# Engauge Digitizer

2

Generated by Doxygen 1.8.13

# Contents

# Chapter 1

# Engauge Digitizer II

The Engauge Digitizer II application quickly extracts numeric data from images containing graphs with curves and two axes drawn. Converting an image into data may be described as doing the opposite of graphing - which converts numeric data into graph images.

### For users

Major features added since version Engauge Digitizer 5.2 include:

- Sub-pixel point placement increases accuracy
- Undo/redo makes recovering from mistakes easy
- Easier and more powerful zooming
- Improved drag-and-drop
- Wizard provides an interactive tutorial to explain the basic steps
- Wizard creates a checklist guide to interactively leads user through steps from file import to file export
- MSI installer for Windows operating system

### For developers

Engauge Digitizer Version 2 uses the new `Qt5` library, rather than the old `Qt3` library used by Version 1. The Qt3 library is disappearing from most operating systems, but Qt5 should be available for many years.

The code takes advantage of some powerful open source toolkits:

- **`CCCC`** provides important metrics to identify possible problem areas (run docccc in src directory)
- **`Doxygen`** documents all C++ classes (run doxygen in src directory)
- **`FFTW`** provides a fast-fourier transform (FFT) for faster image processing, especially cross-correlations
- **`Log4cpp`** provides configurable logging
- **`OpenJPEG`** supports JPEG2000 images on systems without support for that format

The code is architected with some important design patterns:

- **Command pattern** provides Undo/Redo using commands and a command stack, and also provides robust data transfer between threads using commands and a FIFO command queue
- **Factory pattern** generates points with the details encapsulated in the factory class
- **Functor pattern** provides efficient processing of Curve and Point data from outside the Document class, without violating encapsulation of performed by generic iteration through the Curves with functors
- **Model/View pattern** separates graphical object management in the GraphicsScene, and graphical presentation and interaction in the GraphicsView. Delegates, representing the document, interact with the Model and View
- **State pattern** isolates each digitizing mode into one state, with a context class acting as a container and single class for interfacing across the state machine boundary
- **Strategy pattern** encapsulates code chunks when the chunk to be used depends on the current context

Code development has been moved from sourceforge.net to github.com, and Doxygen documentation has been added.

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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

# Chapter 3

# Class Index

## 3.1 Class List

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

# Chapter 4

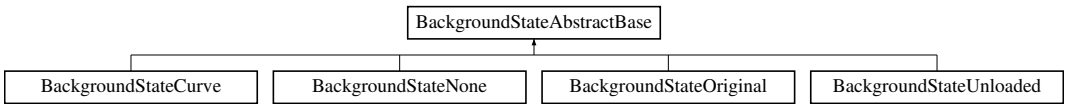
# Class Documentation

## 4.1 BackgroundStateAbstractBase Class Reference

Background image state machine state base class.

`#include <BackgroundStateAbstractBase.h>`

Inheritance diagram for BackgroundStateAbstractBase:




### Public Member Functions

- BackgroundStateAbstractBase (BackgroundStateContext &context, GraphicsScene &scene)

  *Single constructor.*
- virtual void begin ()=0

  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- BackgroundStateContext & context ()

  *Reference to the BackgroundStateContext that contains all the BackgroundStateAbstractBase subclasses, without const.*
- const BackgroundStateContext & context () const

  *Reference to the BackgroundStateContext that contains all the BackgroundStateAbstractBase subclasses, without const.*
- virtual void end ()=0

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void fitInView (GraphicsView &view)=0

  *Zoom so background fills the window.*
- QImage image () const

  *Image for the current state.*
- QGraphicsPixmapItem & imageItem () const

  *Graphics image item for the current state.*
- GraphicsScene & scene ()

  *Reference to the GraphicsScene, without const.*

- const GraphicsScene & scene () const

  *Reference to the GraphicsScene, without const.*

- virtual void setCurveSelected (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)=0

  *Update the currently selected curve name.*

- virtual void setPixmap (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &modelColorFilter, const QPixmap &pixmap, const QString &curveSelected)=0

  *Update the image for this state, after the leaf class processes it appropriately.*

- virtual QString state () const =0

  *State name for debugging.*

- virtual void updateColorFilter (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &modelColorFilter, const QString &curveSelected)=0

  *Apply color filter settings.*

### Protected Member Functions

- void setImageVisible (bool visible)

  *Show/hide background image.*

- void setProcessedPixmap (const QPixmap &pixmap)

  *Save the image for this state after it has been processed by the leaf class.*

#### 4.1.1 Detailed Description

Background image state machine state base class.

Definition at line 30 of file BackgroundStateAbstractBase.h.

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

- Background/BackgroundStateAbstractBase.h
- Background/BackgroundStateAbstractBase.cpp

## 4.2 BackgroundStateContext Class Reference

Context class that manages the background image state machine.

```
#include <BackgroundStateContext.h>
```

### Public Member Functions

- BackgroundStateContext (MainWindow &mainWindow)

  *Single constructor.*
- void close ()

  *Open Document is being closed so remove the background.*
- void fitInView (GraphicsView &view)

  *Zoom so background fills the window.*
- QImage imageForCurveState () const

  *Image for the Curve state, even if the current state is different.*
- void requestStateTransition (BackgroundState backgroundState)

  *Initiate state transition to be performed later, when BackgroundState is off the stack.*
- void setBackgroundImage (BackgroundImage backgroundImage)

  *Transition to the specified state. This method is used by classes outside of the state machine to trigger transitions.*
- void setCurveSelected (const Transformation &transformation, const DocumentModelGridRemoval &model↩GridRemoval, const DocumentModelColorFilter &modelColorFilter, const QString &curveSelected)

  *Update the selected curve.*
- void setPixmap (const Transformation &transformation, const DocumentModelGridRemoval &modelGrid↩Removal, const DocumentModelColorFilter &modelColorFilter, const QPixmap &pixmapOriginal, const Q↩String &curveSelected)

  *Update the images of all states, rather than just the current state.*
- void updateColorFilter (const Transformation &transformation, const DocumentModelGridRemoval &model↩GridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Apply color filter settings.*

#### 4.2.1 Detailed Description

Context class that manages the background image state machine.

Overall strategy is that changing the currently selected curve should not affect the background image if the original image is being shown, or no image is being shown. However, if the curve-specific color filter image is being shown, then it should be replaced by the filtered image specific to the new curve.

Other considerations are that the processing should be robust in terms of ordering of the following incoming events:

1. State transitions
2. Setting of the background image
3. Setting of the currently selected curve name

Definition at line 32 of file BackgroundStateContext.h.

#### 4.2.2 Member Function Documentation

#### 4.2.2.1 setCurveSelected()

```
void BackgroundStateContext::setCurveSelected (
            const Transformation & transformation,
            const DocumentModelGridRemoval & modelGridRemoval,
            const DocumentModelColorFilter & modelColorFilter,
            const QString & curveSelected )
```

Update the selected curve.

Although this probably affects only the BACKGROUND_STATE_CURVE state, we will forward it to all states (consistent with setPixmap)

Definition at line 129 of file BackgroundStateContext.cpp.

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

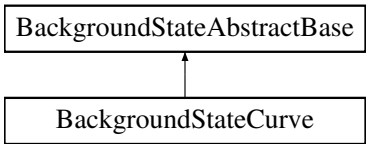
- Background/BackgroundStateContext.h
- Background/BackgroundStateContext.cpp

## 4.3 BackgroundStateCurve Class Reference

Background image state for showing filter image from current curve.

`#include <BackgroundStateCurve.h>`

Inheritance diagram for BackgroundStateCurve:




### Public Member Functions

- BackgroundStateCurve (BackgroundStateContext &context, GraphicsScene &scene)

  *Single constructor.*
- virtual void begin ()

  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void fitInView (GraphicsView &view)

  *Zoom so background fills the window.*
- virtual void setCurveSelected (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Update the currently selected curve name.*
- virtual void setPixmap (const Transformation &transformation, const DocumentModelGridRemoval &model↩ GridRemoval, const DocumentModelColorFilter &modelColorFilter, const QPixmap &pixmapOriginal, const QString &curveSelected)

  *Update the image for this state, after the leaf class processes it appropriately.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateColorFilter (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Apply color filter settings.*

### Additional Inherited Members

#### 4.3.1 Detailed Description

Background image state for showing filter image from current curve.

Definition at line 13 of file BackgroundStateCurve.h.

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

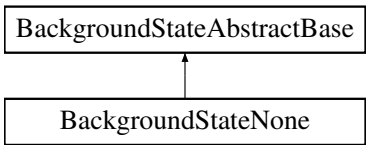
- Background/BackgroundStateCurve.h
- Background/BackgroundStateCurve.cpp

## 4.4 BackgroundStateNone Class Reference

Background image state for showing no image.

`#include <BackgroundStateNone.h>`

Inheritance diagram for BackgroundStateNone:




### Public Member Functions

- BackgroundStateNone (BackgroundStateContext &context, GraphicsScene &scene)

  *Single constructor.*
- virtual void begin ()

  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void fitInView (GraphicsView &view)

  *Zoom so background fills the window.*
- virtual void setCurveSelected (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Update the currently selected curve name.*
- virtual void setPixmap (const Transformation &transformation, const DocumentModelGridRemoval &model↩GridRemoval, const DocumentModelColorFilter &modelColorFilter, const QPixmap &pixmap, const QString &curveSelected)

  *Update the image for this state, after the leaf class processes it appropriately.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateColorFilter (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Apply color filter settings.*

### Additional Inherited Members

#### 4.4.1 Detailed Description

Background image state for showing no image.

Definition at line 13 of file BackgroundStateNone.h.

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

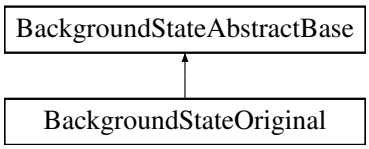
- Background/BackgroundStateNone.h
- Background/BackgroundStateNone.cpp

## 4.5 BackgroundStateOriginal Class Reference

Background image state for showing original (=unfiltered) image.

`#include <BackgroundStateOriginal.h>`

Inheritance diagram for BackgroundStateOriginal:




### Public Member Functions

- BackgroundStateOriginal (BackgroundStateContext &context, GraphicsScene &scene)

  *Single constructor.*
- virtual void begin ()

  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void fitInView (GraphicsView &view)

  *Zoom so background fills the window.*
- virtual void setCurveSelected (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Update the currently selected curve name.*
- virtual void setPixmap (const Transformation &transformation, const DocumentModelGridRemoval &model↩GridRemoval, const DocumentModelColorFilter &modelColorFilter, const QPixmap &pixmap, const QString &curveSelected)

  *Update the image for this state, after the leaf class processes it appropriately.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateColorFilter (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Apply color filter settings.*

### Additional Inherited Members

#### 4.5.1 Detailed Description

Background image state for showing original (=unfiltered) image.

Definition at line 13 of file BackgroundStateOriginal.h.

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

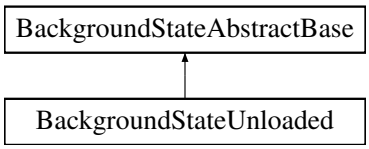
- Background/BackgroundStateOriginal.h
- Background/BackgroundStateOriginal.cpp

## 4.6 BackgroundStateUnloaded Class Reference

Background image state for interval between startup and loading of the image.

`#include <BackgroundStateUnloaded.h>`

Inheritance diagram for BackgroundStateUnloaded:




### Public Member Functions

- BackgroundStateUnloaded (BackgroundStateContext &context, GraphicsScene &scene)

  *Single constructor.*
- virtual void begin ()

  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void fitInView (GraphicsView &view)

  *Zoom so background fills the window.*
- virtual void setCurveSelected (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Update the currently selected curve name.*
- virtual void setPixmap (const Transformation &transformation, const DocumentModelGridRemoval &model↩GridRemoval, const DocumentModelColorFilter &modelColorFilter, const QPixmap &pixmap, const QString &curveSelected)

  *Update the image for this state, after the leaf class processes it appropriately.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateColorFilter (const Transformation &transformation, const DocumentModelGridRemoval &modelGridRemoval, const DocumentModelColorFilter &colorFilter, const QString &curveSelected)

  *Apply color filter settings.*

### Additional Inherited Members

#### 4.6.1 Detailed Description

Background image state for interval between startup and loading of the image.

Definition at line 13 of file BackgroundStateUnloaded.h.

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

- Background/BackgroundStateUnloaded.h
- Background/BackgroundStateUnloaded.cpp

## 4.7 CallbackAddPointsInCurvesGraphs Class Reference

Callback that is used when iterating through a read-only CurvesGraphs to add corresponding points in Document.

`#include <CallbackAddPointsInCurvesGraphs.h>`

### Public Member Functions

- CallbackAddPointsInCurvesGraphs (CoordSystem &coordSystem)
  *Single constructor.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)
  *Callback method.*

#### 4.7.1 Detailed Description

Callback that is used when iterating through a read-only CurvesGraphs to add corresponding points in Document.

Definition at line 17 of file CallbackAddPointsInCurvesGraphs.h.

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

- Callback/CallbackAddPointsInCurvesGraphs.h
- Callback/CallbackAddPointsInCurvesGraphs.cpp

## 4.8 CallbackAxesCheckerFromAxesPoints Class Reference

Callback for positioning the axes indicator according to the axes points.

`#include <CallbackAxesCheckerFromAxesPoints.h>`

### Public Member Functions

- CallbackAxesCheckerFromAxesPoints ()
  *Single constructor.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)
  *Callback method.*
- Points points () const
  *Return the collected points as a polygon, in screen coordinates.*

#### 4.8.1 Detailed Description

Callback for positioning the axes indicator according to the axes points.

Definition at line 17 of file CallbackAxesCheckerFromAxesPoints.h.

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

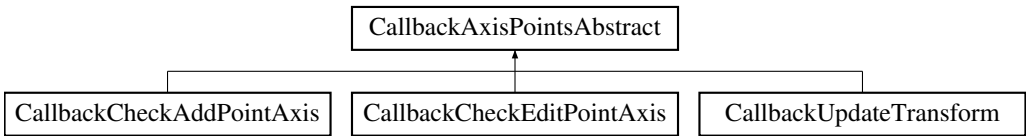
- Callback/CallbackAxesCheckerFromAxesPoints.h
- Callback/CallbackAxesCheckerFromAxesPoints.cpp

## 4.9 CallbackAxisPointsAbstract Class Reference

Callback for collecting axis points and then performing common calculations on those axis points.

`#include <CallbackAxisPointsAbstract.h>`

Inheritance diagram for CallbackAxisPointsAbstract:




### Public Member Functions

- CallbackAxisPointsAbstract (const DocumentModelCoords &modelCoords, DocumentAxesPointsRequired documentAxesPointsRequired)
  *Constructor for when all of the existing axis points are to be processed as is.*
- CallbackAxisPointsAbstract (const DocumentModelCoords &modelCoords, const QString pointIdentifier↩ Override, const QPointF &posGraphOverride, const QPointF &posScreenOverride, DocumentAxesPoints↩ Required documentAxesPointsRequired)
  *Constructor for when the data for one of the existing axis points is to be locally overwritten.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)
  *Callback method.*
- QTransform matrixGraph () const
  *Returns graph coordinates matrix after transformIsDefined has already indicated success.*
- QTransform matrixScreen () const
  *Returns screen coordinates matrix after transformIsDefined has already indicated success.*
- double xGraphRange () const
  *Return the range of the x graph coordinate from low to high, after the transform is defined.*
- double yGraphRange () const
  *Return the range of the y graph coordinate from low to high, after the transform is defined.*

### Protected Member Functions

- DocumentAxesPointsRequired documentAxesPointsRequired () const

  *Number of axes points required for the transformation.*
- QString errorMessage () const

  *This value is checked after iterating to see what was wrong if the axis data was incorrect.*
- bool isError () const

  *This value is checked after iterating to see if the axis data is correct.*
- unsigned int numberAxisPoints () const

  *Number of axis points which is less than 3 if the axes curve is incomplete.*

### Friends

- class TestGraphCoords

  *For unit testing.*

## 4.9.1 Detailed Description

Callback for collecting axis points and then performing common calculations on those axis points.

This class collects 3x3 matrix G which contains columns of graph coordinates, and 3x3 matrix S which contains columns of screen coordinates. Although it goes almost as far as solving (G) = (T) (S) for the transformation T, that is left for the Transformation class. This class does, however, do the sanity checking (like for collinear points) so the gui can provide immediate feedback to the user well before the Transformation class gets involved

This class is versatile. The cases are:

1. Use all existing axis points, and then the subclass can effectively append one more point to check if that additional point would violate any constraints (prior to adding the point)
2. Use all existing axis points, but override the details of one existing axis point to see if those details violate any constraint (prior to editing the point)
3. Use all existing axis points as is. This is for computing the transformation after axis points are added/edited

Definition at line 35 of file CallbackAxisPointsAbstract.h.

## 4.9.2 Member Function Documentation

### 4.9.2.1 isError()

```
bool CallbackAxisPointsAbstract::isError ( ) const  [inline], [protected]
```

This value is checked after iterating to see if the axis data is correct.

The error state does NOT include the case when there are not enough axis points

Definition at line 80 of file CallbackAxisPointsAbstract.h.

#### 4.9.2.2 matrixGraph()

```
QTransform CallbackAxisPointsAbstract::matrixGraph ( ) const
```

Returns graph coordinates matrix after transformIsDefined has already indicated success.

Since QMatrix is deprecated the results are returned as QTransform

Definition at line 459 of file CallbackAxisPointsAbstract.cpp.

#### 4.9.2.3 matrixScreen()

```
QTransform CallbackAxisPointsAbstract::matrixScreen ( ) const
```

Returns screen coordinates matrix after transformIsDefined has already indicated success.

Since QMatrix is deprecated the results are returned as QTransform

Definition at line 464 of file CallbackAxisPointsAbstract.cpp.

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

- Callback/CallbackAxisPointsAbstract.h
- Callback/CallbackAxisPointsAbstract.cpp

## 4.10 CallbackBoundingRects Class Reference

Callback for computing the bounding rectangles of the screen and graph coordinates of the points in the Document.

```
#include <CallbackBoundingRects.h>
```

### Public Member Functions

- CallbackBoundingRects (const Transformation &transformation)

  *Single constructor.*
- QRectF boundingRectGraph (bool &isEmpty) const

  *Graph coordinate bounding rectangle.*
- QRectF boundingRectScreen (bool &isEmpty) const

  *Screen coordinate bounding rectangle.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)

  *Callback method.*

### 4.10.1 Detailed Description

Callback for computing the bounding rectangles of the screen and graph coordinates of the points in the Document.

Definition at line 19 of file CallbackBoundingRects.h.

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

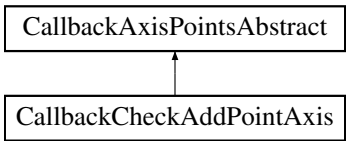
- Callback/CallbackBoundingRects.h
- Callback/CallbackBoundingRects.cpp

## 4.11 CallbackCheckAddPointAxis Class Reference

Callback for sanity checking the screen and graph coordinates of an axis point, before it is added to the axes curve.

`#include <CallbackCheckAddPointAxis.h>`

Inheritance diagram for CallbackCheckAddPointAxis:




### Public Member Functions

- CallbackCheckAddPointAxis (const DocumentModelCoords &modelCoords, const QPointF &posScreen, const QPointF &posGraph, DocumentAxesPointsRequired documentAxesPointsRequired, bool isXOnly)

  *Single constructor.*
- bool isError () const

  *True if an error occurred during iteration.*
- QString errorMessage () const

  *Error message that explains the problem indicated by isError.*

### Additional Inherited Members

### 4.11.1 Detailed Description

Callback for sanity checking the screen and graph coordinates of an axis point, before it is added to the axes curve.

Definition at line 18 of file CallbackCheckAddPointAxis.h.

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

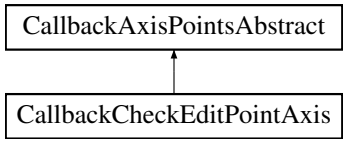
- Callback/CallbackCheckAddPointAxis.h
- Callback/CallbackCheckAddPointAxis.cpp

## 4.12 CallbackCheckEditPointAxis Class Reference

Callback for sanity checking the screen and graph coordinates of an axis point that is in the axes curve, before changing its graph coordinates.

`#include <CallbackCheckEditPointAxis.h>`

Inheritance diagram for CallbackCheckEditPointAxis:




### Public Member Functions

- CallbackCheckEditPointAxis (const DocumentModelCoords &modelCoords, const QString &pointIdentifier, const QPointF &posScreen, const QPointF &posGraph, DocumentAxesPointsRequired documentAxes↩PointsRequired)

  *Single constructor.*
- bool isError () const

  *True if an error occurred during iteration.*
- QString errorMessage () const

  *Error message that explains the problem indicated by isError.*

### Additional Inherited Members

#### 4.12.1 Detailed Description

Callback for sanity checking the screen and graph coordinates of an axis point that is in the axes curve, before changing its graph coordinates.

Definition at line 18 of file CallbackCheckEditPointAxis.h.

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

- Callback/CallbackCheckEditPointAxis.h
- Callback/CallbackCheckEditPointAxis.cpp

## 4.13 CallbackDocumentHash Class Reference

Callback for DocumentHash value for a Document.

`#include <CallbackDocumentHash.h>`

### Public Member Functions

- CallbackDocumentHash (DocumentAxesPointsRequired documentAxesPointsRequired)
  *Single constructor.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)
  *Callback method.*
- DocumentHash hash () const
  *Computed hash value.*

#### 4.13.1 Detailed Description

Callback for DocumentHash value for a Document.

Definition at line 19 of file CallbackDocumentHash.h.

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

- Callback/CallbackDocumentHash.h
- Callback/CallbackDocumentHash.cpp

## 4.14 CallbackGatherXThetaValuesFunctions Class Reference

Callback for collecting X/Theta independent variables, for functions, in preparation for exporting.

`#include <CallbackGatherXThetaValuesFunctions.h>`

### Public Member Functions

- CallbackGatherXThetaValuesFunctions (const DocumentModelExportFormat &modelExport, const QStringList &curveNamesIncluded, const Transformation &transformation)
  *Single constructor.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)
  *Callback method.*
- ValuesVectorXOrY xThetaValuesRaw () const
  *Resulting x/theta values for all included functions.*

#### 4.14.1 Detailed Description

Callback for collecting X/Theta independent variables, for functions, in preparation for exporting.

Definition at line 27 of file CallbackGatherXThetaValuesFunctions.h.

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

- Callback/CallbackGatherXThetaValuesFunctions.h
- Callback/CallbackGatherXThetaValuesFunctions.cpp

## 4.15 CallbackNextOrdinal Class Reference

Callback for computing the next ordinal for a new point.

`#include <CallbackNextOrdinal.h>`

### Public Member Functions

- CallbackNextOrdinal (const QString &curveName)

  *Single constructor.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)

  *Callback method.*
- double nextOrdinal () const

  *Computed next ordinal.*

### 4.15.1 Detailed Description

Callback for computing the next ordinal for a new point.

Definition at line 17 of file CallbackNextOrdinal.h.

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

- Callback/CallbackNextOrdinal.h
- Callback/CallbackNextOrdinal.cpp

## 4.16 CallbackPointOrdinal Class Reference

Callback for computing the ordinal for a specified point, as a function of the LineStyle and curve geometry.

`#include <CallbackPointOrdinal.h>`

### Public Member Functions

- CallbackPointOrdinal (const LineStyle &lineStyle, const Transformation &transformation, const QPointF &posScreen)

  *Single constructor.*
- CallbackSearchReturn callback (const Point &pointStart, const Point &pointStop)

  *Callback method.*
- double ordinal () const

  *Computed ordinal.*

### 4.16.1 Detailed Description

Callback for computing the ordinal for a specified point, as a function of the LineStyle and curve geometry.

Definition at line 19 of file CallbackPointOrdinal.h.

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

- Callback/CallbackPointOrdinal.h
- Callback/CallbackPointOrdinal.cpp

## 4.17 CallbackRemovePointsInCurvesGraphs Class Reference

Callback that is used when iterating through a read-only CurvesGraphs to remove corresponding points in Document.

`#include <CallbackRemovePointsInCurvesGraphs.h>`

### Public Member Functions

- CallbackRemovePointsInCurvesGraphs (CoordSystem &coordSystem)

  *Single constructor.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)

  *Callback method.*

### 4.17.1 Detailed Description

Callback that is used when iterating through a read-only CurvesGraphs to remove corresponding points in Document.

Definition at line 17 of file CallbackRemovePointsInCurvesGraphs.h.

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

- Callback/CallbackRemovePointsInCurvesGraphs.h
- Callback/CallbackRemovePointsInCurvesGraphs.cpp

## 4.18 CallbackScaleBar Class Reference

Callback for identifying, for the scale bar of a map, various quantities.

`#include <CallbackScaleBar.h>`

### Public Member Functions

- CallbackScaleBar ()
  *Single constructor.*
- QStringList axisCurvePointIdentifiers () const
  *Points in axis curve.*
- CallbackSearchReturn callback (const QString &curveName, const Point &point)
  *Callback method.*
- double scaleBarLength () const
  *Length of scale bar.*
- QString scaleBarPointIdentifier () const
  *Identified axis point.*

#### 4.18.1 Detailed Description

Callback for identifying, for the scale bar of a map, various quantities.

Definition at line 17 of file CallbackScaleBar.h.

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

- Callback/CallbackScaleBar.h
- Callback/CallbackScaleBar.cpp

## 4.19 CallbackSceneUpdateAfterCommand Class Reference

Callback for updating the QGraphicsItems in the scene after a command may have modified Points in Curves.

`#include <CallbackSceneUpdateAfterCommand.h>`

### Public Member Functions

- CallbackSceneUpdateAfterCommand (GraphicsLinesForCurves &graphicsLinesForCurves, GraphicsScene &scene, const Document &document, GeometryWindow *geometryWindow)
  *Single constructor.*
- CallbackSearchReturn callback (const QString &, const Point &point)
  *Callback method.*

#### 4.19.1 Detailed Description

Callback for updating the QGraphicsItems in the scene after a command may have modified Points in Curves.

Definition at line 20 of file CallbackSceneUpdateAfterCommand.h.

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

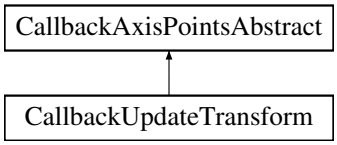
- Callback/CallbackSceneUpdateAfterCommand.h
- Callback/CallbackSceneUpdateAfterCommand.cpp

## 4.20 CallbackUpdateTransform Class Reference

Callback for collecting axis points and then calculating the current transform from those axis points.

`#include <CallbackUpdateTransform.h>`

Inheritance diagram for CallbackUpdateTransform:




### Public Member Functions

- CallbackUpdateTransform (const DocumentModelCoords &modelCoords, DocumentAxesPointsRequired documentAxesPointsRequired)

  *Single constructor.*
- bool transformIsDefined () const

  *True if enough Points were available to create a Transformation.*

### Additional Inherited Members

### 4.20.1 Detailed Description

Callback for collecting axis points and then calculating the current transform from those axis points.

Sanity checking of the axis points was applied earlier when the axis points were added/edited.

Definition at line 19 of file CallbackUpdateTransform.h.

### 4.20.2 Member Function Documentation

#### 4.20.2.1 transformIsDefined()

```
bool CallbackUpdateTransform::transformIsDefined ( ) const
```

True if enough Points were available to create a Transformation.

Except for the node count, all other failure modes are caught externally so user gets immediate feedback as soon as bad axis point data appears

Definition at line 18 of file CallbackUpdateTransform.cpp.

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

- Callback/CallbackUpdateTransform.h
- Callback/CallbackUpdateTransform.cpp

## 4.21 Checker Class Reference

Box shape that is drawn through the three axis points, to temporarily (usually) or permanently (rarely) highlight the local up/down/left/right directions when all axis points have been defined.

`#include <Checker.h>`

### Public Member Functions

- Checker (QGraphicsScene &scene)

  *Single constructor for DlgSettingsAxesChecker, which does not have an explicit transformation. The identity transformation is assumed.*
- void prepareForDisplay (const QPolygonF &polygon, int pointRadius, const DocumentModelAxesChecker &modelAxesChecker, const DocumentModelCoords &modelCoords, DocumentAxesPointsRequired documentAxesPointsRequired)

  *Create the polygon from current information, including pixel coordinates, just prior to display.*
- void prepareForDisplay (const QList< Point > &Points, int pointRadius, const DocumentModelAxesChecker &modelAxesChecker, const DocumentModelCoords &modelCoords, const Transformation &transformation, DocumentAxesPointsRequired documentAxesPointsRequired)

  *Create the polygon from current information, including pixel and graph coordinates, just prior to display.*
- void setVisible (bool visible)

  *Show/hide this axes checker.*
- virtual void updateModelAxesChecker (const DocumentModelAxesChecker &modelAxesChecker)

  *Apply the new DocumentModelAxesChecker, to the points already associated with this object.*

### 4.21.1 Detailed Description

Box shape that is drawn through the three axis points, to temporarily (usually) or permanently (rarely) highlight the local up/down/left/right directions when all axis points have been defined.

The goal of the checker is to make it obvious when a mistake has happened so the screen-to-graph transformation is currently wrong - since the expected up/down/left/right directions will be awry which will distort the checker somehow. Unfortunately, errors in scale are not revealed by the checker.

Definition at line 33 of file Checker.h.

### 4.21.2 Member Function Documentation

#### 4.21.2.1 prepareForDisplay() [1/2]

```
void Checker::prepareForDisplay (
            const QPolygonF & polygon,
            int pointRadius,
            const DocumentModelAxesChecker & modelAxesChecker,
            const DocumentModelCoords & modelCoords,
            DocumentAxesPointsRequired documentAxesPointsRequired )
```

Create the polygon from current information, including pixel coordinates, just prior to display.

This is for DlgSettingsAxesChecker. The identity matrix is used for the transformations between screen and graph coordinates. The point radius is used to exclude the lines from the axes points for clarity

Definition at line 130 of file Checker.cpp.

#### 4.21.2.2 prepareForDisplay() [2/2]

```
void Checker::prepareForDisplay (
            const QList< Point > & Points,
            int pointRadius,
            const DocumentModelAxesChecker & modelAxesChecker,
            const DocumentModelCoords & modelCoords,
            const Transformation & transformation,
            DocumentAxesPointsRequired documentAxesPointsRequired )
```

Create the polygon from current information, including pixel and graph coordinates, just prior to display.

This is for TransformationStateDefined. The point radius is used to exclude the lines from the axes points for clarity

Definition at line 168 of file Checker.cpp.

#### 4.21.2.3 updateModelAxesChecker()

```
void Checker::updateModelAxesChecker (
            const DocumentModelAxesChecker & modelAxesChecker ) [virtual]
```

Apply the new DocumentModelAxesChecker, to the points already associated with this object.

This method starts the timer unless the mode is never or forever

Definition at line 247 of file Checker.cpp.

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

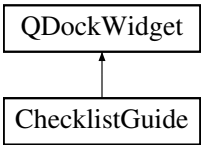
- Checker/Checker.h
- Checker/Checker.cpp

## 4.22 ChecklistGuide Class Reference

Dockable text window containing checklist guide.

`#include <ChecklistGuide.h>`

Inheritance diagram for ChecklistGuide:




### Signals

- void signalChecklistClosed ()

  *Signal that this QDockWidget was just closed.*

### Public Member Functions

- ChecklistGuide (QWidget *parent)

  *Single constructor. Parent is needed or else this widget cannot be redocked after being undocked.*
- bool browserIsEmpty () const

  *When browser is empty, it is pointless to show it.*
- virtual void closeEvent (QCloseEvent *event)

  *Catch close event so corresponding menu item in MainWindow can be updated accordingly.*
- void setTemplateHtml (const QString &html, const QStringList &curveNames)

  *Populate the browser with template html.*
- void update (const CmdMediator &cmdMediator, bool documentIsExported)

  *Update using current CmdMediator/Document state.*

#### 4.22.1 Detailed Description

Dockable text window containing checklist guide.

Definition at line 16 of file ChecklistGuide.h.

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

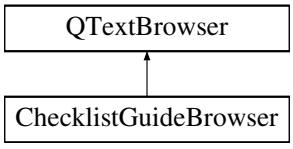
- Checklist/ChecklistGuide.h
- Checklist/ChecklistGuide.cpp

## 4.23 ChecklistGuideBrowser Class Reference

Class that adds rudimentary tree collapse/expand functionality to QTextBrowser.

```
#include <ChecklistGuideBrowser.h>
```

Inheritance diagram for ChecklistGuideBrowser:




### Public Member Functions

- ChecklistGuideBrowser ()

  *Single constructor.*
- virtual void setTemplateHtml (const QString &html, const QStringList &curveNames)

  *Populate the browser with template html. The template html will be converted to real html.*
- void update (const CmdMediator &cmdMediator, bool documentIsExported)

  *Update using current CmdMediator/Document state.*

### 4.23.1 Detailed Description

Class that adds rudimentary tree collapse/expand functionality to QTextBrowser.

Definition at line 15 of file ChecklistGuideBrowser.h.

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

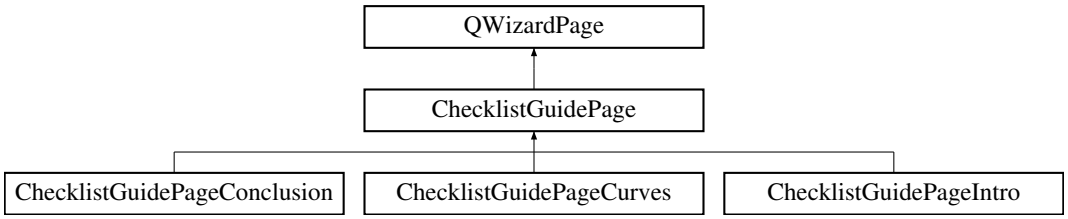
- Checklist/ChecklistGuideBrowser.h
- Checklist/ChecklistGuideBrowser.cpp

## 4.24 ChecklistGuidePage Class Reference

This class customizes QWizardPage for ChecklistGuideWizard.

`#include <ChecklistGuidePage.h>`

Inheritance diagram for ChecklistGuidePage:




### Public Member Functions

- ChecklistGuidePage (const QString &title)
  *Single constructor.*
- void addHtml (const QString &html)
  *Insert html for display.*
- QRadioButton * addLabelAndRadioButton (const QString &label, const QString &whatsThis)
  *Insert radio button and corresponding label.*
- void addLineEdit (ChecklistLineEdit *edit, const QString &whatsThis)
  *Insert line edit.*

### 4.24.1 Detailed Description

This class customizes QWizardPage for ChecklistGuideWizard.

Definition at line 19 of file ChecklistGuidePage.h.

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

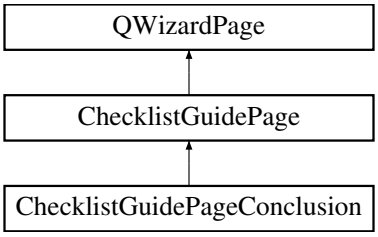
- Checklist/ChecklistGuidePage.h
- Checklist/ChecklistGuidePage.cpp

## 4.25 ChecklistGuidePageConclusion Class Reference

This class uses the validation method of the Conclusion page to perform final processing for ChecklistGuideWizard.

`#include <ChecklistGuidePageConclusion.h>`

Inheritance diagram for ChecklistGuidePageConclusion:




### Public Member Functions

- ChecklistGuidePageConclusion ()
  *Single constructor.*
- virtual bool validatePage ()
  *Perform final processing.*

### 4.25.1 Detailed Description

This class uses the validation method of the Conclusion page to perform final processing for ChecklistGuideWizard.

Definition at line 13 of file ChecklistGuidePageConclusion.h.

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

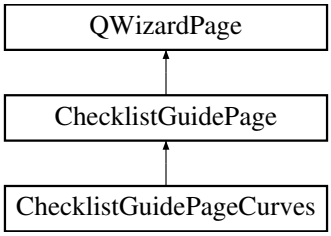
- Checklist/ChecklistGuidePageConclusion.h
- Checklist/ChecklistGuidePageConclusion.cpp

## 4.26 ChecklistGuidePageCurves Class Reference

This class adds validation to the Curves page.

`#include <ChecklistGuidePageCurves.h>`

Inheritance diagram for ChecklistGuidePageCurves:



### Public Slots

- void slotTableChanged ()

  *Update after curve table update.*
- bool withLines () const

  *Drawn with lines, else points.*

### Public Member Functions

- ChecklistGuidePageCurves (const QString &title)

  *Single constructor.*
- QStringList curveNames () const

  *Wizard selection for curve names.*
- virtual bool isComplete () const

  *Validate the contents of this page.*

#### 4.26.1 Detailed Description

This class adds validation to the Curves page.

Definition at line 17 of file ChecklistGuidePageCurves.h.

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

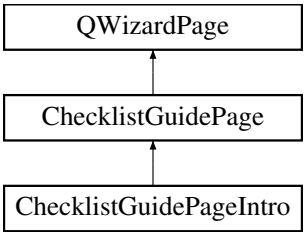
- Checklist/ChecklistGuidePageCurves.h
- Checklist/ChecklistGuidePageCurves.cpp

## 4.27 ChecklistGuidePageIntro Class Reference

This class sets up the introduction page.

`#include <ChecklistGuidePageIntro.h>`

Inheritance diagram for ChecklistGuidePageIntro:




### Public Member Functions

- ChecklistGuidePageIntro ()

  *Single constructor.*

### 4.27.1 Detailed Description

This class sets up the introduction page.

Definition at line 13 of file ChecklistGuidePageIntro.h.

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

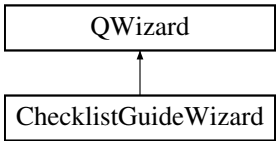
- Checklist/ChecklistGuidePageIntro.h
- Checklist/ChecklistGuidePageIntro.cpp

## 4.28 ChecklistGuideWizard Class Reference

Wizard for setting up the checklist guide.

`#include <ChecklistGuideWizard.h>`

Inheritance diagram for ChecklistGuideWizard:




### Public Member Functions

- ChecklistGuideWizard (MainWindow &mainWindow, unsigned int numberCoordSystem)
  *Single constructor.*
- QStringList curveNames (CoordSystemIndex coordSystemIndex) const
  *Curve names to be placed into Document.*
- void populateCurvesGraphs (CoordSystemIndex coordSystemIndex, CurvesGraphs &curvesGraphs)
  *Create entries in CurvesGraphs for each curve name that user provided.*
- QString templateHtml (CoordSystemIndex coordSystemIndex) const
  *Template html comprising the checklist for display.*

### 4.28.1 Detailed Description

Wizard for setting up the checklist guide.

Definition at line 23 of file ChecklistGuideWizard.h.

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

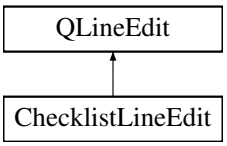
- Checklist/ChecklistGuideWizard.h
- Checklist/ChecklistGuideWizard.cpp

## 4.29 ChecklistLineEdit Class Reference

Adds key event handling to QLineEdit.

`#include <ChecklistLineEdit.h>`

Inheritance diagram for ChecklistLineEdit:




### Signals

- void signalKeyRelease ()

  *Signal that user has just released a key.*

### Public Member Functions

- ChecklistLineEdit ()

  *Single constructor.*
- virtual void keyReleaseEvent (QKeyEvent ∗event)

  *Intercept the key press event.*

### 4.29.1 Detailed Description

Adds key event handling to QLineEdit.

Definition at line 13 of file ChecklistLineEdit.h.

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

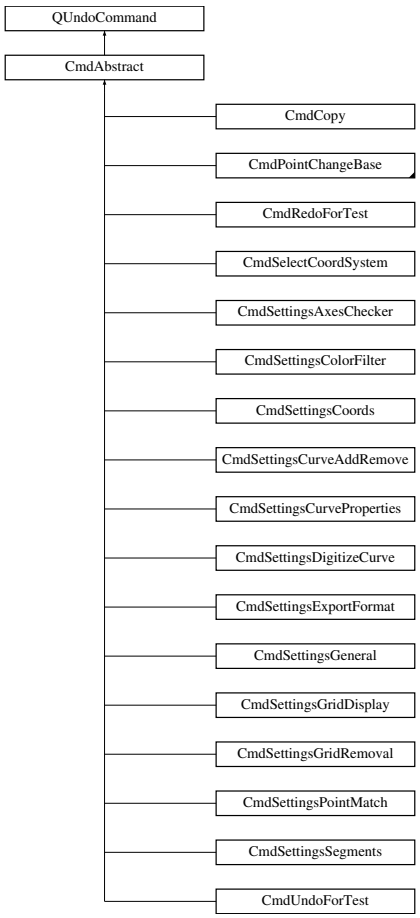
- Checklist/ChecklistLineEdit.h
- Checklist/ChecklistLineEdit.cpp

## 4.30 CmdAbstract Class Reference

Wrapper around QUndoCommand. This simplifies the more complicated feature set of QUndoCommand.

`#include <CmdAbstract.h>`

Inheritance diagram for CmdAbstract:




### Public Member Functions

- CmdAbstract (MainWindow &mainWindow, Document &document, const QString &cmdDescription)

  *Single constructor.*
- virtual void cmdRedo ()=0

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()=0

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const =0

  *Save commands as xml for later uploading.*

### Protected Member Functions

- Document & document ()

  *Return the Document that this command will modify during redo and undo.*
- const Document & document () const

  *Return a const copy of the Document for non redo/undo interaction.*
- MainWindow & mainWindow ()

  *Return the MainWindow so it can be updated by this command as a last step.*
- void resetSelection (const PointIdentifiers &pointIdentifiersToSelect)

  *Since the set of selected points has probably changed, changed that set back to the specified set.*
- void saveOrCheckPostCommandDocumentStateHash (const Document &document)

  *Save, when called the first time, a hash value representing the state of the Document.*
- void saveOrCheckPreCommandDocumentStateHash (const Document &document)

  *Save, when called the first time, a hash value representing the state of the Document.*

## 4.30.1 Detailed Description

Wrapper around QUndoCommand. This simplifies the more complicated feature set of QUndoCommand.

Definition at line 19 of file CmdAbstract.h.

## 4.30.2 Member Function Documentation

### 4.30.2.1 resetSelection()

```
void CmdAbstract::resetSelection (
            const PointIdentifiers & pointIdentifiersToSelect )  [protected]
```

Since the set of selected points has probably changed, changed that set back to the specified set.

This lets the user move selected point(s) repeatedly using arrow keys. Also provides expected behavior when pasting

Definition at line 81 of file CmdAbstract.cpp.

### 4.30.2.2 saveOrCheckPostCommandDocumentStateHash()

```
void CmdAbstract::saveOrCheckPostCommandDocumentStateHash (
            const Document & document )  [protected]
```

Save, when called the first time, a hash value representing the state of the Document.

Then on succeeding calls the hash is recomputed and compared to the original value to check for consistency. This "post" method is called immediately after the redo method of the subclass has done its processing. See also saveOrCheckPreCommandDocumentState

Definition at line 102 of file CmdAbstract.cpp.

#### 4.30.2.3 saveOrCheckPreCommandDocumentStateHash()

```
void CmdAbstract::saveOrCheckPreCommandDocumentStateHash (
            const Document & document )  [protected]
```

Save, when called the first time, a hash value representing the state of the Document.

Then on succeeding calls the hash is recomputed and compared to the original value to check for consistency. This "pre" method is called immediately after the redo method of the subclass has done its processing. See also saveOrCheckPostCommandDocumentState

Definition at line 125 of file CmdAbstract.cpp.

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

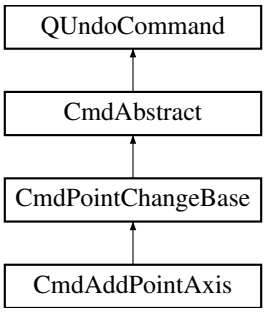
- Cmd/CmdAbstract.h
- Cmd/CmdAbstract.cpp

## 4.31 CmdAddPointAxis Class Reference

Command for adding one axis point.

```
#include <CmdAddPointAxis.h>
```

Inheritance diagram for CmdAddPointAxis:




### Public Member Functions

- CmdAddPointAxis (MainWindow &mainWindow, Document &document, const QPointF &posScreen, const QPointF &posGraph, double ordinal, bool isXOnly)

  *Constructor for normal creation.*
- CmdAddPointAxis (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.31.1 Detailed Description

Command for adding one axis point.

Definition at line 16 of file CmdAddPointAxis.h.

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

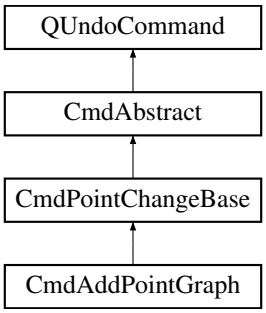
- Cmd/CmdAddPointAxis.h
- Cmd/CmdAddPointAxis.cpp

## 4.32 CmdAddPointGraph Class Reference

Command for adding one graph point.

`#include <CmdAddPointGraph.h>`

Inheritance diagram for CmdAddPointGraph:




### Public Member Functions

- CmdAddPointGraph (MainWindow &mainWindow, Document &document, const QString &curveName, const QPointF &posScreen, double ordinal)

  *Constructor for normal creation.*
- CmdAddPointGraph (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.32.1 Detailed Description

Command for adding one graph point.

