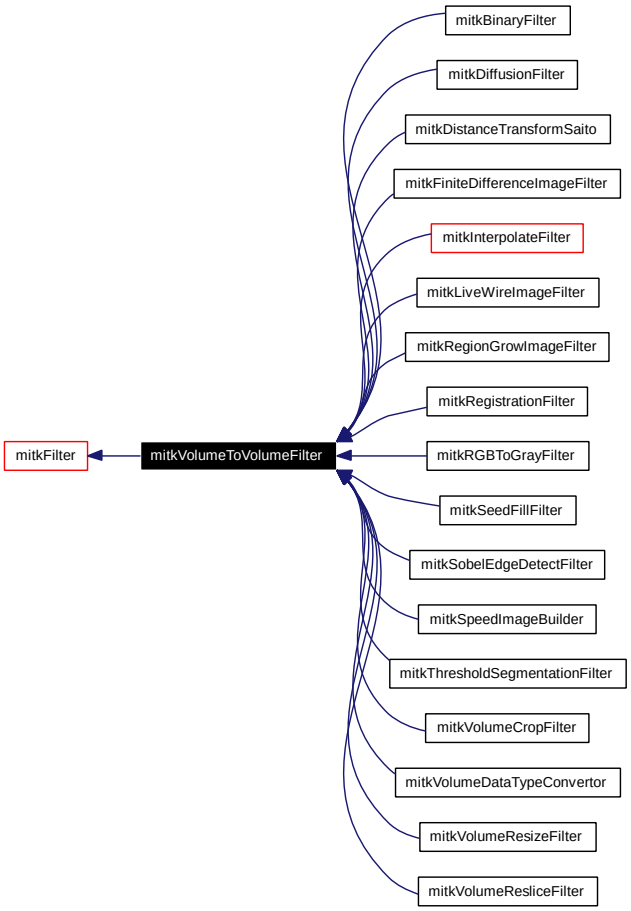
- mitkVolumeToMeshFilter.h

6.155 mitkVolumeToVolumeFilter Class Reference

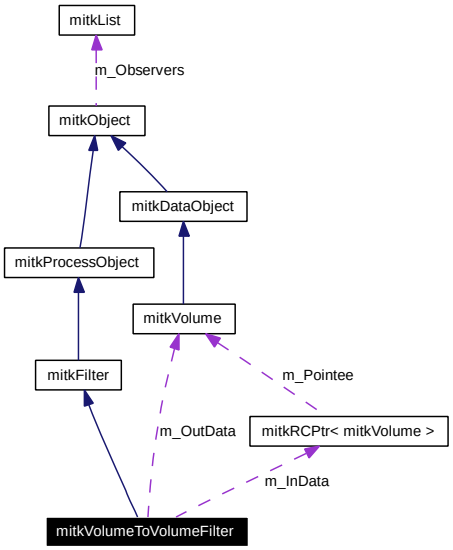
mitkVolumeToVolumeFilter - abstract class specifies interface for volume to volume filter

`#include <mitkVolumeToVolumeFilter.h>`

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, unsigned int bufSliceNum=32, bool supportOoC=true)
- mitkVolume * GetOutput ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_InData**
- mitkVolume * **m_OutData**
- mitkString * **m_DiskPath**
- unsigned int **m_BufferedSliceNum**
- bool **m_NeedOoC**

6.155.1 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.

6.155.2 Member Function Documentation

6.155.2.1 mitkVolume* mitkVolumeToVolumeFilter::GetInput () [inline]

Get the input volume

Returns:
Return the input volume

6.155.2.2 mitkVolume* mitkVolumeToVolumeFilter::GetOutput ()

Get the output volume

Returns:
Return the output volume

6.155.2.3 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, mitkInterpolateFilter, mitkLiveWireImageFilter, mitkNearestNeighborInterpolateFilter, mitkRegionGrowImageFilter, mitkRegistrationFilter, mitkRGBToGrayFilter, mitkSeedFillFilter, mitkSobelEdgeDetectFilter, mitkThresholdSegmentationFilter, mitkVolumeCropFilter, mitkVolumeDataTypeConvertor, mitkVolumeResizeFilter, and mitkVolumeResliceFilter.

