MediGL

Generated by Doxygen 1.6.3

Thu Jul 1 20:13:42 2010

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 C4UBV3F Struct Reference

Public Attributes

- unsigned char **color** [4]
- float **vcoords** [3]

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

- glwidget.cpp

2.2 DICOMImageFile Class Reference

```
#include <dicomimagefile.h>
```

Public Member Functions

- DICOMImageFile (string filename)
- FastImage ∗ getFastImage (uint frame)
- uint getWidth ()
- uint getHeight ()
- uint getFrameCount ()
- ∼DICOMImageFile ()

Protected Attributes

- DicomImage ∗ **image**
- uint **width**
- uint **height**
- uint **frameCount**

2.2.1 Detailed Description

Wrapper class for easy use of the DICOM file format (DICOM images only) Uses the DCMTK library to process DICOM files.

2.2.2 Constructor & Destructor Documentation

2.2.2.1 DICOMImageFile::DICOMImageFile (string *filename*)

Constructs a new DICOMImageFile instance by a given filename Occuring errors are logged to cerr.

2.2.2.2 DICOMImageFile::∼DICOMImageFile ()

Releases all resources acquired by this DICOMImageFile instance

2.2.3 Member Function Documentation

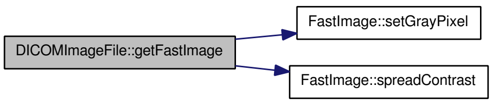
2.2.3.1 FastImage ∗ DICOMImageFile::getFastImage (uint *frame*)

Constructs a new FastImage instance with the contents of a specific file this DICOM image file

Parameters

frame	The number of the frame (beginning with 0) to use

Here is the call graph for this function:



2.2.3.2 uint DICOMImageFile::getFrameCount () `[inline]`

Returns

The frame count of this image

2.2.3.3 uint DICOMImageFile::getHeight () `[inline]`

Returns

The height of this image

2.2.3.4 uint DICOMImageFile::getWidth () `[inline]`

Returns

The width of this image

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

- dicomimagefile.h
- dicomimagefile.cpp

2.3 FastImage Class Reference

`#include <fastimage.h>`

Public Member Functions

- FastImage (QImage ∗img, bool enableGrayCache=true)
- FastImage (uint width, uint height, bool enableGrayCache=true)
- ∼FastImage ()
- uint32_t getRgba (uint x, uint y)
- char getGray (uint x, uint y)
- double getGray32bit (uint x, uint y)
- void setPixel (uint x, uint y, uint32_t val)
- void setGrayPixel (uint x, uint y, unsigned char val)
- void setGrayPixel (uint x, uint y, uint32_t val)
- void spreadContrast ()
- uint getWidth ()
- uint getHeight ()

Protected Attributes

- uint **width**
- uint **height**
- bool **grayCacheEnabled**
- uint32_t ∗ **colorData**
- double ∗ **grayData**

2.3.1 Detailed Description

Image wrapper class internally using arrays for fast read access Optimized for read access Compiling with -fno-strict-aliasing should make this class faster

One instance of this class represents exactly one image. FastImage can build a gray cache to be able to serve getGray() requests very fast. This feature is enabled by default but can be disabled. Users are not encouraged to change the data using setPixel(...).

FastImage internally stores the data using 64 bit floating point numbers

FastImage is optimized to process grayscale images - some functions only work on grayscale images

The purpose of this class is to have a fast, scalable abstraction layer between MediGL input data and OpenGL in order to be able to support a variety of image formats.

2.3.2 Constructor & Destructor Documentation

2.3.2.1 FastImage::FastImage (QImage ∗ *img*, bool *enableGrayCache* = `true`)

Creates a new FastImage instance from a QImage. Uses the data from the QImage instance to fill the cache

Parameters

img	The image to process
enableGrayCache	Whether to enable a separate gray cache

2.3.2.2 FastImage::FastImage (uint *width*, uint *height*, bool *enableGrayCache* = `true`)

Creates a new FastImage instance with given width and height but without any content. Use setPixel(...) to set the pixels

Parameters

width The widi of the new image

height The height of the new image

enableGrayCache Whether to enable a separate gray cache

2.3.2.3 FastImage::∼FastImage ()

Releases all memory occupied by this FastImage instance

2.3.3 Member Function Documentation

2.3.3.1 char FastImage::getGray (uint *x*, uint *y*)

Gets the gray value (8 bit) for specific x and y pixel coordinates. The request is served from the gray cache if it has been enabled for this instance

2.3.3.2 double FastImage::getGray32bit (uint *x*, uint *y*)

Gets the gray value (8 bit) for specific x and y pixel coordinates. The request is served from the gray cache if it has been enabled for this instance

2.3.3.3 uint FastImage::getHeight () `[inline]`

Returns

The height of this image

2.3.3.4 uint32_t FastImage::getRgba (uint *x*, uint *y*)

Gets the RGBA value for a specific pixel in this FastImage instance.