Definition at line 17 of file CmdAddPointGraph.h.

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

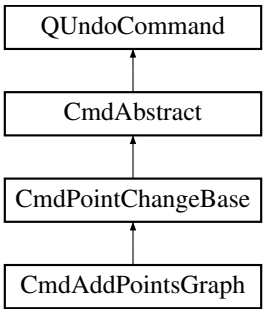
- Cmd/CmdAddPointGraph.h
- Cmd/CmdAddPointGraph.cpp

## 4.33 CmdAddPointsGraph Class Reference

Command for adding one or more graph points. This is for Segment Fill mode.

`#include <CmdAddPointsGraph.h>`

Inheritance diagram for CmdAddPointsGraph:




### Public Member Functions

- CmdAddPointsGraph (MainWindow &mainWindow, Document &document, const QString &curveName, const QList< QPoint > &points, const QList< double > &ordinals)

  *Constructor for normal creation.*
- CmdAddPointsGraph (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.33.1 Detailed Description

Command for adding one or more graph points. This is for Segment Fill mode.

Definition at line 19 of file CmdAddPointsGraph.h.

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

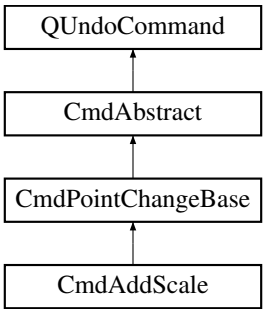
- Cmd/CmdAddPointsGraph.h
- Cmd/CmdAddPointsGraph.cpp

## 4.34 CmdAddScale Class Reference

Command for adding one scale point.

`#include <CmdAddScale.h>`

Inheritance diagram for CmdAddScale:




### Public Member Functions

- CmdAddScale (MainWindow &mainWindow, Document &document, const QPointF &posScreen0, const QPointF &posScreen1, double scaleLength, double ordinal0, double ordinal1)

  *Constructor for normal creation.*
- CmdAddScale (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.34.1 Detailed Description

Command for adding one scale point.

Definition at line 16 of file CmdAddScale.h.

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

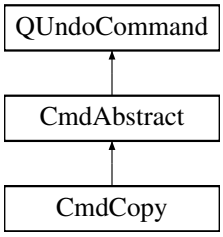
- Cmd/CmdAddScale.h
- Cmd/CmdAddScale.cpp

## 4.35 CmdCopy Class Reference

Command for moving all selected Points by a specified translation.

`#include <CmdCopy.h>`

Inheritance diagram for CmdCopy:




### Public Member Functions

- CmdCopy (MainWindow &mainWindow, Document &document, const QStringList &selectedPointIdentifiers)

  *Constructor for normal creation.*
- CmdCopy (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXml↩ StreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.35.1 Detailed Description

Command for moving all selected Points by a specified translation.

Definition at line 18 of file CmdCopy.h.

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

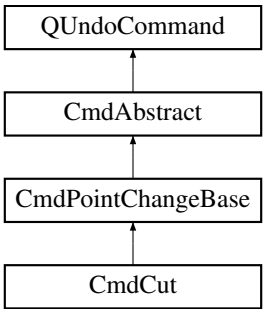
- Cmd/CmdCopy.h
- Cmd/CmdCopy.cpp

## 4.36 CmdCut Class Reference

Command for cutting all selected Points.

`#include <CmdCut.h>`

Inheritance diagram for CmdCut:




### Public Member Functions

- CmdCut (MainWindow &mainWindow, Document &document, const QStringList &selectedPointIdentifiers)

  *Constructor for normal creation.*
- CmdCut (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXml↩ StreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.36.1 Detailed Description

Command for cutting all selected Points.

Definition at line 18 of file CmdCut.h.

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

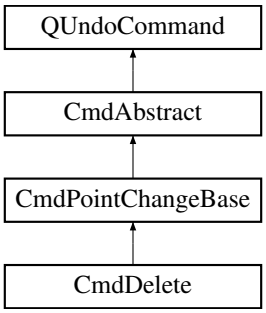
- Cmd/CmdCut.h
- Cmd/CmdCut.cpp

## 4.37 CmdDelete Class Reference

Command for deleting all selected Points.

`#include <CmdDelete.h>`

Inheritance diagram for CmdDelete:




### Public Member Functions

- CmdDelete (MainWindow &mainWindow, Document &document, const QStringList &selectedPoint↩ Identifiers)

  *Constructor for normal creation.*
- CmdDelete (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXml↩ StreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.37.1 Detailed Description

Command for deleting all selected Points.

Definition at line 18 of file CmdDelete.h.

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

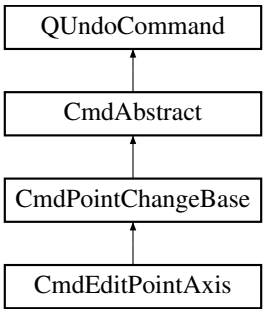
- Cmd/CmdDelete.h
- Cmd/CmdDelete.cpp

## 4.38 CmdEditPointAxis Class Reference

Command for editing the graph coordinates one axis point.

`#include <CmdEditPointAxis.h>`

Inheritance diagram for CmdEditPointAxis:




### Public Member Functions

- CmdEditPointAxis (MainWindow &mainWindow, Document &document, const QString &pointIdentifier, const QPointF &posGraphBefore, const QPointF &posGraphAfter, bool isXOnly)

  *Constructor for normal creation.*
- CmdEditPointAxis (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.38.1 Detailed Description

Command for editing the graph coordinates one axis point.

The screen coordinates are handled by another command

Definition at line 18 of file CmdEditPointAxis.h.

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

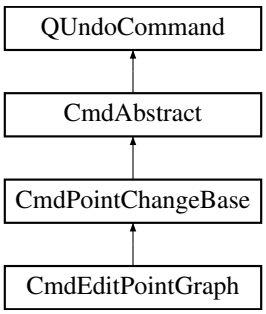
- Cmd/CmdEditPointAxis.h
- Cmd/CmdEditPointAxis.cpp

## 4.39 CmdEditPointGraph Class Reference

Command for editing the graph coordinates of one or more graph points.

`#include <CmdEditPointGraph.h>`

Inheritance diagram for CmdEditPointGraph:




### Public Member Functions

- CmdEditPointGraph (MainWindow &mainWindow, Document &document, const QStringList &point↩ Identifiers, bool isX, bool isY, double x, double y)

  *Constructor for normal creation.*
- CmdEditPointGraph (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.39.1 Detailed Description

Command for editing the graph coordinates of one or more graph points.

The screen coordinates are handled by another command

Definition at line 18 of file CmdEditPointGraph.h.

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

- Cmd/CmdEditPointGraph.h
- Cmd/CmdEditPointGraph.cpp

## 4.40 CmdFactory Class Reference

Factory for CmdAbstractBase objects.

`#include <CmdFactory.h>`

### Public Member Functions

- CmdFactory ()
  *Single constructor.*
- CmdAbstract * createCmd (MainWindow &mainWindow, Document &document, QXmlStreamReader &reader)
  *Factory method. Input is the xml node from an error report file.*

#### 4.40.1 Detailed Description

Factory for CmdAbstractBase objects.

Definition at line 16 of file CmdFactory.h.

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

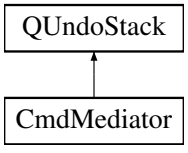
- Cmd/CmdFactory.h
- Cmd/CmdFactory.cpp

## 4.41 CmdMediator Class Reference

Command queue stack.

`#include <CmdMediator.h>`

Inheritance diagram for CmdMediator:



## Public Member Functions

- CmdMediator (MainWindow &mainWindow, const QImage &image)

  *Constructor for imported images and dragged images. Only one coordinate system is created but others can be added later.*
- CmdMediator (MainWindow &mainWindow, const QString &fileName)

  *Constructor for opened Documents and error report files. The specified xml file is opened and read.*
- ~CmdMediator ()

  *Destructor.*
- const CoordSystem & coordSystem () const

  *Provide the current CoordSystem to commands with read-only access, primarily for undo/redo processing.*
- const Curve & curveAxes () const

  *See Document::curveAxes.*
- QStringList curvesGraphsNames () const

  *See CurvesGraphs::curvesGraphsNames.*
- int curvesGraphsNumPoints (const QString &curveName) const

  *See CurvesGraphs::curvesGraphsNumPoints.*
- Document & document ()

  *Provide the Document to commands, primarily for undo/redo processing.*
- const Document & document () const

  *Provide the Document to commands with read-only access, primarily for undo/redo processing.*
- bool isModified () const

  *Dirty flag.*
- void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, Callback↩ SearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for the single axes curve.*
- void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, Callback↩ SearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurvePoints, for the single axes curve.*
- void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, Callback↩ SearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*
- QPixmap pixmap () const

  *See Document::pixmap.*
- QString reasonForUnsuccessfulRead () const

  *See Document::reasonForUnsuccessfulRead.*
- void saveXml (QXmlStreamWriter &writer) const

  *Serialize to xml.*
- QString selectedCurveName () const

  *Currently selected curve name. This is used to set the selected curve combobox in MainWindow.*
- void setDocumentAxesPointsRequired (DocumentAxesPointsRequired documentAxesPointsRequired)

  *Set the number of axes points required.*
- void setSelectedCurveName (const QString &selectedCurveName)

  *Save curve name that is selected for the current coordinate system, for the next time the coordinate system reappears.*
- bool successfulRead () const

  *Wrapper for Document::successfulRead.*

### 4.41.1 Detailed Description

Command queue stack.

This class lies between the Document and the rest of the application. This approach is attractive because the command stack and Document are born together, work together, and deleted together. Also, wrapping this class around Document helps to encapsulate Document that much more.

Definition at line 23 of file CmdMediator.h.

### 4.41.2 Member Function Documentation

#### 4.41.2.1 isModified()

```
bool CmdMediator::isModified ( ) const
```

Dirty flag.

Document is dirty if there are any unsaved changes. The dirty flag is pushed (rather than pulled from this method) through the QUndoStack::cleanChanged signal

Definition at line 82 of file CmdMediator.cpp.

#### 4.41.2.2 setDocumentAxesPointsRequired()

```
void CmdMediator::setDocumentAxesPointsRequired (
            DocumentAxesPointsRequired documentAxesPointsRequired )
```

Set the number of axes points required.

This is called during the Document creation process, after imported images have been previewed or loaded files have had at least some xml parsing

Definition at line 132 of file CmdMediator.cpp.

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

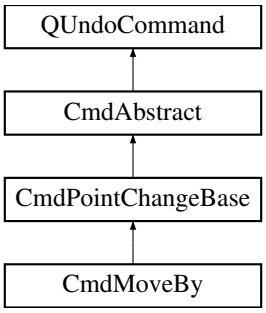
- Cmd/CmdMediator.h
- Cmd/CmdMediator.cpp

## 4.42 CmdMoveBy Class Reference

Command for moving all selected Points by a specified translation.

`#include <CmdMoveBy.h>`

Inheritance diagram for CmdMoveBy:




### Public Member Functions

- CmdMoveBy (MainWindow &mainWindow, Document &document, const QPointF &deltaScreen, const QString &moveText, const QStringList &selectedPointIdentifiers)

  *Constructor for normal creation.*
- CmdMoveBy (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.42.1 Detailed Description

Command for moving all selected Points by a specified translation.

Definition at line 18 of file CmdMoveBy.h.

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

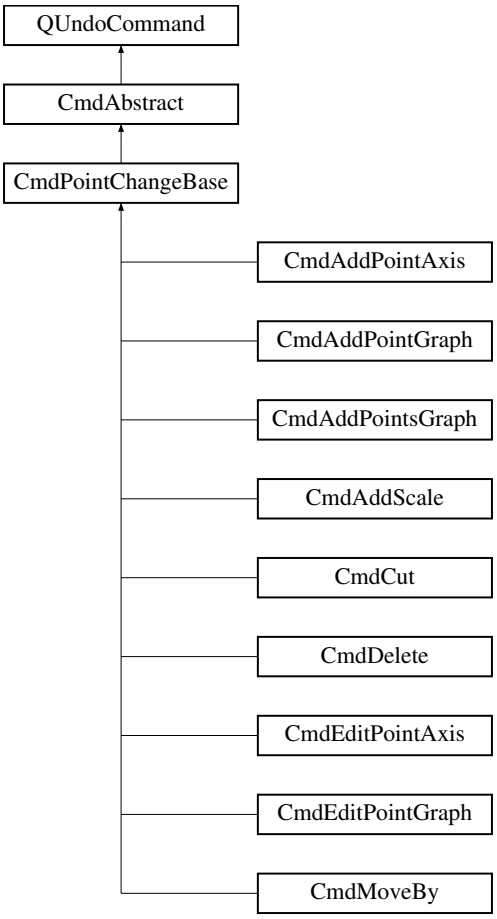
- Cmd/CmdMoveBy.h
- Cmd/CmdMoveBy.cpp

## 4.43 CmdPointChangeBase Class Reference

Base class for CmdBase leaf subclasses that involve point additions, deletions and/or modifications.

`#include <CmdPointChangeBase.h>`

Inheritance diagram for CmdPointChangeBase:




### Public Member Functions

- CmdPointChangeBase (MainWindow &mainWindow, Document &document, const QString &cmd↩ Description)

  *Single constructor.*

### Protected Member Functions

- void restoreDocumentState (Document &document) const

  *Restore the document previously saved by saveDocumentState.*
- void saveDocumentState (const Document &document)

  *Save the document state for restoration by restoreDocumentState.*

### 4.43.1 Detailed Description

Base class for CmdBase leaf subclasses that involve point additions, deletions and/or modifications.

This class uses a strategy of taking a snapshot of all points before the redo, and then applying that snapshot to the Document to (later) perform the undo. Before this strategy, the strategy was to just do the opposite steps of the redo, but that strategy was too fragile since it implicity assumed no point changes occurred after the redo of this command and before the redo of the next command. However, point updates like "ordinal maintenance" do occur during that time period

Definition at line 22 of file CmdPointChangeBase.h.

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

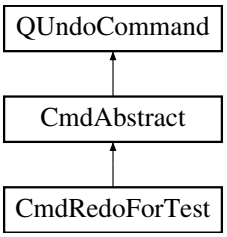
- Cmd/CmdPointChangeBase.h
- Cmd/CmdPointChangeBase.cpp

## 4.44 CmdRedoForTest Class Reference

Command for performing Redo during testing.

`#include <CmdRedoForTest.h>`

Inheritance diagram for CmdRedoForTest:




### Public Member Functions

- CmdRedoForTest (MainWindow &mainWindow, Document &document)

  *Constructor for normal creation.*
- CmdRedoForTest (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.44.1 Detailed Description

Command for performing Redo during testing.

This command is never created automatically, since when the user triggers an Redo that just results in the command stack getting backed up by one command. This command is manually created by editing an xml test file

Definition at line 20 of file CmdRedoForTest.h.

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

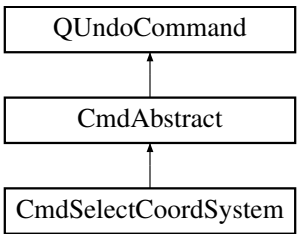
- Cmd/CmdRedoForTest.h
- Cmd/CmdRedoForTest.cpp

## 4.45 CmdSelectCoordSystem Class Reference

Command for changing the currently selected CoordSystem.

`#include <CmdSelectCoordSystem.h>`

Inheritance diagram for CmdSelectCoordSystem:




### Public Member Functions

- CmdSelectCoordSystem (MainWindow &mainWindow, Document &document, CoordSystemIndex coordSystem)

  *Constructor for normal creation.*
- CmdSelectCoordSystem (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.45.1 Detailed Description

Command for changing the currently selected CoordSystem.

Definition at line 16 of file CmdSelectCoordSystem.h.

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

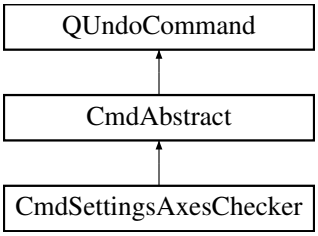
- Cmd/CmdSelectCoordSystem.h
- Cmd/CmdSelectCoordSystem.cpp

## 4.46 CmdSettingsAxesChecker Class Reference

Command for DlgSettingsAxesChecker.

```
#include <CmdSettingsAxesChecker.h>
```

Inheritance diagram for CmdSettingsAxesChecker:




### Public Member Functions

- CmdSettingsAxesChecker (MainWindow &mainWindow, Document &document, const DocumentModelAxesChecker &modelAxesCheckerBefore, const DocumentModelAxesChecker &modelAxesCheckerAfter)

  *Constructor for normal creation.*
- CmdSettingsAxesChecker (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.46.1 Detailed Description

Command for DlgSettingsAxesChecker.

Definition at line 16 of file CmdSettingsAxesChecker.h.

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

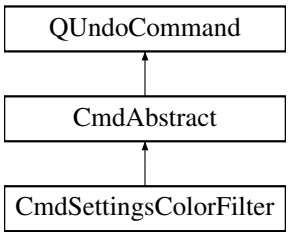
- Cmd/CmdSettingsAxesChecker.h
- Cmd/CmdSettingsAxesChecker.cpp

## 4.47 CmdSettingsColorFilter Class Reference

Command for DlgSettingsColorFilter.

`#include <CmdSettingsColorFilter.h>`

Inheritance diagram for CmdSettingsColorFilter:




### Public Member Functions

- CmdSettingsColorFilter (MainWindow &mainWindow, Document &document, const DocumentModelColorFilter &modelColorFilterBefore, const DocumentModelColorFilter &modelColorFilterAfter)

  *Constructor for normal creation.*
- CmdSettingsColorFilter (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.47.1 Detailed Description

Command for DlgSettingsColorFilter.

Definition at line 16 of file CmdSettingsColorFilter.h.

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

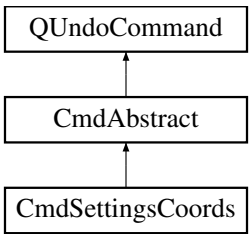
- Cmd/CmdSettingsColorFilter.h
- Cmd/CmdSettingsColorFilter.cpp

## 4.48 CmdSettingsCoords Class Reference

Command for DlgSettingsCoords.

```
#include <CmdSettingsCoords.h>
```

Inheritance diagram for CmdSettingsCoords:




### Public Member Functions

- CmdSettingsCoords (MainWindow &mainWindow, Document &document, const DocumentModelCoords &modelCoordsBefore, const DocumentModelCoords &modelCoordsAfter)

  *Constructor for normal creation.*
- CmdSettingsCoords (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.48.1 Detailed Description

Command for DlgSettingsCoords.

Definition at line 16 of file CmdSettingsCoords.h.

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

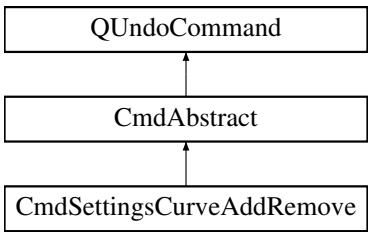
- Cmd/CmdSettingsCoords.h
- Cmd/CmdSettingsCoords.cpp

## 4.49 CmdSettingsCurveAddRemove Class Reference

Command for DlgSettingsCurveAddRemove.

`#include <CmdSettingsCurveAddRemove.h>`

Inheritance diagram for CmdSettingsCurveAddRemove:




### Public Member Functions

- CmdSettingsCurveAddRemove (MainWindow &mainWindow, Document &document, const CurveNameList &modelCurves)

  *Constructor for normal creation.*
- CmdSettingsCurveAddRemove (MainWindow &mainWindow, Document &document, const QString &cmd↩Description, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.49.1 Detailed Description

Command for DlgSettingsCurveAddRemove.

Definition at line 17 of file CmdSettingsCurveAddRemove.h.

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

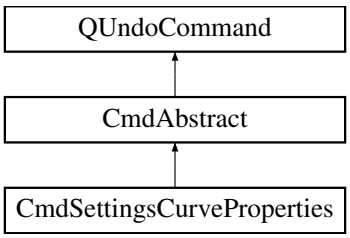
- Cmd/CmdSettingsCurveAddRemove.h
- Cmd/CmdSettingsCurveAddRemove.cpp

## 4.50 CmdSettingsCurveProperties Class Reference

Command for DlgSettingsCurveProperties.

`#include <CmdSettingsCurveProperties.h>`

Inheritance diagram for CmdSettingsCurveProperties:




### Public Member Functions

- CmdSettingsCurveProperties (MainWindow &mainWindow, Document &document, const CurveStyles &modelCurveStylesBefore, const CurveStyles &modelCurveStylesAfter)

  *Constructor for normal creation.*
- CmdSettingsCurveProperties (MainWindow &mainWindow, Document &document, const QString &cmd↩ Description, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.50.1 Detailed Description

Command for DlgSettingsCurveProperties.

Definition at line 19 of file CmdSettingsCurveProperties.h.

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

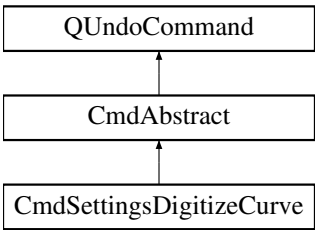
- Cmd/CmdSettingsCurveProperties.h
- Cmd/CmdSettingsCurveProperties.cpp

## 4.51 CmdSettingsDigitizeCurve Class Reference

Command for DlgSettingsDigitizeCurve.

`#include <CmdSettingsDigitizeCurve.h>`

Inheritance diagram for CmdSettingsDigitizeCurve:




### Public Member Functions

- CmdSettingsDigitizeCurve (MainWindow &mainWindow, Document &document, const DocumentModelDigitizeCurve &modelDigitizeCurveBefore, const DocumentModelDigitizeCurve &modelDigitizeCurveAfter)

  *Constructor for normal creation.*
- CmdSettingsDigitizeCurve (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.51.1 Detailed Description

Command for DlgSettingsDigitizeCurve.

Definition at line 16 of file CmdSettingsDigitizeCurve.h.

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

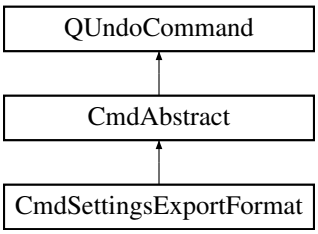
- Cmd/CmdSettingsDigitizeCurve.h
- Cmd/CmdSettingsDigitizeCurve.cpp

## 4.52 CmdSettingsExportFormat Class Reference

Command for DlgSettingsExportFormat.

```
#include <CmdSettingsExportFormat.h>
```

Inheritance diagram for CmdSettingsExportFormat:




### Public Member Functions

- CmdSettingsExportFormat (MainWindow &mainWindow, Document &document, const DocumentModelExportFormat &modelExportBefore, const DocumentModelExportFormat &modelExportAfter)

  *Constructor for normal creation.*
- CmdSettingsExportFormat (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.52.1 Detailed Description

Command for DlgSettingsExportFormat.

Definition at line 16 of file CmdSettingsExportFormat.h.

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

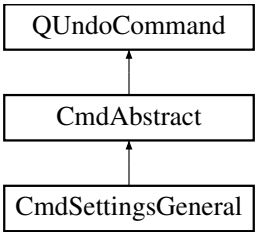
- Cmd/CmdSettingsExportFormat.h
- Cmd/CmdSettingsExportFormat.cpp

## 4.53 CmdSettingsGeneral Class Reference

Command for DlgSettingsGeneral.

`#include <CmdSettingsGeneral.h>`

Inheritance diagram for CmdSettingsGeneral:




### Public Member Functions

- CmdSettingsGeneral (MainWindow &mainWindow, Document &document, const DocumentModelGeneral &modelGeneralBefore, const DocumentModelGeneral &modelGeneralAfter)

  *Constructor for normal creation.*
- CmdSettingsGeneral (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.53.1 Detailed Description

Command for DlgSettingsGeneral.

Definition at line 16 of file CmdSettingsGeneral.h.

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

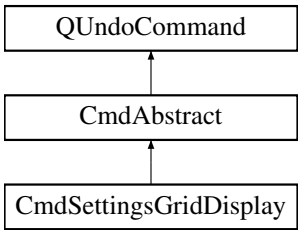
- Cmd/CmdSettingsGeneral.h
- Cmd/CmdSettingsGeneral.cpp

## 4.54 CmdSettingsGridDisplay Class Reference

Command for DlgSettingsGridDisplay.

```
#include <CmdSettingsGridDisplay.h>
```

Inheritance diagram for CmdSettingsGridDisplay:




### Public Member Functions

- CmdSettingsGridDisplay (MainWindow &mainWindow, Document &document, const DocumentModelGrid↩ Display &modelGridDisplayBefore, const DocumentModelGridDisplay &modelGridDisplayAfter)

  *Constructor for normal creation.*
- CmdSettingsGridDisplay (MainWindow &mainWindow, Document &document, const QString &cmd↩ Description, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.54.1 Detailed Description

Command for DlgSettingsGridDisplay.

Definition at line 16 of file CmdSettingsGridDisplay.h.

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

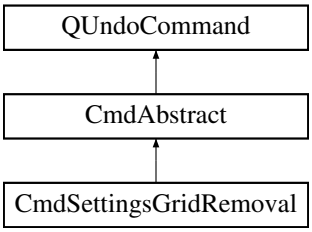
- Cmd/CmdSettingsGridDisplay.h
- Cmd/CmdSettingsGridDisplay.cpp

## 4.55 CmdSettingsGridRemoval Class Reference

Command for DlgSettingsGridRemoval.

```
#include <CmdSettingsGridRemoval.h>
```

Inheritance diagram for CmdSettingsGridRemoval:




### Public Member Functions

- CmdSettingsGridRemoval (MainWindow &mainWindow, Document &document, const DocumentModelGridRemoval &modelGridRemovalBefore, const DocumentModelGridRemoval &modelGridRemovalAfter)

  *Constructor for normal creation.*
- CmdSettingsGridRemoval (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.55.1 Detailed Description

Command for DlgSettingsGridRemoval.

Definition at line 16 of file CmdSettingsGridRemoval.h.

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

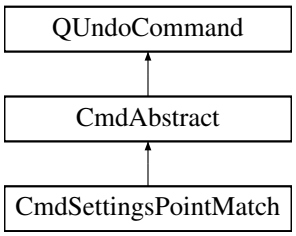
- Cmd/CmdSettingsGridRemoval.h
- Cmd/CmdSettingsGridRemoval.cpp

## 4.56 CmdSettingsPointMatch Class Reference

Command for DlgSettingsPointMatch.

`#include <CmdSettingsPointMatch.h>`

Inheritance diagram for CmdSettingsPointMatch:




### Public Member Functions

- CmdSettingsPointMatch (MainWindow &mainWindow, Document &document, const DocumentModelPoint↩ Match &modelPointMatchBefore, const DocumentModelPointMatch &modelPointMatchAfter)

  *Constructor for normal creation.*
- CmdSettingsPointMatch (MainWindow &mainWindow, Document &document, const QString &cmd↩ Description, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

#### 4.56.1 Detailed Description

Command for DlgSettingsPointMatch.

Definition at line 16 of file CmdSettingsPointMatch.h.

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

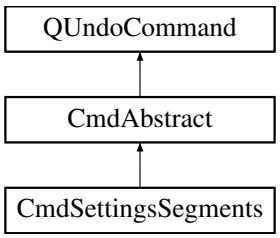
- Cmd/CmdSettingsPointMatch.h
- Cmd/CmdSettingsPointMatch.cpp

## 4.57 CmdSettingsSegments Class Reference

Command for DlgSettingsSegments.

```
#include <CmdSettingsSegments.h>
```

Inheritance diagram for CmdSettingsSegments:




### Public Member Functions

- CmdSettingsSegments (MainWindow &mainWindow, Document &document, const DocumentModelSegments &modelSegmentsBefore, const DocumentModelSegments &modelSegmentsAfter)

  *Constructor for normal creation.*
- CmdSettingsSegments (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.57.1 Detailed Description

Command for DlgSettingsSegments.

Definition at line 16 of file CmdSettingsSegments.h.

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

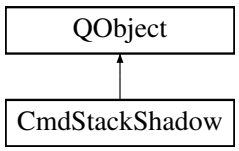
- Cmd/CmdSettingsSegments.h
- Cmd/CmdSettingsSegments.cpp

## 4.58 CmdStackShadow Class Reference

Command stack that shadows the CmdMediator command stack at startup when reading commands from an error report file.

`#include <CmdStackShadow.h>`

Inheritance diagram for CmdStackShadow:




### Public Slots

- void slotRedo ()

  *Move next command from list to CmdMediator. Noop if there are no more commands.*
- void slotUndo ()

  *Throw away every command since trying to reconcile two different command stacks after an undo is too dangerous.*

### Signals

- void signalRedo ()

  *Signal used to emulate a shift-control-z redo command from user during testing.*
- void signalUndo ()

  *Signal used to emulate a shift-z undo command from user during testing.*

### Public Member Functions

- CmdStackShadow ()

  *Single constructor.*
- bool canRedo () const

  *Return true if there is a command available.*
- void loadCommands (MainWindow &mainWindow, Document &document, QXmlStreamReader &reader)

  *Load commands from serialized xml.*

### 4.58.1 Detailed Description

Command stack that shadows the CmdMediator command stack at startup when reading commands from an error report file.

The commands are loaded into this container rather than CmdMediator, since CmdMediator would try to execute all the commands immediately. For the best debugging, we want to be able to execute each command one by one. This container nicely stores commands until we want to copy them to CmdMediator so they can be executed.

This class is not subclassed from QUndoStack since that class is designed to prevent access to individual commands, to preserve their integrity

This class is not named CmdMediatorShadow since does not maintain a Document like CmdMediator, although in some ways that name might be a useful alias

Definition at line 30 of file CmdStackShadow.h.

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

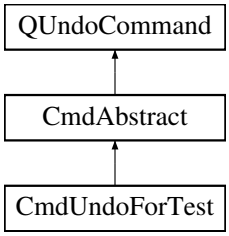
- Cmd/CmdStackShadow.h
- Cmd/CmdStackShadow.cpp

## 4.59 CmdUndoForTest Class Reference

Command for performing Undo during testing.

`#include <CmdUndoForTest.h>`

Inheritance diagram for CmdUndoForTest:




### Public Member Functions

- CmdUndoForTest (MainWindow &mainWindow, Document &document)

  *Constructor for normal creation.*
- CmdUndoForTest (MainWindow &mainWindow, Document &document, const QString &cmdDescription, QXmlStreamReader &reader)

  *Constructor for parsing error report file xml.*
- virtual void cmdRedo ()

  *Redo method that is called when QUndoStack is moved one command forward.*
- virtual void cmdUndo ()

  *Undo method that is called when QUndoStack is moved one command backward.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save commands as xml for later uploading.*

### Additional Inherited Members

### 4.59.1 Detailed Description

Command for performing Undo during testing.

This command is never created automatically, since when the user triggers an Undo that just results in the command stack getting backed up by one command. This command is manually created by editing an xml test file

Definition at line 20 of file CmdUndoForTest.h.

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

- Cmd/CmdUndoForTest.h
- Cmd/CmdUndoForTest.cpp

## 4.60 ColorFilter Class Reference

Class for filtering image to remove unimportant information.

```
#include <ColorFilter.h>
```

### Public Member Functions

- ColorFilter ()

  *Single constructor.*
- bool colorCompare (QRgb rgb1, QRgb rgb2) const

  *See if the two color values are close enough to be considered to be the same.*
- void filterImage (const QImage &imageOriginal, QImage &imageFiltered, ColorFilterMode colorFilterMode, double low, double high, QRgb rgbBackground)

  *Filter the original image according to the specified filtering parameters.*
- QRgb marginColor (const QImage *image) const

  *Identify the margin color of the image, which is defined as the most common color in the four margins.*
- bool pixelFilteredIsOn (const QImage &image, int x, int y) const

  *Return true if specified filtered pixel is on.*
- double pixelToZeroToOneOrMinusOne (ColorFilterMode colorFilterMode, const QColor &pixel, QRgb rgb↩ Background) const

  *Return pixel converted according to the current filter parameter, normalized to zero to one.*
- bool pixelUnfilteredIsOn (ColorFilterMode colorFilterMode, const QColor &pixel, QRgb rgbBackground, double low0To1, double high0To1) const

  *Return true if specified unfiltered pixel is on.*
- int zeroToOneToValue (ColorFilterMode colorFilterMode, double s) const

  *Inverse of pixelToZeroToOneOrMinusOne.*

### 4.60.1 Detailed Description

Class for filtering image to remove unimportant information.

Definition at line 20 of file ColorFilter.h.

### 4.60.2 Member Function Documentation

#### 4.60.2.1 marginColor()

```
QRgb ColorFilter::marginColor (
            const QImage * image ) const
```

Identify the margin color of the image, which is defined as the most common color in the four margins.

For speed, only pixels in the four borders are examined, with the results from those borders safely representing the most common color of the entire margin areas.

Definition at line 73 of file ColorFilter.cpp.

#### 4.60.2.2 pixelToZeroToOneOrMinusOne()

```
double ColorFilter::pixelToZeroToOneOrMinusOne (
            ColorFilterMode colorFilterMode,
            const QColor & pixel,
            QRgb rgbBackground ) const
```

Return pixel converted according to the current filter parameter, normalized to zero to one.

Special case is -1 for a pixel that cannot be converted, like finding hue value for gray scale pixel

Definition at line 171 of file ColorFilter.cpp.

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

- Color/ColorFilter.h
- Color/ColorFilter.cpp

## 4.61 ColorFilterEntry Struct Reference

Helper class so ColorFilter class can compute the background color.

```
#include <ColorFilterEntry.h>
```

### Public Attributes

- QColor color

  *Unique color entry.*
- unsigned int count

  *Number of times this color has appeared.*

### 4.61.1 Detailed Description

Helper class so ColorFilter class can compute the background color.

Definition at line 13 of file ColorFilterEntry.h.

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

- Color/ColorFilterEntry.h

## 4.62 ColorFilterHistogram Class Reference

Class that generates a histogram according to the current filter.

`#include <ColorFilterHistogram.h>`

### Public Member Functions

- ColorFilterHistogram ()

  *Single constructor.*
- int binFromPixel (const ColorFilter &filter, ColorFilterMode colorFilterMode, const QColor &pixel, const QRgb &rgbBackground) const

  *Compute histogram bin number from pixel according to filter.*
- void generate (const ColorFilter &filter, double histogramBins [ ], ColorFilterMode colorFilterMode, const QImage &image, int &maxBinCount) const

  *Generate the histogram.*
- int valueFromBin (const ColorFilter &filter, ColorFilterMode colorFilterMode, int bin)

  *Inverse of binFromPixel.*

### Static Public Member Functions

- static int HISTOGRAM_BINS ()

  *Number of histogram bins.*

### 4.62.1 Detailed Description

Class that generates a histogram according to the current filter.

Definition at line 17 of file ColorFilterHistogram.h.

### 4.62.2 Member Function Documentation

#### 4.62.2.1 generate()

```
void ColorFilterHistogram::generate (
            const ColorFilter & filter,
            double histogramBins[],
            ColorFilterMode colorFilterMode,
            const QImage & image,
            int & maxBinCount ) const
```

Generate the histogram.

The resolution is coarse since

1. finer resolution is not needed
2. this smooths out the curve

Definition at line 40 of file ColorFilterHistogram.cpp.

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

- Color/ColorFilterHistogram.h
- Color/ColorFilterHistogram.cpp

## 4.63 ColorFilterSettings Class Reference

Color filter parameters for one curve. For a class, this is handled the same as LineStyle and PointStyle.

```
#include <ColorFilterSettings.h>
```

### Public Member Functions

- ColorFilterSettings ()

  *Default constructor only for use when this class is being stored by a container that requires the default constructor.*
- ColorFilterSettings (ColorFilterMode colorFilterMode, int intensityLow, int intensityHigh, int foregroundLow, int foregroundHigh, int hueLow, int hueHigh, int saturationLow, int saturationHigh, int valueLow, int valueHigh)

  *Normal constructor. The style type and radius are determined by the currently selected Curve.*
- ColorFilterSettings (const ColorFilterSettings &other)

  *Copy constructor.*
- ColorFilterSettings (QXmlStreamReader &reader)

  *Constructor when loading from serialized xml.*
- ColorFilterSettings & operator= (const ColorFilterSettings &other)

  *Assignment operator.*
- ColorFilterMode colorFilterMode () const

  *Get method for filter mode.*
- int foregroundHigh () const

  *Get method for foreground higher bound.*
- int foregroundLow () const

  *Get method for foreground lower bound.*
- double high () const

*High value of foreground, hue, intensity, saturation or value according to current filter mode, normalized to 0 to 1.*

- int hueHigh () const

  *Get method for hue higher bound.*
- int hueLow () const

  *Get method for hue lower bound.*
- int intensityHigh () const

  *Get method for intensity higher bound.*
- int intensityLow () const

  *Get method for intensity lower bound.*
- void loadXml (QXmlStreamReader &reader)

  *Load curve filter to stream.*
- double low () const

  *Low value of foreground, hue, intensity, saturation or value according to current filter mode, normalized to 0 to 1.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- int saturationHigh () const

  *Get method for saturation higher bound.*
- int saturationLow () const

  *Get method for saturation lower bound.*
- void saveXml (QXmlStreamWriter &writer, const QString &curveName) const

  *Save curve filter to stream.*
- void setColorFilterMode (ColorFilterMode colorFilterMode)

  *Set method for filter mode.*
- void setForegroundHigh (int foregroundHigh)

  *Set method for foreground higher bound.*
- void setForegroundLow (int foregroundLow)

  *Set method for foreground lower bound.*
- void setHigh (double s0To1)

  *Set the high value for the current filter mode.*
- void setHueHigh (int hueHigh)

  *Set method for hue higher bound.*
- void setHueLow (int hueLow)

  *Set method for hue lower bound.*
- void setIntensityHigh (int intensityHigh)

  *Set method for intensity higher bound.*
- void setIntensityLow (int intensityLow)

  *Set method for intensity lower bound.*
- void setLow (double s0To1)

  *Set the low value for the current filter mode.*
- void setSaturationHigh (int saturationHigh)

  *Set method for saturation high.*
- void setSaturationLow (int saturationLow)

  *Set method for saturation low.*
- void setValueHigh (int valueHigh)

  *Set method for value high.*
- void setValueLow (int valueLow)

  *Set method for value low.*
- int valueHigh () const

  *Get method for value high.*
- int valueLow () const

  *Get method for value low.*

### Static Public Member Functions

- static ColorFilterSettings defaultFilter ()
  *Initial default for any Curve.*

### 4.63.1 Detailed Description

Color filter parameters for one curve. For a class, this is handled the same as LineStyle and PointStyle.

Definition at line 19 of file ColorFilterSettings.h.

### 4.63.2 Member Function Documentation

#### 4.63.2.1 high()

```
double ColorFilterSettings::high ( ) const
```

High value of foreground, hue, intensity, saturation or value according to current filter mode, normalized to 0 to 1.

Definition at line 136 of file ColorFilterSettings.cpp.

#### 4.63.2.2 low()

```
double ColorFilterSettings::low ( ) const
```

Low value of foreground, hue, intensity, saturation or value according to current filter mode, normalized to 0 to 1.

Definition at line 218 of file ColorFilterSettings.cpp.

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

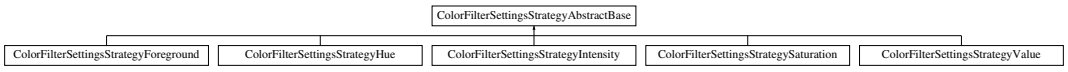
- Color/ColorFilterSettings.h
- Color/ColorFilterSettings.cpp

## 4.64 ColorFilterSettingsStrategyAbstractBase Class Reference

Base class for strategy pattern whose subclasses process the different color filter settings modes (one strategy per mode).

```
#include <ColorFilterSettingsStrategyAbstractBase.h>
```

Inheritance diagram for ColorFilterSettingsStrategyAbstractBase:



### Public Member Functions

- ColorFilterSettingsStrategyAbstractBase ()

  *Single constructor.*
- virtual double high (const ColorFilterSettings &colorFilterSettings) const =0

  *Return the high value normalized to 0 to 1.*
- virtual double low (const ColorFilterSettings &colorFilterSettings) const =0

  *Return the low value normalized to 0 to 1.*
- virtual void printStream (const ColorFilterSettings &colorFilterSettings, QString indentation, QTextStream &str) const =0

  *Print the low and high values.*
- virtual void setHigh (ColorFilterSettings &colorFilterSettings, double s0To1)=0

  *Set the high value given the normalized value.*
- virtual void setLow (ColorFilterSettings &colorFilterSettings, double s0To1)=0

  *Set the low value given the normalized value.*

#### 4.64.1 Detailed Description

Base class for strategy pattern whose subclasses process the different color filter settings modes (one strategy per mode).

The strategy pattern nicely removes cyclomatic complexity from ColorFilterSettings

Definition at line 17 of file ColorFilterSettingsStrategyAbstractBase.h.

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

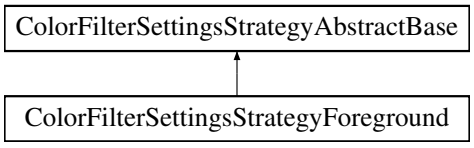
- Color/ColorFilterSettingsStrategyAbstractBase.h
- Color/ColorFilterSettingsStrategyAbstractBase.cpp

## 4.65 ColorFilterSettingsStrategyForeground Class Reference

Leaf class for foreground strategy for ColorFilterSettings.

```
#include <ColorFilterSettingsStrategyForeground.h>
```

Inheritance diagram for ColorFilterSettingsStrategyForeground:



### Public Member Functions

- ColorFilterSettingsStrategyForeground ()
  *Single constructor.*
- virtual double high (const ColorFilterSettings &colorFilterSettings) const
  *Return the high value normalized to 0 to 1.*
- virtual double low (const ColorFilterSettings &colorFilterSettings) const
  *Return the low value normalized to 0 to 1.*
- virtual void printStream (const ColorFilterSettings &colorFilterSettings, QString indentation, QTextStream &str) const
  *Print the low and high values.*
- virtual void setHigh (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the high value given the normalized value.*
- virtual void setLow (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the low value given the normalized value.*

#### 4.65.1 Detailed Description

Leaf class for foreground strategy for ColorFilterSettings.

Definition at line 13 of file ColorFilterSettingsStrategyForeground.h.

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

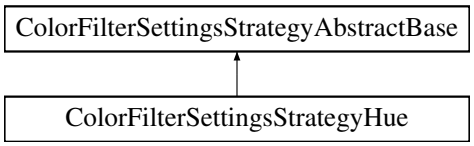
- Color/ColorFilterSettingsStrategyForeground.h
- Color/ColorFilterSettingsStrategyForeground.cpp

## 4.66 ColorFilterSettingsStrategyHue Class Reference

Leaf class for hue strategy for ColorFilterSettings.

`#include <ColorFilterSettingsStrategyHue.h>`

Inheritance diagram for ColorFilterSettingsStrategyHue:




### Public Member Functions

- ColorFilterSettingsStrategyHue ()
  *Single constructor.*
- virtual double high (const ColorFilterSettings &colorFilterSettings) const
  *Return the high value normalized to 0 to 1.*
- virtual double low (const ColorFilterSettings &colorFilterSettings) const
  *Return the low value normalized to 0 to 1.*
- virtual void printStream (const ColorFilterSettings &colorFilterSettings, QString indentation, QTextStream &str) const
  *Print the low and high values.*
- virtual void setHigh (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the high value given the normalized value.*
- virtual void setLow (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the low value given the normalized value.*

### 4.66.1 Detailed Description

Leaf class for hue strategy for ColorFilterSettings.

Definition at line 13 of file ColorFilterSettingsStrategyHue.h.

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

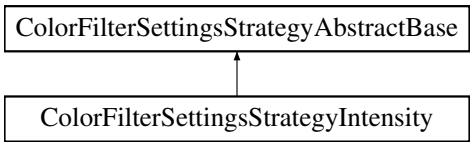
- Color/ColorFilterSettingsStrategyHue.h
- Color/ColorFilterSettingsStrategyHue.cpp

## 4.67 ColorFilterSettingsStrategyIntensity Class Reference

Leaf class for intensity strategy for ColorFilterSettings.

`#include <ColorFilterSettingsStrategyIntensity.h>`

Inheritance diagram for ColorFilterSettingsStrategyIntensity:




### Public Member Functions

- ColorFilterSettingsStrategyIntensity ()
  *Single constructor.*
- virtual double high (const ColorFilterSettings &colorFilterSettings) const
  *Return the high value normalized to 0 to 1.*
- virtual double low (const ColorFilterSettings &colorFilterSettings) const
  *Return the low value normalized to 0 to 1.*
- virtual void printStream (const ColorFilterSettings &colorFilterSettings, QString indentation, QTextStream &str) const
  *Print the low and high values.*
- virtual void setHigh (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the high value given the normalized value.*
- virtual void setLow (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the low value given the normalized value.*

### 4.67.1 Detailed Description

Leaf class for intensity strategy for ColorFilterSettings.

Definition at line 13 of file ColorFilterSettingsStrategyIntensity.h.

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

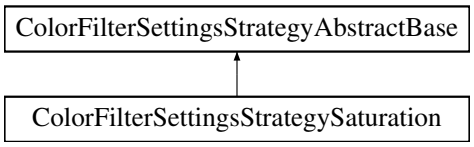
- Color/ColorFilterSettingsStrategyIntensity.h
- Color/ColorFilterSettingsStrategyIntensity.cpp

## 4.68 ColorFilterSettingsStrategySaturation Class Reference

Leaf class for saturation strategy for ColorFilterSettings.