6.155.2.4 void mitkVolumeToVolumeFilter::SetInput (mitkVolume * *inData*) [inline]

Set the input volume

Parameters:
inData Specify the input volume

6.155.2.5 void mitkVolumeToVolumeFilter::SetOoCSupport (char const * *diskPath*, unsigned int *bufSliceNum* = 32, 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:

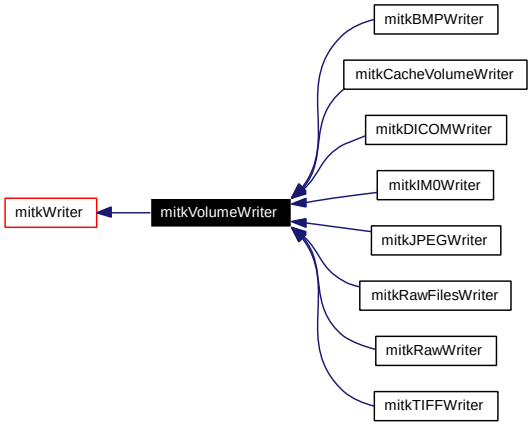
- mitkVolumeToVolumeFilter.h

6.156 mitkVolumeWriter Class Reference

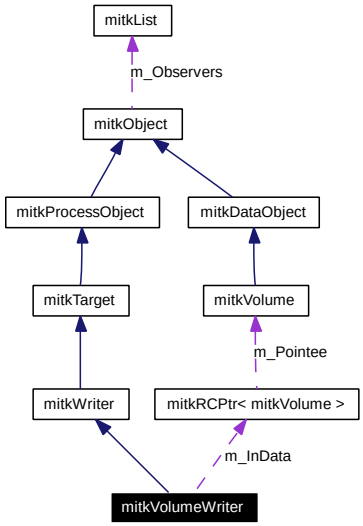
mitkVolumeWriter - an abstract class represents a volume writer for writing image/volume files to disk

`#include <mitkVolumeWriter.h>`

Inheritance diagram for mitkVolumeWriter:



Collaboration diagram for mitkVolumeWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void SetInput (mitkVolume *inData)
- mitkVolume * GetInput ()