This function does NOT check if the x and y parameters are in the bounds of this FastImage instance for sake of performance

2.3.3.5 uint FastImage::getWidth () `[inline]`

Returns

The width of this image

2.3.3.6 void FastImage::setGrayPixel (uint *x*, uint *y*, uint32_t *val*)

Sets a pixel to a specific gray value. The color buffer is not affected.

Parameters

x The x coordinate of the pixel to set
y The x coordinate of the pixel to set
val The 32-bit grayscale value to set the pixel to

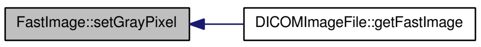
2.3.3.7 void FastImage::setGrayPixel (uint *x*, uint *y*, unsigned char *val*)

Sets a pixel to a specific gray value. The color buffer is not affected.

Parameters

x The x coordinate of the pixel to set
y The x coordinate of the pixel to set
val The grayscale value to set the pixel to

Here is the caller graph for this function:



2.3.3.8 void FastImage::setPixel (uint *x*, uint *y*, uint32_t *val*)

Sets a pixel in this FastImage instance to a specific value and updates the gray cache if it is enabled

Parameters

x The x coordinate of the pixel to set
y The x coordinate of the pixel to set
val The RGBA value to set the pixel to

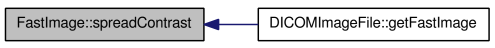
2.3.3.9 void FastImage::spreadContrast ()

Performs a constrast spreading on the gray data of this FastImage.

The algorithm used has a linear complexity

Note: The color data is not affected!

Here is the caller graph for this function:



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

- fastimage.h
- fastimage.cpp

2.4 GLWidget Class Reference

```
#include <glwidget.h>
```

Public Slots

- void **setXRotation** (int angle)
- void **setYRotation** (int angle)
- void **setZRotation** (int angle)

Signals

- void **xRotationChanged** (int angle)
- void **yRotationChanged** (int angle)
- void **zRotationChanged** (int angle)

Public Member Functions

- **GLWidget** (QWidget ∗parent=0)
- void updateImages (vector< FastImage ∗ > imagesParam, uint width, uint height)
- void resetView ()
- void setZExtent (float newZExtent)
- QSize **minimumSizeHint** () const
- QSize **sizeHint** () const
- void keyPressEvent (QKeyEvent ∗)

Protected Member Functions

- void **initializeGL** ()
- void **paintGL** ()
- void **resizeGL** (int width, int height)
- void mousePressEvent (QMouseEvent ∗event)
- void mouseMoveEvent (QMouseEvent ∗event)
- void wheelEvent (QWheelEvent ∗)

2.4.1 Detailed Description

MediGL OpenGL widget Controls and the OpenGL IO, displays the rendered data, reacts to user events and processes images

2.4.2 Member Function Documentation

2.4.2.1 void GLWidget::keyPressEvent (QKeyEvent ∗ *event*)

Reacts to a key event. Key events are translated into translation commands.

All translations are absolute and not dependent on the rotation. The translation amount is dependent on the zoom factor.

Left/right arrow keys: x coordinates Up/down arrow keys: y coordinates PageUp/PageDown: z coordinates +/-: zoom

2.4.2.2 void GLWidget::mouseMoveEvent (QMouseEvent ∗ *event*) `[protected]`

Reacts to a mouse move event. This is part of the rotation code which rotates the data when the user uses drag-and-drop

2.4.2.3 void GLWidget::mousePressEvent (QMouseEvent ∗ *event*) `[protected]`

Reacts to a mouse press event. This is part of the rotation code which rotates the data when the user uses drag-and-drop

2.4.2.4 void GLWidget::resetView () `[inline]`

Resets rotation, translation and zoom and re-renders the data.

2.4.2.5 void GLWidget::setZExtent (float *newZExtent*) `[inline]`

Sets the z (depth) extent of the rendered image cuboid 1.0 is the same as the maximum of with and height

2.4.2.6 void GLWidget::updateImages (vector< FastImage ∗ > *imagesParam*, uint *width*, uint *height*) `[inline]`

Updates the image cache with new images (represented by a vector of FastImage pointers) with a given width and height.

The images must be checked for equal width and height before - the GLWidget class does not check them for performance reasons.

2.4.2.7 void GLWidget::wheelEvent (QWheelEvent ∗ *event*) `[protected]`

Reacts to a mouse wheel event. Mouse wheel events are translated into zoom factor changes.

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

- glwidget.h
- glwidget.cpp

2.5 MediDialog Class Reference

Public Member Functions

- **MediDialog** (QWidget ∗parent, vector< FastImage ∗ > images, uint width, uint height)

Protected Member Functions

- void **changeEvent** (QEvent ∗e)
- void **keyPressEvent** (QKeyEvent ∗)

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

- medidialog.h
- medidialog.cpp

Index