`#include <ColorFilterSettingsStrategySaturation.h>`

Inheritance diagram for ColorFilterSettingsStrategySaturation:




### Public Member Functions

- ColorFilterSettingsStrategySaturation ()
  *Single constructor.*
- virtual double high (const ColorFilterSettings &colorFilterSettings) const
  *Return the high value normalized to 0 to 1.*
- virtual double low (const ColorFilterSettings &colorFilterSettings) const
  *Return the low value normalized to 0 to 1.*
- virtual void printStream (const ColorFilterSettings &colorFilterSettings, QString indentation, QTextStream &str) const
  *Print the low and high values.*
- virtual void setHigh (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the high value given the normalized value.*
- virtual void setLow (ColorFilterSettings &colorFilterSettings, double s0To1)
  *Set the low value given the normalized value.*

### 4.68.1 Detailed Description

Leaf class for saturation strategy for ColorFilterSettings.

Definition at line 13 of file ColorFilterSettingsStrategySaturation.h.

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

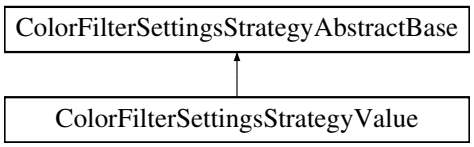
- Color/ColorFilterSettingsStrategySaturation.h
- Color/ColorFilterSettingsStrategySaturation.cpp

## 4.69 ColorFilterSettingsStrategyValue Class Reference

Leaf class for value strategy for ColorFilterSettings.

`#include <ColorFilterSettingsStrategyValue.h>`

Inheritance diagram for ColorFilterSettingsStrategyValue:



### Public Member Functions

- ColorFilterSettingsStrategyValue ()

  *Single constructor.*
- virtual double high (const ColorFilterSettings &colorFilterSettings) const

  *Return the high value normalized to 0 to 1.*
- virtual double low (const ColorFilterSettings &colorFilterSettings) const

  *Return the low value normalized to 0 to 1.*
- virtual void printStream (const ColorFilterSettings &colorFilterSettings, QString indentation, QTextStream &str) const

  *Print the low and high values.*
- virtual void setHigh (ColorFilterSettings &colorFilterSettings, double s0To1)

  *Set the high value given the normalized value.*
- virtual void setLow (ColorFilterSettings &colorFilterSettings, double s0To1)

  *Set the low value given the normalized value.*

#### 4.69.1 Detailed Description

Leaf class for value strategy for ColorFilterSettings.

Definition at line 13 of file ColorFilterSettingsStrategyValue.h.

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

- Color/ColorFilterSettingsStrategyValue.h
- Color/ColorFilterSettingsStrategyValue.cpp

## 4.70 ColorFilterStrategyAbstractBase Class Reference

Base class for strategy pattern whose subclasses process the different color filter settings modes (one strategy per mode).

```
#include <ColorFilterStrategyAbstractBase.h>
```

Inheritance diagram for ColorFilterStrategyAbstractBase:


### Public Member Functions

- ColorFilterStrategyAbstractBase ()

  *Single constructor.*
- virtual double pixelToZeroToOne (const QColor &pixel, QRgb rgbBackground) const =0

  *Return a normalized value of 0 to 1 given input pixel.*
- virtual int zeroToOneToValue (double s) const =0

  *Return the low value normalized to 0 to 1.*

### 4.70.1 Detailed Description

Base class for strategy pattern whose subclasses process the different color filter settings modes (one strategy per mode).

The strategy pattern nicely removes cyclomatic complexity from ColorFilter

Definition at line 19 of file ColorFilterStrategyAbstractBase.h.

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

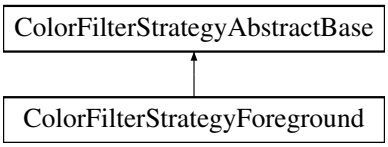
- Color/ColorFilterStrategyAbstractBase.h
- Color/ColorFilterStrategyAbstractBase.cpp

## 4.71 ColorFilterStrategyForeground Class Reference

Leaf class for foreground strategy for ColorFilter.

`#include <ColorFilterStrategyForeground.h>`

Inheritance diagram for ColorFilterStrategyForeground:




### Public Member Functions

- ColorFilterStrategyForeground ()

  *Single constructor.*
- virtual double pixelToZeroToOne (const QColor &pixel, QRgb rgbBackground) const

  *Return a normalized value of 0 to 1 given input pixel.*
- virtual int zeroToOneToValue (double s) const

  *Return the low value normalized to 0 to 1.*

### 4.71.1 Detailed Description

Leaf class for foreground strategy for ColorFilter.

Definition at line 13 of file ColorFilterStrategyForeground.h.

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

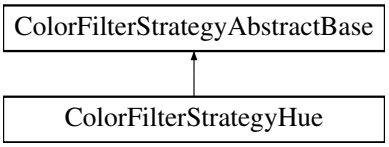
- Color/ColorFilterStrategyForeground.h
- Color/ColorFilterStrategyForeground.cpp

## 4.72 ColorFilterStrategyHue Class Reference

Leaf class for hue strategy for ColorFilter.

`#include <ColorFilterStrategyHue.h>`

Inheritance diagram for ColorFilterStrategyHue:




### Public Member Functions

- ColorFilterStrategyHue ()
  *Single constructor.*
- virtual double pixelToZeroToOne (const QColor &pixel, QRgb rgbBackground) const
  *Return a normalized value of 0 to 1 given input pixel.*
- virtual int zeroToOneToValue (double s) const
  *Return the low value normalized to 0 to 1.*

### 4.72.1 Detailed Description

Leaf class for hue strategy for ColorFilter.

Definition at line 13 of file ColorFilterStrategyHue.h.

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

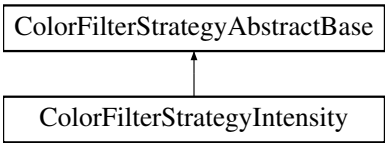
- Color/ColorFilterStrategyHue.h
- Color/ColorFilterStrategyHue.cpp

## 4.73 ColorFilterStrategyIntensity Class Reference

Leaf class for intensity strategy for ColorFilter.

`#include <ColorFilterStrategyIntensity.h>`

Inheritance diagram for ColorFilterStrategyIntensity:



### Public Member Functions

- ColorFilterStrategyIntensity ()

  *Single constructor.*
- virtual double pixelToZeroToOne (const QColor &pixel, QRgb rgbBackground) const

  *Return a normalized value of 0 to 1 given input pixel.*
- virtual int zeroToOneToValue (double s) const

  *Return the low value normalized to 0 to 1.*

#### 4.73.1 Detailed Description

Leaf class for intensity strategy for ColorFilter.

Definition at line 13 of file ColorFilterStrategyIntensity.h.

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

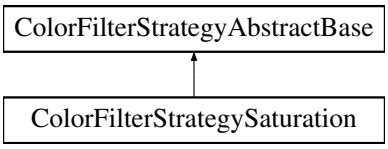
- Color/ColorFilterStrategyIntensity.h
- Color/ColorFilterStrategyIntensity.cpp

## 4.74 ColorFilterStrategySaturation Class Reference

Leaf class for saturation strategy for ColorFilter.

```
#include <ColorFilterStrategySaturation.h>
```

Inheritance diagram for ColorFilterStrategySaturation:




### Public Member Functions

- ColorFilterStrategySaturation ()

  *Single constructor.*
- virtual double pixelToZeroToOne (const QColor &pixel, QRgb rgbBackground) const

  *Return a normalized value of 0 to 1 given input pixel.*
- virtual int zeroToOneToValue (double s) const

  *Return the low value normalized to 0 to 1.*

### 4.74.1 Detailed Description

Leaf class for saturation strategy for ColorFilter.

Definition at line 13 of file ColorFilterStrategySaturation.h.

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

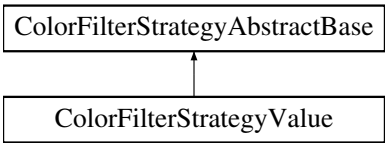
- Color/ColorFilterStrategySaturation.h
- Color/ColorFilterStrategySaturation.cpp

## 4.75 ColorFilterStrategyValue Class Reference

Leaf class for value strategy for ColorFilter.

`#include <ColorFilterStrategyValue.h>`

Inheritance diagram for ColorFilterStrategyValue:




### Public Member Functions

- ColorFilterStrategyValue ()

  *Single constructor.*
- virtual double pixelToZeroToOne (const QColor &pixel, QRgb rgbBackground) const

  *Return a normalized value of 0 to 1 given input pixel.*
- virtual int zeroToOneToValue (double s) const

  *Return the low value normalized to 0 to 1.*

### 4.75.1 Detailed Description

Leaf class for value strategy for ColorFilter.

Definition at line 13 of file ColorFilterStrategyValue.h.

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

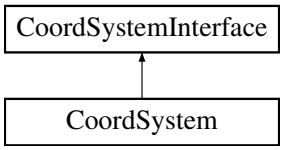
- Color/ColorFilterStrategyValue.h
- Color/ColorFilterStrategyValue.cpp

## 4.76 CoordSystem Class Reference

Storage of data belonging to one coordinate system.

`#include <CoordSystem.h>`

Inheritance diagram for CoordSystem:




### Public Member Functions

- CoordSystem ()

  *Single constructor.*
- CoordSystem (const QString &fileName)

  *Constructor for opened Graphs, and error report files. The specified file is opened and read.*
- virtual void addGraphCurveAtEnd (const QString &curveName)

  *Add new graph curve to the list of existing graph curves.*
- virtual void addPointAxisWithGeneratedIdentifier (const QPointF &posScreen, const QPointF &posGraph, QString &identifier, double ordinal, bool isXOnly)

  *Add a single axis point with a generated point identifier.*
- virtual void addPointAxisWithSpecifiedIdentifier (const QPointF &posScreen, const QPointF &posGraph, const QString &identifier, double ordinal, bool isXOnly)

  *Add a single axis point with the specified point identifier.*
- virtual void addPointGraphWithGeneratedIdentifier (const QString &curveName, const QPointF &posScreen, QString &generatedIentifier, double ordinal)

  *Add a single graph point with a generated point identifier.*
- virtual void addPointGraphWithSpecifiedIdentifier (const QString &curveName, const QPointF &posScreen, const QString &identifier, double ordinal)

  *Add a single graph point with the specified point identifer. Note that PointStyle is not applied to the point within the Graph.*
- virtual void addPointsInCurvesGraphs (CurvesGraphs &curvesGraphs)

  *Add all points identified in the specified CurvesGraphs. See also removePointsInCurvesGraphs.*
- virtual void checkAddPointAxis (const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage, bool isXOnly, DocumentAxesPointsRequired documentAxesPointsRequired)

  *Check before calling addPointAxis. Also returns the next available ordinal number (to prevent clashes)*
- virtual void checkEditPointAxis (const QString &pointIdentifier, const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage, DocumentAxesPointsRequired documentAxesPointsRequired)

  *Check before calling editPointAxis.*
- virtual const Curve & curveAxes () const

  *Get method for axis curve.*
- virtual Curve * curveForCurveName (const QString &curveName)

  *See CurvesGraphs::curveForCurveName, although this also works for AXIS_CURVE_NAME.*
- virtual const Curve * curveForCurveName (const QString &curveName) const

  *See CurvesGraphs::curveForCurveNames, although this also works for AXIS_CURVE_NAME.*
- virtual const CurvesGraphs & curvesGraphs () const

*Make all Curves available, read only, for CmdAbstract classes only.*

- virtual QStringList curvesGraphsNames () const

  *See CurvesGraphs::curvesGraphsNames.*

- virtual int curvesGraphsNumPoints (const QString &curveName) const

  *See CurvesGraphs::curvesGraphsNumPoints.*

- virtual void editPointAxis (const QPointF &posGraph, const QString &identifier)

  *Edit the graph coordinates of a single axis point. Call this after checkAddPointAxis to guarantee success in this call.*

- virtual void editPointGraph (bool isX, bool isY, double x, double y, const QStringList &identifiers, const Transformation &transformation)

  *Edit the graph coordinates of one or more graph points.*

- bool isXOnly (const QString &pointIdentifier) const

  *Return true if y coordinate is undefined, otherwise x coordinae is undefined in DOCUMENT_AXES_POINT_REQ↩ UIRE_4 mode.*

- virtual void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for the axes curve.*

- virtual void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurvePoints, for the axes curve.*

- virtual void iterateThroughCurveSegments (const QString &curveName, const Functor2wRet< const Point &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurveSegments, for any axes or graph curve.*

- virtual void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*

- virtual void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*

- virtual bool loadCurvesFile (const QString &curvesFile)

  *Load the curve names in the specified Engauge file into the current graph. This is called near the end of the import process only.*

- void loadPreVersion6 (QDataStream &str, double version, DocumentAxesPointsRequired &documentAxes↩ PointsRequired)

  *Load from file in pre-version 6 format. Number of axes points is read in and passed to Document.*

- void loadVersion6 (QXmlStreamReader &reader, DocumentAxesPointsRequired &documentAxesPoints↩ Required)

  *Load from file in version 6 format. Number of axes points is read in and passed to Document.*

- void loadVersions7AndUp (QXmlStreamReader &reader)

  *Load from file in versions 7 and 8 formats. Number of axes points is already defined at Document level.*

- virtual DocumentModelAxesChecker modelAxesChecker () const

  *Get method for DocumentModelAxesChecker.*

- virtual DocumentModelColorFilter modelColorFilter () const

  *Get method for DocumentModelColorFilter.*

- virtual DocumentModelCoords modelCoords () const

  *Get method for DocumentModelCoords.*

- virtual CurveStyles modelCurveStyles () const

  *Get method for CurveStyles.*

- virtual DocumentModelDigitizeCurve modelDigitizeCurve () const

  *Get method for DocumentModelDigitizeCurve.*

- virtual DocumentModelExportFormat modelExport () const

  *Get method for DocumentModelExportFormat.*

- virtual DocumentModelGeneral modelGeneral () const

*Get method for DocumentModelGeneral.*

- virtual DocumentModelGridDisplay modelGridDisplay () const

  *Get method for DocumentModelGridDisplay.*
- virtual DocumentModelGridRemoval modelGridRemoval () const

  *Get method for DocumentModelGridRemoval.*
- virtual DocumentModelPointMatch modelPointMatch () const

  *Get method for DocumentModelPointMatch.*
- virtual DocumentModelSegments modelSegments () const

  *Get method for DocumentModelSegments.*
- virtual void movePoint (const QString &pointIdentifier, const QPointF &deltaScreen)

  *See Curve::movePoint.*
- virtual int nextOrdinalForCurve (const QString &curveName) const

  *Default next ordinal value for specified curve.*
- virtual QPointF positionGraph (const QString &pointIdentifier) const

  *See Curve::positionGraph.*
- virtual QPointF positionScreen (const QString &pointIdentifier) const

  *See Curve::positionScreen.*
- virtual void print () const

  *Debugging method for printing directly from symbolic debugger.*
- virtual void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual QString reasonForUnsuccessfulRead () const

  *Return an informative text message explaining why startup loading failed. Applies if successfulRead returns false.*
- virtual void removePointAxis (const QString &identifier)

  *Perform the opposite of addPointAxis.*
- virtual void removePointGraph (const QString &identifier)

  *Perform the opposite of addPointGraph.*
- virtual void removePointsInCurvesGraphs (CurvesGraphs &curvesGraphs)

  *Remove all points identified in the specified CurvesGraphs. See also addPointsInCurvesGraphs.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save graph to xml.*
- virtual QString selectedCurveName () const

  *Currently selected curve name. This is used to set the selected curve combobox in MainWindow.*
- virtual void setCurveAxes (const Curve &curveAxes)

  *Let CmdAbstract classes overwrite axes Curve. Applies to current coordinate system.*
- virtual void setCurvesGraphs (const CurvesGraphs &curvesGraphs)

  *Let CmdAbstract classes overwrite CurvesGraphs. Applies to current coordinate system.*
- virtual void setModelAxesChecker (const DocumentModelAxesChecker &modelAxesChecker)

  *Set method for DocumentModelAxesChecker.*
- virtual void setModelColorFilter (const DocumentModelColorFilter &modelColorFilter)

  *Set method for DocumentModelColorFilter.*
- virtual void setModelCoords (const DocumentModelCoords &modelCoords)

  *Set method for DocumentModelCoords.*
- virtual void setModelCurveStyles (const CurveStyles &modelCurveStyles)

  *Set method for CurveStyles.*
- virtual void setModelDigitizeCurve (const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Set method for DocumentModelDigitizeCurve.*
- virtual void setModelExport (const DocumentModelExportFormat &modelExport)

  *Set method for DocumentModelExportFormat.*
- virtual void setModelGeneral (const DocumentModelGeneral &modelGeneral)

  *Set method for DocumentModelGeneral.*

- virtual void setModelGridDisplay (const DocumentModelGridDisplay &modelGridDisplay)

  *Set method for DocumentModelGridDisplay.*
- virtual void setModelGridRemoval (const DocumentModelGridRemoval &modelGridRemoval)

  *Set method for DocumentModelGridRemoval.*
- void setModelPointMatch (const DocumentModelPointMatch &modelPointMatch)

  *Set method for DocumentModelPointMatch.*
- virtual void setModelSegments (const DocumentModelSegments &modelSegments)

  *Set method for DocumentModelSegments.*
- virtual void setSelectedCurveName (const QString &selectedCurveName)

  *Save curve name that is selected for the current coordinate system, for the next time the coordinate system reappears.*
- virtual bool successfulRead () const

  *Return true if startup loading succeeded. If the loading failed then reasonForUnsuccessfulRed will explain why.*
- virtual void updatePointOrdinals (const Transformation &transformation)

  *Update point ordinals after point addition/removal or dragging.*

### 4.76.1 Detailed Description

Storage of data belonging to one coordinate system.

There can be one or more coordinate systems per graph, and one or more graphs in the image belonging to a Document

Definition at line 42 of file CoordSystem.h.

### 4.76.2 Member Function Documentation

#### 4.76.2.1 addPointAxisWithGeneratedIdentifier()

```
void CoordSystem::addPointAxisWithGeneratedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            QString & identifier,
            double ordinal,
            bool isXOnly )  [virtual]
```

Add a single axis point with a generated point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if graph coordinates have only x coordinate |

Implements CoordSystemInterface.

Definition at line 72 of file CoordSystem.cpp.

#### 4.76.2.2 addPointAxisWithSpecifiedIdentifier()

```
void CoordSystem::addPointAxisWithSpecifiedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            const QString & identifier,
            double ordinal,
            bool isXOnly )  [virtual]
```

Add a single axis point with the specified point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if graph coordinates have only x coordinate |

Implements CoordSystemInterface.

Definition at line 94 of file CoordSystem.cpp.

#### 4.76.2.3 isXOnly()

```
bool CoordSystem::isXOnly (
            const QString & pointIdentifier ) const
```

Return true if y coordinate is undefined, otherwise x coordinae is undefined in DOCUMENT_AXES_POINT_RE↩ QUIRE_4 mode.

Applies to axes points only

Definition at line 302 of file CoordSystem.cpp.

#### 4.76.2.4 updatePointOrdinals()

```
void CoordSystem::updatePointOrdinals (
            const Transformation & transformation )  [virtual]
```

Update point ordinals after point addition/removal or dragging.

See GraphicsScene::updatePointOrdinalsAfterDrag. Graph coordinates of point must be up to date

Implements CoordSystemInterface.

Definition at line 961 of file CoordSystem.cpp.

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

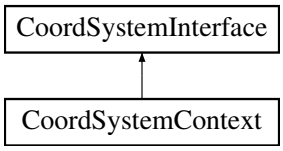
- CoordSystem/CoordSystem.h
- CoordSystem/CoordSystem.cpp

## 4.77 CoordSystemContext Class Reference

This class plays the role of context class in a state machine, although the 'states' are actually different instantiations of the CoordSystem class.

```
#include <CoordSystemContext.h>
```

Inheritance diagram for CoordSystemContext:




### Public Member Functions

- CoordSystemContext ()

  *Default constructor for constructing from opened file.*
- void addCoordSystems (unsigned int numberCoordSystemToAdd)

  *Add specified number of coordinate systems to the original one created by the constructor.*
- virtual void addGraphCurveAtEnd (const QString &curveName)

  *Add new graph curve to the list of existing graph curves.*
- virtual void addPointAxisWithGeneratedIdentifier (const QPointF &posScreen, const QPointF &posGraph, QString &identifier, double ordinal, bool isXOnly)

  *Add a single axis point with a generated point identifier.*
- virtual void addPointAxisWithSpecifiedIdentifier (const QPointF &posScreen, const QPointF &posGraph, const QString &identifier, double ordinal, bool isXOnly)

  *Add a single axis point with the specified point identifier.*
- virtual void addPointGraphWithGeneratedIdentifier (const QString &curveName, const QPointF &posScreen, QString &generatedIentifier, double ordinal)

  *Add a single graph point with a generated point identifier.*

- virtual void addPointGraphWithSpecifiedIdentifier (const QString &curveName, const QPointF &posScreen, const QString &identifier, double ordinal)

  *Add a single graph point with the specified point identifer. Note that PointStyle is not applied to the point within the Graph.*
- virtual void addPointsInCurvesGraphs (CurvesGraphs &curvesGraphs)

  *Add all points identified in the specified CurvesGraphs. See also removePointsInCurvesGraphs.*
- virtual void checkAddPointAxis (const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage, bool isXOnly, DocumentAxesPointsRequired documentAxesPointsRequired)

  *Check before calling addPointAxis. Also returns the next available ordinal number (to prevent clashes)*
- virtual void checkEditPointAxis (const QString &pointIdentifier, const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage, DocumentAxesPointsRequired documentAxesPointsRequired)

  *Check before calling editPointAxis.*
- const CoordSystem & coordSystem () const

  *Current CoordSystem.*
- unsigned int coordSystemCount () const

  *Number of CoordSystem.*
- CoordSystemIndex coordSystemIndex () const

  *Index of current CoordSystem.*
- virtual const Curve & curveAxes () const

  *Get method for axis curve.*
- virtual Curve * curveForCurveName (const QString &curveName)

  *See CurvesGraphs::curveForCurveName, although this also works for AXIS_CURVE_NAME.*
- virtual const Curve * curveForCurveName (const QString &curveName) const

  *See CurvesGraphs::curveForCurveNames, although this also works for AXIS_CURVE_NAME.*
- virtual const CurvesGraphs & curvesGraphs () const

  *Make all Curves available, read only, for CmdAbstract classes only.*
- virtual QStringList curvesGraphsNames () const

  *See CurvesGraphs::curvesGraphsNames.*
- virtual int curvesGraphsNumPoints (const QString &curveName) const

  *See CurvesGraphs::curvesGraphsNumPoints.*
- virtual void editPointAxis (const QPointF &posGraph, const QString &identifier)

  *Edit the graph coordinates of a single axis point. Call this after checkAddPointAxis to guarantee success in this call.*
- virtual void editPointGraph (bool isX, bool isY, double x, double y, const QStringList &identifiers, const Transformation &transformation)

  *Edit the graph coordinates of one or more graph points.*
- bool isXOnly (const QString &pointIdentifier) const

  *True/false if y/x value is empty.*
- virtual void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for the axes curve.*
- virtual void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurvePoints, for the axes curve.*
- virtual void iterateThroughCurveSegments (const QString &curveName, const Functor2wRet< const Point &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurveSegments, for any axes or graph curve.*
- virtual void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*
- virtual void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

*See Curve::iterateThroughCurvePoints, for all the graphs curves.*

- virtual bool loadCurvesFile (const QString &curvesFile)

  *Load the curve names in the specified Engauge file into the current graph. This is called near the end of the import process only.*
- void loadPreVersion6 (QDataStream &str, double version, DocumentAxesPointsRequired &documentAxes↩ PointsRequired)

  *Load from file in pre-version 6 format.*
- void loadVersion6 (QXmlStreamReader &reader, DocumentAxesPointsRequired &documentAxesPoints↩ Required)

  *Load from file in version 6 format, into the single CoordSystem.*
- void loadVersions7AndUp (QXmlStreamReader &reader)

  *Load one CoordSystem from file in version 7 format or newer, into the most recent CoordSystem which was just created before the call to this method.*
- virtual DocumentModelAxesChecker modelAxesChecker () const

  *Get method for DocumentModelAxesChecker.*
- virtual DocumentModelColorFilter modelColorFilter () const

  *Get method for DocumentModelColorFilter.*
- virtual DocumentModelCoords modelCoords () const

  *Get method for DocumentModelCoords.*
- virtual CurveStyles modelCurveStyles () const

  *Get method for CurveStyles.*
- virtual DocumentModelDigitizeCurve modelDigitizeCurve () const

  *Get method for DocumentModelDigitizeCurve.*
- virtual DocumentModelExportFormat modelExport () const

  *Get method for DocumentModelExportFormat.*
- virtual DocumentModelGeneral modelGeneral () const

  *Get method for DocumentModelGeneral.*
- virtual DocumentModelGridDisplay modelGridDisplay () const

  *Get method for DocumentModelGridDisplay.*
- virtual DocumentModelGridRemoval modelGridRemoval () const

  *Get method for DocumentModelGridRemoval.*
- virtual DocumentModelPointMatch modelPointMatch () const

  *Get method for DocumentModelPointMatch.*
- virtual DocumentModelSegments modelSegments () const

  *Get method for DocumentModelSegments.*
- virtual void movePoint (const QString &pointIdentifier, const QPointF &deltaScreen)

  *See Curve::movePoint.*
- virtual int nextOrdinalForCurve (const QString &curveName) const

  *Default next ordinal value for specified curve.*
- virtual QPointF positionGraph (const QString &pointIdentifier) const

  *See Curve::positionGraph.*
- virtual QPointF positionScreen (const QString &pointIdentifier) const

  *See Curve::positionScreen.*
- virtual void print () const

  *Debugging method for printing directly from symbolic debugger.*
- virtual void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual QString reasonForUnsuccessfulRead () const

  *Return an informative text message explaining why startup loading failed. Applies if successfulRead returns false.*
- virtual void removePointAxis (const QString &identifier)

  *Perform the opposite of addPointAxis.*

- virtual void removePointGraph (const QString &identifier)

  *Perform the opposite of addPointGraph.*
- virtual void removePointsInCurvesGraphs (CurvesGraphs &curvesGraphs)

  *Remove all points identified in the specified CurvesGraphs. See also addPointsInCurvesGraphs.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save graph to xml.*
- virtual QString selectedCurveName () const

  *Currently selected curve name. This is used to set the selected curve combobox in MainWindow.*
- void setCoordSystemIndex (CoordSystemIndex coordSystemIndex)

  *Index of current CoordSystem.*
- virtual void setCurveAxes (const Curve &curveAxes)

  *Let CmdAbstract classes overwrite axes Curve. Applies to current coordinate system.*
- virtual void setCurvesGraphs (const CurvesGraphs &curvesGraphs)

  *Let CmdAbstract classes overwrite CurvesGraphs. Applies to current coordinate system.*
- virtual void setModelAxesChecker (const DocumentModelAxesChecker &modelAxesChecker)

  *Set method for DocumentModelAxesChecker.*
- virtual void setModelColorFilter (const DocumentModelColorFilter &modelColorFilter)

  *Set method for DocumentModelColorFilter.*
- virtual void setModelCoords (const DocumentModelCoords &modelCoords)

  *Set method for DocumentModelCoords.*
- virtual void setModelCurveStyles (const CurveStyles &modelCurveStyles)

  *Set method for CurveStyles.*
- virtual void setModelDigitizeCurve (const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Set method for DocumentModelDigitizeCurve.*
- virtual void setModelExport (const DocumentModelExportFormat &modelExport)

  *Set method for DocumentModelExportFormat.*
- virtual void setModelGeneral (const DocumentModelGeneral &modelGeneral)

  *Set method for DocumentModelGeneral.*
- virtual void setModelGridDisplay (const DocumentModelGridDisplay &modelGridDisplay)

  *Set method for DocumentModelGridDisplay.*
- virtual void setModelGridRemoval (const DocumentModelGridRemoval &modelGridRemoval)

  *Set method for DocumentModelGridRemoval.*
- void setModelPointMatch (const DocumentModelPointMatch &modelPointMatch)

  *Set method for DocumentModelPointMatch.*
- virtual void setModelSegments (const DocumentModelSegments &modelSegments)

  *Set method for DocumentModelSegments.*
- virtual void setSelectedCurveName (const QString &selectedCurveName)

  *Save curve name that is selected for the current coordinate system, for the next time the coordinate system reappears.*
- virtual bool successfulRead () const

  *Return true if startup loading succeeded. If the loading failed then reasonForUnsuccessfulRed will explain why.*
- virtual void updatePointOrdinals (const Transformation &transformation)

  *Update point ordinals after point addition/removal or dragging.*

### 4.77.1 Detailed Description

This class plays the role of context class in a state machine, although the 'states' are actually different instantiations of the CoordSystem class.

At any point in time, one CoordSystem is active (as selected by the user)

Definition at line 24 of file CoordSystemContext.h.

## 4.77.2 Member Function Documentation

### 4.77.2.1 addPointAxisWithGeneratedIdentifier()

```
void CoordSystemContext::addPointAxisWithGeneratedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            QString & identifier,
            double ordinal,
            bool isXOnly )  [virtual]
```

Add a single axis point with a generated point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if graph coordinates have only x coordinate |

Implements CoordSystemInterface.

Definition at line 50 of file CoordSystemContext.cpp.

### 4.77.2.2 addPointAxisWithSpecifiedIdentifier()

```
void CoordSystemContext::addPointAxisWithSpecifiedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            const QString & identifier,
            double ordinal,
            bool isXOnly )  [virtual]
```

Add a single axis point with the specified point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if graph coordinates have only x coordinate |

Implements CoordSystemInterface.

Definition at line 65 of file CoordSystemContext.cpp.

#### 4.77.2.3 updatePointOrdinals()

```
void CoordSystemContext::updatePointOrdinals (
            const Transformation & transformation )  [virtual]
```

Update point ordinals after point addition/removal or dragging.

See GraphicsScene::updatePointOrdinalsAfterDrag. Graph coordinates of point must be up to date

Implements CoordSystemInterface.

Definition at line 589 of file CoordSystemContext.cpp.

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

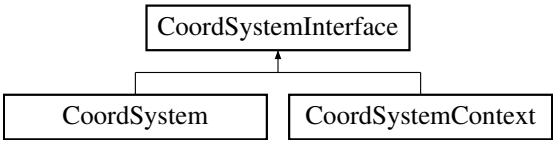
- CoordSystem/CoordSystemContext.h
- CoordSystem/CoordSystemContext.cpp

## 4.78 CoordSystemInterface Class Reference

Interface common to CoordSystemContext and CoordSystem classes.

```
#include <CoordSystemInterface.h>
```

Inheritance diagram for CoordSystemInterface:



## Public Member Functions

- CoordSystemInterface ()

  *Single constructor.*
- virtual void addGraphCurveAtEnd (const QString &curveName)=0

  *Add new graph curve to the list of existing graph curves.*
- virtual void addPointAxisWithGeneratedIdentifier (const QPointF &posScreen, const QPointF &posGraph, QString &identifier, double ordinal, bool isXOnly)=0

  *Add a single axis point with a generated point identifier.*
- virtual void addPointAxisWithSpecifiedIdentifier (const QPointF &posScreen, const QPointF &posGraph, const QString &identifier, double ordinal, bool isXOnly)=0

  *Add a single axis point with the specified point identifier.*
- virtual void addPointGraphWithGeneratedIdentifier (const QString &curveName, const QPointF &posScreen, QString &generatedIentifier, double ordinal)=0

  *Add a single graph point with a generated point identifier.*
- virtual void addPointGraphWithSpecifiedIdentifier (const QString &curveName, const QPointF &posScreen, const QString &identifier, double ordinal)=0

  *Add a single graph point with the specified point identifer. Note that PointStyle is not applied to the point within the Graph.*
- virtual void addPointsInCurvesGraphs (CurvesGraphs &curvesGraphs)=0

  *Add all points identified in the specified CurvesGraphs. See also removePointsInCurvesGraphs.*
- virtual void checkAddPointAxis (const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage, bool isXOnly, DocumentAxesPointsRequired documentAxesPointsRequired)=0

  *Check before calling addPointAxis. Also returns the next available ordinal number (to prevent clashes)*
- virtual void checkEditPointAxis (const QString &pointIdentifier, const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage, DocumentAxesPointsRequired documentAxesPointsRequired)=0

  *Check before calling editPointAxis.*
- virtual const Curve & curveAxes () const =0

  *Get method for axis curve.*
- virtual Curve * curveForCurveName (const QString &curveName)=0

  *See CurvesGraphs::curveForCurveName, although this also works for AXIS_CURVE_NAME.*
- virtual const Curve * curveForCurveName (const QString &curveName) const =0

  *See CurvesGraphs::curveForCurveNames, although this also works for AXIS_CURVE_NAME.*
- virtual const CurvesGraphs & curvesGraphs () const =0

  *Make all Curves available, read only, for CmdAbstract classes only.*
- virtual QStringList curvesGraphsNames () const =0

  *See CurvesGraphs::curvesGraphsNames.*
- virtual int curvesGraphsNumPoints (const QString &curveName) const =0

  *See CurvesGraphs::curvesGraphsNumPoints.*
- virtual void editPointAxis (const QPointF &posGraph, const QString &identifier)=0

  *Edit the graph coordinates of a single axis point. Call this after checkAddPointAxis to guarantee success in this call.*
- virtual void editPointGraph (bool isX, bool isY, double x, double y, const QStringList &identifiers, const Transformation &transformation)=0

  *Edit the graph coordinates of one or more graph points.*
- virtual void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)=0

  *See Curve::iterateThroughCurvePoints, for the axes curve.*
- virtual void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const =0

  *See Curve::iterateThroughCurvePoints, for the axes curve.*
- virtual void iterateThroughCurveSegments (const QString &curveName, const Functor2wRet< const Point &, const Point &, CallbackSearchReturn > &ftorWithCallback) const =0

*See Curve::iterateThroughCurveSegments, for any axes or graph curve.*

- virtual void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)=0

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*

- virtual void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const =0

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*

- virtual bool loadCurvesFile (const QString &curvesFile)=0

  *Load the curve names in the specified Engauge file into the current graph. This is called near the end of the import process only.*

- virtual DocumentModelAxesChecker modelAxesChecker () const =0

  *Get method for DocumentModelAxesChecker.*

- virtual DocumentModelColorFilter modelColorFilter () const =0

  *Get method for DocumentModelColorFilter.*

- virtual DocumentModelCoords modelCoords () const =0

  *Get method for DocumentModelCoords.*

- virtual CurveStyles modelCurveStyles () const =0

  *Get method for CurveStyles.*

- virtual DocumentModelDigitizeCurve modelDigitizeCurve () const =0

  *Get method for DocumentModelDigitizeCurve.*

- virtual DocumentModelExportFormat modelExport () const =0

  *Get method for DocumentModelExportFormat.*

- virtual DocumentModelGeneral modelGeneral () const =0

  *Get method for DocumentModelGeneral.*

- virtual DocumentModelGridDisplay modelGridDisplay () const =0

  *Get method for DocumentModelGridDisplay.*

- virtual DocumentModelGridRemoval modelGridRemoval () const =0

  *Get method for DocumentModelGridRemoval.*

- virtual DocumentModelPointMatch modelPointMatch () const =0

  *Get method for DocumentModelPointMatch.*

- virtual DocumentModelSegments modelSegments () const =0

  *Get method for DocumentModelSegments.*

- virtual void movePoint (const QString &pointIdentifier, const QPointF &deltaScreen)=0

  *See Curve::movePoint.*

- virtual int nextOrdinalForCurve (const QString &curveName) const =0

  *Default next ordinal value for specified curve.*

- virtual QPointF positionGraph (const QString &pointIdentifier) const =0

  *See Curve::positionGraph.*

- virtual QPointF positionScreen (const QString &pointIdentifier) const =0

  *See Curve::positionScreen.*

- virtual void print () const =0

  *Debugging method for printing directly from symbolic debugger.*

- virtual void printStream (QString indentation, QTextStream &str) const =0

  *Debugging method that supports print method of this class and printStream method of some other class(es)*

- virtual QString reasonForUnsuccessfulRead () const =0

  *Return an informative text message explaining why startup loading failed. Applies if successfulRead returns false.*

- virtual void removePointAxis (const QString &identifier)=0

  *Perform the opposite of addPointAxis.*

- virtual void removePointGraph (const QString &identifier)=0

  *Perform the opposite of addPointGraph.*

- virtual void removePointsInCurvesGraphs (CurvesGraphs &curvesGraphs)=0

*Remove all points identified in the specified CurvesGraphs. See also addPointsInCurvesGraphs.*

- virtual void saveXml (QXmlStreamWriter &writer) const =0

  *Save graph to xml.*
- virtual QString selectedCurveName () const =0

  *Currently selected curve name. This is used to set the selected curve combobox in MainWindow.*
- virtual void setCurveAxes (const Curve &curveAxes)=0

  *Let CmdAbstract classes overwrite axes Curve. Applies to current coordinate system.*
- virtual void setCurvesGraphs (const CurvesGraphs &curvesGraphs)=0

  *Let CmdAbstract classes overwrite CurvesGraphs. Applies to current coordinate system.*
- virtual void setModelAxesChecker (const DocumentModelAxesChecker &modelAxesChecker)=0

  *Set method for DocumentModelAxesChecker.*
- virtual void setModelColorFilter (const DocumentModelColorFilter &modelColorFilter)=0

  *Set method for DocumentModelColorFilter.*
- virtual void setModelCoords (const DocumentModelCoords &modelCoords)=0

  *Set method for DocumentModelCoords.*
- virtual void setModelCurveStyles (const CurveStyles &modelCurveStyles)=0

  *Set method for CurveStyles.*
- virtual void setModelDigitizeCurve (const DocumentModelDigitizeCurve &modelDigitizeCurve)=0

  *Set method for DocumentModelDigitizeCurve.*
- virtual void setModelExport (const DocumentModelExportFormat &modelExport)=0

  *Set method for DocumentModelExportFormat.*
- virtual void setModelGeneral (const DocumentModelGeneral &modelGeneral)=0

  *Set method for DocumentModelGeneral.*
- virtual void setModelGridDisplay (const DocumentModelGridDisplay &modelGridDisplay)=0

  *Set method for DocumentModelGridDisplay.*
- virtual void setModelGridRemoval (const DocumentModelGridRemoval &modelGridRemoval)=0

  *Set method for DocumentModelGridRemoval.*
- virtual void setModelPointMatch (const DocumentModelPointMatch &modelPointMatch)=0

  *Set method for DocumentModelPointMatch.*
- virtual void setModelSegments (const DocumentModelSegments &modelSegments)=0

  *Set method for DocumentModelSegments.*
- virtual void setSelectedCurveName (const QString &selectedCurveName)=0

  *Save curve name that is selected for the current coordinate system, for the next time the coordinate system reappears.*
- virtual bool successfulRead () const =0

  *Return true if startup loading succeeded. If the loading failed then reasonForUnsuccessfulRed will explain why.*
- virtual void updatePointOrdinals (const Transformation &transformation)=0

  *Update point ordinals after point addition/removal or dragging.*

### 4.78.1 Detailed Description

Interface common to CoordSystemContext and CoordSystem classes.

Definition at line 34 of file CoordSystemInterface.h.

### 4.78.2 Member Function Documentation

#### 4.78.2.1 addPointAxisWithGeneratedIdentifier()

```
virtual void CoordSystemInterface::addPointAxisWithGeneratedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            QString & identifier,
            double ordinal,
            bool isXOnly )  [pure virtual]
```

Add a single axis point with a generated point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if graph coordinates have only x coordinate |

Implemented in CoordSystem, and CoordSystemContext.

#### 4.78.2.2 addPointAxisWithSpecifiedIdentifier()

```
virtual void CoordSystemInterface::addPointAxisWithSpecifiedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            const QString & identifier,
            double ordinal,
            bool isXOnly )  [pure virtual]
```

Add a single axis point with the specified point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if graph coordinates have only x coordinate |

Implemented in CoordSystem, and CoordSystemContext.

#### 4.78.2.3 updatePointOrdinals()

```
virtual void CoordSystemInterface::updatePointOrdinals (
            const Transformation & transformation )  [pure virtual]
```

Update point ordinals after point addition/removal or dragging.

See GraphicsScene::updatePointOrdinalsAfterDrag. Graph coordinates of point must be up to date

Implemented in CoordSystem, and CoordSystemContext.

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

- CoordSystem/CoordSystemInterface.h
- CoordSystem/CoordSystemInterface.cpp

## 4.79 Correlation Class Reference

Fast cross correlation between two functions.

```
#include <Correlation.h>
```

### Public Member Functions

- Correlation (int N)

  *Single constructor. Slow memory allocations are done once and then reused repeatedly.*
- void correlateWithShift (int N, const double function1 [ ], const double function2 [ ], int &binStartMax, double &corrMax, double correlations [ ]) const

  *Return the shift in function1 that best aligns that function with function2.*
- void correlateWithoutShift (int N, const double function1 [ ], const double function2 [ ], double &corrMax) const

  *Return the correlation of the two functions, without any shift.*

### 4.79.1 Detailed Description

Fast cross correlation between two functions.

We do not use complex.h along with fftw3.h since then the complex numbers will be native, which would then require platform-dependent code

Definition at line 14 of file Correlation.h.

### 4.79.2 Member Function Documentation

#### 4.79.2.1 correlateWithoutShift()

```
void Correlation::correlateWithoutShift (
            int N,
            const double function1[],
            const double function2[],
            double & corrMax ) const
```

Return the correlation of the two functions, without any shift.

The functions are normalized internally.

Definition at line 131 of file Correlation.cpp.

#### 4.79.2.2 correlateWithShift()

```
void Correlation::correlateWithShift (
            int N,
            const double function1[],
            const double function2[],
            int & binStartMax,
            double & corrMax,
            double correlations[] ) const
```

Return the shift in function1 that best aligns that function with function2.

The functions are normalized internally. The correlations vector, as a function of shift, is returned for logging

Definition at line 44 of file Correlation.cpp.

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

- Correlation/Correlation.h
- Correlation/Correlation.cpp

## 4.80 CursorFactory Class Reference

Create standard cross cursor, or custom cursor, according to settings.

```
#include <CursorFactory.h>
```

### Public Member Functions

- CursorFactory ()
  *Single constructor.*
- QCursor generate (const DocumentModelDigitizeCurve &modelDigitizeCurve) const
  *Factory method to generate standard or custom cursor.*

### 4.80.1 Detailed Description

Create standard cross cursor, or custom cursor, according to settings.

Definition at line 15 of file CursorFactory.h.

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

- Cursor/CursorFactory.h
- Cursor/CursorFactory.cpp

## 4.81 Curve Class Reference

Container for one set of digitized Points.

`#include <Curve.h>`

### Public Member Functions

- Curve (const QString &curveName, const ColorFilterSettings &colorFilterSettings, const CurveStyle &curveStyle)

  *Constructor from scratch.*
- Curve (QDataStream &str)

  *Constructor from serialized binary pre-version 6 file.*
- Curve (QXmlStreamReader &reader)

  *Constructor for use when loading from serialized xml.*
- Curve (const Curve &curve)

  *Copy constructor. Copying a Curve only helps for making a copy, since access to any Points inside must be via functor.*
- Curve & operator= (const Curve &curve)

  *Assignment constructor.*
- void addPoint (Point point)

  *Add Point to this Curve.*
- ColorFilterSettings colorFilterSettings () const

  *Return the color filter.*
- QString curveName () const

  *Name of this Curve.*
- CurveStyle curveStyle () const

  *Return the curve style.*
- void editPointAxis (const QPointF &posGraph, const QString &identifier)

  *Edit the graph coordinates of an axis point. This method does not apply to a graph point.*
- void editPointGraph (bool isX, bool isY, double x, double y, const QStringList &identifiers, const Transformation &transformation)

  *Edit the graph coordinates of one or more graph points. This method does not apply to an axis point.*
- void exportToClipboard (const QHash< QString, bool > &selectedHash, const Transformation &transformation, QTextStream &strCsv, QTextStream &strHtml, CurvesGraphs &curvesGraphs) const

  *Export points in this Curve found in the specified point list.*
- bool isXOnly (const QString &pointIdentifier) const

  *Determine if specified point has just x coordinate. Otherwise has just y coordinate, or both x and y coordinates.*

- void iterateThroughCurvePoints (const Functor2wRet< const QString &, const Point &, CallbackSearch↩ Return > &ftorWithCallback) const

  *Apply functor to Points on Curve.*
- void iterateThroughCurveSegments (const Functor2wRet< const Point &, const Point &, CallbackSearch↩ Return > &ftorWithCallback) const

  *Apply functor to successive Points, as line segments, on Curve. This could be a bit slow.*
- void movePoint (const QString &pointIdentifier, const QPointF &deltaScreen)

  *Translate the position of a point by the specified distance vector.*
- int numPoints () const

  *Number of points.*
- const Points points () const

  *Return a shallow copy of the Points.*
- QPointF positionGraph (const QString &pointIdentifier) const

  *Return the position, in graph coordinates, of the specified Point.*
- QPointF positionScreen (const QString &pointIdentifier) const

  *Return the position, in screen coordinates, of the specified Point.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void removePoint (const QString &identifier)

  *Perform the opposite of addPointAtEnd.*
- void saveXml (QXmlStreamWriter &writer) const

  *Serialize curve.*
- void setColorFilterSettings (const ColorFilterSettings &colorFilterSettings)

  *Set color filter.*
- void setCurveName (const QString &curveName)

  *Change the curve name.*
- void setCurveStyle (const CurveStyle &curveStyle)

  *Set curve style.*
- void updatePointOrdinals (const Transformation &transformation)

  *See CurveGraphs::updatePointOrdinals.*

### 4.81.1 Detailed Description

Container for one set of digitized Points.