Protected Attributes

- mitkRCPtr< mitkVolume > **m_InData**

6.156.1 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();
```

6.156.2 Member Function Documentation

6.156.2.1 mitkVolume* mitkVolumeWriter::GetInput () [inline]

Get input volume.

Returns:
Return the input volume

6.156.2.2 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 mitkBMPWriter, mitkCacheVolumeWriter, mitkDICOMWriter, mitkJPEGWriter, mitkRawFilesWriter, mitkRawWriter, and mitkTIFFWriter.

6.156.2.3 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:

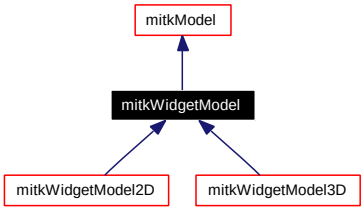
- mitkVolumeWriter.h

6.157 mitkWidgetModel Class Reference

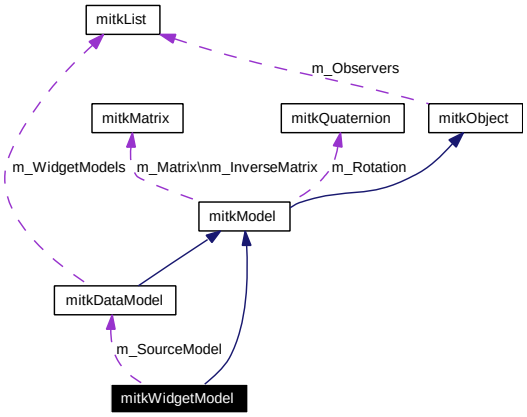
mitkWidgetModel - abstract class used to represent a widget entity (e.g. a line or an angle) in a rendering scene

`#include <mitkWidgetModel.h>`

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 (mitkView *view)
- 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 SetView (mitkView *view)=0
- 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
- void **_pushName** ()
- void **_popNames** ()

Protected Attributes

- mitkDataModel * **m_SourceModel**
- GLuint **m_PickedObj**
- bool **m_LButtonDown**
- bool **m_MButtonDown**
- bool **m_RButtonDown**
- bool **m_IsOpaque**
- bool **m_IsSelectMode**
- bool **m_IsActive**

6.157.1 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.

6.157.2 Member Function Documentation

6.157.2.1 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.

6.157.2.2 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.

6.157.2.3 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.

6.157.2.4 mitkDataModel* mitkWidgetModel::GetSourceModel () [inline]

Get the source model to which this widget attach.

Returns:

Return the pointer to the source model.

6.157.2.5 virtual bool mitkWidgetModel::IsActive () [inline, virtual]

Whether this widget is active.

Returns:

Return true if this widget is active (been selected).

6.157.2.6 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.

6.157.2.7 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.

6.157.2.8 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.

6.157.2.9 virtual bool mitkWidgetModel::IsOpaque () [inline, virtual]

Whether this model is opaque.

Returns:
Return true if this model is opaque.

Implements mitkModel.

6.157.2.10 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

6.157.2.11 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

6.157.2.12 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

6.157.2.13 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.

6.157.2.14 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.

6.157.2.15 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.

6.157.2.16 virtual void mitkWidgetModel::Select (mitkView * *view*) [virtual]

Use select mode to render this model.

Parameters:

view the pointer of an mitkView in which this model is contained.

Reimplemented from mitkModel.

6.157.2.17 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.

6.157.2.18 virtual void mitkWidgetModel::SetView (mitkView * *view*) [pure virtual]

Set the view which contains this model.

Parameters:

view pointer to the mitkView contains this model

Note:

This is a pure virtual function and this class dose not has data member of view 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 view.

Implemented in mitkWidgetModel2D, and mitkWidgetModel3D.

6.157.2.19 virtual void mitkWidgetModel::Update () [pure virtual]

Update the parameters of the widget.

Implemented in mitkAngleWidgetModel3D, mitkClippingPlaneWidgetModel, mitkLineWidgetModel3D, mitkPseudocolorWidgetModelEx, mitkReslicePlaneWidgetModel, mitkWidgetModel2D, and mitkWidgetModel3D.

6.157.2.20 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:

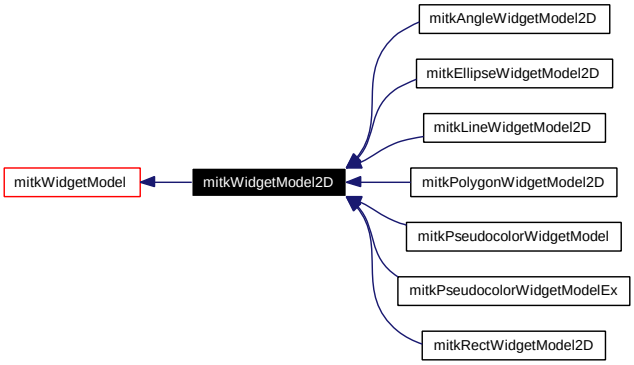
- mitkWidgetModel.h

6.158 mitkWidgetModel2D Class Reference

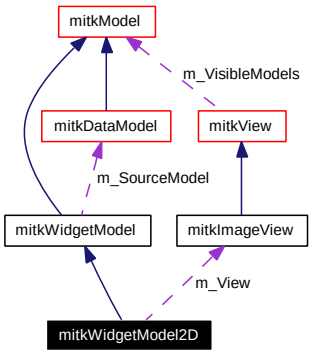
mitkWidgetModel2D - abstract class used to represent a 2D widget entity

`#include <mitkWidgetModel2D.h>`

Inheritance diagram for mitkWidgetModel2D:



Collaboration diagram for mitkWidgetModel2D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void SetView (mitkView ∗view)
- 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 ()

Protected Member Functions

- void **_widgetInit** ()

- void **_widgetFree** ()
- void **_getOriginalCoordinates** (int sx, int sy, float &ox, float &oy)
- void **_getOriginalCoordinates** (float objX, float objY, float &ox, float &oy)
- void **_drawLine** (float startPos[2], float endPos[2], unsigned int name)
- void **_drawLine** (float x1, float y1, float x2, float y2, unsigned int name)
- void **_drawArrow** (float pos[2], float direction[2], float length, float width, unsigned int name)
- void **_drawDisk** (float center[2], float radius, unsigned int name)
- void **_drawDisk** (float cx, float cy, float radius, unsigned int name)
- void **_drawSolidSquare** (float center[2], float sideLength, unsigned int name)
- void **_drawSolidSquare** (float cx, float cy, float sideLength, unsigned int name)

Protected Attributes

- mitkImageView * **m_View**
- GLUquadricObj * **m_QObj**
- float **m_LineWidth**
- float **m_ScaleLength**
- float **m_RotationMatrix** [16]
- float **m_UsualColor** [4]
- float **m_Units** [2]
- int **m_DiskSlice**

6.158.1 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.

6.158.2 Member Function Documentation

6.158.2.1 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.

6.158.2.2 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.

6.158.2.3 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.

6.158.2.4 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)

6.158.2.5 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)

6.158.2.6 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.

6.158.2.7 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

6.158.2.8 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

6.158.2.9 virtual void mitkWidgetModel2D::SetView (mitkView * *view*) [virtual]

Set the view which contains this model.

Parameters:
view pointer to the mitkView contains this model

Implements mitkWidgetModel.

6.158.2.10 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:

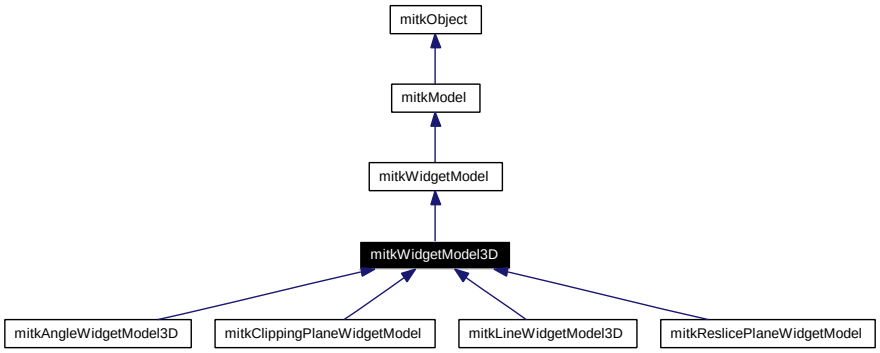
- mitkWidgetModel2D.h

6.159 mitkWidgetModel3D Class Reference

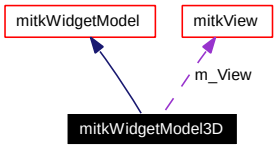
mitkWidgetModel3D - abstract class used to represent a 3D widget entity

`#include <mitkWidgetModel3D.h>`

Inheritance diagram for mitkWidgetModel3D:



Collaboration diagram for mitkWidgetModel3D:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- virtual void SetView (mitkView *view)
- virtual void SetSourceModel (mitkDataModel *model)
- virtual void Update ()

Protected Member Functions

- void **_widgetInit** ()
- void **_widgetFree** ()
- void **_defaultMaterial** ()
- void **_undefaultMaterial** ()
- void **_disableClippingPlanes** ()
- void **_drawLine** (float startPos[3], float endPos[3], unsigned int name)
- void **_drawCone** (float pos[3], float direction[3], float radius, float length, unsigned int name)
- void **_drawCylinder** (float startPos[3], float endPos[3], float radius, bool close, unsigned int name)
- void **_drawSphere** (float pos[3], float radius, unsigned int name)

Protected Attributes

- mitkView * **m_View**
- GLUquadricObj * **m_QObj**
- float **m_LineWidth**
- float **m_CubeSideLength**
- float **m_BallRadius**
- float **m_CylinderRadius**
- float **m_ConeRadius**
- float **m_RotationMatrix** [16]
- float **m_ScaleLength**
- int **m_BallSlice**
- int **m_CylinderSlice**
- int **m_CylinderStack**

6.159.1 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.

6.159.2 Member Function Documentation

6.159.2.1 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.

6.159.2.2 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.

6.159.2.3 virtual void mitkWidgetModel3D::SetView (mitkView * *view*) [inline, virtual]

Set the view which contains this model.

Parameters:
view pointer to the mitkView contains this model

Implements mitkWidgetModel.

6.159.2.4 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:

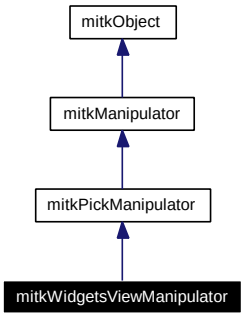
- mitkWidgetModel3D.h