Definition at line 33 of file Curve.h.

### 4.81.2 Member Function Documentation

#### 4.81.2.1 updatePointOrdinals()

```
void Curve::updatePointOrdinals (
            const Transformation & transformation )
```

See CurveGraphs::updatePointOrdinals.

Same algorithm as GraphicsLinesForCurve::updatePointOrdinalsAfterDrag, although graph coordinates of points have been updated before this is called so the graph coordinates are not updated by this method

Definition at line 568 of file Curve.cpp.

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

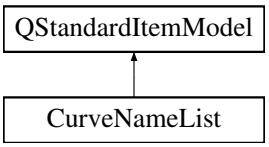
- Curve/Curve.h
- Curve/Curve.cpp

## 4.82 CurveNameList Class Reference

Model for DlgSettingsCurveAddRemove and CmdSettingsCurveAddRemove.

`#include <CurveNameList.h>`

Inheritance diagram for CurveNameList:




### Public Member Functions

- CurveNameList ()
  *Default constructor.*
- virtual int columnCount (const QModelIndex &parent) const
  *One column.*
- bool containsCurveNameCurrent (const QString &curveName) const
  *Return true if specified curve name is already in the list.*
- QString currentCurvesAsString () const
  *For debugging we dump the curve names.*
- QString currentCurveToOriginalCurve (const QString &currentCurve) const
  *Return the original curve for the specified current curve.*
- unsigned int currentCurveToPointCount (const QString &currentCurve) const
  *Return the point count for the specified current curve.*
- virtual Qt::ItemFlags flags (const QModelIndex &index) const
  *Override normal flags with additional editing flags.*
- void insertRow (int row, const QString &curveCurrent, const QString &curveOriginal, unsigned int pointCount)
  *Create a new entry at the specified row.*
- virtual QStandardItem * item (int row, int column=0) const
  *Retrieve data from model.*
- unsigned int numPointsForSelectedCurves (const QList< unsigned int > &rowsSelected) const
  *Return the number of points associated with the selected curves, as specified by their row numbers.*
- virtual bool removeRows (int row, int count, const QModelIndex &parent)
  *Remove one row.*
- void reset ()
  *Clear all information.*
- virtual int rowCount (const QModelIndex &parent=QModelIndex()) const
  *One row per curve name.*
- virtual bool setData (const QModelIndex &index, const QVariant &value, int role)
  *Store data for one curve name.*
- virtual void setItem (int row, int column, QStandardItem *item)
  *Store one curve name data.*
- virtual Qt::DropActions supportedDropActions () const
  *Allow dragging for reordering.*

### 4.82.1 Detailed Description

Model for DlgSettingsCurveAddRemove and CmdSettingsCurveAddRemove.

This is displayed as a QListView, with visible first column showing current curve name. Second column is hidden with curve name at the start of editing, or empty if none.

Definition at line 27 of file CurveNameList.h.

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

- Curve/CurveNameList.h
- Curve/CurveNameList.cpp

## 4.83 CurveSettingsInt Class Reference

Internal settings for one curve, such as LineStyle, PointStyle and CurveFilter.

`#include <CurveSettingsInt.h>`

### Public Member Functions

- CurveSettingsInt (const ColorFilterSettings &colorFilterSettings, const PointStyle &pointStyle, const LineStyle &lineStyle, CurveConnectAs curveConnectAs)

  *Single constructor.*
- CurveConnectAs curveConnectAs () const

  *Get method for connection method.*
- ColorFilterSettings colorFilterSettings () const

  *Get method for color filter.*
- LineStyle lineStyle () const

  *Get method for line style.*
- PointStyle pointStyle () const

  *Get method for point style.*

### 4.83.1 Detailed Description

Internal settings for one curve, such as LineStyle, PointStyle and CurveFilter.

These settings are used only internally by Curve, and are not related to the DlgSettings classes at all

Definition at line 17 of file CurveSettingsInt.h.

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

- Curve/CurveSettingsInt.h
- Curve/CurveSettingsInt.cpp

## 4.84 CurvesGraphs Class Reference

Container for all graph curves. The axes point curve is external to this class.

`#include <CurvesGraphs.h>`

### Public Member Functions

- void addGraphCurveAtEnd (Curve curve)

  *Append new graph Curve to end of Curve list.*
- void addPoint (const Point &point)

  *Append new Point to the specified Curve.*
- Curve * curveForCurveName (const QString &curveName)

  *Return the axis or graph curve for the specified curve name.*
- const Curve * curveForCurveName (const QString &curveName) const

  *Return the axis or graph curve for the specified curve name.*
- QStringList curvesGraphsNames () const

  *List of graph curve names.*
- int curvesGraphsNumPoints (const QString &curveName) const

  *Point count.*
- void editPointGraph (bool isX, bool isY, double x, double y, const QStringList &identifiers, const Transformation &transformation)

  *Set the x and/or y coordinate values of the specified points.*
- void iterateThroughCurvePoints (const QString &curveNameWanted, const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)

  *Apply functor to Points in the specified axis or graph Curve.*
- void iterateThroughCurveSegments (const QString &curveNameWanted, const Functor2wRet< const Point &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *Apply functor to segments on the specified axis or graph Curve.*
- void iterateThroughCurvesPoints (const Functor2wRet< const QString &, const Point &, CallbackSearch↩ Return > &ftorWithCallback)

  *Apply functor to Points on all of the Curves.*
- void iterateThroughCurvesPoints (const Functor2wRet< const QString &, const Point &, CallbackSearch↩ Return > &ftorWithCallback) const

  *Apply functor to Points on all of the Curves.*
- void loadPreVersion6 (QDataStream &str)

  *Load from serialized binary pre-version 6 file.*
- void loadXml (QXmlStreamReader &reader)

  *Load from serialized xml post-version 5 file.*
- int numCurves () const

  *Current number of graphs curves.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void removePoint (const QString &pointIdentifier)

  *Remove the Point from its Curve.*
- void saveXml (QXmlStreamWriter &writer) const

  *Serialize curves.*
- void updatePointOrdinals (const Transformation &transformation)

  *Update point ordinals to be consistent with their CurveStyle and x/theta coordinate.*

### 4.84.1 Detailed Description

Container for all graph curves. The axes point curve is external to this class.

Definition at line 24 of file CurvesGraphs.h.

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

- Curve/CurvesGraphs.h
- Curve/CurvesGraphs.cpp

## 4.85 CurveStyle Class Reference

Container for LineStyle and PointStyle for one Curve.

`#include <CurveStyle.h>`

### Public Member Functions

- CurveStyle ()
  *Default constructor.*
- CurveStyle (const LineStyle &lineStyle, const PointStyle &pointStyle)
  *Constructor with styles.*
- LineStyle lineStyle () const
  *Get method for LineStyle.*
- QString loadXml (QXmlStreamReader &reader)
  *Load from serialized xml. Returns the curve name.*
- PointStyle pointStyle () const
  *Get method for PointStyle.*
- void printStream (QString indentation, QTextStream &str) const
  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void saveXml (QXmlStreamWriter &writer, const QString &curveName) const
  *Serialize to xml.*
- void setLineColor (ColorPalette lineColor)
  *Set method for line color in specified curve.*
- void setLineConnectAs (CurveConnectAs curveConnectAs)
  *Set method for connect as method for lines in specified curve.*
- void setLineStyle (const LineStyle &lineStyle)
  *Set method for LineStyle.*
- void setLineWidth (int width)
  *Set method for line width in specified curve.*
- void setPointColor (ColorPalette curveColor)
  *Set method curve point color in specified curve.*
- void setPointLineWidth (int width)
  *Set method for curve point perimeter line width.*
- void setPointRadius (int radius)
  *Set method for curve point radius.*
- void setPointShape (PointShape shape)
  *Set method for curve point shape in specified curve.*
- void setPointStyle (const PointStyle &pointStyle)
  *Set method for PointStyle.*

### 4.85.1 Detailed Description

Container for LineStyle and PointStyle for one Curve.

Definition at line 18 of file CurveStyle.h.

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

- Curve/CurveStyle.h
- Curve/CurveStyle.cpp

## 4.86 CurveStyles Class Reference

Model for DlgSettingsCurveProperties and CmdSettingsCurveProperties.

`#include <CurveStyles.h>`

### Public Member Functions

- CurveStyles ()
  *Default constructor.*
- CurveStyles (const CoordSystem &coordSystem)
  *Initial constructor from Document.*
- CurveStyles (const CurveStyles &other)
  *Copy constructor.*
- CurveStyles & operator= (const CurveStyles &other)
  *Assignment constructor.*
- QStringList curveNames () const
  *List of all curve names.*
- CurveStyle curveStyle (const QString &curveName) const
  *CurveStyle in specified curve.*
- ColorPalette lineColor (const QString &curveName) const
  *Get method for line color in specified curve.*
- CurveConnectAs lineConnectAs (const QString &curveName) const
  *Get method for connect as method for lines in specified curve.*
- const LineStyle lineStyle (const QString &curveName) const
  *Get method for copying one line style in one step.*
- int lineWidth (const QString &curveName) const
  *Get method for line width in specified curve.*
- void loadXml (QXmlStreamReader &reader)
  *Load from serialized xml.*
- ColorPalette pointColor (const QString &curveName) const
  *Get method for curve point color in specified curve.*
- bool pointIsCircle (const QString &curveName) const
  *Get method for curve point is circle in specified curve.*
- int pointLineWidth (const QString &curveName) const
  *Get method for curve point line width.*
- QPolygonF pointPolygon (const QString &curveName) const
  *Get method for curve point polygon in specified curve.*

- int pointRadius (const QString &curveName) const

  *Get method for curve point radius.*
- PointShape pointShape (const QString &curveName) const

  *Get method for curve point shape.*
- const PointStyle pointStyle (const QString &curveName) const

  *Get method for copying one point style. Cannot return just a reference or else there is a warning about returning reference to temporary.*
- void saveXml (QXmlStreamWriter &writer) const

  *Serialize to xml.*
- void setCurveStyle (const QString &curveName, const CurveStyle &curveStyle)

  *Set method for curve style.*
- void setLineColor (const QString &curveName, ColorPalette lineColor)

  *Set method for line color in specified curve.*
- void setLineConnectAs (const QString &curveName, CurveConnectAs curveConnectAs)

  *Set method for connect as method for lines in specified curve.*
- void setLineWidth (const QString &curveName, int width)

  *Set method for line width in specified curve.*
- void setPointColor (const QString &curveName, ColorPalette curveColor)

  *Set method curve point color in specified curve.*
- void setPointIsCircle (const QString &curveName, bool pointIsCircle)

  *Set method for curve point is circle in specified curve.*
- void setPointLineWidth (const QString &curveName, int width)

  *Set method for curve point perimeter line width.*
- void setPointRadius (const QString &curveName, int radius)

  *Set method for curve point radius.*
- void setPointShape (const QString &curveName, PointShape shape)

  *Set method for curve point shape in specified curve.*

### 4.86.1 Detailed Description

Model for DlgSettingsCurveProperties and CmdSettingsCurveProperties.

Definition at line 22 of file CurveStyles.h.

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

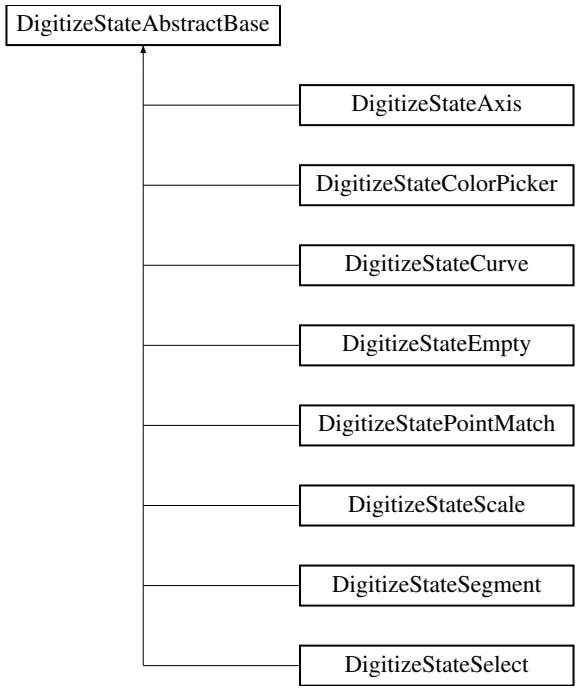
- Curve/CurveStyles.h
- Curve/CurveStyles.cpp

## 4.87 DigitizeStateAbstractBase Class Reference

Base class for all digitizing states. This serves as an interface to DigitizeStateContext.

`#include <DigitizeStateAbstractBase.h>`

Inheritance diagram for DigitizeStateAbstractBase:




## Public Member Functions

- DigitizeStateAbstractBase (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const =0

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)=0

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const =0

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- DigitizeStateContext & context ()

  *Reference to the DigitizeStateContext that contains all the DigitizeStateAbstractBase subclasses, without const.*
- const DigitizeStateContext & context () const

  *Reference to the DigitizeStateContext that contains all the DigitizeStateAbstractBase subclasses, without const.*
- virtual void end ()=0

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)=0

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)=0

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)=0

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)=0

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)=0

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF pos)=0

  *Handle a mouse press that was intercepted earlier.*

- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF pos)=0

  *Handle a mouse release that was intercepted earlier.*
- void setCursor (CmdMediator *cmdMediator)

  *Update the cursor according to the current state.*
- virtual QString state () const =0

  *State name for debugging.*
- virtual void updateAfterPointAddition ()=0

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)=0

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)=0

  *Update the segments given the new settings.*

### Protected Member Functions

- bool canPasteProtected (const Transformation &transformation, const QSize &viewSize) const

  *Protected version of canPaste method. Some, but not all, leaf classes use this method.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const =0

  *Returns the state-specific cursor shape.*

## 4.87.1 Detailed Description

Base class for all digitizing states. This serves as an interface to DigitizeStateContext.

Definition at line 37 of file DigitizeStateAbstractBase.h.

## 4.87.2 Member Function Documentation

### 4.87.2.1 begin()

```
virtual void DigitizeStateAbstractBase::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [pure virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implemented in DigitizeStateScale, DigitizeStateColorPicker, DigitizeStatePointMatch, DigitizeStateSegment, DigitizeStateSelect, DigitizeStateAxis, DigitizeStateCurve, and DigitizeStateEmpty.

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

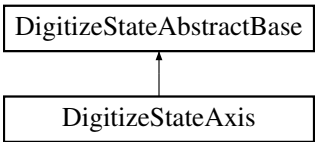
- DigitizeState/DigitizeStateAbstractBase.h
- DigitizeState/DigitizeStateAbstractBase.cpp

## 4.88 DigitizeStateAxis Class Reference

Digitizing state for digitizing one axis point at a time.

`#include <DigitizeStateAxis.h>`

Inheritance diagram for DigitizeStateAxis:




### Public Member Functions

- DigitizeStateAxis (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

### Additional Inherited Members

### 4.88.1 Detailed Description

Digitizing state for digitizing one axis point at a time.

Once three axis points are defined, those points define an affine transformation from pixel screen coordinates to graph coordinates.

Definition at line 14 of file DigitizeStateAxis.h.

### 4.88.2 Member Function Documentation

#### 4.88.2.1 begin()

```
void DigitizeStateAxis::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 38 of file DigitizeStateAxis.cpp.

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

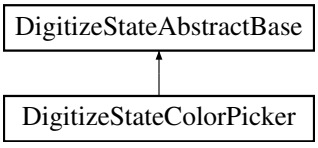
- DigitizeState/DigitizeStateAxis.h
- DigitizeState/DigitizeStateAxis.cpp

## 4.89 DigitizeStateColorPicker Class Reference

Digitizing state for selecting a color for DigitizeStateSegment.

```
#include <DigitizeStateColorPicker.h>
```

Inheritance diagram for DigitizeStateColorPicker:



### Public Member Functions

- DigitizeStateColorPicker (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

### Additional Inherited Members

#### 4.89.1 Detailed Description

Digitizing state for selecting a color for DigitizeStateSegment.

The basic strategy is that this class acts like a special case of DlgSettingsFilter. Specifically, the pixel just selected by a mouse click is used to change the segment filter for the currently specified curve

Definition at line 21 of file DigitizeStateColorPicker.h.

### 4.89.2 Member Function Documentation

#### 4.89.2.1 begin()

```
void DigitizeStateColorPicker::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 37 of file DigitizeStateColorPicker.cpp.

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

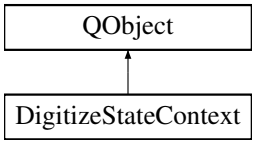
- DigitizeState/DigitizeStateColorPicker.h
- DigitizeState/DigitizeStateColorPicker.cpp

## 4.90 DigitizeStateContext Class Reference

Container for all DigitizeStateAbstractBase subclasses. This functions as the context class in a standard state machine implementation.

```
#include <DigitizeStateContext.h>
```

Inheritance diagram for DigitizeStateContext:



## Public Member Functions

- DigitizeStateContext (MainWindow &mainWindow, QGraphicsView &view, bool isGnuplot)

  *Single constructor.*
- QString activeCurve () const

  *Curve name for active Curve. This can include AXIS_CURVE_NAME, and empty string.*
- void appendNewCmd (CmdMediator *cmdMediator, QUndoCommand *cmd)

  *Append just-created QUndoCommand to command stack. This is called from DigitizeStateAbstractBase subclasses.*
- bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that operation is compatible with the current state.*
- void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *See DigitizeStateAbstractBase::handleContextMenuEventAxis.*
- void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *See DigitizeStateAbstractBase::handleContextMenuEventGraph.*
- void handleCurveChange (CmdMediator *cmdMediator)

  *See DigitizeStateAbstractBase::handleCurveChange.*
- void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *See DigitizeStateAbstractBase::handleKeyPress.*
- void handleMouseMove (CmdMediator *cmdMediator, QPointF pos)

  *See DigitizeStateAbstractBase::handleMouseMove.*
- void handleMousePress (CmdMediator *cmdMediator, QPointF pos)

  *See DigitizeStateAbstractBase::handleMousePress.*
- void handleMouseRelease (CmdMediator *cmdMediator, QPointF pos)

  *See DigitizeStateAbstractBase::handleMouseRelease.*
- bool isGnuplot () const

  *Get method for gnuplot flag.*
- MainWindow & mainWindow ()

  *Reference to the MainWindow, without const.*
- const MainWindow & mainWindow () const

  *Reference to the MainWindow, with const.*
- void requestDelayedStateTransition (DigitizeState digitizeState)

  *Initiate state transition to be performed later, when DigitizeState is off the stack.*
- void requestImmediateStateTransition (CmdMediator *cmdMediator, DigitizeState digitizeState)

  *Perform immediate state transition. Called from outside state machine.*
- void resetOnLoad (CmdMediator *cmdMediator)

  *Resetting makes re-initializes for documents after the first.*
- void setCursor (CmdMediator *cmdMediator)

  *Set cursor after asking state for the new cursor shape.*
- void setDragMode (QGraphicsView::DragMode dragMode)

  *Set QGraphicsView drag mode (in m_view). Called from DigitizeStateAbstractBase subclasses.*
- void setImageIsLoaded (CmdMediator *cmdMediator, bool imageIsLoaded)

  *Set the image so QGraphicsView cursor and drag mode are accessible.*
- QString state () const

  *State name for debugging.*
- void updateAfterPointAddition ()

  *Update the graphics attributes.*
- void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*
- QGraphicsView & view ()

  *QGraphicsView for use by DigitizeStateAbstractBase subclasses.*

### 4.90.1 Detailed Description

Container for all DigitizeStateAbstractBase subclasses. This functions as the context class in a standard state machine implementation.

Definition at line 27 of file DigitizeStateContext.h.

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

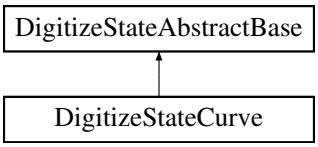
- DigitizeState/DigitizeStateContext.h
- DigitizeState/DigitizeStateContext.cpp

## 4.91 DigitizeStateCurve Class Reference

Digitizing state for creating Curve Points, one at a time.

`#include <DigitizeStateCurve.h>`

Inheritance diagram for DigitizeStateCurve:




### Public Member Functions

- DigitizeStateCurve (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*

- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

## Additional Inherited Members

### 4.91.1 Detailed Description

Digitizing state for creating Curve Points, one at a time.

Definition at line 13 of file DigitizeStateCurve.h.

### 4.91.2 Member Function Documentation

#### 4.91.2.1 begin()

```
void DigitizeStateCurve::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 35 of file DigitizeStateCurve.cpp.

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

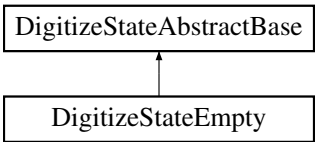
- DigitizeState/DigitizeStateCurve.h
- DigitizeState/DigitizeStateCurve.cpp

## 4.92 DigitizeStateEmpty Class Reference

Digitizing state before a Document has been created. In this state, the cursor is Qt::ArrowCursor.

`#include <DigitizeStateEmpty.h>`

Inheritance diagram for DigitizeStateEmpty:




### Public Member Functions

- DigitizeStateEmpty (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

### Additional Inherited Members

### 4.92.1 Detailed Description

Digitizing state before a Document has been created. In this state, the cursor is Qt::ArrowCursor.

Definition at line 13 of file DigitizeStateEmpty.h.

### 4.92.2 Member Function Documentation

#### 4.92.2.1 begin()

```
void DigitizeStateEmpty::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 29 of file DigitizeStateEmpty.cpp.

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

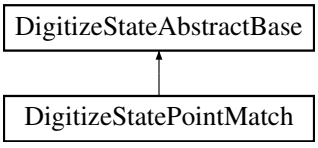
- DigitizeState/DigitizeStateEmpty.h
- DigitizeState/DigitizeStateEmpty.cpp

## 4.93 DigitizeStatePointMatch Class Reference

Digitizing state for matching Curve Points, one at a time.

```
#include <DigitizeStatePointMatch.h>
```

Inheritance diagram for DigitizeStatePointMatch:



### Public Member Functions

- DigitizeStatePointMatch (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

### Additional Inherited Members

#### 4.93.1 Detailed Description

Digitizing state for matching Curve Points, one at a time.

Definition at line 21 of file DigitizeStatePointMatch.h.

### 4.93.2 Member Function Documentation

#### 4.93.2.1 begin()

```
void DigitizeStatePointMatch::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 52 of file DigitizeStatePointMatch.cpp.

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

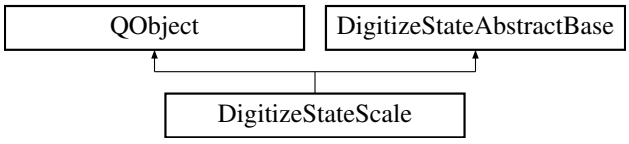
- DigitizeState/DigitizeStatePointMatch.h
- DigitizeState/DigitizeStatePointMatch.cpp

## 4.94 DigitizeStateScale Class Reference

Digitizing state for creating the scale bar.

```
#include <DigitizeStateScale.h>
```

Inheritance diagram for DigitizeStateScale:



### Public Member Functions

- DigitizeStateScale (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

### Additional Inherited Members

#### 4.94.1 Detailed Description

Digitizing state for creating the scale bar.

A scale bar is preferred over an approach using two axis points and DigitizeStateAxis since:

1. Fewer clicks are involved to move the scale bar. One click and drag operation moves the scale bar, but two click and drag operations would be needed to move two axis points
2. Clicking on a large scale bar is easier than clicking on much smaller axis points
3. DigitizeStateAxis solution would have a line, representing the scale bar, that would be unselectable - confusing to users

Definition at line 23 of file DigitizeStateScale.h.

### 4.94.2 Member Function Documentation

#### 4.94.2.1 begin()

```
void DigitizeStateScale::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 44 of file DigitizeStateScale.cpp.

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

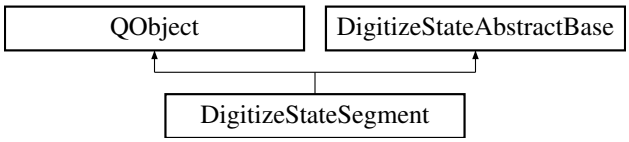
- DigitizeState/DigitizeStateScale.h
- DigitizeState/DigitizeStateScale.cpp

## 4.95 DigitizeStateSegment Class Reference

Digitizing state for creating multiple Points along a highlighted segment.

```
#include <DigitizeStateSegment.h>
```

Inheritance diagram for DigitizeStateSegment:




### Public Slots

- void slotMouseClickOnSegment (QPointF)

  *Receive signal from Segment that has been clicked on. The CmdMediator from the begin method will be used.*

### Public Member Functions

- DigitizeStateSegment (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

### Additional Inherited Members

#### 4.95.1 Detailed Description

Digitizing state for creating multiple Points along a highlighted segment.

Definition at line 17 of file DigitizeStateSegment.h.

### 4.95.2 Member Function Documentation

#### 4.95.2.1 begin()

```
void DigitizeStateSegment::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 36 of file DigitizeStateSegment.cpp.

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

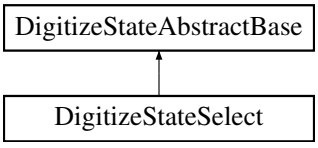
- DigitizeState/DigitizeStateSegment.h
- DigitizeState/DigitizeStateSegment.cpp

## 4.96 DigitizeStateSelect Class Reference

Digitizing state for selecting one or more Points in the Document.

```
#include <DigitizeStateSelect.h>
```

Inheritance diagram for DigitizeStateSelect:



### Public Member Functions

- DigitizeStateSelect (DigitizeStateContext &context)

  *Single constructor.*
- virtual QString activeCurve () const

  *Name of the active Curve. This can include AXIS_CURVE_NAME.*
- virtual void begin (CmdMediator *cmdMediator, DigitizeState previousState)

  *Method that is called at the exact moment a state is entered.*
- virtual bool canPaste (const Transformation &transformation, const QSize &viewSize) const

  *Return true if there is good data in the clipboard for pasting, and that is compatible with the current state.*
- virtual QCursor cursor (CmdMediator *cmdMediator) const

  *Returns the state-specific cursor shape.*
- virtual void end ()

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void handleContextMenuEventAxis (CmdMediator *cmdMediator, const QString &pointIdentifier)

  *Handle a right click, on an axis point, that was intercepted earlier.*
- virtual void handleContextMenuEventGraph (CmdMediator *cmdMediator, const QStringList &pointIdentifiers)

  *Handle a right click, on a graph point, that was intercepted earlier.*
- virtual void handleCurveChange (CmdMediator *cmdMediator)

  *Handle the selection of a new curve. At a minimum, DigitizeStateSegment will generate a new set of Segments.*
- virtual void handleKeyPress (CmdMediator *cmdMediator, Qt::Key key, bool atLeastOneSelectedItem)

  *Handle a key press that was intercepted earlier.*
- virtual void handleMouseMove (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse move. This is part of an experiment to see if augmenting the cursor in Point Match mode is worthwhile.*
- virtual void handleMousePress (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse press that was intercepted earlier.*
- virtual void handleMouseRelease (CmdMediator *cmdMediator, QPointF posScreen)

  *Handle a mouse release that was intercepted earlier.*
- virtual QString state () const

  *State name for debugging.*
- virtual void updateAfterPointAddition ()

  *Update graphics attributes after possible new points. This is useful for highlight opacity.*
- virtual void updateModelDigitizeCurve (CmdMediator *cmdMediator, const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update the digitize curve settings.*
- virtual void updateModelSegments (const DocumentModelSegments &modelSegments)

  *Update the segments given the new settings.*

### Additional Inherited Members

#### 4.96.1 Detailed Description

Digitizing state for selecting one or more Points in the Document.

Originally this class set the cursor for each QGraphicsItem at the beginning of the state, but that triggered Qt bug 4190 which has the description 'If you have set the cursor for some QGraphicsItems you can no longer change the cursor for the view in for example a mouseReleaseEvent'. In turn, that lead to Engauge issue #155. Unfortunately, this means the user no longer has need feedback that suggests the user can do something with the QGraphicsItems.

Definition at line 19 of file DigitizeStateSelect.h.

### 4.96.2 Member Function Documentation

#### 4.96.2.1 begin()

```
void DigitizeStateSelect::begin (
            CmdMediator * cmdMediator,
            DigitizeState previousState )  [virtual]
```

Method that is called at the exact moment a state is entered.

Typically called just after end for the previous state. The previousState value is used by DigitizeStateColorPicker to return to the previous state

Implements DigitizeStateAbstractBase.

Definition at line 69 of file DigitizeStateSelect.cpp.

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

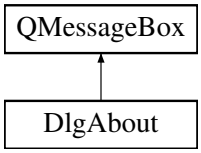
- DigitizeState/DigitizeStateSelect.h
- DigitizeState/DigitizeStateSelect.cpp

## 4.97 DlgAbout Class Reference

About Engauge dialog. This provides a hidden shortcut for triggering ENGAUGE_ASSERT.

```
#include <DlgAbout.h>
```

Inheritance diagram for DlgAbout:




### Public Member Functions

- DlgAbout (MainWindow &mainWindow)
  *Single constructor.*

### 4.97.1 Detailed Description

About Engauge dialog. This provides a hidden shortcut for triggering ENGAUGE_ASSERT.

Definition at line 15 of file DlgAbout.h.

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

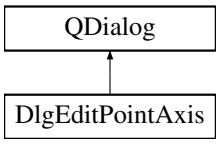
- Dlg/DlgAbout.h
- Dlg/DlgAbout.cpp

## 4.98 DlgEditPointAxis Class Reference

Dialog box for editing the information of one axis point, in a graph with two axes.

`#include <DlgEditPointAxis.h>`

Inheritance diagram for DlgEditPointAxis:




### Public Member Functions

- DlgEditPointAxis (MainWindow &mainWindow, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, DocumentAxesPointsRequired documentAxesPointsRequired, bool isXOnly=false, const double *xInitialValue=0, const double *yInitialValue=0)

  *Constructor for existing point which already has graph coordinates (which may be changed using this dialog).*
- QPointF posGraph (bool &isXOnly) const

  *Return the graph coordinates position specified by the user. Only applies if dialog was accepted.*

### 4.98.1 Detailed Description

Dialog box for editing the information of one axis point, in a graph with two axes.

Definition at line 24 of file DlgEditPointAxis.h.

### 4.98.2 Constructor & Destructor Documentation

#### 4.98.2.1 DlgEditPointAxis()

```
DlgEditPointAxis::DlgEditPointAxis (
            MainWindow & mainWindow,
            const DocumentModelCoords & modelCoords,
            const DocumentModelGeneral & modelGeneral,
            const MainWindowModel & modelMainWindow,
            const Transformation & transformation,
            DocumentAxesPointsRequired documentAxesPointsRequired,
            bool isXOnly = false,
            const double * xInitialValue = 0,
            const double * yInitialValue = 0 )
```

Constructor for existing point which already has graph coordinates (which may be changed using this dialog).

If initial values are unspecified then the value fields will be initially empty

Definition at line 38 of file DlgEditPointAxis.cpp.

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

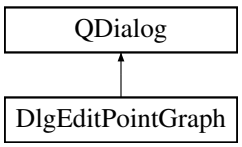
- Dlg/DlgEditPointAxis.h
- Dlg/DlgEditPointAxis.cpp

## 4.99 DlgEditPointGraph Class Reference

Dialog box for editing the information of one or more points.

`#include <DlgEditPointGraph.h>`

Inheritance diagram for DlgEditPointGraph:




### Public Member Functions

- DlgEditPointGraph (MainWindow &mainWindow, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, const double *xInitialValue=0, const double *yInitialValue=0)

  *Constructor for existing point which already has graph coordinates (which may be changed using this dialog).*
- void posGraph (bool &isX, double &x, bool &isY, double &y) const

  *Return one or both coordinates. Only applies if dialog was accepted.*

### 4.99.1 Detailed Description

Dialog box for editing the information of one or more points.

Definition at line 25 of file DlgEditPointGraph.h.

### 4.99.2 Constructor & Destructor Documentation

#### 4.99.2.1 DlgEditPointGraph()

```
DlgEditPointGraph::DlgEditPointGraph (
            MainWindow & mainWindow,
            const DocumentModelCoords & modelCoords,
            const DocumentModelGeneral & modelGeneral,
            const MainWindowModel & modelMainWindow,
            const Transformation & transformation,
            const double * xInitialValue = 0,
            const double * yInitialValue = 0 )
```

Constructor for existing point which already has graph coordinates (which may be changed using this dialog).

If initial values are unspecified then the value fields will be initially empty

Definition at line 28 of file DlgEditPointGraph.cpp.

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

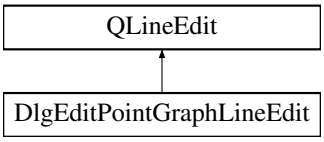
- Dlg/DlgEditPointGraph.h
- Dlg/DlgEditPointGraph.cpp

## 4.100 DlgEditPointGraphLineEdit Class Reference

Adds hover highlighting to QLineEdit.

`#include <DlgEditPointGraphLineEdit.h>`

Inheritance diagram for DlgEditPointGraphLineEdit:




### Public Member Functions

- DlgEditPointGraphLineEdit (QWidget *widget=0)

  *Single constructor.*
- virtual void enterEvent (QEvent *)

  *Hover entry triggers clearing of the background color so user does not think of widget as disabled and is encouraged to enter text.*
- virtual void leaveEvent (QEvent *)

  *Hover exit triggers restoration of the background color.*
- void updateBackground ()

  *Update background given the current state.*

### 4.100.1 Detailed Description

Adds hover highlighting to QLineEdit.

Definition at line 15 of file DlgEditPointGraphLineEdit.h.

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

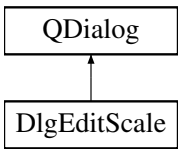
- Dlg/DlgEditPointGraphLineEdit.h
- Dlg/DlgEditPointGraphLineEdit.cpp

## 4.101 DlgEditScale Class Reference

Dialog box for editing the information of the map scale.

`#include <DlgEditScale.h>`

Inheritance diagram for DlgEditScale:



### Public Member Functions

- DlgEditScale (MainWindow &mainWindow, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const double *scaleLength=0)

  *Single constructor.*
- double scaleLength () const

  *Return the scale bar length specified by the user. Only applies if dialog was accepted.*

#### 4.101.1 Detailed Description

Dialog box for editing the information of the map scale.

Definition at line 22 of file DlgEditScale.h.

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

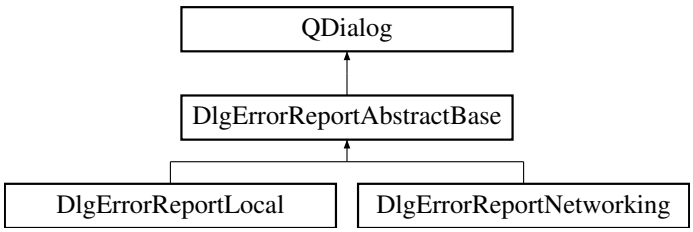
- Dlg/DlgEditScale.h
- Dlg/DlgEditScale.cpp

## 4.102 DlgErrorReportAbstractBase Class Reference

Base class for dialogs that handle the error report.

`#include <DlgErrorReportAbstractBase.h>`

Inheritance diagram for DlgErrorReportAbstractBase:




### Public Member Functions

- DlgErrorReportAbstractBase (QWidget *parent=0)

  *Single constructor.*

### Protected Member Functions

- QString errorFile () const

  *File name for output file containing error report.*
- void saveFile (const QString &xml) const

  *Save xml into output file named by errorFile.*

### 4.102.1 Detailed Description

Base class for dialogs that handle the error report.

Definition at line 13 of file DlgErrorReportAbstractBase.h.

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

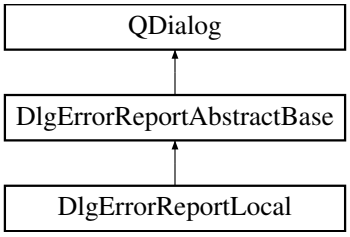
- Dlg/DlgErrorReportAbstractBase.h
- Dlg/DlgErrorReportAbstractBase.cpp

## 4.103 DlgErrorReportLocal Class Reference

Dialog for saving error report to local hard drive.

`#include <DlgErrorReportLocal.h>`

Inheritance diagram for DlgErrorReportLocal:




### Public Member Functions

- DlgErrorReportLocal (const QString &xmlWithImage, QWidget *parent=0)

  *Single constructor. With the original data, the extra context improves debugging. With anonymization, user privacy is maintained.*

### Additional Inherited Members

### 4.103.1 Detailed Description

Dialog for saving error report to local hard drive.

Definition at line 15 of file DlgErrorReportLocal.h.

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

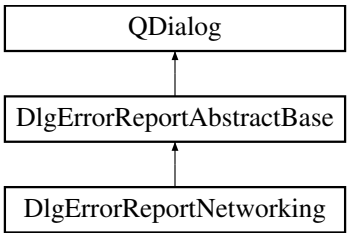
- Dlg/DlgErrorReportLocal.h
- Dlg/DlgErrorReportLocal.cpp

## 4.104 DlgErrorReportNetworking Class Reference

Dialog for sending error report with networking.

`#include <DlgErrorReportNetworking.h>`

Inheritance diagram for DlgErrorReportNetworking:




### Public Member Functions

- DlgErrorReportNetworking (const QString &xmlWithImage, QWidget *parent=0)

  *Single constructor. With the original data, the extra context improves debugging. With anonymization, user privacy is maintained.*
- QString xmlToUpload () const

  *Xml to be uploaded. Includes document if user has approved.*

### Additional Inherited Members

### 4.104.1 Detailed Description

Dialog for sending error report with networking.

Even if it is not sent, the information is available while this dialog is open, as a file in the executable directory

Definition at line 17 of file DlgErrorReportNetworking.h.

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

- Dlg/DlgErrorReportNetworking.h
- Dlg/DlgErrorReportNetworking.cpp

## 4.105 DlgFilterCommand Class Reference

Command pattern object for receiving new parameters in DlgFilterWorker from GUI thread.

`#include <DlgFilterCommand.h>`

### Public Member Functions

- DlgFilterCommand (ColorFilterMode colorFilterMode, double low0To1, double high0To1)

  *Initial constructor.*
- DlgFilterCommand (const DlgFilterCommand &other)

  *Copy constructor.*
- DlgFilterCommand & operator= (const DlgFilterCommand &other)

  *Assignment operator.*
- ColorFilterMode colorFilterMode () const

  *Get method for filter mode.*
- double high0To1 () const

  *Get method for high value.*
- double low0To1 () const

  *Get method for low value.*

#### 4.105.1 Detailed Description

Command pattern object for receiving new parameters in DlgFilterWorker from GUI thread.

Definition at line 13 of file DlgFilterCommand.h.

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

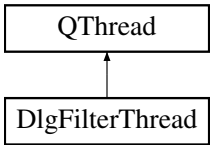
- Dlg/DlgFilterCommand.h
- Dlg/DlgFilterCommand.cpp

## 4.106 DlgFilterThread Class Reference

Class for processing new filter settings. This is based on http://blog.debao.me/2013/08/how-to-use-qthread-in-

`#include <DlgFilterThread.h>`

Inheritance diagram for DlgFilterThread:




### Signals

- void signalTransferPiece (int xLeft, QImage image)

  *Send a processed vertical piece of the original pixmap. The destination is between xLeft and xLeft+pixmap.width()*

### Public Member Functions

- DlgFilterThread (const QPixmap &pixmapOriginal, QRgb rgbBackground, DlgSettingsColorFilter &dlg↩ SettingsColorFilter)

  *Single constructor.*
- virtual void run ()

  *Run this thread.*

#### 4.106.1 Detailed Description

Class for processing new filter settings. This is based on http://blog.debao.me/2013/08/how-to-use-qthread-in-

Definition at line 18 of file DlgFilterThread.h.

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

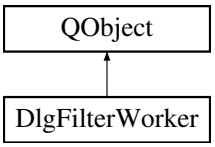
- Dlg/DlgFilterThread.h
- Dlg/DlgFilterThread.cpp

## 4.107 DlgFilterWorker Class Reference

Class for processing new filter settings. This is based on http://blog.debao.me/2013/08/how-to-use-qworker-in-

```
#include <DlgFilterWorker.h>
```

Inheritance diagram for DlgFilterWorker:




### Public Slots

- void slotNewParameters (ColorFilterMode colorFilterMode, double low, double high)

  *Start processing with a new set of parameters. Any ongoing processing is interrupted when m_filterMode changes.*

### Signals

- void signalTransferPiece (int xLeft, QImage image)

  *Send a processed vertical piece of the original pixmap. The destination is between xLeft and xLeft+pixmap.width()*

### Public Member Functions

- DlgFilterWorker (const QPixmap &pixmapOriginal, QRgb m_rgbBackground)

  *Single constructor.*

### 4.107.1 Detailed Description

Class for processing new filter settings. This is based on http://blog.debao.me/2013/08/how-to-use-qworker-in-

Definition at line 22 of file DlgFilterWorker.h.

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

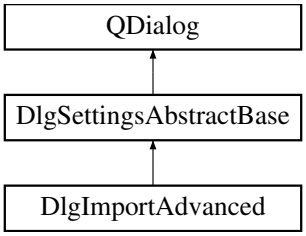
- Dlg/DlgFilterWorker.h
- Dlg/DlgFilterWorker.cpp

## 4.108 DlgImportAdvanced Class Reference

Dialog for setting the advanced parameters in a newly imported Document.

`#include <DlgImportAdvanced.h>`

Inheritance diagram for DlgImportAdvanced:




### Public Member Functions

- DlgImportAdvanced (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- DocumentAxesPointsRequired documentAxesPointsRequired () const

  *Number of axes points selected by user.*
- virtual void handleOk ()

  *Process slotOk.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- unsigned int numberCoordSystem () const

  *Number of coordinate systems selected by user.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Additional Inherited Members

#### 4.108.1 Detailed Description

Dialog for setting the advanced parameters in a newly imported Document.

Definition at line 19 of file DlgImportAdvanced.h.

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

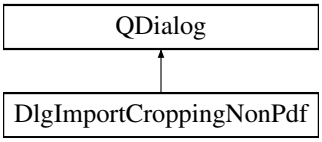
- Dlg/DlgImportAdvanced.h
- Dlg/DlgImportAdvanced.cpp

## 4.109 DlgImportCroppingNonPdf Class Reference

Dialog for selecting a page and frame on that page when importing an image from a non-pdf file.

`#include <DlgImportCroppingNonPdf.h>`

Inheritance diagram for DlgImportCroppingNonPdf:




### Public Member Functions

- DlgImportCroppingNonPdf (const QString &fileName)
  *Single constructor.*
- QImage image () const
  *Image that was selected. Value is null if loading failed.*
- virtual void showEvent (QShowEvent ∗event)
  *Do preparation before dialog is displayed.*

#### 4.109.1 Detailed Description

Dialog for selecting a page and frame on that page when importing an image from a non-pdf file.

Definition at line 24 of file DlgImportCroppingNonPdf.h.

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

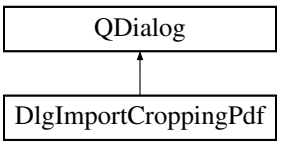
- Dlg/DlgImportCroppingNonPdf.h
- Dlg/DlgImportCroppingNonPdf.cpp

## 4.110 DlgImportCroppingPdf Class Reference

Dialog for selecting a page and frame on that page when importing an image from a pdf file.

`#include <DlgImportCroppingPdf.h>`

Inheritance diagram for DlgImportCroppingPdf:




### Public Member Functions

- DlgImportCroppingPdf (const Poppler::Document &document, int resolution)
  *Single constructor.*
- QImage image () const
  *Image that was selected. Value is null if loading failed.*
- virtual void showEvent (QShowEvent ∗event)
  *Do preparation before dialog is displayed.*

### 4.110.1 Detailed Description

Dialog for selecting a page and frame on that page when importing an image from a pdf file.

Definition at line 28 of file DlgImportCroppingPdf.h.

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

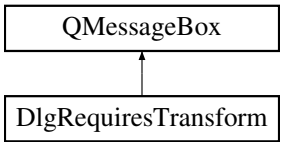
- Dlg/DlgImportCroppingPdf.h
- Dlg/DlgImportCroppingPdf.cpp

## 4.111 DlgRequiresTransform Class Reference

Dialog to be displayed whenever some operation or processing cannot be performed since the axis points are not defined.

`#include <DlgRequiresTransform.h>`

Inheritance diagram for DlgRequiresTransform:



### Public Member Functions

- DlgRequiresTransform (const QString &context)
  *Single constructor.*

### 4.111.1 Detailed Description

Dialog to be displayed whenever some operation or processing cannot be performed since the axis points are not defined.

Definition at line 13 of file DlgRequiresTransform.h.

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

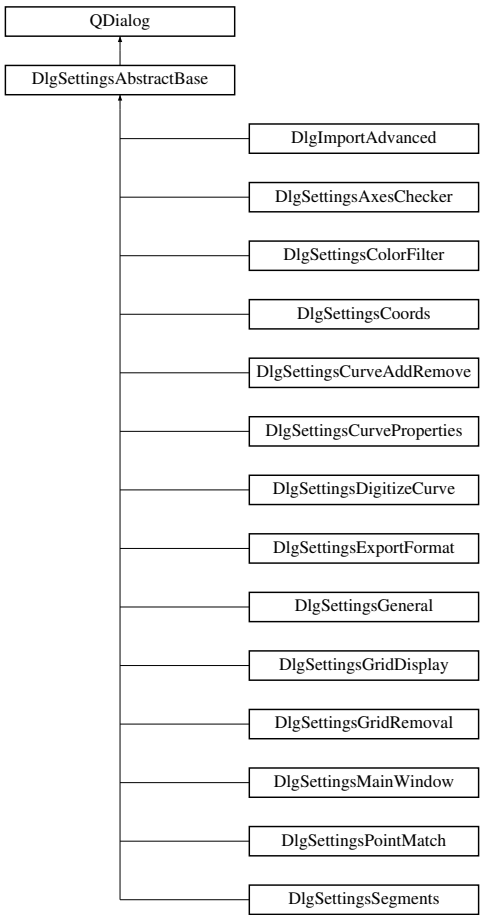
- Dlg/DlgRequiresTransform.h
- Dlg/DlgRequiresTransform.cpp

## 4.112 DlgSettingsAbstractBase Class Reference

Abstract base class for all Settings dialogs.

`#include <DlgSettingsAbstractBase.h>`

Inheritance diagram for DlgSettingsAbstractBase:



### Public Member Functions

- DlgSettingsAbstractBase (const QString &title, const QString &dialogName, MainWindow &mainWindow)

  *Single constructor.*

### Protected Member Functions

- CmdMediator & cmdMediator ()

  *Provide access to Document information wrapped inside CmdMediator.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)=0

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()=0

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- void enableOk (bool enable)

  *Let leaf subclass control the Ok button.*
- void finishPanel (QWidget *subPanel, int minimumWidth=MINIMUM_DIALOG_WIDTH, int minimumHeightOrZero=0)

  *Add Ok and Cancel buttons to subpanel to get the whole dialog.*
- virtual void handleOk ()=0

  *Process slotOk.*
- virtual void load (CmdMediator &cmdMediator)=0

  *Load settings from Document.*
- MainWindow & mainWindow ()

  *Get method for MainWindow.*
- const MainWindow & mainWindow () const

  *Const get method for MainWindow.*
- void populateColorComboWithoutTransparent (QComboBox &combo)

  *Add colors in color palette to combobox, without transparent entry at end.*
- void populateColorComboWithTransparent (QComboBox &combo)

  *Add colors in color palette to combobox, with transparent entry at end.*
- void setCmdMediator (CmdMediator &cmdMediator)

  *Store CmdMediator for easy access by the leaf class.*
- void setDisableOkAtStartup (bool disableOkAtStartup)

  *Override the default Ok button behavior applied in showEvent.*
- virtual void setSmallDialogs (bool smallDialogs)=0

  *If false then dialogs have a minimum size so all controls are visible.*

### Static Protected Attributes

- static int MINIMUM_DIALOG_WIDTH = 380

  *Dialog layout constant that guarantees every widget has sufficient room. Can be increased by finishPanel.*
- static int MINIMUM_PREVIEW_HEIGHT = 100

  *Dialog layout constant that guarantees preview has sufficent room.*

#### 4.112.1 Detailed Description

Abstract base class for all Settings dialogs.

Definition at line 20 of file DlgSettingsAbstractBase.h.

### 4.112.2 Member Function Documentation

#### 4.112.2.1 enableOk()

```
void DlgSettingsAbstractBase::enableOk (
            bool enable )  [protected]
```

Let leaf subclass control the Ok button.

This method is separate from the subclasses' updateControls, rather than part of that method since updateControls is not aware of when it is called at startup - at which point the ok button should ALWAYS be disabled since there are not yet any changes. In other words, we call this method at startup to override the ok button state that was just set by updateControls

Note - if this method is called with a constant value of true from updateControls, one of two cases applies: 1) There are no constraints to worry about (like a required text field cannot be empty) 2) There are constraints, but they are already handled by validators and/or other constraint logic

Definition at line 52 of file DlgSettingsAbstractBase.cpp.

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

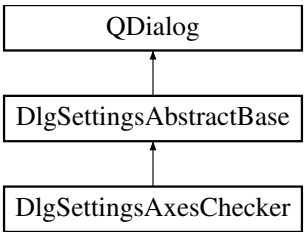
- Dlg/DlgSettingsAbstractBase.h
- Dlg/DlgSettingsAbstractBase.cpp

## 4.113 DlgSettingsAxesChecker Class Reference

Dialog for editing axes checker settings.

```
#include <DlgSettingsAxesChecker.h>
```

Inheritance diagram for DlgSettingsAxesChecker:




### Public Member Functions

- DlgSettingsAxesChecker (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.113.1 Detailed Description

Dialog for editing axes checker settings.

Definition at line 24 of file DlgSettingsAxesChecker.h.

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

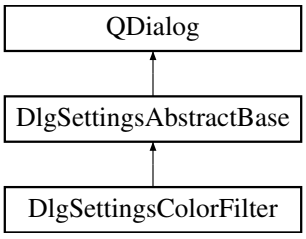
- Dlg/DlgSettingsAxesChecker.h
- Dlg/DlgSettingsAxesChecker.cpp

## 4.114 DlgSettingsColorFilter Class Reference

Dialog for editing filtering settings.

```
#include <DlgSettingsColorFilter.h>
```

Inheritance diagram for DlgSettingsColorFilter:




### Public Slots

- void slotTransferPiece (int xLeft, QImage image)

  *Receive processed piece of preview image, to be inserted at xLeft to xLeft+pixmap.width().*

### Signals

- void signalApplyFilter (ColorFilterMode colorFilterMode, double low, double high)

  *Send filter parameters to DlgFilterThread and DlgFilterWorker for processing.*

### Public Member Functions

- DlgSettingsColorFilter (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.114.1 Detailed Description

Dialog for editing filtering settings.

Definition at line 29 of file DlgSettingsColorFilter.h.

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

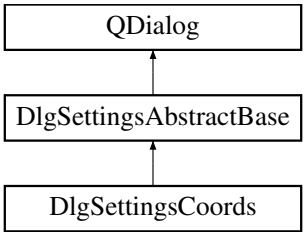
- Dlg/DlgSettingsColorFilter.h
- Dlg/DlgSettingsColorFilter.cpp

## 4.115 DlgSettingsCoords Class Reference

Dialog for editing coordinates settings.

```
#include <DlgSettingsCoords.h>
```

Inheritance diagram for DlgSettingsCoords:



### Public Member Functions

- DlgSettingsCoords (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.115.1 Detailed Description

Dialog for editing coordinates settings.

Definition at line 27 of file DlgSettingsCoords.h.

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

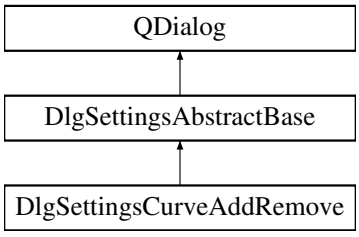
- Dlg/DlgSettingsCoords.h
- Dlg/DlgSettingsCoords.cpp

## 4.116 DlgSettingsCurveAddRemove Class Reference

Dialog for editing curve names settings.

```
#include <DlgSettingsCurveAddRemove.h>
```

Inheritance diagram for DlgSettingsCurveAddRemove:



### Public Slots

- void slotRowsAboutToBeRemoved (const QModelIndex &parent, int rowFirst, int rowLast)

  *Cleanup after rows have been removed in the model. We remove the corresponding rows in the QListView.*

### Public Member Functions

- DlgSettingsCurveAddRemove (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.116.1 Detailed Description

Dialog for editing curve names settings.

Definition at line 24 of file DlgSettingsCurveAddRemove.h.

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

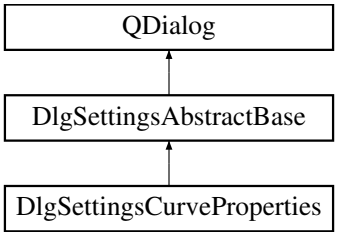
- Dlg/DlgSettingsCurveAddRemove.h
- Dlg/DlgSettingsCurveAddRemove.cpp

## 4.117 DlgSettingsCurveProperties Class Reference

Dialog for editing curve properties settings.

`#include <DlgSettingsCurveProperties.h>`

Inheritance diagram for DlgSettingsCurveProperties:



### Public Member Functions

- DlgSettingsCurveProperties (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- void setCurveName (const QString &curveName)

  *Load information for the specified curve name. When called externally, the load method must have been called first.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.117.1 Detailed Description

Dialog for editing curve properties settings.

Definition at line 23 of file DlgSettingsCurveProperties.h.

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

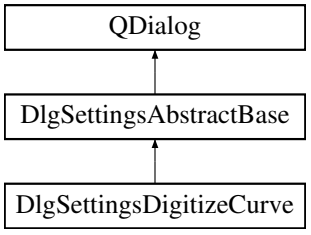
- Dlg/DlgSettingsCurveProperties.h
- Dlg/DlgSettingsCurveProperties.cpp

## 4.118 DlgSettingsDigitizeCurve Class Reference

Dialog for editing DigitizeStateCurve settings.

```
#include <DlgSettingsDigitizeCurve.h>
```

Inheritance diagram for DlgSettingsDigitizeCurve:



### Public Member Functions

- DlgSettingsDigitizeCurve (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.118.1 Detailed Description

Dialog for editing DigitizeStateCurve settings.

The preview window would should the selected cursor in the center, but there is no way to access the image of Qt::CrossCursor (QCursor::pixmap only works for custom cursors that were defined by a QPixmap)

Definition at line 26 of file DlgSettingsDigitizeCurve.h.

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

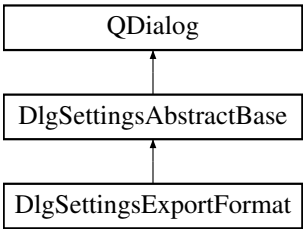
- Dlg/DlgSettingsDigitizeCurve.h
- Dlg/DlgSettingsDigitizeCurve.cpp

## 4.119 DlgSettingsExportFormat Class Reference

Dialog for editing exporting settings.

```
#include <DlgSettingsExportFormat.h>
```

Inheritance diagram for DlgSettingsExportFormat:



### Public Member Functions

- DlgSettingsExportFormat (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.119.1 Detailed Description

Dialog for editing exporting settings.

Definition at line 28 of file DlgSettingsExportFormat.h.

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

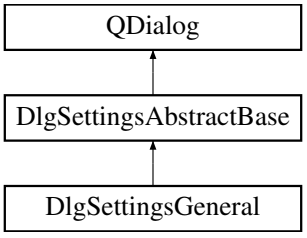
- Dlg/DlgSettingsExportFormat.h
- Dlg/DlgSettingsExportFormat.cpp

## 4.120 DlgSettingsGeneral Class Reference

Dialog for editing general settings.

```
#include <DlgSettingsGeneral.h>
```

Inheritance diagram for DlgSettingsGeneral:



### Public Member Functions

- DlgSettingsGeneral (MainWindow &mainWindow)
  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)
  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()
  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)
  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)
  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()
  *Process slotOk.*

### Additional Inherited Members

#### 4.120.1 Detailed Description

Dialog for editing general settings.

Definition at line 18 of file DlgSettingsGeneral.h.

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

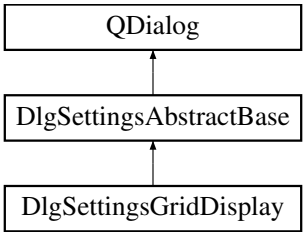
- Dlg/DlgSettingsGeneral.h
- Dlg/DlgSettingsGeneral.cpp

## 4.121 DlgSettingsGridDisplay Class Reference

Dialog for editing grid display settings.

```
#include <DlgSettingsGridDisplay.h>
```

Inheritance diagram for DlgSettingsGridDisplay:



### Public Member Functions

- DlgSettingsGridDisplay (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.121.1 Detailed Description

Dialog for editing grid display settings.

Definition at line 26 of file DlgSettingsGridDisplay.h.

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

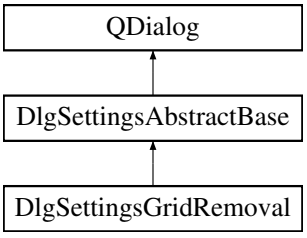
- Dlg/DlgSettingsGridDisplay.h
- Dlg/DlgSettingsGridDisplay.cpp

## 4.122 DlgSettingsGridRemoval Class Reference

Dialog for editing grid removal settings.

```
#include <DlgSettingsGridRemoval.h>
```

Inheritance diagram for DlgSettingsGridRemoval:



### Public Member Functions

- DlgSettingsGridRemoval (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.122.1 Detailed Description

Dialog for editing grid removal settings.

Definition at line 23 of file DlgSettingsGridRemoval.h.

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

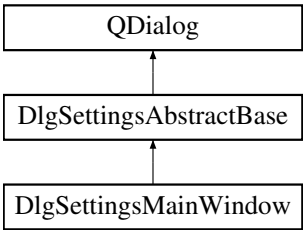
- Dlg/DlgSettingsGridRemoval.h
- Dlg/DlgSettingsGridRemoval.cpp

## 4.123 DlgSettingsMainWindow Class Reference

Dialog for editing main window settings, which are entirely independent of all documents.

```
#include <DlgSettingsMainWindow.h>
```

Inheritance diagram for DlgSettingsMainWindow:



### Public Member Functions

- DlgSettingsMainWindow (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- void loadMainWindowModel (CmdMediator &cmdMediator, const MainWindowModel &modelMainWindow)

  *Replaced load method since the main window settings are independent of document, unlike other DlgSettings* classes.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.123.1 Detailed Description

Dialog for editing main window settings, which are entirely independent of all documents.

Definition at line 22 of file DlgSettingsMainWindow.h.

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

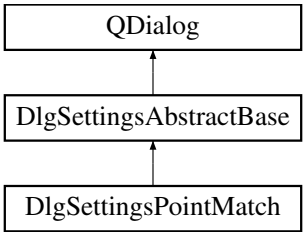
- Dlg/DlgSettingsMainWindow.h
- Dlg/DlgSettingsMainWindow.cpp

## 4.124 DlgSettingsPointMatch Class Reference

Dialog for editing point match settings, for DigitizeStatePointMatch.

`#include <DlgSettingsPointMatch.h>`

Inheritance diagram for DlgSettingsPointMatch:



### Public Member Functions

- DlgSettingsPointMatch (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.124.1 Detailed Description

Dialog for editing point match settings, for DigitizeStatePointMatch.

Definition at line 24 of file DlgSettingsPointMatch.h.

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

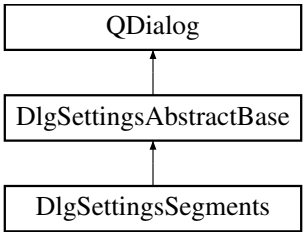
- Dlg/DlgSettingsPointMatch.h
- Dlg/DlgSettingsPointMatch.cpp

## 4.125 DlgSettingsSegments Class Reference

Dialog for editing Segments settings, for DigitizeStateSegment.

```
#include <DlgSettingsSegments.h>
```

Inheritance diagram for DlgSettingsSegments:



### Public Member Functions

- DlgSettingsSegments (MainWindow &mainWindow)

  *Single constructor.*
- virtual void createOptionalSaveDefault (QHBoxLayout *layout)

  *Let subclass define an optional Save As Default button.*
- virtual QWidget * createSubPanel ()

  *Create dialog-specific panel to which base class will add Ok and Cancel buttons.*
- virtual void load (CmdMediator &cmdMediator)

  *Load settings from Document.*
- virtual void setSmallDialogs (bool smallDialogs)

  *If false then dialogs have a minimum size so all controls are visible.*

### Protected Member Functions

- virtual void handleOk ()

  *Process slotOk.*

### Additional Inherited Members

#### 4.125.1 Detailed Description

Dialog for editing Segments settings, for DigitizeStateSegment.

Definition at line 27 of file DlgSettingsSegments.h.

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

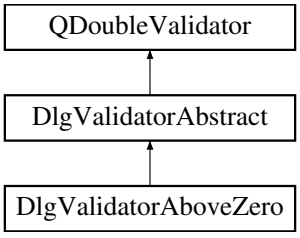
- Dlg/DlgSettingsSegments.h
- Dlg/DlgSettingsSegments.cpp

## 4.126 DlgValidatorAboveZero Class Reference

Validator for generic (=simple) numbers that must be greater than zero.

`#include <DlgValidatorAboveZero.h>`

Inheritance diagram for DlgValidatorAboveZero:



### Public Member Functions

- DlgValidatorAboveZero (const QLocale &locale, QObject ∗parent=0)
  *Single constructor.*
- virtual QValidator::State validate (QString &input, int &pos) const
  *Apply the standard validation with 0 as the exclusive minimum. Call setCoordScale just before calling this method.*

### 4.126.1 Detailed Description

Validator for generic (=simple) numbers that must be greater than zero.

Definition at line 14 of file DlgValidatorAboveZero.h.

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

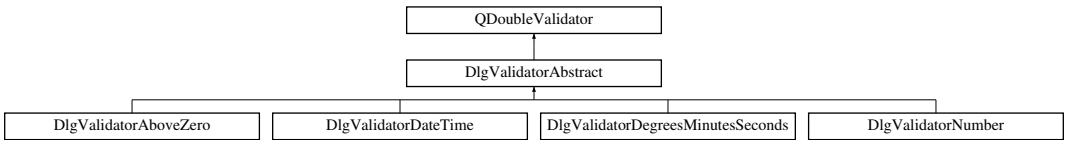
- Dlg/DlgValidatorAboveZero.h
- Dlg/DlgValidatorAboveZero.cpp

## 4.127 DlgValidatorAbstract Class Reference

Abstract validator for all numeric formats.

`#include <DlgValidatorAbstract.h>`

Inheritance diagram for DlgValidatorAbstract:




### Public Member Functions

- DlgValidatorAbstract (QObject ∗parent=0)
  *Single constructor.*
- virtual QValidator::State validate (QString &input, int &pos) const =0
  *Validate according to the numeric format specific to the leaf class.*

### 4.127.1 Detailed Description

Abstract validator for all numeric formats.

Definition at line 14 of file DlgValidatorAbstract.h.

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

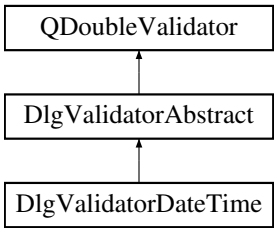
- Dlg/DlgValidatorAbstract.h
- Dlg/DlgValidatorAbstract.cpp

## 4.128 DlgValidatorDateTime Class Reference

Validator for numeric value expressed as date and/or time.

`#include <DlgValidatorDateTime.h>`

Inheritance diagram for DlgValidatorDateTime:




### Public Member Functions

- DlgValidatorDateTime (CoordScale coordScale, CoordUnitsDate coordUnitsDate, CoordUnitsTime coord↩ UnitsTime, QObject ∗parent=0)

  *Single constructor.*
- virtual QValidator::State validate (QString &input, int &pos) const

  *Validate according to the numeric format specific to the leaf class.*

### 4.128.1 Detailed Description

Validator for numeric value expressed as date and/or time.

Definition at line 16 of file DlgValidatorDateTime.h.

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

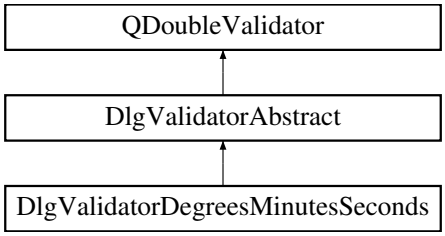
- Dlg/DlgValidatorDateTime.h
- Dlg/DlgValidatorDateTime.cpp

## 4.129 DlgValidatorDegreesMinutesSeconds Class Reference

Validator for angles in real degrees, integer degrees and real minutes, or integer degrees with integer minutes with real seconds.

`#include <DlgValidatorDegreesMinutesSeconds.h>`

Inheritance diagram for DlgValidatorDegreesMinutesSeconds:



### Public Member Functions

- DlgValidatorDegreesMinutesSeconds (CoordScale coordScale, QObject *parent=0)

  *Single constructor.*
- virtual QValidator::State validate (QString &input, int &pos) const

  *Validate according to the numeric format specific to the leaf class.*

#### 4.129.1 Detailed Description

Validator for angles in real degrees, integer degrees and real minutes, or integer degrees with integer minutes with real seconds.

Definition at line 17 of file DlgValidatorDegreesMinutesSeconds.h.

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

- Dlg/DlgValidatorDegreesMinutesSeconds.h
- Dlg/DlgValidatorDegreesMinutesSeconds.cpp

## 4.130 DlgValidatorFactory Class Reference

Validator factory.

`#include <DlgValidatorFactory.h>`

### Public Member Functions

- DlgValidatorFactory ()

  *Single constructor.*
- DlgValidatorAbstract * createAboveZero (const QLocale &locale) const

  *Factory method for generating validators for scale length which must be a number greater than zero.*
- DlgValidatorAbstract * createCartesianOrPolarWithNonPolarPolar (CoordScale coordScale, bool isCartesian, CoordUnitsNonPolarTheta coordUnitsCartesian, CoordUnitsNonPolarTheta coordUnitsPolar, CoordUnitsDate coordUnitsDate, CoordUnitsTime coordUnitsTime, const QLocale &locale) const

  *Factory method for generating validators for either cartesian or polar case, when polar format is specified by CoordUnitsNonPolarTheta.*
- DlgValidatorAbstract * createCartesianOrPolarWithPolarPolar (CoordScale coordScale, bool isCartesian, CoordUnitsNonPolarTheta coordUnitsCartesian, CoordUnitsPolarTheta coordUnitsPolar, CoordUnitsDate coordUnitsDate, CoordUnitsTime coordUnitsTime, const QLocale &locale) const

  *Factory method for generating validators for either cartesian or polar case, when polar format is specified by CoordUnitsPolarTheta.*
- DlgValidatorAbstract * createWithNonPolar (CoordScale coordScale, CoordUnitsNonPolarTheta coordUnits, CoordUnitsDate coordUnitsDate, CoordUnitsTime coordUnitsTime, const QLocale &locale) const

  *Factory method for generating validators when cartesian/polar case handling is handled externally, and format is specified by CoordUnitsNonPolarTheta.*
- DlgValidatorAbstract * createWithPolar (CoordScale coordScale, CoordUnitsPolarTheta coordUnits, const QLocale &locale) const

  *Factory method for generating validators when cartesian/polar case handling is handled externally, and format is specified by CoordUnitsNonPolarTheta.*

### 4.130.1 Detailed Description

Validator factory.

Definition at line 18 of file DlgValidatorFactory.h.

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

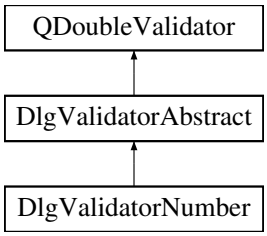
- Dlg/DlgValidatorFactory.h
- Dlg/DlgValidatorFactory.cpp

## 4.131 DlgValidatorNumber Class Reference

Validator for generic (=simple) numbers.

`#include <DlgValidatorNumber.h>`

Inheritance diagram for DlgValidatorNumber:




### Public Member Functions

- DlgValidatorNumber (CoordScale coordScale, const QLocale &locale, QObject ∗parent=0)
  *Single constructor.*
- virtual QValidator::State validate (QString &input, int &pos) const
  *Apply the standard validation with 0 as the exclusive minimum. Call setCoordScale just before calling this method.*

### 4.131.1 Detailed Description

Validator for generic (=simple) numbers.

Definition at line 17 of file DlgValidatorNumber.h.

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

- Dlg/DlgValidatorNumber.h
- Dlg/DlgValidatorNumber.cpp

## 4.132 Document Class Reference

Storage of one imported image and the data attached to that image.

`#include <Document.h>`

### Public Member Functions

- Document (const QImage &image)

  *Constructor for imported images and dragged images. Only one coordinate system is create - others are added later externally.*
- Document (const QString &fileName)

  *Constructor for opened Documents, and error report files. The specified file is opened and read.*
- void addCoordSystems (unsigned int numberCoordSystemToAdd)

  *Add some number (0 or more) of additional coordinate systems.*
- void addGraphCurveAtEnd (const QString &curveName)

  *Add new graph curve to the list of existing graph curves.*
- void addPointAxisWithGeneratedIdentifier (const QPointF &posScreen, const QPointF &posGraph, QString &identifier, double ordinal, bool isXOnly)

  *Add a single axis point with a generated point identifier.*
- void addPointAxisWithSpecifiedIdentifier (const QPointF &posScreen, const QPointF &posGraph, const QString &identifier, double ordinal, bool isXOnly)

  *Add a single axis point with the specified point identifier.*
- void addPointGraphWithGeneratedIdentifier (const QString &curveName, const QPointF &posScreen, QString &generatedIentifier, double ordinal)

  *Add a single graph point with a generated point identifier.*
- void addPointGraphWithSpecifiedIdentifier (const QString &curveName, const QPointF &posScreen, const QString &identifier, double ordinal)

  *Add a single graph point with the specified point identifer. Note that PointStyle is not applied to the point within the Document.*
- void addPointsInCurvesGraphs (CurvesGraphs &curvesGraphs)

  *Add all points identified in the specified CurvesGraphs. See also removePointsInCurvesGraphs.*
- void addScaleWithGeneratedIdentifier (const QPointF &posScreen0, const QPointF &posScreen1, double scaleLength, QString &identifier0, QString &identifier1, double ordinal0, double ordinal1)

  *Add scale with a generated point identifier.*
- void checkAddPointAxis (const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage, bool isXOnly)

  *Check before calling addPointAxis. Also returns the next available ordinal number (to prevent clashes)*
- void checkEditPointAxis (const QString &pointIdentifier, const QPointF &posScreen, const QPointF &posGraph, bool &isError, QString &errorMessage)

  *Check before calling editPointAxis.*
- const CoordSystem & coordSystem () const

  *Currently active CoordSystem.*
- unsigned int coordSystemCount () const

  *Number of CoordSystem.*
- CoordSystemIndex coordSystemIndex () const

  *Index of current active CoordSystem.*
- const Curve & curveAxes () const

  *Get method for axis curve.*
- const Curve * curveForCurveName (const QString &curveName) const

  *See CurvesGraphs::curveForCurveNames, although this also works for AXIS_CURVE_NAME.*

- const CurvesGraphs & curvesGraphs () const

  *Make all Curves available, read only, for CmdAbstract classes only.*
- QStringList curvesGraphsNames () const

  *See CurvesGraphs::curvesGraphsNames.*
- int curvesGraphsNumPoints (const QString &curveName) const

  *See CurvesGraphs::curvesGraphsNumPoints.*
- DocumentAxesPointsRequired documentAxesPointsRequired () const

  *Get method for DocumentAxesPointsRequired.*
- void editPointAxis (const QPointF &posGraph, const QString &identifier)

  *Edit the graph coordinates of a single axis point. Call this after checkAddPointAxis to guarantee success in this call.*
- void editPointGraph (bool isX, bool isY, double x, double y, const QStringList &identifiers, const Transformation &transformation)

  *Edit the graph coordinates of one or more graph points.*
- void initializeGridDisplay (const Transformation &transformation)

  *Initialize grid display. This is called immediately after the transformation has been defined for the first time.*
- bool isXOnly (const QString &pointIdentifier) const

  *See Curve::isXOnly.*
- void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for the axes curve.*
- void iterateThroughCurvePointsAxes (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurvePoints, for the axes curve.*
- void iterateThroughCurveSegments (const QString &curveName, const Functor2wRet< const Point &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurveSegments, for any axes or graph curve.*
- void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback)

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*
- void iterateThroughCurvesPointsGraphs (const Functor2wRet< const QString &, const Point &, CallbackSearchReturn > &ftorWithCallback) const

  *See Curve::iterateThroughCurvePoints, for all the graphs curves.*
- bool loadCurvesFile (const QString &curvesFile)

  *Load the curve names in the specified Engauge file into the current document. This is called near the end of the import process only.*
- DocumentModelAxesChecker modelAxesChecker () const

  *Get method for DocumentModelAxesChecker.*
- DocumentModelColorFilter modelColorFilter () const

  *Get method for DocumentModelColorFilter.*
- DocumentModelCoords modelCoords () const

  *Get method for DocumentModelCoords.*
- CurveStyles modelCurveStyles () const

  *Get method for CurveStyles.*
- DocumentModelDigitizeCurve modelDigitizeCurve () const

  *Get method for DocumentModelDigitizeCurve.*
- DocumentModelExportFormat modelExport () const

  *Get method for DocumentModelExportFormat.*
- DocumentModelGeneral modelGeneral () const

  *Get method for DocumentModelGeneral.*
- DocumentModelGridDisplay modelGridDisplay () const

  *Get method for DocumentModelGridDisplay.*
- DocumentModelGridRemoval modelGridRemoval () const

*Get method for DocumentModelGridRemoval.*

- DocumentModelPointMatch modelPointMatch () const

  *Get method for DocumentModelPointMatch.*
- DocumentModelSegments modelSegments () const

  *Get method for DocumentModelSegments.*
- void movePoint (const QString &pointIdentifier, const QPointF &deltaScreen)

  *See Curve::movePoint.*
- int nextOrdinalForCurve (const QString &curveName) const

  *Default next ordinal value for specified curve.*
- QPixmap pixmap () const

  *Return the image that is being digitized.*
- QPointF positionGraph (const QString &pointIdentifier) const

  *See Curve::positionGraph.*
- QPointF positionScreen (const QString &pointIdentifier) const

  *See Curve::positionScreen.*
- void print () const

  *Debugging method for printing directly from symbolic debugger.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- QString reasonForUnsuccessfulRead () const

  *Return an informative text message explaining why startup loading failed. Applies if successfulRead returns false.*
- void removePointAxis (const QString &identifier)

  *Perform the opposite of addPointAxis.*
- void removePointGraph (const QString &identifier)

  *Perform the opposite of addPointGraph.*
- void removePointsInCurvesGraphs (CurvesGraphs &curvesGraphs)

  *Remove all points identified in the specified CurvesGraphs. See also addPointsInCurvesGraphs.*
- void saveXml (QXmlStreamWriter &writer) const

  *Save document to xml.*
- QString selectedCurveName () const

  *Currently selected curve name. This is used to set the selected curve combobox in MainWindow.*
- void setCoordSystemIndex (CoordSystemIndex coordSystemIndex)

  *Set the index of current active CoordSystem.*
- void setCurveAxes (const Curve &curveAxes)

  *Let CmdAbstract classes overwrite axes Curve.*
- void setCurvesGraphs (const CurvesGraphs &curvesGraphs)

  *Let CmdAbstract classes overwrite CurvesGraphs.*
- void setDocumentAxesPointsRequired (DocumentAxesPointsRequired documentAxesPointsRequired)

  *Set the number of axes points required.*
- void setModelAxesChecker (const DocumentModelAxesChecker &modelAxesChecker)

  *Set method for DocumentModelAxesChecker.*
- void setModelColorFilter (const DocumentModelColorFilter &modelColorFilter)

  *Set method for DocumentModelColorFilter.*
- void setModelCoords (const DocumentModelCoords &modelCoords)

  *Set method for DocumentModelCoords.*
- void setModelCurveStyles (const CurveStyles &modelCurveStyles)

  *Set method for CurveStyles.*
- void setModelDigitizeCurve (const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Set method for DocumentModelDigitizeCurve.*
- void setModelExport (const DocumentModelExportFormat &modelExport)

  *Set method for DocumentModelExportFormat.*

- void setModelGeneral (const DocumentModelGeneral &modelGeneral)

  *Set method for DocumentModelGeneral.*
- void setModelGridDisplay (const DocumentModelGridDisplay &modelGridDisplay)

  *Set method for DocumentModelGridDisplay.*
- void setModelGridRemoval (const DocumentModelGridRemoval &modelGridRemoval)

  *Set method for DocumentModelGridRemoval.*
- void setModelPointMatch (const DocumentModelPointMatch &modelPointMatch)

  *Set method for DocumentModelPointMatch.*
- void setModelSegments (const DocumentModelSegments &modelSegments)

  *Set method for DocumentModelSegments.*
- void setPixmap (const QImage &image)

  *Set method for the background pixmap.*
- void setSelectedCurveName (const QString &selectedCurveName)

  *Save curve name that is selected for the current coordinate system, for the next time the coordinate system reappears.*
- bool successfulRead () const

  *Return true if startup loading succeeded. If the loading failed then reasonForUnsuccessfulRed will explain why.*
- void updatePointOrdinals (const Transformation &transformation)

  *Update point ordinals after point addition/removal or dragging.*

### 4.132.1 Detailed Description

Storage of one imported image and the data attached to that image.

Definition at line 41 of file Document.h.

### 4.132.2 Member Function Documentation

#### 4.132.2.1 addCoordSystems()

```
void Document::addCoordSystems (
            unsigned int numberCoordSystemToAdd )
```

Add some number (0 or more) of additional coordinate systems.

This is only safe to call during import and before any changes have been made to the Document

Definition at line 147 of file Document.cpp.

#### 4.132.2.2 addPointAxisWithGeneratedIdentifier()

```
void Document::addPointAxisWithGeneratedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            QString & identifier,
            double ordinal,
            bool isXOnly )
```

Add a single axis point with a generated point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if point has only an x coordinate |

Definition at line 162 of file Document.cpp.

#### 4.132.2.3 addPointAxisWithSpecifiedIdentifier()

```
void Document::addPointAxisWithSpecifiedIdentifier (
            const QPointF & posScreen,
            const QPointF & posGraph,
            const QString & identifier,
            double ordinal,
            bool isXOnly )
```

Add a single axis point with the specified point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen* | Screen coordinates from QGraphicsView |
| *posGraph* | Graph coordiantes from user |
| *identifier* | Identifier for new axis point |
| *ordinal* | Unique, for curve, ordinal number |
| *isXOnly* | True if point has only an x coordinate |

Definition at line 177 of file Document.cpp.

#### 4.132.2.4 addScaleWithGeneratedIdentifier()

```
void Document::addScaleWithGeneratedIdentifier (
            const QPointF & posScreen0,
            const QPointF & posScreen1,
            double scaleLength,
            QString & identifier0,
            QString & identifier1,
            double ordinal0,
            double ordinal1 )
```

Add scale with a generated point identifier.

Call this after checkAddPointAxis to guarantee success in this call.

**Parameters**

| | |
|---|---|
| *posScreen0* | Screen coordinates of first point from QGraphicsView |
| *posScreen1* | Screen coordinates of second point from QGraphicsView |
| *scaleLength* | Scale bar length in graph coordinates |
| *identifier0* | Identifier for first new axis point |
| *identifier1* | Identifier for second new axis point |
| *ordinal0* | Unique, for curve, ordinal number of first point |
| *ordinal1* | Unique, for curve, ordinal number of second point |

Definition at line 225 of file Document.cpp.

#### 4.132.2.5 setDocumentAxesPointsRequired()

```
void Document::setDocumentAxesPointsRequired (
            DocumentAxesPointsRequired documentAxesPointsRequired )
```

Set the number of axes points required.

This is called during the Document creation process, after imported images have been previewed or loaded files have had at least some xml parsing

Definition at line 933 of file Document.cpp.

#### 4.132.2.6 updatePointOrdinals()

```
void Document::updatePointOrdinals (
            const Transformation & transformation )
```

Update point ordinals after point addition/removal or dragging.

See GraphicsScene::updatePointOrdinalsAfterDrag. Graph coordinates of point must be up to date

Definition at line 1060 of file Document.cpp.

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

- Document/Document.h
- Document/Document.cpp

## 4.133 DocumentHashGenerator Class Reference

Generates a DocumentHash value representing the state of the entire Document.

```
#include <DocumentHashGenerator.h>
```

### Public Member Functions

- DocumentHashGenerator ()
  *Single constructor.*
- DocumentHash generate (const Document &document) const
  *Generate the hash for external storage.*

### 4.133.1 Detailed Description

Generates a DocumentHash value representing the state of the entire Document.

Definition at line 15 of file DocumentHashGenerator.h.

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

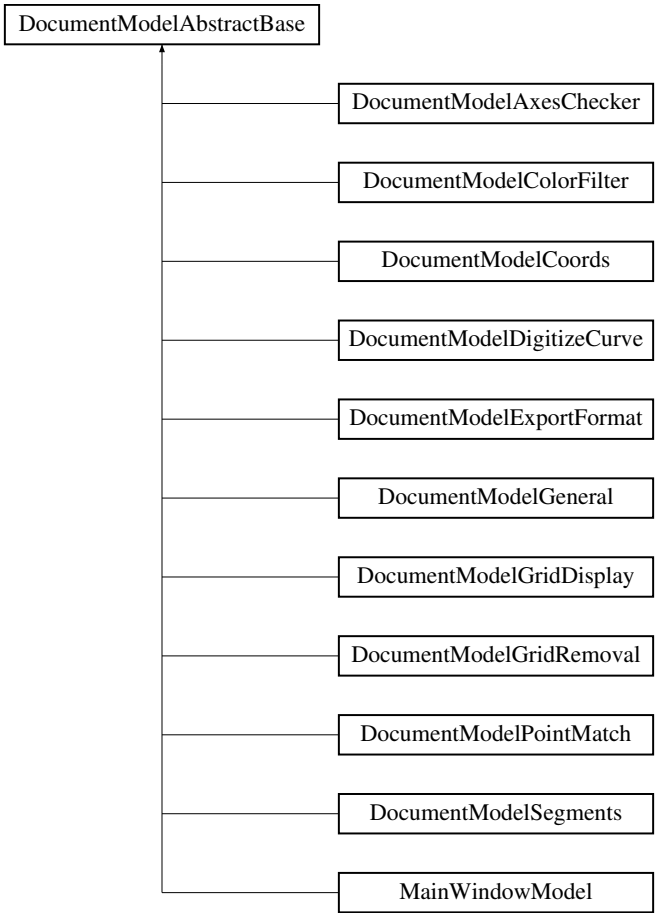
- Document/DocumentHashGenerator.h
- Document/DocumentHashGenerator.cpp

## 4.134 DocumentModelAbstractBase Class Reference

Abstract base class for document models. This class enforces a common interface for the leaf subclasses.

`#include <DocumentModelAbstractBase.h>`

Inheritance diagram for DocumentModelAbstractBase:



### Public Member Functions

- DocumentModelAbstractBase ()

  *Single constructor.*
- virtual ~DocumentModelAbstractBase ()

  *Single destructor.*

### Protected Member Functions

- virtual void loadXml (QXmlStreamReader &reader)=0

  *Load model from serialized xml.*
- virtual void saveXml (QXmlStreamWriter &writer) const =0

  *Save entire model as xml into stream.*

#### 4.134.1 Detailed Description

Abstract base class for document models. This class enforces a common interface for the leaf subclasses.

Definition at line 16 of file DocumentModelAbstractBase.h.

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

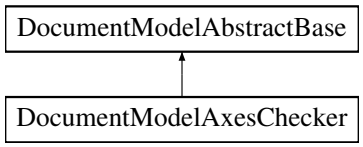
- Document/DocumentModelAbstractBase.h
- Document/DocumentModelAbstractBase.cpp

## 4.135 DocumentModelAxesChecker Class Reference

Model for DlgSettingsAxesChecker and CmdSettingsAxesChecker.

```
#include <DocumentModelAxesChecker.h>
```

Inheritance diagram for DocumentModelAxesChecker:



### Public Member Functions

- DocumentModelAxesChecker ()
  *Default constructor.*
- DocumentModelAxesChecker (const Document &document)
  *Initial constructor from Document.*
- DocumentModelAxesChecker (const DocumentModelAxesChecker &other)
  *Copy constructor.*
- DocumentModelAxesChecker & operator= (const DocumentModelAxesChecker &other)
  *Assignment constructor.*
- CheckerMode checkerMode () const
  *Get method for checker lifetime mode.*
- int checkerSeconds () const
  *Get method for checker lifetime in seconds.*
- ColorPalette lineColor () const
  *Get method for line color.*
- virtual void loadXml (QXmlStreamReader &reader)
  *Load model from serialized xml.*
- void printStream (QString indentation, QTextStream &str) const
  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const
  *Save entire model as xml into stream.*
- void setCheckerMode (CheckerMode checkerMode)
  *Set method for checker mode.*
- void setCheckerSeconds (int seconds)
  *Set method for checker lifetime in seconds.*
- void setLineColor (ColorPalette lineColor)
  *Set method for line color.*

### Additional Inherited Members

#### 4.135.1 Detailed Description

Model for DlgSettingsAxesChecker and CmdSettingsAxesChecker.

Definition at line 18 of file DocumentModelAxesChecker.h.

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

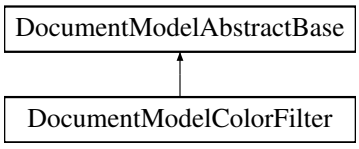
- Document/DocumentModelAxesChecker.h
- Document/DocumentModelAxesChecker.cpp

## 4.136 DocumentModelColorFilter Class Reference

Model for DlgSettingsColorFilter and CmdSettingsColorFilter.

`#include <DocumentModelColorFilter.h>`

Inheritance diagram for DocumentModelColorFilter:



## Public Member Functions

- DocumentModelColorFilter ()

  *Default constructor.*
- DocumentModelColorFilter (const DocumentModelColorFilter &other)

  *Copy constructor.*
- DocumentModelColorFilter (const CoordSystem &coordSystem)

  *Initial constructor from CoordSystem.*
- DocumentModelColorFilter & operator= (const DocumentModelColorFilter &other)

  *Assignment constructor.*
- ColorFilterMode colorFilterMode (const QString &curveName) const

  *Get method for filter mode.*
- const ColorFilterSettings colorFilterSettings (const QString &curveName) const

  *Get method for copying one color filter. Cannot return just a reference or else there is a warning about returning reference to temporary.*
- const ColorFilterSettingsList & colorFilterSettingsList () const

  *Get method for copying all color filters in one step.*
- int foregroundHigh (const QString &curveName) const

  *Get method for foreground higher bound.*
- int foregroundLow (const QString &curveName) const

  *Get method for foreground lower bound.*
- double high (const QString &curveName) const

  *High value of foreground, hue, intensity, saturation or value according to current filter mode.*
- int hueHigh (const QString &curveName) const

  *Get method for hue higher bound.*
- int hueLow (const QString &curveName) const

  *Get method for hue lower bound.*
- int intensityHigh (const QString &curveName) const

  *Get method for intensity higher bound.*
- int intensityLow (const QString &curveName) const

  *Get method for intensity lower bound.*
- virtual void loadXml (QXmlStreamReader &reader)

  *Load model from serialized xml.*
- double low (const QString &curveName) const

  *Low value of foreground, hue, intensity, saturation or value according to current filter mode normalized to 0 to 1.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- int saturationHigh (const QString &curveName) const

  *Get method for saturation higher bound.*
- int saturationLow (const QString &curveName) const

  *Get method for saturation lower bound.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save entire model as xml into stream.*
- void setColorFilterMode (const QString &curveName, ColorFilterMode colorFilterMode)

  *Set method for filter mode.*
- void setForegroundHigh (const QString &curveName, int foregroundHigh)

  *Set method for foreground higher bound.*
- void setForegroundLow (const QString &curveName, int foregroundLow)

  *Set method for foreground lower bound.*
- void setHigh (const QString &curveName, double s0To1)

  *Set the high value for the current filter mode.*

- void setHueHigh (const QString &curveName, int hueHigh)

  *Set method for hue higher bound.*
- void setHueLow (const QString &curveName, int hueLow)

  *Set method for hue lower bound.*
- void setIntensityHigh (const QString &curveName, int intensityHigh)

  *Set method for intensity higher bound.*
- void setIntensityLow (const QString &curveName, int intensityLow)

  *Set method for intensity lower bound.*
- void setLow (const QString &curveName, double s0To1)

  *Set the low value for the current filter mode.*
- void setSaturationHigh (const QString &curveName, int saturationHigh)

  *Set method for saturation high.*
- void setSaturationLow (const QString &curveName, int saturationLow)

  *Set method for saturation low.*
- void setValueHigh (const QString &curveName, int valueHigh)

  *Set method for value high.*
- void setValueLow (const QString &curveName, int valueLow)

  *Set method for value low.*
- int valueHigh (const QString &curveName) const

  *Get method for value high.*
- int valueLow (const QString &curveName) const

  *Get method for value low.*

## Additional Inherited Members

### 4.136.1 Detailed Description

Model for DlgSettingsColorFilter and CmdSettingsColorFilter.

Definition at line 21 of file DocumentModelColorFilter.h.

### 4.136.2 Member Function Documentation

#### 4.136.2.1 high()

```
double DocumentModelColorFilter::high (
            const QString & curveName ) const
```

High value of foreground, hue, intensity, saturation or value according to current filter mode.

normalized to 0 to 1.

Definition at line 105 of file DocumentModelColorFilter.cpp.