6.160 mitkWidgetsViewManipulator Class Reference

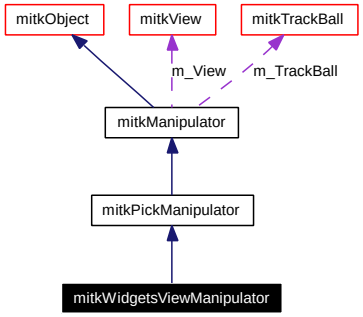
mitkWidgetsViewManipulator - manipulator of a view contains widgets

`#include <mitkWidgetsViewManipulator.h>`

Inheritance diagram for mitkWidgetsViewManipulator:



Collaboration diagram for mitkWidgetsViewManipulator:



Public Member Functions

- **mitkWidgetsViewManipulator** (mitkView *view)
- 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)

6.160.1 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.

6.160.2 Member Function Documentation

6.160.2.1 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.

6.160.2.2 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.

6.160.2.3 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.

6.160.2.4 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:

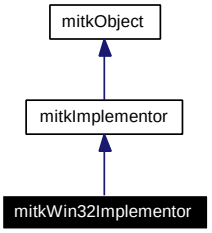
- mitkWidgetsViewManipulator.h

6.161 mitkWin32Implementor Class Reference

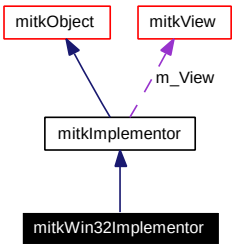
mitkWin32Implementor - a concrete implementor class to implement some OS dependent operations

`#include <mitkWin32Implementor.h>`

Inheritance diagram for mitkWin32Implementor:



Collaboration diagram for mitkWin32Implementor:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- **mitkWin32Implementor** (mitkView *view)
- virtual void * **GetWindowId** (void)
- virtual void **SetParent** (void *parentId)
- virtual void * **GetParent** (void)
- virtual void * **GetApplicationId** (void)
- virtual void * **GetDeviceContext** (void)
- virtual void * **GetRenderContext** (void)
- virtual void **SetPosition** (int leftPos, int topPos)
- virtual void **SetSize** (int widthSize, int heightSize)
- virtual void **SetWindowName** (const char *winName)
- virtual void **Show** ()
- virtual void **Hide** ()
- virtual void **Update** ()
- virtual void **MakeCurrent** ()
- virtual void **SwapBuffers** ()
- virtual bool **HasUnprocessedMouseMessage** ()

Protected Member Functions

- virtual LRESULT **_messageHandler** (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

Static Protected Member Functions

- static LRESULT APIENTRY **WndProc** (HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

Protected Attributes

- HINSTANCE **m_ApplicationInstance**
- HGLRC **m_RenderContext**
- HDC **m_DeviceContext**
- HWND **m_WindowHandle**
- HWND **m_ParentHandle**

6.161.1 Detailed Description

mitkWin32Implementor - a concrete implementor class to implement some OS dependent operations

mitkWin32Implementor is a concrete implementor class to implement some OS dependent operations. It is created automatically if the current OS is Windows. User needn't call its member function.

6.161.2 Member Function Documentation

6.161.2.1 virtual void mitkWin32Implementor::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 mitkImplementor.

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

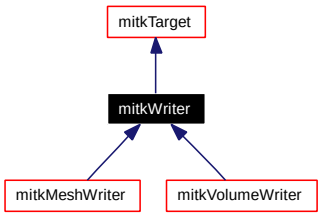
- mitkWin32Implementor.h

6.162 mitkWriter Class Reference

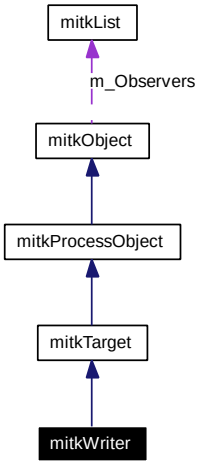
mitkWriter - an abstract class represents a writer

`#include <mitkWriter.h>`

Inheritance diagram for mitkWriter:



Collaboration diagram for mitkWriter:



Public Member Functions

- virtual void PrintSelf (ostream &os)
- void AddFileName (const char *inFileName)
- void SortFileNames ()

Protected Member Functions

- int **_getFileCount** (void)
- const char * **_getFileName** (int nIndex)

Protected Attributes

- mitkStringList * **m_FileNames**

6.162.1 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();
```

6.162.2 Member Function Documentation

6.162.2.1 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.

6.162.2.2 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 mitkBMPWriter, mitkCacheVolumeWriter, mitkDICOMWriter, mitkJPEGWriter, mitkMeshWriter, mitkPLYASCIIWriter, mitkPLYBinaryWriter, mitkRawFilesWriter, mitkRawWriter, mitkSTLASCIIWriter, mitkSTLBinaryWriter, mitkTIFFWriter, and mitkVolumeWriter.

6.162.2.3 void mitkWriter::SortFileNames ()

Sort all the file names alphabetically.

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

- mitkWriter.h