#### 4.136.2.2 low()

```
double DocumentModelColorFilter::low (
            const QString & curveName ) const
```

Low value of foreground, hue, intensity, saturation or value according to current filter mode normalized to 0 to 1.

Definition at line 212 of file DocumentModelColorFilter.cpp.

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

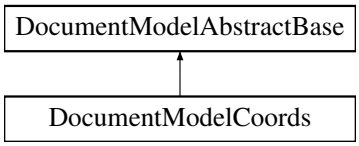
- Document/DocumentModelColorFilter.h
- Document/DocumentModelColorFilter.cpp

## 4.137 DocumentModelCoords Class Reference

Model for DlgSettingsCoords and CmdSettingsCoords.

```
#include <DocumentModelCoords.h>
```

Inheritance diagram for DocumentModelCoords:




### Public Member Functions

- DocumentModelCoords ()
  *Default constructor.*
- DocumentModelCoords (const Document &document)
  *Initial constructor from Document.*
- DocumentModelCoords (const DocumentModelCoords &other)
  *Copy constructor.*
- DocumentModelCoords & operator= (const DocumentModelCoords &other)
  *Assignment constructor.*
- CoordScale coordScaleXTheta () const
  *Get method for linear/log scale on x/theta.*
- CoordScale coordScaleYRadius () const
  *Get method for linear/log scale on y/radius.*
- CoordsType coordsType () const
  *Get method for coordinates type.*
- CoordUnitsDate coordUnitsDate () const
  *Get method for date format when used.*
- CoordUnitsNonPolarTheta coordUnitsRadius () const
  *Get method for radius units.*
- CoordUnitsPolarTheta coordUnitsTheta () const
  *Get method for theta unit.*

- CoordUnitsTime coordUnitsTime () const

  *Get method for time format when used.*
- CoordUnitsNonPolarTheta coordUnitsX () const

  *Get method for x units.*
- CoordUnitsNonPolarTheta coordUnitsY () const

  *Get method for x units.*
- virtual void loadXml (QXmlStreamReader &reader)

  *Load model from serialized xml.*
- double originRadius () const

  *Get method for origin radius in polar mode.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save entire model as xml into stream.*
- void setCoordScaleXTheta (CoordScale coordScale)

  *Set method for linear/log scale on x/theta.*
- void setCoordScaleYRadius (CoordScale coordScale)

  *Set method for linear/log scale on y/radius.*
- void setCoordsType (CoordsType coordsType)

  *Set method for coordinates type.*
- void setCoordUnitsDate (CoordUnitsDate coordUnits)

  *Set method for date units.*
- void setCoordUnitsRadius (CoordUnitsNonPolarTheta coordUnits)

  *Set method for radius units.*
- void setCoordUnitsTheta (CoordUnitsPolarTheta coordUnits)

  *Set method for theta units.*
- void setCoordUnitsTime (CoordUnitsTime coordUnits)

  *Set method for time units.*
- void setCoordUnitsX (CoordUnitsNonPolarTheta coordUnits)

  *Set method for x units.*
- void setCoordUnitsY (CoordUnitsNonPolarTheta coordUnits)

  *Set method for y units.*
- void setOriginRadius (double originRadius)

  *Set method for origin radius in polar mode.*
- double thetaPeriod () const

  *Return the period of the theta value for polar coordinates, consistent with CoordThetaUnits.*

### Additional Inherited Members

### 4.137.1 Detailed Description

Model for DlgSettingsCoords and CmdSettingsCoords.

Definition at line 20 of file DocumentModelCoords.h.

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

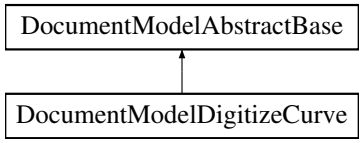
- Document/DocumentModelCoords.h
- Document/DocumentModelCoords.cpp

## 4.138 DocumentModelDigitizeCurve Class Reference

Model for DlgSettingsDigitizeCurve and CmdSettingsDigitizeCurve.

`#include <DocumentModelDigitizeCurve.h>`

Inheritance diagram for DocumentModelDigitizeCurve:




### Public Member Functions

- DocumentModelDigitizeCurve ()
  *Default constructor.*
- DocumentModelDigitizeCurve (const Document &document)
  *Initial constructor from Document.*
- DocumentModelDigitizeCurve (const DocumentModelDigitizeCurve &other)
  *Copy constructor.*
- DocumentModelDigitizeCurve & operator= (const DocumentModelDigitizeCurve &other)
  *Assignment constructor.*
- int cursorInnerRadius () const
  *Get method for cursor inner radius.*
- int cursorLineWidth () const
  *Get method for cursor line width.*
- CursorSize cursorSize () const
  *Get method for cursor size.*
- bool cursorStandardCross () const
  *Get method for cursor type.*
- virtual void loadXml (QXmlStreamReader &reader)
  *Load model from serialized xml.*
- void printStream (QString indentation, QTextStream &str) const
  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const
  *Save entire model as xml into stream.*
- void setCursorInnerRadius (int innerRadius)
  *Set method for cursor inner radius.*
- void setCursorLineWidth (int lineWidth)
  *Set method for cursor line width.*
- void setCursorSize (CursorSize cursorSize)
  *Set method for cursor size.*
- void setCursorStandardCross (bool cursorStandardCross)
  *Set method for cursor type.*

### Additional Inherited Members

#### 4.138.1 Detailed Description

Model for DlgSettingsDigitizeCurve and CmdSettingsDigitizeCurve.

No color is involved because the documentation in QCursor suggests that not all platforms support colored cursors

Definition at line 18 of file DocumentModelDigitizeCurve.h.

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

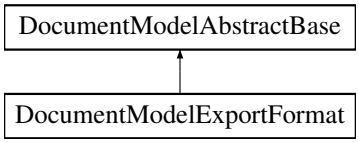
- Document/DocumentModelDigitizeCurve.h
- Document/DocumentModelDigitizeCurve.cpp

## 4.139 DocumentModelExportFormat Class Reference

Model for DlgSettingsExportFormat and CmdSettingsExportFormat.

`#include <DocumentModelExportFormat.h>`

Inheritance diagram for DocumentModelExportFormat:




### Public Member Functions

- DocumentModelExportFormat ()
  *Default constructor.*
- DocumentModelExportFormat (const Document &document)
  *Initial constructor from Document.*
- DocumentModelExportFormat (const DocumentModelExportFormat &other)
  *Copy constructor.*
- DocumentModelExportFormat & operator= (const DocumentModelExportFormat &other)
  *Assignment constructor.*
- QStringList curveNamesNotExported () const
  *Get method for curve names not exported.*
- ExportDelimiter delimiter () const
  *Get method for delimiter.*
- ExportHeader header () const
  *Get method for header.*
- ExportLayoutFunctions layoutFunctions () const
  *Get method for functions layout.*
- virtual void loadXml (QXmlStreamReader &reader)
  *Load model from serialized xml.*

- bool overrideCsvTsv () const

  *Get method for csv/tsv format override.*
- double pointsIntervalFunctions () const

  *Get method for points interval for functions.*
- double pointsIntervalRelations () const

  *Get method for relations interval for relations.*
- ExportPointsIntervalUnits pointsIntervalUnitsFunctions () const

  *Get method for points interval units for functions.*
- ExportPointsIntervalUnits pointsIntervalUnitsRelations () const

  *Get method for points interval units for relations.*
- ExportPointsSelectionFunctions pointsSelectionFunctions () const

  *Get method for point selection for functions.*
- ExportPointsSelectionRelations pointsSelectionRelations () const

  *Get method for point selection for relations.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save entire model as xml into stream.*
- void setCurveNamesNotExported (const QStringList &curveNamesNotExported)

  *Set method for curve names not exported.*
- void setDelimiter (ExportDelimiter exportDelimiter)

  *Set method for delimiter.*
- void setHeader (ExportHeader exportHeader)

  *Set method for header.*
- void setLayoutFunctions (ExportLayoutFunctions exportLayoutFunctions)

  *Set method for functions layout.*
- void setOverrideCsvTsv (bool overrideCsvTsv)

  *Set method for csv/tsv format override.*
- void setPointsIntervalFunctions (double pointsIntervalFunctions)

  *Set method for points interval for functions.*
- void setPointsIntervalRelations (double pointsIntervalRelations)

  *Set method for relations interval for relations.*
- void setPointsIntervalUnitsFunctions (ExportPointsIntervalUnits pointsIntervalUnitsFunctions)

  *Set method for points interval units for functions.*
- void setPointsIntervalUnitsRelations (ExportPointsIntervalUnits pointsIntervalUnitsRelations)

  *Set method for points interval units for relations.*
- void setPointsSelectionFunctions (ExportPointsSelectionFunctions exportPointsSelectionFunctions)

  *Set method for point selection for functions.*
- void setPointsSelectionRelations (ExportPointsSelectionRelations exportPointsSelectionRelations)

  *Set method for point selection for relations.*
- void setXLabel (const QString &xLabel)

  *Set method for x label.*
- QString xLabel () const

  *Get method for x label.*

### Additional Inherited Members

#### 4.139.1 Detailed Description

Model for DlgSettingsExportFormat and CmdSettingsExportFormat.

Definition at line 23 of file DocumentModelExportFormat.h.

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

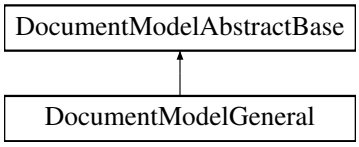
- Document/DocumentModelExportFormat.h
- Document/DocumentModelExportFormat.cpp

## 4.140 DocumentModelGeneral Class Reference

Model for DlgSettingsGeneral and CmdSettingsGeneral.

`#include <DocumentModelGeneral.h>`

Inheritance diagram for DocumentModelGeneral:




### Public Member Functions

- DocumentModelGeneral ()
  *Default constructor.*
- DocumentModelGeneral (const Document &document)
  *Initial constructor from Document.*
- DocumentModelGeneral (const DocumentModelGeneral &other)
  *Copy constructor.*
- DocumentModelGeneral & operator= (const DocumentModelGeneral &other)
  *Assignment constructor.*
- int cursorSize () const
  *Get method for effective cursor size.*
- int extraPrecision () const
  *Get method for extra digits of precsion.*
- virtual void loadXml (QXmlStreamReader &reader)
  *Load model from serialized xml.*
- void printStream (QString indentation, QTextStream &str) const
  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const
  *Save entire model as xml into stream.*
- void setCursorSize (int cursorSize)
  *Set method for effective cursor size.*
- void setExtraPrecision (int extraPrecision)
  *Set method for extra digits of precision.*

### Additional Inherited Members

#### 4.140.1 Detailed Description

Model for DlgSettingsGeneral and CmdSettingsGeneral.

Definition at line 16 of file DocumentModelGeneral.h.

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

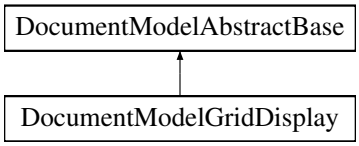
- Document/DocumentModelGeneral.h
- Document/DocumentModelGeneral.cpp

## 4.141 DocumentModelGridDisplay Class Reference

Model for DlgSettingsGridDisplay and CmdSettingsGridDisplay.

`#include <DocumentModelGridDisplay.h>`

Inheritance diagram for DocumentModelGridDisplay:




### Public Member Functions

- DocumentModelGridDisplay ()
  *Default constructor.*
- DocumentModelGridDisplay (const Document &document)
  *Initial constructor from Document.*
- DocumentModelGridDisplay (const DocumentModelGridDisplay &other)
  *Copy constructor.*
- DocumentModelGridDisplay & operator= (const DocumentModelGridDisplay &other)
  *Assignment constructor.*
- unsigned int countX () const
  *Get method for x grid line count.*
- unsigned int countY () const
  *Get method for y grid line count.*
- GridCoordDisable disableX () const
  *Get method for x grid line disabled variable.*
- GridCoordDisable disableY () const
  *Get method for y grid line disabled variable.*
- virtual void loadXml (QXmlStreamReader &reader)
  *Load model from serialized xml.*
- ColorPalette paletteColor () const
  *Get method for color.*

- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save entire model as xml into stream.*
- void setCountX (unsigned int countX)

  *Set method for x grid line count.*
- void setCountY (unsigned int countY)

  *Set method for y grid line count.*
- void setDisableX (GridCoordDisable disableX)

  *Set method for x grid line disabled variable.*
- void setDisableY (GridCoordDisable disableY)

  *Set method for y grid line disabled variable.*
- void setPaletteColor (ColorPalette paletteColor)

  *Set method for color.*
- void setStable (bool stable)

  *Set method for stable flag.*
- void setStartX (double startX)

  *Set method for x grid line lower bound (inclusive).*
- void setStartY (double yStart)

  *Set method for y grid line lower bound (inclusive).*
- void setStepX (double stepX)

  *Set method for x grid line increment.*
- void setStepY (double yStep)

  *Set method for y grid line increment.*
- void setStopX (double stopX)

  *Set method for x grid line upper bound (inclusive).*
- void setStopY (double yStop)

  *Set method for y grid line upper bound (inclusive).*
- bool stable () const

  *Get method for stable flag.*
- double startX () const

  *Get method for x grid line lower bound (inclusive).*
- double startY () const

  *Get method for y grid line lower bound (inclusive).*
- double stepX () const

  *Get method for x grid line increment.*
- double stepY () const

  *Get method for y grid line increment.*
- double stopX () const

  *Get method for x grid line upper bound (inclusive).*
- double stopY () const

  *Get method for y grid line upper bound (inclusive).*

### Additional Inherited Members

### 4.141.1 Detailed Description

Model for DlgSettingsGridDisplay and CmdSettingsGridDisplay.

Definition at line 18 of file DocumentModelGridDisplay.h.

### 4.141.2 Member Function Documentation

#### 4.141.2.1 stable()

```
bool DocumentModelGridDisplay::stable ( ) const
```

Get method for stable flag.

The flag is false to let the settings get automatically updated, until the user selects settings - at which point the stable flag is set to true

Definition at line 268 of file DocumentModelGridDisplay.cpp.

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

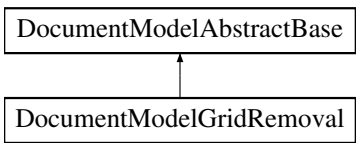
- Document/DocumentModelGridDisplay.h
- Document/DocumentModelGridDisplay.cpp

## 4.142 DocumentModelGridRemoval Class Reference

Model for DlgSettingsGridRemoval and CmdSettingsGridRemoval. The settings are unstable until the user approves.

```
#include <DocumentModelGridRemoval.h>
```

Inheritance diagram for DocumentModelGridRemoval:




### Public Member Functions

- DocumentModelGridRemoval ()
  *Default constructor.*
- DocumentModelGridRemoval (double startX, double startY, double stepX, double stepY, int countX, int countY)
  *Constructor fed by GridClassifier.*
- DocumentModelGridRemoval (const Document &document)
  *Initial constructor from Document.*
- DocumentModelGridRemoval (const DocumentModelGridRemoval &other)
  *Copy constructor.*
- DocumentModelGridRemoval & operator= (const DocumentModelGridRemoval &other)
  *Assignment constructor.*
- double closeDistance () const

*Get method for close distance.*

- int countX () const

  *Get method for x count.*
- int countY () const

  *Get method for y count.*
- GridCoordDisable gridCoordDisableX () const

  *Get method for x coord parameter to disable.*
- GridCoordDisable gridCoordDisableY () const

  *Get method for y coord parameter to disable.*
- virtual void loadXml (QXmlStreamReader &reader)

  *Load model from serialized xml.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- bool removeDefinedGridLines () const

  *Get method for removing defined grid lines.*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save entire model as xml into stream.*
- void setCloseDistance (double closeDistance)

  *Set method for close distance.*
- void setCountX (int countX)

  *Set method for x count.*
- void setCountY (int countY)

  *Set method for y count.*
- void setGridCoordDisableX (GridCoordDisable gridCoordDisable)

  *Set method for x coord parameter to disable.*
- void setGridCoordDisableY (GridCoordDisable gridCoordDisable)

  *Set method for y coord parameter to disable.*
- void setRemoveDefinedGridLines (bool removeDefinedGridLines)

  *Set method for removing defined grid lines.*
- void setStable ()

  *Set the stable flag to true. This public version has no argument since it cannot be undone.*
- void setStartX (double startX)

  *Set method for x start.*
- void setStartY (double startY)

  *Set method for y start.*
- void setStepX (double stepX)

  *Set method for x step.*
- void setStepY (double stepY)

  *Set method for y step.*
- void setStopX (double stopX)

  *Set method for x stop.*
- void setStopY (double stopY)

  *Set method for y stop.*
- bool stable () const

  *Get method for stable flag.*
- double startX () const

  *Get method for x start.*
- double startY () const

  *Get method for y start.*
- double stepX () const

  *Get method for x step.*

- double stepY () const
  *Get method for y step.*
- double stopX () const
  *Get method for x stop.*
- double stopY () const
  *Get method for y stop.*

### Additional Inherited Members

### 4.142.1 Detailed Description

Model for DlgSettingsGridRemoval and CmdSettingsGridRemoval. The settings are unstable until the user approves.

Definition at line 17 of file DocumentModelGridRemoval.h.

### 4.142.2 Member Function Documentation

#### 4.142.2.1 stable()

```
bool DocumentModelGridRemoval::stable ( ) const
```

Get method for stable flag.

The flag is false to let the settings get automatically updated, until the user selects settings - at which point the stable flag is set to true

Definition at line 320 of file DocumentModelGridRemoval.cpp.

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

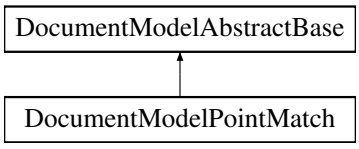
- Document/DocumentModelGridRemoval.h
- Document/DocumentModelGridRemoval.cpp

## 4.143 DocumentModelPointMatch Class Reference

Model for DlgSettingsPointMatch and CmdSettingsPointMatch.

```
#include <DocumentModelPointMatch.h>
```

Inheritance diagram for DocumentModelPointMatch:



### Public Member Functions

- DocumentModelPointMatch ()

  *Default constructor.*
- DocumentModelPointMatch (const Document &document)

  *Initial constructor from Document.*
- DocumentModelPointMatch (const DocumentModelPointMatch &other)

  *Copy constructor.*
- DocumentModelPointMatch & operator= (const DocumentModelPointMatch &other)

  *Assignment constructor.*
- virtual void loadXml (QXmlStreamReader &reader)

  *Load model from serialized xml.*
- double maxPointSize () const

  *Get method for max point size.*
- ColorPalette paletteColorAccepted () const

  *Get method for accepted color.*
- ColorPalette paletteColorCandidate () const

  *Get method for candidate color.*
- ColorPalette paletteColorRejected () const

  *Get method for rejected color.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save entire model as xml into stream.*
- void setMaxPointSize (double maxPointSize)

  *Set method for max point size.*
- void setPaletteColorAccepted (ColorPalette paletteColorAccepted)

  *Set method for accepted color.*
- void setPaletteColorCandidate (ColorPalette paletteColorCandidate)

  *Set method for candidate color.*
- void setPaletteColorRejected (ColorPalette paletteColorRejected)

  *Set method for rejected color.*

### Additional Inherited Members

#### 4.143.1 Detailed Description

Model for DlgSettingsPointMatch and CmdSettingsPointMatch.

Definition at line 17 of file DocumentModelPointMatch.h.

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

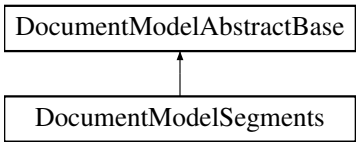
- Document/DocumentModelPointMatch.h
- Document/DocumentModelPointMatch.cpp

## 4.144 DocumentModelSegments Class Reference

Model for DlgSettingsSegments and CmdSettingsSegments.

`#include <DocumentModelSegments.h>`

Inheritance diagram for DocumentModelSegments:




### Public Member Functions

- DocumentModelSegments ()
  *Default constructor.*
- DocumentModelSegments (const Document &document)
  *Initial constructor from Document.*
- DocumentModelSegments (const DocumentModelSegments &other)
  *Copy constructor.*
- DocumentModelSegments & operator= (const DocumentModelSegments &other)
  *Assignment constructor.*
- bool fillCorners () const
  *Get method for fill corners.*
- ColorPalette lineColor () const
  *Get method for line color.*
- double lineWidth () const
  *Get method for line width.*
- virtual void loadXml (QXmlStreamReader &reader)
  *Load model from serialized xml.*
- double minLength () const
  *Get method for min length.*
- double pointSeparation () const
  *Get method for point separation.*
- void printStream (QString indentation, QTextStream &str) const
  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const
  *Save entire model as xml into stream.*
- void setFillCorners (bool fillCorners)
  *Set method for fill corners.*
- void setLineColor (ColorPalette lineColor)
  *Set method for line color.*
- void setLineWidth (double lineWidth)
  *Set method for line width.*
- void setMinLength (double minLength)
  *Set method for min length.*
- void setPointSeparation (double pointSeparation)
  *Set method for point separation.*

### Additional Inherited Members

#### 4.144.1 Detailed Description

Model for DlgSettingsSegments and CmdSettingsSegments.

Definition at line 17 of file DocumentModelSegments.h.

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

- Document/DocumentModelSegments.h
- Document/DocumentModelSegments.cpp

## 4.145 ExportAlignLinear Class Reference

Pick first simplest x value between specified min and max, for linear scaling.

```
#include <ExportAlignLinear.h>
```

### Public Member Functions

- ExportAlignLinear (double xMin, double xMax)
  *Single constructor.*
- double firstSimplestNumber () const
  *Result.*

#### 4.145.1 Detailed Description

Pick first simplest x value between specified min and max, for linear scaling.

A simplest value is defined here as one having the smallest number of significant digits, and is used for aligning periodic values on simple numbers. Examples:

1. 0.4 to 3.4, result is 1
2. 110 to 1100, result is 200 (not 1000 although both have same number of significant digits)
3. 112.123 to 122.456, result is 120

Definition at line 17 of file ExportAlignLinear.h.

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

- Export/ExportAlignLinear.h
- Export/ExportAlignLinear.cpp

## 4.146 ExportAlignLog Class Reference

Pick first simplest x value between specified min and max, for log scaling.

`#include <ExportAlignLog.h>`

### Public Member Functions

- ExportAlignLog (double xMin, double xMax)
  *Single constructor.*
- double firstSimplestNumber () const
  *Result.*

### 4.146.1 Detailed Description

Pick first simplest x value between specified min and max, for log scaling.

A simplest value is defined here as one having the smallest number of significant digits when log value is taken, and is used for aligning periodic values on simple numbers. Examples:

1. 0.9 to 2, result is 1 which is 10^0
2. 1.1 to 9, result is sqrt(10) which is midway between 1 and 10 in log scale, and equal to 10^0.5
3. 9.81 to 9.93, result is 10^0.992 since 9.81=10^0.99166 and 9.93=10^0.9969

Definition at line 17 of file ExportAlignLog.h.

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

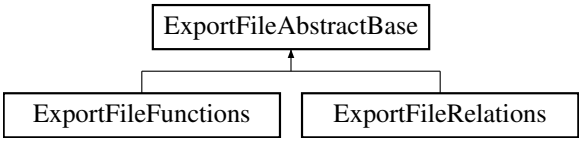
- Export/ExportAlignLog.h
- Export/ExportAlignLog.cpp

## 4.147 ExportFileAbstractBase Class Reference

Strategy base class for exporting to a file. This class provides common methods.

`#include <ExportFileAbstractBase.h>`

Inheritance diagram for ExportFileAbstractBase:



### Public Member Functions

- ExportFileAbstractBase ()

  *Single constructor.*

### Protected Member Functions

- QStringList curvesToInclude (const DocumentModelExportFormat &modelExportOverride, const Document &document, const QStringList &curvesGraphsNames, CurveConnectAs curveConnectAs1, CurveConnectAs curveConnectAs2) const

  *Identify curves to include in export. The specified DocumentModelExportFormat overrides same data in Document for previewing window.*
- void destroy2DArray (QVector< QVector< QString *> > &array) const

  *Deallocate memory for array.*
- QString gnuplotComment () const

  *Gnuplot comment delimiter.*
- void insertLineSeparator (bool isFirst, ExportHeader exportHeader, QTextStream &str) const

  *Insert line(s) between successive sets of curves.*
- QString wrapInDoubleQuotesIfNeeded (const DocumentModelExportFormat &modelExportOverride, const QString &valueString) const

  *RFC 4180 says if values are delimited by a comma AND a value has commas in it (for locale like English/Switzerland when dealing with numbers) then double quotes are required for the value.*

## 4.147.1 Detailed Description

Strategy base class for exporting to a file. This class provides common methods.

Definition at line 24 of file ExportFileAbstractBase.h.

## 4.147.2 Member Function Documentation

### 4.147.2.1 wrapInDoubleQuotesIfNeeded()

```
QString ExportFileAbstractBase::wrapInDoubleQuotesIfNeeded (
            const DocumentModelExportFormat & modelExportOverride,
            const QString & valueString ) const  [protected]
```

RFC 4180 says if values are delimited by a comma AND a value has commas in it (for locale like English/Switzerland when dealing with numbers) then double quotes are required for the value.

In other cases this method is a noop

Definition at line 89 of file ExportFileAbstractBase.cpp.

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

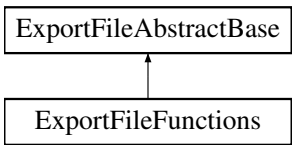
- Export/ExportFileAbstractBase.h
- Export/ExportFileAbstractBase.cpp

## 4.148 ExportFileFunctions Class Reference

Strategy class for exporting to a file. This strategy is external to the Document class so that class is simpler.

`#include <ExportFileFunctions.h>`

Inheritance diagram for ExportFileFunctions:




### Public Member Functions

- ExportFileFunctions ()

  *Single constructor.*
- void exportToFile (const DocumentModelExportFormat &modelExportOverride, const Document &document, const MainWindowModel &modelMainWindow, const Transformation &transformation, QTextStream &str, unsigned int &numWritesSoFar) const

  *Export Document points according to the settings.*

### Friends

- class **TestExport**

### Additional Inherited Members

### 4.148.1 Detailed Description

Strategy class for exporting to a file. This strategy is external to the Document class so that class is simpler.

Definition at line 24 of file ExportFileFunctions.h.

### 4.148.2 Member Function Documentation

#### 4.148.2.1 exportToFile()

```
void ExportFileFunctions::exportToFile (
             const DocumentModelExportFormat & modelExportOverride,
             const Document & document,
             const MainWindowModel & modelMainWindow,
             const Transformation & transformation,
             QTextStream & str,
             unsigned int & numWritesSoFar ) const
```

Export Document points according to the settings.

The DocumentModelExportFormat inside the Document is ignored so DlgSettingsExport can supply its own DocumentModelExportFormat when previewing what would be exported.

Definition at line 127 of file ExportFileFunctions.cpp.

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

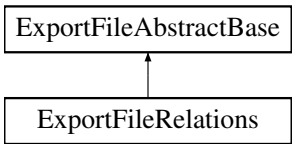
- Export/ExportFileFunctions.h
- Export/ExportFileFunctions.cpp

## 4.149 ExportFileRelations Class Reference

Strategy class for exporting to a file. This strategy is external to the Document class so that class is simpler.

```
#include <ExportFileRelations.h>
```

Inheritance diagram for ExportFileRelations:




### Public Member Functions

- ExportFileRelations ()

  *Single constructor.*
- void exportToFile (const DocumentModelExportFormat &modelExportOverride, const Document &document, const MainWindowModel &modelMainWindow, const Transformation &transformation, QTextStream &str, unsigned int &numWritesSoFar) const

  *Export Document points according to the settings.*

### Friends

- class **TestExport**

### Additional Inherited Members

#### 4.149.1 Detailed Description

Strategy class for exporting to a file. This strategy is external to the Document class so that class is simpler.

Definition at line 25 of file ExportFileRelations.h.

#### 4.149.2 Member Function Documentation

##### 4.149.2.1 exportToFile()

```
void ExportFileRelations::exportToFile (
            const DocumentModelExportFormat & modelExportOverride,
            const Document & document,
            const MainWindowModel & modelMainWindow,
            const Transformation & transformation,
            QTextStream & str,
            unsigned int & numWritesSoFar ) const
```

Export Document points according to the settings.

The DocumentModelExportFormat inside the Document is ignored so DlgSettingsExport can supply its own DocumentModelExportFormat when previewing what would be exported.

Definition at line 223 of file ExportFileRelations.cpp.

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

- Export/ExportFileRelations.h
- Export/ExportFileRelations.cpp

## 4.150 ExportImageForRegression Class Reference

Class for exporting during regression, when the Transformation has not yet been defined.

```
#include <ExportImageForRegression.h>
```

### Public Member Functions

- ExportImageForRegression (const QPixmap &pixmap)

  *Single constructor.*
- void fileExport (const QString &filename) const

  *Export to the specified file. This is called when the Transformation has not been defined.*

### 4.150.1 Detailed Description

Class for exporting during regression, when the Transformation has not yet been defined.

This class just exports the image size

Definition at line 15 of file ExportImageForRegression.h.

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

- Export/ExportImageForRegression.h
- Export/ExportImageForRegression.cpp

## 4.151 ExportOrdinalsSmooth Class Reference

Utility class to interpolate points spaced evenly along a piecewise defined curve with fitted spline.

`#include <ExportOrdinalsSmooth.h>`

### Public Member Functions

- ExportOrdinalsSmooth ()

  *Single constructor.*
- void loadSplinePairsWithoutTransformation (const Points &points, std::vector< double > &t, std::vector< SplinePair > &xy) const

  *Load t (=ordinal) and xy (=screen position) spline pairs, without any conversion to graph coordinates.*
- void loadSplinePairsWithTransformation (const Points &points, const Transformation &transformation, bool isLogXTheta, bool isLogYRadius, std::vector< double > &t, std::vector< SplinePair > &xy) const

  *Load t (=ordinal) and xy (=screen position) spline pairs, converting screen coordinates to graph coordinates.*
- ExportValuesOrdinal ordinalsAtIntervalsGraph (const std::vector< double > &t, const std::vector< SplinePair > &xy, double pointsInterval) const

  *Perform the interpolation on the arrays loaded by the other methods.*

### 4.151.1 Detailed Description

Utility class to interpolate points spaced evenly along a piecewise defined curve with fitted spline.

Definition at line 20 of file ExportOrdinalsSmooth.h.

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

- Export/ExportOrdinalsSmooth.h
- Export/ExportOrdinalsSmooth.cpp

## 4.152 ExportOrdinalsStraight Class Reference

Utility class to interpolate points spaced evenly along a piecewise defined curve with line segments between points.

`#include <ExportOrdinalsStraight.h>`

### Public Member Functions

- ExportOrdinalsStraight ()

  *Single constructor.*
- ExportValuesOrdinal ordinalsAtIntervalsGraphWithoutTransformation (const Points &points, double pointsInterval) const

  *Compute ordinals, without any conversion to graph coordinates.*
- ExportValuesOrdinal ordinalsAtIntervalsGraphWithTransformation (const Points &points, const Transformation &transformation, double pointsInterval) const

  *Compute ordinals, converting screen coordinates to graph coordinates.*

#### 4.152.1 Detailed Description

Utility class to interpolate points spaced evenly along a piecewise defined curve with line segments between points.

Definition at line 19 of file ExportOrdinalsStraight.h.

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

- Export/ExportOrdinalsStraight.h
- Export/ExportOrdinalsStraight.cpp

## 4.153 ExportToClipboard Class Reference

Strategy class for exporting to the clipboard. This strategy is external to the Document class so that class is simpler.

`#include <ExportToClipboard.h>`

### Public Member Functions

- ExportToClipboard ()

  *Single constructor.*
- void exportToClipboard (const QStringList &selected, const Transformation &transformation, QTextStream &strCsv, QTextStream &strHtml, const Curve &curveAxis, const CurvesGraphs &curvesGraphsAll, CurvesGraphs &curvesGraphsSelected) const

  *Export, curve-by-curve, raw data points to a string that will be copied to the clipboard.*

#### 4.153.1 Detailed Description

Strategy class for exporting to the clipboard. This strategy is external to the Document class so that class is simpler.

Definition at line 18 of file ExportToClipboard.h.

#### 4.153.2 Member Function Documentation

#### 4.153.2.1 exportToClipboard()

```
void ExportToClipboard::exportToClipboard (
            const QStringList & selected,
            const Transformation & transformation,
            QTextStream & strCsv,
            QTextStream & strHtml,
            const Curve & curveAxis,
            const CurvesGraphs & curvesGraphsAll,
            CurvesGraphs & curvesGraphsSelected ) const
```

Export, curve-by-curve, raw data points to a string that will be copied to the clipboard.

**Parameters**

| | | |
|---|---|---|
| in | *selected* | Simple list of selected points that will be exported |
| in | *transformation* | Transformation which may or may not be defined |
| out | *strCsv* | Selected points as comma separated value list |
| out | *strHtml* | Selected points as html |
| in | *curveAxis* | Axis curve in the Document and its points |
| in | *curvesGraphsAll* | All graph curves in the Document and their points |
| out | *curvesGraphsSelected* | Selected points as a subset of document.curvesGraphs() |

Definition at line 18 of file ExportToClipboard.cpp.

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

- Export/ExportToClipboard.h
- Export/ExportToClipboard.cpp

## 4.154 ExportToFile Class Reference

Strategy class for exporting to a file. This strategy is external to the Document class so that class is simpler.

```
#include <ExportToFile.h>
```

### Public Member Functions

- ExportToFile ()

  *Single constructor.*
- void exportToFile (const DocumentModelExportFormat &modelExport, const Document &document, const MainWindowModel &modelMainWindow, const Transformation &transformation, QTextStream &str) const

  *Export Document points according to the settings.*
- QString fileExtensionCsv () const

  *File extension for csv export files.*
- QString fileExtensionTsv () const

  *File extension for tsv export files.*
- QString filterCsv () const

  *QFileDialog filter for CSV files.*
- QString filterTsv () const

  *QFileDialog filter for TSV files.*

### 4.154.1 Detailed Description

Strategy class for exporting to a file. This strategy is external to the Document class so that class is simpler.

Definition at line 25 of file ExportToFile.h.

### 4.154.2 Member Function Documentation

#### 4.154.2.1 exportToFile()

```
void ExportToFile::exportToFile (
            const DocumentModelExportFormat & modelExport,
            const Document & document,
            const MainWindowModel & modelMainWindow,
            const Transformation & transformation,
            QTextStream & str ) const
```

Export Document points according to the settings.

The DocumentModelExportFormat inside the Document is ignored so DlgSettingsExport can supply its own DocumentModelExportFormat when previewing what would be exported.

Definition at line 23 of file ExportToFile.cpp.

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

- Export/ExportToFile.h
- Export/ExportToFile.cpp

## 4.155 ExportXThetaValuesMergedFunctions Class Reference

Creates the set of merged x/theta values for exporting functions, using interpolation.

```
#include <ExportXThetaValuesMergedFunctions.h>
```

### Public Member Functions

- ExportXThetaValuesMergedFunctions (const DocumentModelExportFormat &modelExport, const Values↩ VectorXOrY &xThetaValuesRaw, const Transformation &transformation)

  *Single constructor.*
- ExportValuesXOrY xThetaValues () const

  *Resulting x/theta values for all included functions.*

### 4.155.1 Detailed Description

Creates the set of merged x/theta values for exporting functions, using interpolation.

Definition at line 19 of file ExportXThetaValuesMergedFunctions.h.

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

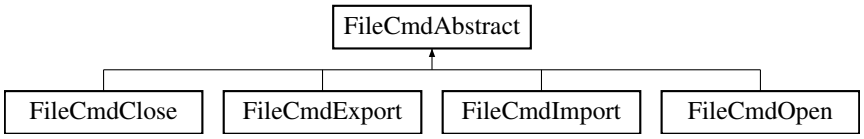
- Export/ExportXThetaValuesMergedFunctions.h
- Export/ExportXThetaValuesMergedFunctions.cpp

## 4.156 FileCmdAbstract Class Reference

Base class for 'file commands' that are used specifically for regression testing of file import/open/export features.

`#include <FileCmdAbstract.h>`

Inheritance diagram for FileCmdAbstract:




#### Public Member Functions

- FileCmdAbstract (const QString &cmdDescription)
  *Single constructor.*
- virtual void redo (MainWindow &mainWindow)=0
  *Apply this command, through MainWindow.*

#### Protected Member Functions

- QString cmdDescription () const
  *Command description for logging.*

### 4.156.1 Detailed Description

Base class for 'file commands' that are used specifically for regression testing of file import/open/export features.

These commands operate outside of the normal undo/redo command framework, since that framework uses commands that are attached to an open Document. The file commands follow special rules:

1. Never generated by the code
2. Created by manually editing a 'file command' xml file, which does NOT have a Document (so error report files cannot be used unless pretty much everything is removed)
3. Are only read during regression testing normally. Although they can be loaded otherwise, there is no point in doing so
4. These commands operate in the forward direction only, since undoing a File Close could be quite messy

Definition at line 22 of file FileCmdAbstract.h.

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

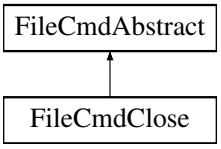
- FileCmd/FileCmdAbstract.h
- FileCmd/FileCmdAbstract.cpp

## 4.157 FileCmdClose Class Reference

Command for closing a file.

`#include <FileCmdClose.h>`

Inheritance diagram for FileCmdClose:




### Public Member Functions

- FileCmdClose (QXmlStreamReader &reader)
  *Constructor for parsing file script xml.*
- virtual void redo (MainWindow &mainWindow)
  *Apply this command, through MainWindow.*

### Additional Inherited Members

### 4.157.1 Detailed Description

Command for closing a file.

Definition at line 15 of file FileCmdClose.h.

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

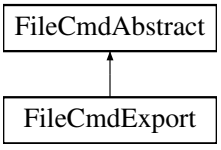
- FileCmd/FileCmdClose.h
- FileCmd/FileCmdClose.cpp

## 4.158 FileCmdExport Class Reference

Command for exporting a file.

`#include <FileCmdExport.h>`

Inheritance diagram for FileCmdExport:



### Public Member Functions

- FileCmdExport (QXmlStreamReader &reader)

  *Constructor for parsing file script xml.*
- virtual void redo (MainWindow &mainWindow)

  *Apply this command, through MainWindow.*

### Additional Inherited Members

#### 4.158.1 Detailed Description

Command for exporting a file.

Definition at line 15 of file FileCmdExport.h.

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

- FileCmd/FileCmdExport.h
- FileCmd/FileCmdExport.cpp

## 4.159 FileCmdFactory Class Reference

Factory that creates FileCmds from a file cmd script file, in xml format.

```
#include <FileCmdFactory.h>
```

### Public Member Functions

- FileCmdFactory ()

  *Single constructor.*
- FileCmdAbstract * createFileCmd (QXmlStreamReader &reader) const

  *Create one FileCmdAbstract from the specified xml subtree.*

#### 4.159.1 Detailed Description

Factory that creates FileCmds from a file cmd script file, in xml format.

Definition at line 15 of file FileCmdFactory.h.

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

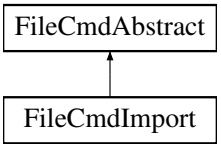
- FileCmd/FileCmdFactory.h
- FileCmd/FileCmdFactory.cpp

## 4.160 FileCmdImport Class Reference

Command for importing a file.

`#include <FileCmdImport.h>`

Inheritance diagram for FileCmdImport:




### Public Member Functions

- FileCmdImport (QXmlStreamReader &reader)
  *Constructor for parsing file script xml.*
- virtual void redo (MainWindow &mainWindow)
  *Apply this command, through MainWindow.*

### Additional Inherited Members

#### 4.160.1 Detailed Description

Command for importing a file.

Definition at line 15 of file FileCmdImport.h.

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

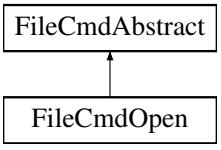
- FileCmd/FileCmdImport.h
- FileCmd/FileCmdImport.cpp

## 4.161 FileCmdOpen Class Reference

Command for opening a file.

`#include <FileCmdOpen.h>`

Inheritance diagram for FileCmdOpen:



### Public Member Functions

- FileCmdOpen (QXmlStreamReader &reader)
  *Constructor for parsing file script xml.*
- virtual void redo (MainWindow &mainWindow)
  *Apply this command, through MainWindow.*

### Additional Inherited Members

#### 4.161.1 Detailed Description

Command for opening a file.

Definition at line 15 of file FileCmdOpen.h.

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

- FileCmd/FileCmdOpen.h
- FileCmd/FileCmdOpen.cpp

## 4.162 FileCmdScript Class Reference

File that manages a command stack for regression testing of file import/open/export/close.

```
#include <FileCmdScript.h>
```

### Public Member Functions

- FileCmdScript (const QString &fileCmdScriptFile)
  *Single constructor.*
- bool canRedo () const
  *Returns true if there is at least one command on the stack.*
- void redo (MainWindow &mainWindow)
  *Apply the next command. Requires non-empty stack.*

#### 4.162.1 Detailed Description

File that manages a command stack for regression testing of file import/open/export/close.

This command stack (with a lifetime the same as the application's) is independent of the command stack in CmdMediator (which is Document-specific)

Definition at line 20 of file FileCmdScript.h.

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

- FileCmd/FileCmdScript.h
- FileCmd/FileCmdScript.cpp

## 4.163 FilterImage Class Reference

Filters an image using a combination of color filtering and grid removal.

`#include <FilterImage.h>`

### Public Member Functions

- FilterImage ()

  *Single constructor.*
- QPixmap filter (const QImage &imageUnfiltered, const Transformation &transformation, const QString &curveSelected, const DocumentModelColorFilter &modelColorFilter, const DocumentModelGridRemoval &modelGridRemoval) const

  *Filter original unfiltered image into filtered pixmap.*

### 4.163.1 Detailed Description

Filters an image using a combination of color filtering and grid removal.

Definition at line 18 of file FilterImage.h.

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

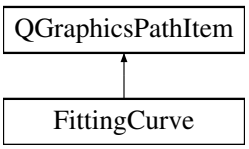
- Filter/FilterImage.h
- Filter/FilterImage.cpp

## 4.164 FittingCurve Class Reference

Curve that overlays the current scene so the regression-fitted curve is visible.

`#include <FittingCurve.h>`

Inheritance diagram for FittingCurve:




### Public Member Functions

- FittingCurve (const FittingCurveCoefficients &fittingCoef, double xMin, double xMax, bool isLogXTheta, bool isLogYRadius, const Transformation &transformation)

  *Single constructor.*

### 4.164.1 Detailed Description

Curve that overlays the current scene so the regression-fitted curve is visible.

Definition at line 16 of file FittingCurve.h.

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

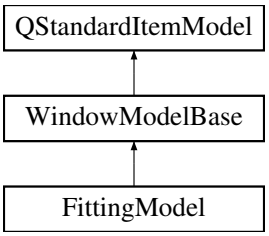
- Fitting/FittingCurve.h
- Fitting/FittingCurve.cpp

## 4.165 FittingModel Class Reference

Model for FittingWindow.

`#include <FittingModel.h>`

Inheritance diagram for FittingModel:




### Public Member Functions

- FittingModel ()
  *Single constructor.*
- virtual QVariant data (const QModelIndex &index, int role=Qt::DisplayRole) const
  *Override for special processing.*

### 4.165.1 Detailed Description

Model for FittingWindow.

Definition at line 14 of file FittingModel.h.

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

- Fitting/FittingModel.h
- Fitting/FittingModel.cpp

## 4.166 FittingStatistics Class Reference

This class does the math to compute statistics for FittingWindow.

`#include <FittingStatistics.h>`

### Public Member Functions

- FittingStatistics ()

  *Single constructor.*
- void calculateCurveFitAndStatistics (unsigned int order, const FittingPointsConvenient &pointsConvenient, FittingCurveCoefficients &coefficients, double &mse, double &rms, double &rSquared)

  *Compute the curve fit and the statistics for that curve fit.*

### 4.166.1 Detailed Description

This class does the math to compute statistics for FittingWindow.

Definition at line 19 of file FittingStatistics.h.

### 4.166.2 Member Function Documentation

#### 4.166.2.1 calculateCurveFitAndStatistics()

```
void FittingStatistics::calculateCurveFitAndStatistics (
            unsigned int order,
            const FittingPointsConvenient & pointsConvenient,
            FittingCurveCoefficients & coefficients,
            double & mse,
            double & rms,
            double & rSquared )
```

Compute the curve fit and the statistics for that curve fit.

**Parameters**

| | |
|---|---|
| *order* | Requested order of the polynomial to be fitted. This will be reduced if there are not enough points just enough to prevent having an undetermined system (=more degrees of freedom than constraints) since otherwise there wil be an infinite number of solutions |
| *pointsConvenient* | Input data consisting of (x,y) points in graph coordinates |
| *coefficients* | Output coefficients a0, a1, and so on in y = a0 + a1 ∗ x + a2 ∗ x^2 + ... |
| *mse* | Mean squared error between the original data and the fitted curve |
| *rms* | Root mean square error between the original data and the fitted curve |
| *rSquared* | R-squared error between the original data and the fitted curve |

Definition at line 68 of file FittingStatistics.cpp.

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

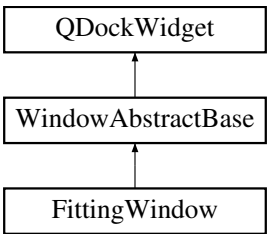
- Fitting/FittingStatistics.h
- Fitting/FittingStatistics.cpp

## 4.167 FittingWindow Class Reference

Window that displays curve fitting as applied to the currently selected curve.

`#include <FittingWindow.h>`

Inheritance diagram for FittingWindow:




### Signals

- void signalCurveFit (FittingCurveCoefficients, double, double, bool, bool)

  *Signal containing coefficients from curve fit.*
- void signalFittingWindowClosed ()

  *Signal that this QDockWidget was just closed.*

### Public Member Functions

- FittingWindow (MainWindow *mainWindow)

  *Single constructor. Parent is needed or else this widget cannot be redocked after being undocked.*
- virtual void clear ()

  *Clear stale information.*
- virtual void closeEvent (QCloseEvent *event)

  *Catch close event so corresponding menu item in MainWindow can be updated accordingly.*
- virtual void doCopy ()

  *Copy the current selection to the clipboard.*
- virtual void update (const CmdMediator &cmdMediator, const MainWindowModel &modelMainWindow, const QString &curveSelected, const Transformation &transformation)

  *Populate the table with the specified Curve.*
- virtual QTableView * view () const

  *QTableView-based class used by child class.*

### Additional Inherited Members

#### 4.167.1 Detailed Description

Window that displays curve fitting as applied to the currently selected curve.

The strategy used assumes no changes to the DIG file format will be made for the original implementation. Since settings cannot be saved for the Document or Curves, this keeps the implementation simple

Definition at line 34 of file FittingWindow.h.

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

- Fitting/FittingWindow.h
- Fitting/FittingWindow.cpp

## 4.168 FormatCoordsUnits Class Reference

Highest-level wrapper around other Formats classes.

`#include <FormatCoordsUnits.h>`

### Public Member Functions

- FormatCoordsUnits ()

  *Single constructor.*
- void formattedToUnformatted (const QString &xThetaFormatted, const QString &yRadiusFormatted, const DocumentModelCoords &modelCoords, const MainWindowModel &mainWindowModel, double &xTheta↩ Unformatted, double &yRadiusUnformatted) const

  *Convert formatted string to unformatted numeric value.*
- void unformattedToFormatted (double xThetaUnformatted, double yRadiusUnformatted, const Document↩ ModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &mainWindowModel, QString &xThetaFormatted, QString &yRadiusFormatted, const Transformation &transformation) const

  *Convert unformatted numeric value to formatted string. Transformation is used to determine best resolution.*

#### 4.168.1 Detailed Description

Highest-level wrapper around other Formats classes.

Definition at line 17 of file FormatCoordsUnits.h.

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

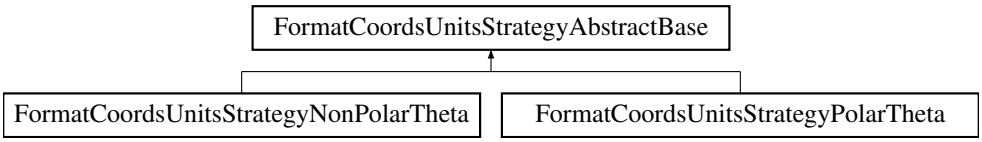
- Format/FormatCoordsUnits.h
- Format/FormatCoordsUnits.cpp

## 4.169 FormatCoordsUnitsStrategyAbstractBase Class Reference

Common methods for formatting strategies.

`#include <FormatCoordsUnitsStrategyAbstractBase.h>`

Inheritance diagram for FormatCoordsUnitsStrategyAbstractBase:




### Public Member Functions

- FormatCoordsUnitsStrategyAbstractBase ()

  *Single constructor.*

### Protected Member Functions

- int precisionDigitsForRawNumber (double valueUnformatted, double valueUnformattedOther, bool isXTheta, const DocumentModelGeneral &modelGeneral, const Transformation &transformation) const

  *Compute precision for outputting an unformatted value, consistent with the resolution at the point where that point lies.*

### 4.169.1 Detailed Description

Common methods for formatting strategies.

Definition at line 14 of file FormatCoordsUnitsStrategyAbstractBase.h.

### 4.169.2 Member Function Documentation

#### 4.169.2.1 precisionDigitsForRawNumber()

```
int FormatCoordsUnitsStrategyAbstractBase::precisionDigitsForRawNumber (
            double valueUnformatted,
            double valueUnformattedOther,
            bool isXTheta,
            const DocumentModelGeneral & modelGeneral,
            const Transformation & transformation ) const  [protected]
```

Compute precision for outputting an unformatted value, consistent with the resolution at the point where that point lies.

This algorithm causes many digits to appear when a graph's dynamic range is relatively small (like -118.4 to -118.2 degrees in longitude), and fewer digits to appear when a graph's dynamic range is relatively large (like 0 to 100)

Definition at line 17 of file FormatCoordsUnitsStrategyAbstractBase.cpp.

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

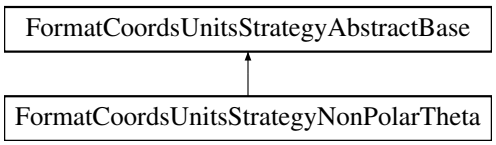
- Format/FormatCoordsUnitsStrategyAbstractBase.h
- Format/FormatCoordsUnitsStrategyAbstractBase.cpp

## 4.170 FormatCoordsUnitsStrategyNonPolarTheta Class Reference

Format conversions between unformatted and formatted for CoordUnitsNonPolarTheta.

`#include <FormatCoordsUnitsStrategyNonPolarTheta.h>`

Inheritance diagram for FormatCoordsUnitsStrategyNonPolarTheta:




### Public Member Functions

- FormatCoordsUnitsStrategyNonPolarTheta ()

  *Single constructor.*
- double formattedToUnformatted (const QString &string, const QLocale &locale, CoordUnitsNonPolarTheta coordUnits, CoordUnitsDate coordUnitsDate, CoordUnitsTime coordUnitsTime) const

  *Convert formatted string to simple unformatted number.*
- QString unformattedToFormatted (double valueUnformatted, const QLocale &locale, CoordUnitsNonPolar↩ Theta coordUnits, CoordUnitsDate coordUnitsDate, CoordUnitsTime coordUnitsTime, bool isXTheta, const DocumentModelGeneral &modelGeneral, const Transformation &transformation, double valueUnformatted↩ Other) const

  *Convert simple unformatted number to formatted string.*

### Additional Inherited Members

#### 4.170.1 Detailed Description

Format conversions between unformatted and formatted for CoordUnitsNonPolarTheta.

Definition at line 21 of file FormatCoordsUnitsStrategyNonPolarTheta.h.

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

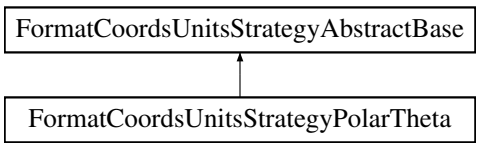
- Format/FormatCoordsUnitsStrategyNonPolarTheta.h
- Format/FormatCoordsUnitsStrategyNonPolarTheta.cpp

## 4.171 FormatCoordsUnitsStrategyPolarTheta Class Reference

Format conversions between unformatted and formatted for CoordUnitsStrategyPolarTheta.

`#include <FormatCoordsUnitsStrategyPolarTheta.h>`

Inheritance diagram for FormatCoordsUnitsStrategyPolarTheta:



### Public Member Functions

- FormatCoordsUnitsStrategyPolarTheta ()

  *Single constructor.*
- double formattedToUnformatted (const QString &string, const QLocale &locale, CoordUnitsPolarTheta coordUnits) const

  *Convert formatted string to simple unformatted number.*
- QString unformattedToFormatted (double valueUnformatted, const QLocale &locale, CoordUnitsPolarTheta coordUnits, const DocumentModelGeneral &modelGeneral, const Transformation &transformation, double valueUnformattedOther) const

  *Convert simple unformatted number to formatted string.*

### Additional Inherited Members

#### 4.171.1 Detailed Description

Format conversions between unformatted and formatted for CoordUnitsStrategyPolarTheta.

Definition at line 19 of file FormatCoordsUnitsStrategyPolarTheta.h.

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

- Format/FormatCoordsUnitsStrategyPolarTheta.h
- Format/FormatCoordsUnitsStrategyPolarTheta.cpp

## 4.172 FormatDateTime Class Reference

Input parsing and output formatting for date/time values.

```
#include <FormatDateTime.h>
```

### Public Member Functions

- FormatDateTime ()

  *Single constructor.*
- QString formatOutput (CoordUnitsDate coordUnitsDate, CoordUnitsTime coordUnitsTime, double value) const

  *Format the date/time value according to date/time format settings.*
- QValidator::State parseInput (CoordUnitsDate coordUnitsDate, CoordUnitsTime coordUnitsTime, const QString &stringUntrimmed, double &value) const

  *Parse the input string into a time value.*

#### 4.172.1 Detailed Description

Input parsing and output formatting for date/time values.

Definition at line 25 of file FormatDateTime.h.

### 4.172.2 Member Function Documentation

#### 4.172.2.1 parseInput()

```
QValidator::State FormatDateTime::parseInput (
            CoordUnitsDate coordUnitsDate,
            CoordUnitsTime coordUnitsTime,
            const QString & stringUntrimmed,
            double & value ) const
```

Parse the input string into a time value.

Success flag is false if parsing failed. Leading/trailing spaces are trimmed (=ignored)

Definition at line 422 of file FormatDateTime.cpp.

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

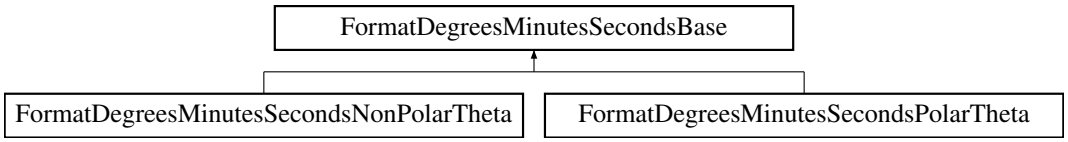
- Format/FormatDateTime.h
- Format/FormatDateTime.cpp

## 4.173 FormatDegreesMinutesSecondsBase Class Reference

Common input parsing and output formatting for degrees/minutes/seconds values.

`#include <FormatDegreesMinutesSecondsBase.h>`

Inheritance diagram for FormatDegreesMinutesSecondsBase:




### Public Member Functions

- FormatDegreesMinutesSecondsBase ()
  *Single constructor.*
- QValidator::State parseInput (const QString &stringUntrimmed, double &value) const
  *Parse the input string into a number value.*

### Protected Member Functions

- QString formatOutputDegreesMinutesSeconds (double value) const
  *Format as degrees, minutes and seconds without hemisphere.*
- QString formatOutputDegreesMinutesSecondsNsew (double value, bool isNsHemisphere) const
  *Format as degrees, minutes and seconds with hemisphere.*

### 4.173.1 Detailed Description

Common input parsing and output formatting for degrees/minutes/seconds values.

Definition at line 14 of file FormatDegreesMinutesSecondsBase.h.

### 4.173.2 Member Function Documentation

#### 4.173.2.1 parseInput()

```
QValidator::State FormatDegreesMinutesSecondsBase::parseInput (
            const QString & stringUntrimmed,
            double & value ) const
```

Parse the input string into a number value.

Success flag is false if the parsing failed. Either signed values or hemisphere (North, South, East, West) values can be accepted irregardless of the output format selected by the user. Leading/trailing spaces are trimmed. Leading/trailing spaces are trimmed (=ignored)

Definition at line 87 of file FormatDegreesMinutesSecondsBase.cpp.

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

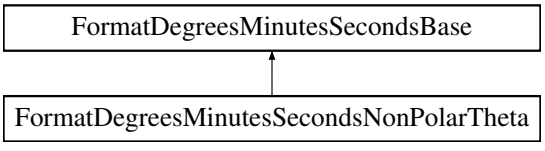
- Format/FormatDegreesMinutesSecondsBase.h
- Format/FormatDegreesMinutesSecondsBase.cpp

## 4.174 FormatDegreesMinutesSecondsNonPolarTheta Class Reference

Angular units according to CoordUnitsNonPolarTheta.

```
#include <FormatDegreesMinutesSecondsNonPolarTheta.h>
```

Inheritance diagram for FormatDegreesMinutesSecondsNonPolarTheta:




### Public Member Functions

- FormatDegreesMinutesSecondsNonPolarTheta ()

  *Single constructor.*
- QString formatOutput (CoordUnitsNonPolarTheta coordUnits, double value, bool isXTheta) const

  *Format the degrees/minutes/seconds value. Distinguishing x/theta versus y/radius is required for N/S/E/W hemispheres.*

### Additional Inherited Members

#### 4.174.1 Detailed Description

Angular units according to CoordUnitsNonPolarTheta.

Definition at line 15 of file FormatDegreesMinutesSecondsNonPolarTheta.h.

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

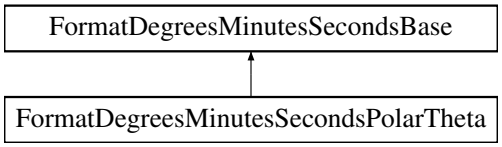
- Format/FormatDegreesMinutesSecondsNonPolarTheta.h
- Format/FormatDegreesMinutesSecondsNonPolarTheta.cpp

## 4.175 FormatDegreesMinutesSecondsPolarTheta Class Reference

Angular units according to CoordUnitsPolarTheta.

`#include <FormatDegreesMinutesSecondsPolarTheta.h>`

Inheritance diagram for FormatDegreesMinutesSecondsPolarTheta:




### Public Member Functions

- FormatDegreesMinutesSecondsPolarTheta ()

  *Single constructor.*
- QString formatOutput (CoordUnitsPolarTheta coordUnits, double value, bool isXTheta) const

  *Format the degrees/minutes/seconds value. Distinguishing x/theta versus y/radius is required for N/S/E/W hemispheres.*

### Additional Inherited Members

#### 4.175.1 Detailed Description

Angular units according to CoordUnitsPolarTheta.

Definition at line 15 of file FormatDegreesMinutesSecondsPolarTheta.h.

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

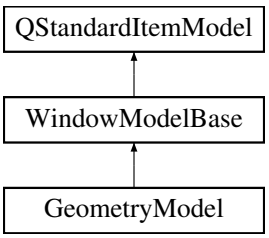
- Format/FormatDegreesMinutesSecondsPolarTheta.h
- Format/FormatDegreesMinutesSecondsPolarTheta.cpp

## 4.176 GeometryModel Class Reference

Model that adds row highlighting according to the currently select point identifier.

`#include <GeometryModel.h>`

Inheritance diagram for GeometryModel:




### Public Member Functions

- GeometryModel ()
  *Single constructor.*
- virtual QVariant data (const QModelIndex &index, int role=Qt::DisplayRole) const
  *Override for special processing.*
- void setCurrentPointIdentifier (const QString &pointIdentifier)
  *Set the point identifier to be highlighted. Value is empty for no highlighting.*

### 4.176.1 Detailed Description

Model that adds row highlighting according to the currently select point identifier.

Definition at line 14 of file GeometryModel.h.

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

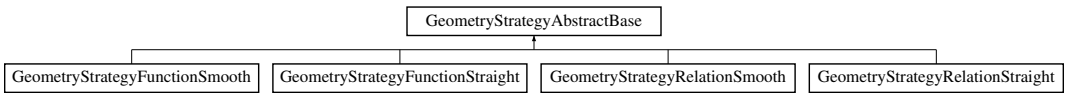
- Geometry/GeometryModel.h
- Geometry/GeometryModel.cpp

## 4.177 GeometryStrategyAbstractBase Class Reference

Base class for all geometry strategies.

`#include <GeometryStrategyAbstractBase.h>`

Inheritance diagram for GeometryStrategyAbstractBase:



### Public Member Functions

- GeometryStrategyAbstractBase ()

  *Single constructor.*

- virtual void calculateGeometry (const Points &points, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, QString &funcArea, QString &polyArea, QVector< QString > &x, QVector< QString > &y, QVector< QString > &distanceGraphForward, QVector< QString > &distancePercentForward, QVector< QString > &distanceGraphBackward, QVector< QString > &distancePercentBackward) const =0

  *Calculate geometry parameters.*

### Protected Member Functions

- void calculatePositionsGraph (const Points &points, const Transformation &transformation, QVector< QPointF > &positionsGraph) const

  *Convert screen positions to graph positions.*

- double functionArea (const QVector< QPointF > &positionsGraph) const

  *Use trapezoidal approximation to compute area under the function. Does not apply to relation.*

- void insertSubintervalsAndLoadDistances (int subintervalsPerInterval, const QVector< QPointF > &positionsGraph, QVector< QPointF > &positionsGraphWithSubintervals, QVector< QString > &distanceGraphForward, QVector< QString > &distancePercentForward, QVector< QString > &distanceGraphBackward, QVector< QString > &distancePercentBackward) const

  *Insert the specified number of subintervals into each interval.*

- void loadXY (const QVector< QPointF > &positionsGraph, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, QVector< QString > &x, QVector< QString > &y) const

  *Load x and y coordinate vectors.*

- double polygonAreaForSimplyConnected (const QVector< QPointF > &points) const

  *Area in polygon using Shoelace formula, which only works if polygon is simply connected.*

#### 4.177.1 Detailed Description

Base class for all geometry strategies.

Each strategy computes geometry parameters according to the curve's settings.

The numbering for the strategies is specified as the CurveConnectAs enumeration

Definition at line 23 of file GeometryStrategyAbstractBase.h.

#### 4.177.2 Member Function Documentation

#### 4.177.2.1 insertSubintervalsAndLoadDistances()

```
void GeometryStrategyAbstractBase::insertSubintervalsAndLoadDistances (
            int subintervalsPerInterval,
            const QVector< QPointF > & positionsGraph,
            QVector< QPointF > & positionsGraphWithSubintervals,
            QVector< QString > & distanceGraphForward,
            QVector< QString > & distancePercentForward,
            QVector< QString > & distanceGraphBackward,
            QVector< QString > & distancePercentBackward ) const  [protected]
```

Insert the specified number of subintervals into each interval.

For straight curves subintervalsPerInterval=1 so the linearity is maintained, and for smooth curves subintervals↩ PerInterval>1 so the geometry calculations take into account the curvature(s) of the line

Definition at line 61 of file GeometryStrategyAbstractBase.cpp.

#### 4.177.2.2 polygonAreaForSimplyConnected()

```
double GeometryStrategyAbstractBase::polygonAreaForSimplyConnected (
            const QVector< QPointF > & points ) const  [protected]
```

Area in polygon using Shoelace formula, which only works if polygon is simply connected.

We do not check to see if the polygon is simply connected since that would be (1) slow and (2) much work

Definition at line 166 of file GeometryStrategyAbstractBase.cpp.

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

- Geometry/GeometryStrategyAbstractBase.h
- Geometry/GeometryStrategyAbstractBase.cpp

## 4.178 GeometryStrategyContext Class Reference

Class for that manages geometry strategies.

```
#include <GeometryStrategyContext.h>
```

### Public Member Functions

- GeometryStrategyContext ()

  *Single constructor.*
- void calculateGeometry (const Points &points, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, CurveConnectAs connectAs, QString &funcArea, QString &polyArea, QVector< QString > &x, QVector< QString > &y, QVector< QString > &distanceGraphForward, QVector< QString > &distancePercentForward, QVector< QString > &distanceGraphBackward, QVector< QString > &distancePercentBackward) const

  *Calculate geometry parameters.*

### 4.178.1 Detailed Description

Class for that manages geometry strategies.

Definition at line 21 of file GeometryStrategyContext.h.

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

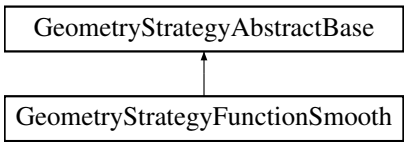
- Geometry/GeometryStrategyContext.h
- Geometry/GeometryStrategyContext.cpp

## 4.179 GeometryStrategyFunctionSmooth Class Reference

Calculate for line through the points that is smoothly connected as a function.

`#include <GeometryStrategyFunctionSmooth.h>`

Inheritance diagram for GeometryStrategyFunctionSmooth:




### Public Member Functions

- GeometryStrategyFunctionSmooth ()

  *Single constructor.*
- virtual void calculateGeometry (const Points &points, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, QString &funcArea, QString &polyArea, QVector< QString > &x, QVector< QString > &y, QVector< QString > &distanceGraphForward, QVector< QString > &distancePercentForward, QVector< QString > &distanceGraphBackward, QVector< QString > &distancePercentBackward) const

  *Calculate geometry parameters.*

### Additional Inherited Members

### 4.179.1 Detailed Description

Calculate for line through the points that is smoothly connected as a function.

Definition at line 16 of file GeometryStrategyFunctionSmooth.h.

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

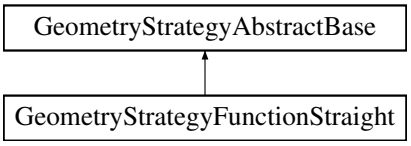
- Geometry/GeometryStrategyFunctionSmooth.h
- Geometry/GeometryStrategyFunctionSmooth.cpp

## 4.180 GeometryStrategyFunctionStraight Class Reference

Calculate for line through the points that is straightly connected as a function.

`#include <GeometryStrategyFunctionStraight.h>`

Inheritance diagram for GeometryStrategyFunctionStraight:




### Public Member Functions

- GeometryStrategyFunctionStraight ()

  *Single constructor.*
- virtual void calculateGeometry (const Points &points, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, QString &funcArea, QString &polyArea, QVector< QString > &x, QVector< QString > &y, QVector< QString > &distanceGraphForward, QVector< QString > &distancePercentForward, QVector< QString > &distanceGraphBackward, QVector< QString > &distancePercentBackward) const

  *Calculate geometry parameters.*

### Additional Inherited Members

### 4.180.1 Detailed Description

Calculate for line through the points that is straightly connected as a function.

Definition at line 16 of file GeometryStrategyFunctionStraight.h.

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

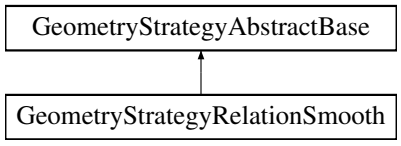
- Geometry/GeometryStrategyFunctionStraight.h
- Geometry/GeometryStrategyFunctionStraight.cpp

## 4.181 GeometryStrategyRelationSmooth Class Reference

Calculate for line through the points that is smoothly connected as a relation.

`#include <GeometryStrategyRelationSmooth.h>`

Inheritance diagram for GeometryStrategyRelationSmooth:



### Public Member Functions

- GeometryStrategyRelationSmooth ()

  *Single constructor.*
- virtual void calculateGeometry (const Points &points, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, QString &funcArea, QString &polyArea, QVector< QString > &x, QVector< QString > &y, QVector< QString > &distanceGraphForward, QVector< QString > &distancePercentForward, QVector< QString > &distanceGraphBackward, QVector< QString > &distancePercentBackward) const

  *Calculate geometry parameters.*

### Additional Inherited Members

#### 4.181.1 Detailed Description

Calculate for line through the points that is smoothly connected as a relation.

Definition at line 16 of file GeometryStrategyRelationSmooth.h.

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

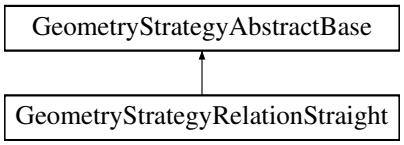
- Geometry/GeometryStrategyRelationSmooth.h
- Geometry/GeometryStrategyRelationSmooth.cpp

## 4.182 GeometryStrategyRelationStraight Class Reference

Calculate for line through the points that is straightly connected as a relation.

`#include <GeometryStrategyRelationStraight.h>`

Inheritance diagram for GeometryStrategyRelationStraight:




### Public Member Functions

- GeometryStrategyRelationStraight ()

  *Single constructor.*
- virtual void calculateGeometry (const Points &points, const DocumentModelCoords &modelCoords, const DocumentModelGeneral &modelGeneral, const MainWindowModel &modelMainWindow, const Transformation &transformation, QString &funcArea, QString &polyArea, QVector< QString > &x, QVector< QString > &y, QVector< QString > &distanceGraphForward, QVector< QString > &distancePercentForward, QVector< QString > &distanceGraphBackward, QVector< QString > &distancePercentBackward) const

  *Calculate geometry parameters.*

### Additional Inherited Members

### 4.182.1 Detailed Description

Calculate for line through the points that is straightly connected as a relation.

Definition at line 16 of file GeometryStrategyRelationStraight.h.

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

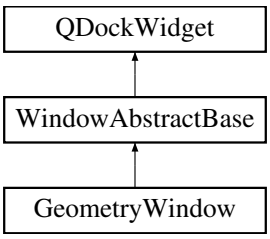
- Geometry/GeometryStrategyRelationStraight.h
- Geometry/GeometryStrategyRelationStraight.cpp

## 4.183 GeometryWindow Class Reference

Window that displays the geometry information, as a table, for the current curve.

`#include <GeometryWindow.h>`

Inheritance diagram for GeometryWindow:




### Public Slots

- void slotPointHoverEnter (QString)

  *Highlight the row for the specified point.*
- void slotPointHoverLeave (QString)

  *Unhighlight the row for the specified point.*

### Signals

- void signalGeometryWindowClosed ()

  *Signal that this QDockWidget was just closed.*

### Public Member Functions

- GeometryWindow (MainWindow *mainWindow)

  *Single constructor. Parent is needed or else this widget cannot be redocked after being undocked.*
- virtual void clear ()

  *Clear stale information.*
- virtual void closeEvent (QCloseEvent *event)

  *Catch close event so corresponding menu item in MainWindow can be updated accordingly.*
- virtual void doCopy ()

  *Copy the current selection to the clipboard.*
- virtual void update (const CmdMediator &cmdMediator, const MainWindowModel &modelMainWindow, const QString &curveSelected, const Transformation &transformation)

  *Populate the table with the specified Curve.*
- virtual QTableView * view () const

  *QTableView-based class used by child class.*

### Static Public Member Functions

- static int columnBodyPointIdentifiers ()

  *Hidden column that has the point identifiers.*

### Additional Inherited Members

#### 4.183.1 Detailed Description

Window that displays the geometry information, as a table, for the current curve.

Column COLUMN_BODY_POINT_IDENTIFIERS is hidden. It contains the point identifiers so we can find the line associated with a point, and then highlight that line

Definition at line 28 of file GeometryWindow.h.

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

- Geometry/GeometryWindow.h
- Geometry/GeometryWindow.cpp

## 4.184 GhostEllipse Class Reference

Ghost for a QGraphicsEllipseItem.

`#include <GhostEllipse.h>`

### Public Member Functions

- GhostEllipse (const QRectF &rect, const QPen &pen, const QBrush &brush)
  *Initial constructor.*
- GhostEllipse (const GhostEllipse &other)
  *Copy constructor.*
- GhostEllipse & operator= (const GhostEllipse &other)
  *Assignment operator.*
- QBrush brush () const
  *Get method for brush.*
- QPen pen () const
  *Get method for pen.*
- QRectF rect () const
  *Get method for bounding rectangle.*

#### 4.184.1 Detailed Description

Ghost for a QGraphicsEllipseItem.

Definition at line 15 of file GhostEllipse.h.

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

- Ghosts/GhostEllipse.h
- Ghosts/GhostEllipse.cpp

## 4.185 GhostPath Class Reference

Ghost for a QGraphicsPathItem.

```
#include <GhostPath.h>
```

### Public Member Functions

- GhostPath (const QPainterPath &path, const QPen &pen, const QBrush &brush)
  *Initial constructor.*
- GhostPath (const GhostPath &other)
  *Copy constructor.*
- GhostPath & operator= (const GhostPath &other)
  *Assignment operator.*
- QBrush brush () const
  *Get method for brush.*
- QPainterPath path () const
  *Get method for path.*
- QPen pen () const
  *Get method for pen.*

### 4.185.1 Detailed Description

Ghost for a QGraphicsPathItem.

Definition at line 15 of file GhostPath.h.

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

- Ghosts/GhostPath.h
- Ghosts/GhostPath.cpp

## 4.186 GhostPolygon Class Reference

Ghost for a QGraphicsPolygonItem.

`#include <GhostPolygon.h>`

### Public Member Functions

- GhostPolygon (const QPolygonF &polygon, const QPen &pen, const QBrush &brush)
  *Initial constructor.*
- GhostPolygon (const GhostPolygon &other)
  *Copy constructor.*
- GhostPolygon & operator= (const GhostPolygon &other)
  *Assignment operator.*
- QBrush brush () const
  *Get method for brush.*
- QPen pen () const
  *Get method for pen.*
- QPolygonF polygon () const
  *Get method for polygon.*

### 4.186.1 Detailed Description

Ghost for a QGraphicsPolygonItem.

Definition at line 15 of file GhostPolygon.h.

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

- Ghosts/GhostPolygon.h
- Ghosts/GhostPolygon.cpp

## 4.187 Ghosts Class Reference

Class for showing points and lines for all coordinate systems simultaneously, even though the code normally only allows graphical items for once coordinate system to be visible at a time.

`#include <Ghosts.h>`

### Public Member Functions

- Ghosts (unsigned int coordSystemIndexToBeRestored)
  *Single constructor.*
- unsigned int coordSystemIndexToBeRestored () const
  *Coordinate system index that was active before the ghosts.*
- void captureGraphicsItems (QGraphicsScene &scene)
  *Take a snapshot of the graphics items.*
- void createGhosts (QGraphicsScene &scene)
  *Create ghosts from the path/rect/polygon lists.*
- void destroyGhosts (QGraphicsScene &scene)
  *Destory ghosts. Called at end of algorithm.*

### 4.187.1 Detailed Description

Class for showing points and lines for all coordinate systems simultaneously, even though the code normally only allows graphical items for once coordinate system to be visible at a time.

QGraphicsLineItems are ignored since those are just used for the AxesChecker, and QGraphicsPixmapItems are ignored since those are just used for the background. The other QGraphicsItem subclasses are captured and converted into ghosts.

Definition at line 26 of file Ghosts.h.

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

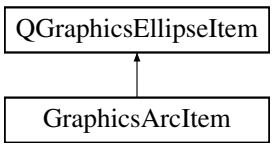
- Ghosts/Ghosts.h
- Ghosts/Ghosts.cpp

## 4.188 GraphicsArcItem Class Reference

Draw an arc as an ellipse but without lines from the center to the start and end points.

`#include <GraphicsArcItem.h>`

Inheritance diagram for GraphicsArcItem:




### Public Member Functions

- GraphicsArcItem (double x, double y, double width, double height, QGraphicsItem *parent=0)
  *Constructor with individual coordinates.*
- GraphicsArcItem (const QRectF &rect, QGraphicsItem *parent=0)
  *Constructor with coordinates specified as rectangle.*
- virtual void paint (QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget)
  *Paint without interior fill.*

### 4.188.1 Detailed Description

Draw an arc as an ellipse but without lines from the center to the start and end points.

Originally this class overrode QGraphicsEllipseItem::boundingRect and called QGraphicsScene::boundingRect. However, that led to an infinite loop since QGraphicsScene::boundingRect looped back around to QGraphics↩ EllipseItem::boundingRect

Definition at line 17 of file GraphicsArcItem.h.

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

- Graphics/GraphicsArcItem.h
- Graphics/GraphicsArcItem.cpp

## 4.189 GraphicsItemsExtractor Class Reference

This class consolidates utility routines that deal with graphics items that are getting extracted from the scene.

`#include <GraphicsItemsExtractor.h>`

### Public Member Functions

- GraphicsItemsExtractor ()

  *Single constructor.*
- bool allSelectedItemsAreEitherAxisOrGraph (const QList< QGraphicsItem * > &items, AxisOrGraph axis↩ OrGraph) const

  *Return true if all selected points are of the specified axis or graph type.*
- QStringList selectedPointIdentifiers (const QList< QGraphicsItem * > &items) const

  *Return list of selected point identifiers.*

### 4.189.1 Detailed Description

This class consolidates utility routines that deal with graphics items that are getting extracted from the scene.

Definition at line 20 of file GraphicsItemsExtractor.h.

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

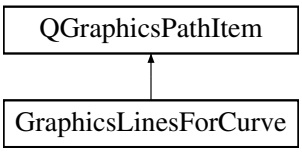
- Graphics/GraphicsItemsExtractor.h
- Graphics/GraphicsItemsExtractor.cpp

## 4.190 GraphicsLinesForCurve Class Reference

This class stores the GraphicsLine objects for one Curve.

`#include <GraphicsLinesForCurve.h>`

Inheritance diagram for GraphicsLinesForCurve:




### Public Member Functions

- GraphicsLinesForCurve (const QString &curveName)
  *Single constructor.*
- void addPoint (const QString &pointIdentifier, double ordinal, GraphicsPoint &point)
  *Add new line.*
- double identifierToOrdinal (const QString &identifier) const
  *Get ordinal for specified identifier.*
- void lineMembershipPurge (const LineStyle &lineStyle)
  *Mark the end of addPoint calls. Remove stale lines, insert missing lines, and draw the graphics lines.*
- void lineMembershipReset ()
  *Mark points as unwanted. Afterwards, lineMembershipPurge gets called.*
- void printStream (QString indentation, QTextStream &str) const
  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void removePoint (double ordinal)
  *Remove the specified point. The act of deleting it will automatically remove it from the GraphicsScene.*
- void removeTemporaryPointIfExists ()
  *Remove temporary point if it exists.*
- void updateAfterCommand (GraphicsScene &scene, const PointStyle &pointStyle, const Point &point, GeometryWindow *geometryWindow)
  *Update the GraphicsScene with the specified Point from the Document. If it does not exist yet in the scene, we add it.*
- void updateCurveStyle (const CurveStyle &curveStyle)
  *Update the curve style for this curve.*
- void updateGraphicsLinesToMatchGraphicsPoints (const LineStyle &lineStyle)
  *Calls to moveLinesWithDraggedPoint have finished so update the lines correspondingly.*
- void updateHighlightOpacity (double highlightOpacity)
  *Update the highlight opacity value. This may or may not affect the current display immediately depending on the state.*
- void updatePointOrdinalsAfterDrag (const LineStyle &lineStyle, const Transformation &transformation)
  *See GraphicsScene::updateOrdinalsAfterDrag. Pretty much the same steps as Curve::updatePointOrdinals.*

### 4.190.1 Detailed Description

This class stores the GraphicsLine objects for one Curve.

The container is a QMap since that container maintains order by key

Definition at line 25 of file GraphicsLinesForCurve.h.

### 4.190.2 Member Function Documentation

#### 4.190.2.1 addPoint()

```
void GraphicsLinesForCurve::addPoint (
            const QString & pointIdentifier,
            double ordinal,
            GraphicsPoint & point )
```

Add new line.

The GraphicsPoint arguments are not const since this line binds to the points, so dragging points also drags the lines. The ordinal is already in the GraphicsPoint as DATA_KEY_ORDINAL

Definition at line 53 of file GraphicsLinesForCurve.cpp.

#### 4.190.2.2 removeTemporaryPointIfExists()

```
void GraphicsLinesForCurve::removeTemporaryPointIfExists ( )
```

Remove temporary point if it exists.

Temporary point handling is so complicated that this method quietly allows redundant calls to this method, without complaining that the point has already been removed when called again

Definition at line 281 of file GraphicsLinesForCurve.cpp.

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

- Graphics/GraphicsLinesForCurve.h
- Graphics/GraphicsLinesForCurve.cpp

## 4.191 GraphicsLinesForCurves Class Reference

This class stores the GraphicsLinesForCurves objects, one per Curve.

```
#include <GraphicsLinesForCurves.h>
```

### Public Member Functions

- GraphicsLinesForCurves ()

  *Single constructor.*
- void addPoint (const QString &curveName, const QString &pointIdentifier, double ordinal, GraphicsPoint &point)

  *Add new point.*
- void addRemoveCurves (GraphicsScene &scene, const QStringList &curveNames)

  *Add new curves and remove expired curves to match the specified list.*
- void lineMembershipPurge (const CurveStyles &curveStyles)

  *Mark the end of addPoint calls. Remove stale lines, insert missing lines, and draw the graphics lines.*
- void lineMembershipReset ()

  *Mark points as unwanted. Afterwards, lineMembershipPurge gets called.*
- void print () const

  *Debugging method for printing directly from symbolic debugger.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void removePoint (const QString &identifier)

  *Remove the specified point. The act of deleting it will automatically remove it from the GraphicsScene.*
- void removeTemporaryPointIfExists ()

  *Remove temporary point if it exists.*
- void resetOnLoad ()

  *Reset, when loading a document after the first, to same state that first document was at when loaded.*
- void updateAfterCommand (GraphicsScene &scene, const CurveStyles &curveStyles, const QString &curveName, const Point &point, GeometryWindow ∗geometryWindow)

  *Update the GraphicsScene with the specified Point from the Document. If it does not exist yet in the scene, we add it.*
- void updateCurveStyles (const CurveStyles &modelCurveStyles)

  *Update the curve style for every curve.*
- void updateGraphicsLinesToMatchGraphicsPoints (const CurveStyles &curveStyles)

  *Calls to moveLinesWithDraggedPoint have finished so update the lines correspondingly.*
- void updateHighlightOpacity (double highlightOpacity)

  *Update the highlight opacity value. This may or may not affect the current display immediately depending on the state.*
- void updatePointOrdinalsAfterDrag (const CurveStyles &curveStyles, const Transformation &transformation)

  *See GraphicsScene::updateOrdinalsAfterDrag.*

## 4.191.1 Detailed Description

This class stores the GraphicsLinesForCurves objects, one per Curve.

Definition at line 26 of file GraphicsLinesForCurves.h.

## 4.191.2 Member Function Documentation

#### 4.191.2.1 addPoint()

```
void GraphicsLinesForCurves::addPoint (
            const QString & curveName,
            const QString & pointIdentifier,
            double ordinal,
            GraphicsPoint & point )
```

Add new point.

The ordinal is already in the GraphicsPoint as DATA_KEY_ORDINAL

Definition at line 28 of file GraphicsLinesForCurves.cpp.

#### 4.191.2.2 removeTemporaryPointIfExists()

```
void GraphicsLinesForCurves::removeTemporaryPointIfExists ( )
```

Remove temporary point if it exists.

Temporary point handling is so complicated that this method quietly allows redundant calls to this method, without complaining that the point has already been removed when called again

Definition at line 149 of file GraphicsLinesForCurves.cpp.

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

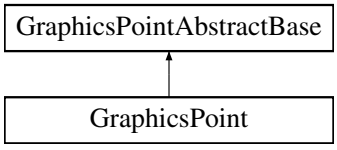
- Graphics/GraphicsLinesForCurves.h
- Graphics/GraphicsLinesForCurves.cpp

## 4.192 GraphicsPoint Class Reference

Graphics item for drawing a circular or polygonal Point.

```
#include <GraphicsPoint.h>
```

Inheritance diagram for GraphicsPoint:



### Public Member Functions

- GraphicsPoint (QGraphicsScene &scene, const QString &identifier, const QPointF &posScreen, const QColor &color, unsigned int radius, double lineWidth, GeometryWindow *geometryWindow)

  *Constructor of circular point.*
- GraphicsPoint (QGraphicsScene &scene, const QString &identifier, const QPointF &posScreen, const QColor &color, const QPolygonF &polygon, double lineWidth, GeometryWindow *geometryWindow)

  *Constructor of polygon point.*
- ~GraphicsPoint ()

  *Destructor. This remove the graphics item from the scene.*
- QRectF boundingRect () const

  *Proxy method for QGraphicsItem::boundingRect.*
- QVariant data (int key) const

  *Proxy method for QGraphicsItem::data.*
- double highlightOpacity () const

  *Get method for highlight opacity.*
- QPointF pos () const

  *Proxy method for QGraphicsItem::pos.*
- void printStream (QString indentation, QTextStream &str, double ordinalKey) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void reset ()

  *Mark point as unwanted, and unbind any bound lines.*
- void setData (int key, const QVariant &data)

  *Proxy method for QGraphicsItem::setData.*
- void setHighlightOpacity (double highlightOpacity)

  *Set method for highlight opacity.*
- void setPointStyle (const PointStyle &pointStyle)

  *Update the point style.*
- void setPos (const QPointF pos)

  *Update the position.*
- void setPassive ()

  *Prevent automatic focus on point (=make it passive) for scale bar so drags can be made to work properly.*
- void setWanted ()

  *Mark point as wanted. Marking as unwanted is done by the reset function.*
- void updateCurveStyle (const CurveStyle &curveStyle)

  *Update point and line styles that comprise the curve style.*
- bool wanted () const

  *Identify point as wanted//unwanted.*

## 4.192.1 Detailed Description

Graphics item for drawing a circular or polygonal Point.

In this class, lines are drawn twice: 1) As nonzero-width lines so user can have thick, and highly visible, points 2) As a 'shadow' with zero-width lines since these always appear even when zooming results in some pixel rows/columns disappearing This dual-line approach is better than using QGraphicsItem::ItemIgnoresTransformations to prevent horrible aliasing problems, since that approach involves complicated transformation matrix manipulations

Layering is used for the single graphics item contained by this class. External code only has to deal with this single class, and there is no multiple inheritance involved. If inheritance was used, we would have one class based on QGraphicsEllipseItem and another on QGraphicsPolygonItem, so having a single class (for the convenience of the

external code) would involve multiple inheritance (of those two classes). With the inheritance approach, using just the methods supplied by QGraphicsItem would be inadequate.

Definition at line 43 of file GraphicsPoint.h.

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

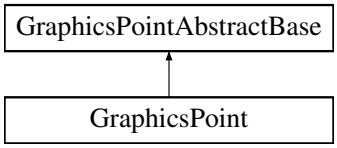
- Graphics/GraphicsPoint.h
- Graphics/GraphicsPoint.cpp

## 4.193 GraphicsPointAbstractBase Class Reference

Base class for adding identifiers to graphics items that represent Points.

`#include <GraphicsPointAbstractBase.h>`

Inheritance diagram for GraphicsPointAbstractBase:




### Public Member Functions

- GraphicsPointAbstractBase ()

  *Single constructor.*

### 4.193.1 Detailed Description

Base class for adding identifiers to graphics items that represent Points.

Identifiers are needed to distinguish which nodes are selected from those that are not selected. Each identifier is stored as a data item in QGraphicsItem.

This abstract base class no longer does anything.

Definition at line 18 of file GraphicsPointAbstractBase.h.

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

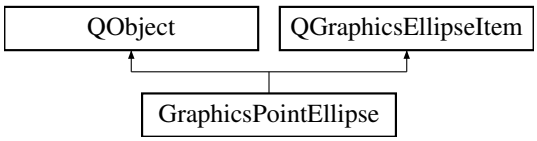
- Graphics/GraphicsPointAbstractBase.h
- Graphics/GraphicsPointAbstractBase.cpp

## 4.194 GraphicsPointEllipse Class Reference

This class add event handling to QGraphicsEllipseItem.

`#include <GraphicsPointEllipse.h>`

Inheritance diagram for GraphicsPointEllipse:




### Signals

- void signalPointHoverEnter (QString)

  *Signal for geometry window to highlight the current point upon hover enter.*
- void signalPointHoverLeave (QString)

  *Signal for geometry window to unhighlight the current point upon hover leave.*

### Public Member Functions

- GraphicsPointEllipse (GraphicsPoint &graphicsPoint, const QRect &rect)

  *Single constructor.*
- QVariant itemChange (GraphicsItemChange change, const QVariant &value)

  *Intercept moves by dragging so moved items can be identified. This replaces unreliable hit tests.*
- virtual void hoverEnterEvent (QGraphicsSceneHoverEvent ∗event)

  *Accept hover so point can be highlighted when cursor is over it as a guide to user.*
- virtual void hoverLeaveEvent (QGraphicsSceneHoverEvent ∗event)

  *Unhighlight this point.*
- void setRadius (int radius)

  *Update the radius.*
- void setShadow (GraphicsPointEllipse ∗shadow)

  *Bind this graphics item to its shadow.*

### 4.194.1 Detailed Description

This class add event handling to QGraphicsEllipseItem.

Definition at line 17 of file GraphicsPointEllipse.h.

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

- Graphics/GraphicsPointEllipse.h
- Graphics/GraphicsPointEllipse.cpp

## 4.195 GraphicsPointFactory Class Reference

Factor for generating GraphicsPointAbstractBase class objects.

`#include <GraphicsPointFactory.h>`

### Public Member Functions

- GraphicsPointFactory ()

  *Single constructor.*
- GraphicsPoint * createPoint (QGraphicsScene &scene, const QString &identifier, const QPointF &posScreen, const PointStyle &pointStyle, GeometryWindow *geometryWindow)

  *Create circle or polygon point according to the PointStyle.*

### 4.195.1 Detailed Description

Factor for generating GraphicsPointAbstractBase class objects.

Definition at line 19 of file GraphicsPointFactory.h.

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

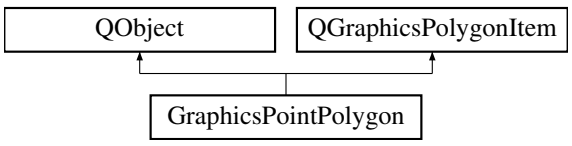
- Graphics/GraphicsPointFactory.h
- Graphics/GraphicsPointFactory.cpp

## 4.196 GraphicsPointPolygon Class Reference

This class add event handling to QGraphicsPolygonItem.

`#include <GraphicsPointPolygon.h>`

Inheritance diagram for GraphicsPointPolygon:




### Signals

- void signalPointHoverEnter (QString)

  *Signal for geometry window to highlight the current point upon hover enter.*
- void signalPointHoverLeave (QString)

  *Signal for geometry window to unhighlight the current point upon hover leave.*

### Public Member Functions

- GraphicsPointPolygon (GraphicsPoint &graphicsPoint, const QPolygonF &polygon)

  *Single constructor.*
- QVariant itemChange (GraphicsItemChange change, const QVariant &value)

  *Intercept moves by dragging so moved items can be identified. This replaces unreliable hit tests.*
- virtual void hoverEnterEvent (QGraphicsSceneHoverEvent *event)

  *Accept hover so point can be highlighted when cursor is over it as a guide to user.*
- virtual void hoverLeaveEvent (QGraphicsSceneHoverEvent *event)

  *Unhighlight this point.*
- void setRadius (int radius)

  *Update the radius.*
- void setShadow (GraphicsPointPolygon *shadow)

  *Bind this graphics item to its shadow.*

#### 4.196.1 Detailed Description

This class add event handling to QGraphicsPolygonItem.

Definition at line 17 of file GraphicsPointPolygon.h.

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

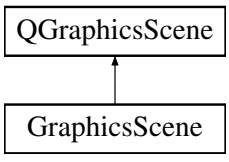
- Graphics/GraphicsPointPolygon.h
- Graphics/GraphicsPointPolygon.cpp

## 4.197 GraphicsScene Class Reference

Add point and line handling to generic QGraphicsScene.

```
#include <GraphicsScene.h>
```

Inheritance diagram for GraphicsScene:



### Public Member Functions

- GraphicsScene (MainWindow *mainWindow)

  *Single constructor.*
- void addTemporaryPoint (const QString &identifier, GraphicsPoint *point)

  *Add one temporary point to m_graphicsLinesForCurves. Non-temporary points are handled by the updateLine↩ Membership functions.*
- void addTemporaryScaleBar (GraphicsPoint *point0, GraphicsPoint *point1, const QString &pointIdentifier0, const QString &pointIdentifier1)

  *Add temporary scale bar to scene.*
- GraphicsPoint * createPoint (const QString &identifier, const PointStyle &pointStyle, const QPointF &pos↩ Screen, GeometryWindow *geometryWindow)

  *Create one QGraphicsItem-based object that represents one Point. It is NOT added to m_graphicsLinesForCurves (see addPoint)*
- void hideAllItemsExceptImage ()

  *Hide all graphics items, except background image, in preparation for preview during IMPORT_TYPE_ADVANCED.*
- QStringList positionHasChangedPointIdentifiers () const

  *Return a list of identifiers for the points that have moved since the last call to resetPositionHasChanged.*
- void printStream (QString indentation, QTextStream &str)

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void removePoint (const QString &identifier)

  *Remove specified point. This aborts if the point does not exist.*
- void removeTemporaryPointIfExists ()

  *Remove temporary point if it exists.*
- void removeTemporaryScaleBarIfExists ()

  *Remove temporary scale bar, composed of two points and the line between them.*
- void resetOnLoad ()

  *Reset, when loading a document after the first, to same state that first document was at when loaded.*
- void resetPositionHasChangedFlags ()

  *Reset positionHasChanged flag for all items. Typically this is done as part of mousePressEvent.*
- void showCurves (bool show, bool showAll=false, const QString &curveName="")

  *Show or hide all Curves (if showAll is true) or just the selected Curve (if showAll is false);.*
- void updateAfterCommand (CmdMediator &cmdMediator, double highlightOpacity, GeometryWindow *geometryWindow)

  *Update the Points and their Curves after executing a command.*
- void updateCurveStyles (const CurveStyles &modelCurveStyles)

  *Update curve styles after settings changed.*
- void updateGraphicsLinesToMatchGraphicsPoints (const CurveStyles &modelCurveStyles, const Transformation &transformation)

  *A mouse move has just occurred so move the selected points, since they were dragged.*

#### 4.197.1 Detailed Description

Add point and line handling to generic QGraphicsScene.

The primary tasks are:

1. update the graphics items to stay in sync with the explicit Points in the Document
2. update the graphics items to stay in sync with the implicit lines between the Points, according to Document settings

This class stores points and lines as QGraphicsItems, but also maintains identifier-to-QGraphicsItems mappings to the points and lines are accessible for updates (like when dragging points around and we need to update the attached lines).

Definition at line 33 of file GraphicsScene.h.

### 4.197.2 Member Function Documentation

#### 4.197.2.1 addTemporaryScaleBar()

```
void GraphicsScene::addTemporaryScaleBar (
            GraphicsPoint * point0,
            GraphicsPoint * point1,
            const QString & pointIdentifier0,
            const QString & pointIdentifier1 )
```

Add temporary scale bar to scene.

The scale bar is different from points and lines (always a complete set of 2 points and one line, and drawn using different point and line styles) that it is handled outside m_graphisLinesForCurves

Definition at line 45 of file GraphicsScene.cpp.

#### 4.197.2.2 removeTemporaryPointIfExists()

```
void GraphicsScene::removeTemporaryPointIfExists ( )
```

Remove temporary point if it exists.

Temporary point handling is so complicated that this method quietly allows redundant calls to this method, without complaining that the point has already been removed when called again

Definition at line 187 of file GraphicsScene.cpp.

#### 4.197.2.3 updateAfterCommand()

```
void GraphicsScene::updateAfterCommand (
            CmdMediator & cmdMediator,
            double highlightOpacity,
            GeometryWindow * geometryWindow )
```

Update the Points and their Curves after executing a command.

After a mouse drag, the lines are already updated and updating would be done on out of date information (since that would be brought up to date by the NEXT command)

Definition at line 270 of file GraphicsScene.cpp.

#### 4.197.2.4 updateGraphicsLinesToMatchGraphicsPoints()

```
void GraphicsScene::updateGraphicsLinesToMatchGraphicsPoints (
            const CurveStyles & modelCurveStyles,
            const Transformation & transformation )
```

A mouse move has just occurred so move the selected points, since they were dragged.

The transformation is needed so the screen coordinates can be converted to graph coordinates when updating point ordinals

Definition at line 305 of file GraphicsScene.cpp.

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

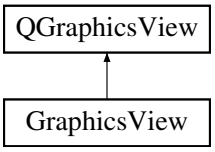
- Graphics/GraphicsScene.h
- Graphics/GraphicsScene.cpp

## 4.198 GraphicsView Class Reference

QGraphicsView class with event handling added. Typically the events are sent to the active digitizing state.

`#include <GraphicsView.h>`

Inheritance diagram for GraphicsView:




### Signals

- void signalContextMenuEventAxis (QString pointIdentifier)

  *Send right click on axis point to MainWindow for editing.*
- void signalContextMenuEventGraph (QStringList pointIdentifiers)

  *Send right click on graph point(s) to MainWindow for editing.*
- void signalDraggedDigFile (QString)

  *Send dragged dig file to MainWindow for import. This comes from dragging an engauge dig file.*
- void signalDraggedImage (QImage)

  *Send dragged image to MainWindow for import. This typically comes from dragging a file.*
- void signalDraggedImageUrl (QUrl)

  *Send dragged url to MainWindow for import. This typically comes from dragging an image from a browser.*
- void signalKeyPress (Qt::Key, bool atLeastOneSelectedItem)

  *Send keypress to MainWindow for eventual processing by DigitizeStateAbstractBase subclasses.*
- void signalMouseMove (QPointF)

  *Send mouse move to MainWindow for eventual display of cursor coordinates in StatusBar.*
- void signalMousePress (QPointF)

  *Send mouse press to MainWindow for creating one or more Points.*
- void signalMouseRelease (QPointF)

  *Send mouse release to MainWindow for moving Points.*
- void signalViewZoomIn ()

  *Send wheel event to MainWindow for zooming in.*
- void signalViewZoomOut ()

  *Send wheel event to MainWindow for zooming out.*

### Public Member Functions

- GraphicsView (QGraphicsScene ∗scene, MainWindow &mainWindow)

  *Single constructor.*
- virtual void contextMenuEvent (QContextMenuEvent ∗event)

  *Intercept context event to support point editing.*
- virtual void dragEnterEvent (QDragEnterEvent ∗event)

  *Intercept mouse drag event to support drag-and-drop.*
- virtual void dragMoveEvent (QDragMoveEvent ∗event)

  *Intercept mouse move event to support drag-and-drop.*
- virtual void dropEvent (QDropEvent ∗event)

  *Intercept mouse drop event to support drag-and-drop. This initiates asynchronous loading of the dragged image.*
- virtual void keyPressEvent (QKeyEvent ∗event)

  *Intercept key press events to handle left/right/up/down moving.*
- virtual void mouseMoveEvent (QMouseEvent ∗event)

  *Intercept mouse move events to populate the current cursor position in StatusBar.*
- virtual void mousePressEvent (QMouseEvent ∗event)

  *Intercept mouse press events to create one or more Points.*
- virtual void mouseReleaseEvent (QMouseEvent ∗event)

  *Intercept mouse release events to move one or more Points.*
- virtual void wheelEvent (QWheelEvent ∗event)

  *Convert wheel events into zoom in/out.*

#### 4.198.1 Detailed Description

QGraphicsView class with event handling added. Typically the events are sent to the active digitizing state.

Definition at line 20 of file GraphicsView.h.

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

- Graphics/GraphicsView.h
- Graphics/GraphicsView.cpp

## 4.199 GridClassifier Class Reference

Classify the grid pattern in an original image.

```
#include <GridClassifier.h>
```

### Public Member Functions

- GridClassifier ()

  *Single constructor.*
- void classify (bool isGnuplot, const QPixmap &originalPixmap, const Transformation &transformation, int &countX, double &startX, double &stepX, int &countY, double &startY, double &stepY)

  *Classify the specified image, and return the most probably x and y grid settings.*

### 4.199.1 Detailed Description

Classify the grid pattern in an original image.

This class uses the following tricks for faster performance:

1. FFT is used for "fast correlations" in frequency space rather than graph space
2. FFT initialization/shutdown housekeeping is done once
3. Rather than a combinatorial search of grid line start, step and count, we exploit the periodicity of the FFT to search start and step as the first step, and then as a separate second step we search count. In the first step, the periodicity means the repeating grid lines wrap around the end of the end of the image back around to the start of the image - so the grid line count is not even relevant. In other words, the searches are START X STEP + COUNT rather than START X STEP X COUNT

Definition at line 26 of file GridClassifier.h.

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

- Grid/GridClassifier.h
- Grid/GridClassifier.cpp

## 4.200 GridHealer Class Reference

Class that 'heals' the curves after grid lines have been removed.

`#include <GridHealer.h>`

### Public Member Functions

- GridHealer (const QImage &imageBefore, const DocumentModelGridRemoval &modelGridRemoval)
  *Single constructor.*
- void erasePixel (int xCol, int yRow)
  *Remember that pixel was erased since it belongs to an grid line.*
- void heal (QImage &imageToHeal)
  *Heal the broken curve lines by spanning the gaps across the newly-removed grid lines.*

### 4.200.1 Detailed Description

Class that 'heals' the curves after grid lines have been removed.

Specifically, gaps that span the pixels in the removed grid lines are filled in, if they are less than some epsilon value

Definition at line 37 of file GridHealer.h.

### 4.200.2 Member Function Documentation

#### 4.200.2.1 erasePixel()

```
void GridHealer::erasePixel (
            int xCol,
            int yRow )
```

Remember that pixel was erased since it belongs to an grid line.

In the image, erasure correponds to a foreground pixel being changed to the background color

Definition at line 96 of file GridHealer.cpp.

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

- Grid/GridHealer.h
- Grid/GridHealer.cpp

## 4.201 GridInitializer Class Reference

This class initializes the count, start, step and stop parameters for one coordinate (either x/theta or y/range)

```
#include <GridInitializer.h>
```

### Public Member Functions

- GridInitializer ()

  *Single constructor.*
- int computeCount (bool linearAxis, double start, double stop, double step) const

  *Compute axis scale count from the other axis parameters.*
- double computeStart (bool linearAxis, double stop, double step, int count) const

  *Compute axis scale start from the other axis parameters.*
- double computeStep (bool linearAxis, double start, double stop, int count) const

  *Compute axis scale step from the other axis parameters.*
- double computeStop (bool linearAxis, double start, double step, int count) const

  *Compute axis scale stop from the other axis parameters.*
- DocumentModelGridDisplay initializeWithNarrowCoverage (const QRectF &boundingRectGraph, const DocumentModelCoords &modelCoords) const

  *Initialize given the boundaries of the graph coordinates. The output is useful for the Checker class.*
- DocumentModelGridDisplay initializeWithWidePolarCoverage (const QRectF &boundingRectGraph, const DocumentModelCoords &modelCoords, const Transformation &transformation, const QSize &imageSize) const

  *Initialize given the boundaries of the graph coordinates, and then extra processing for polar coordinates:*
  1. *radial range expanded to cover the center (to remove hole at center) to the image corners (to guarantee coverage at corners of graph)*
  2. *angular range is expanded to cover the entire circle (so coverage is total for all directions)*
- int valuePower (double value) const

  *Compute power of 10 for input value, rounding down to nearest integer solution of value>=10**solution.*

### 4.201.1 Detailed Description

This class initializes the count, start, step and stop parameters for one coordinate (either x/theta or y/range)

Definition at line 13 of file GridInitializer.h.

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

- Grid/GridInitializer.h
- Grid/GridInitializer.cpp

## 4.202 GridLine Class Reference

Single grid line drawn a straight or curved line.

`#include <GridLine.h>`

### Public Member Functions

- GridLine ()

  *Default constructor for storage in containers.*
- GridLine (const GridLine &other)

  *Copy constructor. This will assert if called since copying of pointer containers is problematic.*
- GridLine & operator= (GridLine &other)

  *Assignment constructor. This will assert if called since copying of pointer containers is problematic.*
- void add (QGraphicsItem *item)

  *Add graphics item which represents one segment of the line.*
- void setPen (const QPen &pen)

  *Set the pen style.*
- void setVisible (bool visible)

  *Set each grid line as visible or hidden.*

### 4.202.1 Detailed Description

Single grid line drawn a straight or curved line.

This is expected to be composed of QGraphicsEllipseItem and QGraphicsLineItem objects

Definition at line 20 of file GridLine.h.

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

- Grid/GridLine.h
- Grid/GridLine.cpp

## 4.203 GridLineFactory Class Reference

Factory class for generating the points, composed of QGraphicsItem objects, along a GridLine.

`#include <GridLineFactory.h>`

### Public Member Functions

- GridLineFactory (QGraphicsScene &scene, const DocumentModelCoords &modelCoords)

  *Simple constructor for general use (i.e. not by Checker)*
- GridLineFactory (QGraphicsScene &scene, int pointRadius, const QList< Point > &pointsToIsolate, const DocumentModelCoords &modelCoords)

  *Constructor for use by Checker, which has points that are isolated.*
- GridLine * createGridLine (double xFrom, double yFrom, double xTo, double yTo, const Transformation &transformation)

  *Create grid line, either along constant X/theta or constant Y/radius side.*
- void createGridLinesForEvenlySpacedGrid (const DocumentModelGridDisplay &modelGridDisplay, const Document &document, const MainWindowModel &modelMainWindow, const Transformation &transformation, GridLines &gridLines)

  *Create a rectangular (cartesian) or annular (polar) grid of evenly spaced grid lines.*

### 4.203.1 Detailed Description

Factory class for generating the points, composed of QGraphicsItem objects, along a GridLine.

For polar coordinates, the grid lines will appear as an annular segments.

For the Checker class, a set of Points can be specified which will be isolated by having grid lines stop at a specified distance (or radius) from each point

Definition at line 29 of file GridLineFactory.h.

### 4.203.2 Member Function Documentation

#### 4.203.2.1 createGridLine()

```
GridLine * GridLineFactory::createGridLine (
            double xFrom,
            double yFrom,
            double xTo,
            double yTo,
            const Transformation & transformation )
```

Create grid line, either along constant X/theta or constant Y/radius side.

Line goes from pointFromGraph to pointToGraph. If the coordinates are polar, we go clockwise from pointFrom↩ Graph to pointToGraph (as set up by adjustPolarAngleRange).

Definition at line 73 of file GridLineFactory.cpp.

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

- Grid/GridLineFactory.h
- Grid/GridLineFactory.cpp

## 4.204 GridLineLimiter Class Reference

Limit the number of grid lines so a bad combination of start/step/stop value will not lead to extremely long delays when the step size is much too small for the start/stop values.

`#include <GridLineLimiter.h>`

### Public Member Functions

- GridLineLimiter ()

  *Single constructor.*
- void limitForXTheta (const Document &document, const Transformation &transformation, const DocumentModelCoords &modelCoords, const MainWindowModel &modelMainWindow, const DocumentModelGridDisplay &modelGrid, double &startX, double &stepX, double &stopX) const

  *Limit step value for x/theta coordinate. This is a noop if the maximum grid line limit in MainWindowModel is not exceeded.*
- void limitForYRadius (const Document &document, const Transformation &transformation, const DocumentModelCoords &modelCoords, const MainWindowModel &modelMainWindow, const DocumentModelGridDisplay &modelGrid, double &startY, double &stepY, double &stopY) const

  *Limit step value for y/range coordinate. This is a noop if the maximum grid line limit in MainWindowModel is not exceeded.*

### 4.204.1 Detailed Description

Limit the number of grid lines so a bad combination of start/step/stop value will not lead to extremely long delays when the step size is much too small for the start/stop values.

Definition at line 23 of file GridLineLimiter.h.

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

- Grid/GridLineLimiter.h
- Grid/GridLineLimiter.cpp

## 4.205 GridLines Class Reference

Container class for GridLine objects.

`#include <GridLines.h>`

### Public Member Functions

- GridLines ()

  *Single constructor.*
- void add (GridLine *gridLine)

  *Add specified grid line. Ownership of all allocated QGraphicsItems is passed to new GridLine.*
- void clear ()

  *Deallocate and remove all grid lines.*
- void setPen (const QPen &pen)

  *Set the pen style of each grid line.*
- void setVisible (bool visible)

  *Make all grid lines visible or hidden.*

### 4.205.1 Detailed Description

Container class for GridLine objects.

Definition at line 18 of file GridLines.h.

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

- Grid/GridLines.h
- Grid/GridLines.cpp

## 4.206 GridRemoval Class Reference

Strategy class for grid removal.

`#include <GridRemoval.h>`

### Public Member Functions

- GridRemoval ()
  *Single constructor.*
- QPixmap remove (const Transformation &transformation, const DocumentModelGridRemoval &modelGrid↩Removal, const QImage &imageBefore)
  *Process QImage into QPixmap, removing the grid lines.*

### 4.206.1 Detailed Description

Strategy class for grid removal.

Definition at line 19 of file GridRemoval.h.

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

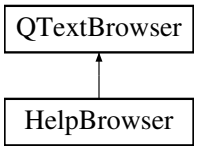
- Grid/GridRemoval.h
- Grid/GridRemoval.cpp

## 4.207 HelpBrowser Class Reference

Text browser with resource loading enhanced for use as help text browser.

`#include <HelpBrowser.h>`

Inheritance diagram for HelpBrowser:



### Public Member Functions

- HelpBrowser (QHelpEngine *engine, QWidget *parent=0)
  *Single constructor.*
- QVariant loadResource (int type, const QUrl &url)
  *Load resources.*

#### 4.207.1 Detailed Description

Text browser with resource loading enhanced for use as help text browser.

Definition at line 15 of file HelpBrowser.h.

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

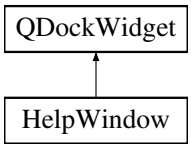
- Help/HelpBrowser.h
- Help/HelpBrowser.cpp

## 4.208 HelpWindow Class Reference

Dockable help window.

`#include <HelpWindow.h>`

Inheritance diagram for HelpWindow:




### Public Member Functions

- HelpWindow (QWidget *parent)
  *Single constructor.*

#### 4.208.1 Detailed Description

Dockable help window.

Despite a lot of work trying to work with the OSX sandbox, support for the sandbox was never completed since QHelpEngine requires WRITE-access to the collection file. Even trying to create a temporary directory does not work since copying would involve QHelpEngine::copyCollectionFile which copys from the CURRENT collection file (versus just some arbitrary file name)

Definition at line 16 of file HelpWindow.h.

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

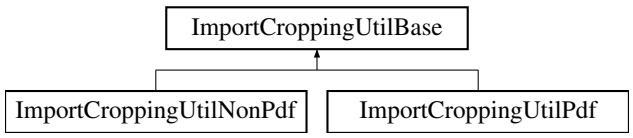
- Help/HelpWindow.h
- Help/HelpWindow.cpp

## 4.209 ImportCroppingUtilBase Class Reference

Utility class for import cropping options.

`#include <ImportCroppingUtilBase.h>`

Inheritance diagram for ImportCroppingUtilBase:




### Public Member Functions

- ImportCroppingUtilBase ()
  *Single constructor.*

### Static Public Member Functions

- static QString importCroppingToString (ImportCropping importCropping)
  *Option as string for display to user.*

### 4.209.1 Detailed Description

Utility class for import cropping options.

Default option is oldest, and simplest, behavior, which is no cropping.

A complication is that a dialog for cropping is not wanted during batch-mode regression testing, so this class and its subclasses offer methods for overriding the current setting during regression testing

Definition at line 17 of file ImportCroppingUtilBase.h.

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

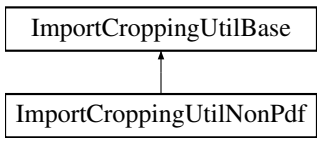
- Import/ImportCroppingUtilBase.h
- Import/ImportCroppingUtilBase.cpp

## 4.210 ImportCroppingUtilNonPdf Class Reference

Import of non-pdf files.

`#include <ImportCroppingUtilNonPdf.h>`

Inheritance diagram for ImportCroppingUtilNonPdf:



### Public Member Functions

- ImportCroppingUtilNonPdf ()

  *Single constructor.*
- bool applyImportCropping (bool isRegression, ImportCropping importCropping) const

  *Skip cropping dialog during regression testing, otherwise crop if it is always turned on.*

### Additional Inherited Members

#### 4.210.1 Detailed Description

Import of non-pdf files.

Definition at line 14 of file ImportCroppingUtilNonPdf.h.

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

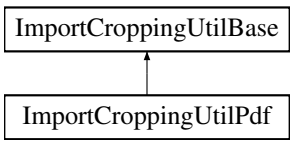
- Import/ImportCroppingUtilNonPdf.h
- Import/ImportCroppingUtilNonPdf.cpp

## 4.211 ImportCroppingUtilPdf Class Reference

Import of pdf files.

`#include <ImportCroppingUtilPdf.h>`

Inheritance diagram for ImportCroppingUtilPdf:




### Public Member Functions

- ImportCroppingUtilPdf ()

  *Single constructor.*
- bool applyImportCropping (bool isRegression, const QString &fileName, ImportCropping importCropping, Poppler::Document ∗&document) const

  *For pdf files, skip cropping dialog during regression testing, otherwise crop if it is always turned on or if there are multiple pages.*

### Additional Inherited Members

#### 4.211.1 Detailed Description

Import of pdf files.

Definition at line 19 of file ImportCroppingUtilPdf.h.

### 4.211.2 Member Function Documentation

#### 4.211.2.1 applyImportCropping()

```
bool ImportCroppingUtilPdf::applyImportCropping (
            bool isRegression,
            const QString & fileName,
            ImportCropping importCropping,
            Poppler::Document *& document ) const
```

For pdf files, skip cropping dialog during regression testing, otherwise crop if it is always turned on or if there are multiple pages.

For speed, the Document is returned if cropping is to be performed so the file needs to be read only once

Definition at line 17 of file ImportCroppingUtilPdf.cpp.

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

- Import/ImportCroppingUtilPdf.h
- Import/ImportCroppingUtilPdf.cpp

## 4.212 Jpeg2000 Class Reference

Wrapper around OpenJPEG library, in C, for opening jpeg2000 files.

```
#include <Jpeg2000.h>
```

### Public Member Functions

- Jpeg2000 ()
  *Single constructor.*
- bool load (const QString &filename, QImage &image) const
  *Load image from jpeg2000 file.*
- QStringList supportedImageWildcards () const
  *List the supported jpeg2000 file extensions, for filtering import files.*

### 4.212.1 Detailed Description

Wrapper around OpenJPEG library, in C, for opening jpeg2000 files.

Definition at line 26 of file Jpeg2000.h.

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

- Jpeg2000/Jpeg2000.h
- Jpeg2000/Jpeg2000.cpp

## 4.213 LinearToLog Class Reference

Warps log coordinates to make them linear before passing them to code that accepts only linear coordinates.

`#include <LinearToLog.h>`

### Public Member Functions

- double delinearize (double value, bool isLog) const

  *Convert linear coordinates to log. This is a noop if the output is supposed to be linear.*
- double linearize (double value, bool isLog) const

  *Convert log coordinates to linear. This is a noop if the input is already linear.*

### 4.213.1 Detailed Description

Warps log coordinates to make them linear before passing them to code that accepts only linear coordinates.

For example, the Spline funtions are already complicated (third order polynomial interpolation) so upgrading them to handle log coordinates in addition to linear coordinates would be painful to implement and debug

Definition at line 7 of file LinearToLog.h.

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

- util/LinearToLog.h
- util/LinearToLog.cpp

## 4.214 LineStyle Class Reference

Details for a specific Line.

`#include <LineStyle.h>`

### Public Member Functions

- LineStyle ()

  *Default constructor only for use when this class is being stored by a container that requires the default constructor.*
- LineStyle (unsigned int width, ColorPalette paletteColor, CurveConnectAs curveConnectAs)

  *Normal constructor.*
- LineStyle (const LineStyle &other)

  *Copy constructor.*
- LineStyle & operator= (const LineStyle &other)

  *Assignment operator.*
- CurveConnectAs curveConnectAs () const

  *Get method for connect type.*
- void loadXml (QXmlStreamReader &reader)

  *Load model from serialized xml. Returns the curve name.*

- ColorPalette paletteColor () const

  *Line color.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void saveXml (QXmlStreamWriter &writer) const

  *Serialize to stream.*
- void setCurveConnectAs (CurveConnectAs curveConnectAs)

  *Set connect as.*
- void setPaletteColor (ColorPalette paletteColor)

  *Set method for line color.*
- void setWidth (int width)

  *Set width of line.*
- unsigned int width () const

  *Width of line.*

### Static Public Member Functions

- static LineStyle defaultAxesCurve ()

  *Initial default for axes curve.*
- static LineStyle defaultGraphCurve (int index)

  *Initial default for index'th graph curve.*

#### 4.214.1 Detailed Description

Details for a specific Line.

Definition at line 19 of file LineStyle.h.

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

- Line/LineStyle.h
- Line/LineStyle.cpp

## 4.215 LoadFileInfo Class Reference

Returns information about files.

`#include <LoadFileInfo.h>`

### Public Member Functions

- LoadFileInfo ()

  *Single constructor.*
- bool loadsAsDigFile (const QString &urlString) const

  *Returns true if specified file name can be loaded as a DIG file.*

### 4.215.1 Detailed Description

Returns information about files.

Definition at line 13 of file LoadFileInfo.h.

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

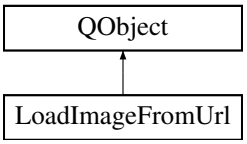
- Load/LoadFileInfo.h
- Load/LoadFileInfo.cpp

## 4.216 LoadImageFromUrl Class Reference

Load QImage from url. This is trivial for a file, but requires an asynchronous download step for http urls.

`#include <LoadImageFromUrl.h>`

Inheritance diagram for LoadImageFromUrl:




### Signals

- void signalImportImage (QString, QImage)

  *Send the imported image to MainWindow. This completes the asynchronous loading of the image.*

### Public Member Functions

- LoadImageFromUrl (MainWindow &mainWindow)

  *Single constructor.*
- void startLoadImage (const QUrl &url)

  *Start the asynchronous loading of an image from the specified url.*

### 4.216.1 Detailed Description

Load QImage from url. This is trivial for a file, but requires an asynchronous download step for http urls.

Definition at line 20 of file LoadImageFromUrl.h.

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

- Load/LoadImageFromUrl.h
- Load/LoadImageFromUrl.cpp

## 4.217 LoggerUpload Class Reference

Upload logging information to website for developer support.

`#include <LoggerUpload.h>`

### Public Member Functions

- LoggerUpload ()
  *Single constructor.*

### Static Public Member Functions

- static void bindToMainWindow (const MainWindow *mainWindow)
  *Bind to MainWindow so this class can access the command stack.*
- static void loggerAssert (const char *condition, const char *file, int line) NO_RETURN_VALUE
  *Smart equivalent to standard assert method and Q_ASSERT (in qglobal.h).*
- static void loggerCheckPtr (const char *pointer, const char *file, int line) NO_RETURN_VALUE
  *Smart equivalent to Q_CHECK_PTR (in qglobal.h). Similar to loggerAssert but for checking newly-allocated pointers.*

### 4.217.1 Detailed Description

Upload logging information to website for developer support.

Definition at line 21 of file LoggerUpload.h.

### 4.217.2 Member Function Documentation

#### 4.217.2.1 loggerAssert()

```
void LoggerUpload::loggerAssert (
            const char * condition,
            const char * file,
            int line )  [static]
```

Smart equivalent to standard assert method and Q_ASSERT (in qglobal.h).

Upon error, an upload is proposed. This is static for easy access from anywhere else in the application

Definition at line 22 of file LoggerUpload.cpp.

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

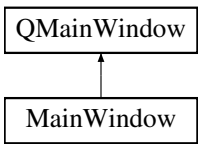
- Logger/LoggerUpload.h
- Logger/LoggerUpload.cpp

## 4.218 MainWindow Class Reference

Main window consisting of menu, graphics scene, status bar and optional toolbars as a Single Document Interface.

`#include <MainWindow.h>`

Inheritance diagram for MainWindow:




### Signals

- void signalZoom (int)

  *Send zoom selection, picked from menu or keystroke, to StatusBar.*

### Public Member Functions

- MainWindow (const QString &errorReportFile, const QString &fileCmdScriptFile, bool isRegressionTest, bool isGnuplot, bool isReset, QStringList loadStartupFiles, QWidget ∗parent=0)

  *Single constructor.*
- void cmdFileClose ()

  *Close file. This is called from a file script command.*
- void cmdFileExport (const QString &fileName)

  *Export file. This is called from a file script command.*
- void cmdFileImport (const QString &fileName)

  *Import file. This is called from a file script command.*
- void cmdFileOpen (const QString &fileName)

  *Open file. This is called from a file script command.*
- CmdMediator ∗ cmdMediator ()

  *Accessor for commands to process the Document.*
- virtual bool eventFilter (QObject ∗, QEvent ∗)

  *Catch secret keypresses.*
- QImage imageFiltered () const

  *Background image that has been filtered for the current curve. This asserts if a curve-specific image is not being shown.*
- bool isGnuplot () const

  *Get method for gnuplot flag.*
- MainWindowModel modelMainWindow () const

  *Get method for main window model.*
- void resizeEvent (QResizeEvent ∗event)

  *Intercept resize event so graphics scene can be appropriately resized when in Fill mode.*
- void saveErrorReportFileAndExit (const char ∗comment, const char ∗file, int line, const char ∗context) const

  *Save error report and exit.*
- GraphicsScene & scene ()

  *Scene container for the QImage and QGraphicsItems.*
- BackgroundImage selectOriginal (BackgroundImage backgroundImage)

*Make original background visible, for DigitizeStateColorPicker.*

- QString selectedGraphCurve () const

  *Curve name that is currently selected in m_cmbCurve.*
- virtual void showEvent (QShowEvent *)

  *Processing performed after gui becomes available.*
- void showTemporaryMessage (const QString &temporaryMessage)

  *Show temporary message in status bar.*
- Transformation transformation () const

  *Return read-only copy of transformation.*
- bool transformIsDefined () const

  *Return true if all three axis points have been defined.*
- void updateAfterCommand ()

  *See GraphicsScene::updateAfterCommand.*
- void updateAfterMouseRelease ()

  *Call MainWindow::updateControls (which is private) after the very specific case - a mouse press/release.*
- void updateCoordSystem (CoordSystemIndex coordSystemIndex)

  *Select a different CoordSystem.*
- void updateDigitizeStateIfSoftwareTriggered (DigitizeState digitizeState)

  *After software-triggered state transition, this method manually triggers the action as if user had clicked on digitize button.*
- void updateGraphicsLinesToMatchGraphicsPoints ()

  *Update the graphics lines so they follow the graphics points, after a drag, addition, removal, and such.*
- void updateSettingsAxesChecker (const DocumentModelAxesChecker &modelAxesChecker)

  *Update with new axes indicator properties.*
- void updateSettingsColorFilter (const DocumentModelColorFilter &modelColorFilter)

  *Update with new color filter properties.*
- void updateSettingsCoords (const DocumentModelCoords &modelCoords)

  *Update with new coordinate properties.*
- void updateSettingsCurveAddRemove (const CurvesGraphs &curvesGraphs)

  *Update with new curves.*
- void updateSettingsCurveStyles (const CurveStyles &modelCurveStyles)

  *Update with new curve styles.*
- void updateSettingsDigitizeCurve (const DocumentModelDigitizeCurve &modelDigitizeCurve)

  *Update with new curve digitization styles.*
- void updateSettingsExportFormat (const DocumentModelExportFormat &modelExport)

  *Update with new export properties.*
- void updateSettingsGeneral (const DocumentModelGeneral &modelGeneral)

  *Update with new general properties.*
- void updateSettingsGridDisplay (const DocumentModelGridDisplay &modelGridDisplay)

  *Update with new grid display properties.*
- void updateSettingsGridRemoval (const DocumentModelGridRemoval &modelGridRemoval)

  *Update with new grid removal properties.*
- void updateSettingsMainWindow (const MainWindowModel &modelMainWindow)

  *Update with new main window properties.*
- void updateSettingsPointMatch (const DocumentModelPointMatch &modelPointMatch)

  *Update with new point match properties.*
- void updateSettingsSegments (const DocumentModelSegments &modelSegments)

  *Update with new segments properties.*
- void updateViewsOfSettings (const QString &activeCurve)

  *Update curve-specific view of settings. Private version gets active curve name from DigitizeStateContext.*
- GraphicsView & view ()

  *View for the QImage and QGraphicsItems, without const.*
- const GraphicsView & view () const

  *View for the QImage and QGraphicsItems, without const.*

### 4.218.1 Detailed Description

Main window consisting of menu, graphics scene, status bar and optional toolbars as a Single Document Interface.

Definition at line 89 of file MainWindow.h.

### 4.218.2 Constructor & Destructor Documentation

#### 4.218.2.1 MainWindow()

```
MainWindow::MainWindow (
             const QString & errorReportFile,
             const QString & fileCmdScriptFile,
             bool isRegressionTest,
             bool isGnuplot,
             bool isReset,
             QStringList loadStartupFiles,
             QWidget * parent = 0 )
```

Single constructor.

**Parameters**

| | |
|---|---|
| *errorReportFile* | Optional error report file to be read at startup. Empty if unused. Incompatible with fileCmdScript |
| *fileCmdScriptFile* | Optional file command script file to be read at startup. Empty if unused. Incompatible with errorReportFile |
| *isRegressionTest* | True if errorReportFile or fileCmdScript is for regression testing, in which case it is executed and the program exits |
| *isGnuplot* | True if diagnostic gnuplot files are generated for math-intense sections of the code. Used for development and debugging |
| *isReset* | True to reset all settings that would otherwise be restored from the previous execution of Engauge |
| *loadStartupFiles* | Zero or more Engauge document files to load at startup. A separate instance of Engauge is created for each file |
| *parent* | Optional parent widget for this widget |

Definition at line 149 of file MainWindow.cpp.

### 4.218.3 Member Function Documentation

#### 4.218.3.1 selectOriginal()

```
BackgroundImage MainWindow::selectOriginal (
             BackgroundImage backgroundImage )
```

Make original background visible, for DigitizeStateColorPicker.

This returns the previous background state for restoring when state finishes

Definition at line 2557 of file MainWindow.cpp.

#### 4.218.3.2 updateGraphicsLinesToMatchGraphicsPoints()

```
void MainWindow::updateGraphicsLinesToMatchGraphicsPoints ( )
```

Update the graphics lines so they follow the graphics points, after a drag, addition, removal, and such.

The points in the Document may (and probably are) out of date with respect to the graphics points

Definition at line 4518 of file MainWindow.cpp.

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

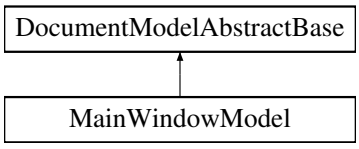
- main/MainWindow.h
- main/MainWindow.cpp

## 4.219 MainWindowModel Class Reference

Model for DlgSettingsMainWindow.

```
#include <MainWindowModel.h>
```

Inheritance diagram for MainWindowModel:



## Public Member Functions

- MainWindowModel ()

  *Default constructor.*
- MainWindowModel (const MainWindowModel &other)

  *Copy constructor.*
- MainWindowModel & operator= (const MainWindowModel &other)

  *Assignment constructor.*
- bool dragDropExport () const

  *Get method for drag and drop export.*
- virtual void loadXml (QXmlStreamReader &reader)

  *Load model from serialized xml.*
- double highlightOpacity () const

  *Get method for highlight opacity.*
- ImportCropping importCropping () const

  *Get method for import cropping.*
- QLocale locale () const

  *Get method for locale.*
- MainTitleBarFormat mainTitleBarFormat () const

  *Get method for MainWindow titlebar filename format.*
- int maximumGridLines () const

  *Maximum number of grid lines.*
- int pdfResolution () const

  *Get method for resolution of imported PDF files, in dots per inch.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- virtual void saveXml (QXmlStreamWriter &writer) const

  *Save entire model as xml into stream.*
- void setDragDropExport (bool dragDropExport)

  *Set method for drag and drop export.*
- void setHighlightOpacity (double highlightOpacity)

  *Set method for highlight opacity.*
- void setLocale (QLocale::Language language, QLocale::Country country)

  *Set method for locale given attributes.*
- void setLocale (const QLocale &locale)

  *Set method for locale given locale object.*
- void setImportCropping (ImportCropping importCropping)

  *Set method for import cropping.*
- void setMainTitleBarFormat (MainTitleBarFormat mainTitleBarFormat)

  *Set method for MainWindow titlebar filename format.*
- void setMaximumGridLines (int maximumGridLines)

  *Set method for maximum number of grid lines.*
- void setPdfResolution (int resolution)

  *Set method for resolution of imported PDF files, in dots per inch.*
- void setSmallDialogs (bool smallDialogs)

  *Set method for small dialogs flag.*
- void setZoomControl (ZoomControl zoomControl)

  *Set method for zoom control.*
- void setZoomFactorInitial (ZoomFactorInitial zoomFactorInitial)

  *Set method for initial zoom factor.*
- bool smallDialogs () const

*Get method for small dialogs flag.*

- ZoomControl zoomControl () const

  *Get method for zoom control.*

- ZoomFactorInitial zoomFactorInitial () const

  *Get method for initial zoom factor.*

### Additional Inherited Members

#### 4.219.1 Detailed Description

Model for DlgSettingsMainWindow.

Unlike the other models (DocumentModel∗) this data is not saved and loaded within the document, so no xml or working with the Document class is involved. Also, there is no associated Cmd. Instead, the settings are saved using QSettings. Method involving xml/Document (from DocumentModelAbstractBase) are stubbed out

Definition at line 27 of file MainWindowModel.h.

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

- main/MainWindowModel.h
- main/MainWindowModel.cpp

## 4.220 Matrix Class Reference

Matrix class that supports arbitrary NxN size.

```
#include <Matrix.h>
```

### Public Member Functions

- Matrix (int N)

  *Simple constructor of square matrix with initialization to identity matrix.*

- Matrix (int rows, int cols)

  *Simple constructor of rectangular matrix with initialization to zero matrix.*

- Matrix (const Matrix &other)

  *Copy constructor.*

- Matrix & operator= (const Matrix &matrix)

  *Assignment operator.*

- int cols () const

  *Width of matrix.*

- double determinant () const

  *Return the determinant of this matrix.*

- double get (int row, int col) const

  *Return (row, col) element.*

- Matrix inverse () const

  *Return the inverse of this matrix.*

- Matrix minorReduced (int rowOmit, int colOmit) const

*Return minor matrix which is the original with the specified row and column omitted. The name 'minor' is a reserved word.*

- Matrix operator∗ (const Matrix &other) const

  *Multiplication operator with a matrix.*
- QVector< double > operator∗ (const QVector< double > other) const

  *Multiplication operator with a vector.*
- int rows () const

  *Height of matrix.*
- void set (int row, int col, double value)

  *Set (row, col) element.*
- QString toString () const

  *Dump matrix to a string.*
- Matrix transpose () const

  *Return the transpose of the current matrix.*

### 4.220.1 Detailed Description

Matrix class that supports arbitrary NxN size.

Definition at line 14 of file Matrix.h.

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

- Matrix/Matrix.h
- Matrix/Matrix.cpp

## 4.221 MigrateToVersion6 Class Reference

Converts old (=pre version 6) enums to new (=version 6) enums, for reading of old document files.

`#include <MigrateToVersion6.h>`

### Public Member Functions

- MigrateToVersion6 ()

  *Single constructor.*
- ColorPalette colorPalette (int preVersion6) const

  *Color from color palette.*
- CurveConnectAs curveConnectAs (int preVersion6) const

  *Line drawn between points.*
- PointShape pointShape (int preVersion6) const

  *Point shape.*
- int pointRadius (int preVersion6) const

  *Point radius.*

### 4.221.1 Detailed Description

Converts old (=pre version 6) enums to new (=version 6) enums, for reading of old document files.

Definition at line 15 of file MigrateToVersion6.h.

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

- util/MigrateToVersion6.h
- util/MigrateToVersion6.cpp

## 4.222 MimePointsDetector Class Reference

Detect if text is acceptable for ingestion by MimePoints.

`#include <MimePointsDetector.h>`

### Public Member Functions

- MimePointsDetector ()
  *Default constructofr.*
- virtual ~MimePointsDetector ()
  *Destructor.*
- bool isMimePointsData (const Transformation &transforation, const QSize &screenSize) const
  *Returns true if text is acceptable mime data.*

### 4.222.1 Detailed Description

Detect if text is acceptable for ingestion by MimePoints.

Definition at line 18 of file MimePointsDetector.h.

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

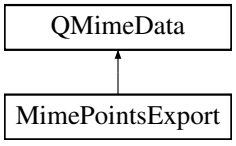
- Mime/MimePointsDetector.h
- Mime/MimePointsDetector.cpp

## 4.223 MimePointsExport Class Reference

Custom mime type for separate treatment of graph coordinates and, when there is no transform, points coordinates.

`#include <MimePointsExport.h>`

Inheritance diagram for MimePointsExport:



### Public Member Functions

- MimePointsExport ()

  *Default constructor. Initial contents are overwritten by other constructors.*
- MimePointsExport (const QString &csvGraph, const QString &htmlGraph)

  *Constructor when graph coordinates are available because the transformation is defined.*
- MimePointsExport (const QString &csvPoints)

  *Constructor when transformation is not defined. This data is not meant to leave this application.*
- MimePointsExport & operator= (const MimePointsExport &other)

  *Assignment operator.*
- virtual ~MimePointsExport ()

  *Destructor.*
- QString csvGraph () const

  *Get method for csvGraph.*
- QString csvPoints () const

  *Get method for csvPoints.*
- virtual QStringList formats () const

  *Available formats, which depend on whether or not the transform is defined.*
- QString htmlGraph () const

  *Get methjod for htmlGraph.*

### Protected Member Functions

- virtual QVariant retrieveData (const QString &format, QVariant::Type preferredType) const

  *Returns a variant with the data for the specified format.*

#### 4.223.1 Detailed Description

Custom mime type for separate treatment of graph coordinates and, when there is no transform, points coordinates.

Used for export only.

Definition at line 16 of file MimePointsExport.h.

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

- Mime/MimePointsExport.h
- Mime/MimePointsExport.cpp

## 4.224 MimePointsImport Class Reference

Import of point data from clipboard.

```
#include <MimePointsImport.h>
```

### Public Member Functions

- MimePointsImport ()

  *Default constructor.*
- virtual ~MimePointsImport ()

  *Destructor.*
- void retrievePoints (const Transformation &transformation, QList< QPoint > &points, QList< double > &ordinals) const

  *Retrieve points from clipboard.*

#### 4.224.1 Detailed Description

Import of point data from clipboard.

Definition at line 16 of file MimePointsImport.h.

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

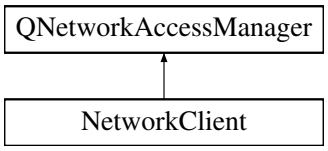
- Mime/MimePointsImport.h
- Mime/MimePointsImport.cpp

## 4.225 NetworkClient Class Reference

Client for interacting with Engauge server.

`#include <NetworkClient.h>`

Inheritance diagram for NetworkClient:




### Public Slots

- void slotFinished (QNetworkReply *)

  *Cleanup after response is returned.*

### Public Member Functions

- NetworkClient (QObject *parent)

  *Single constructor.*
- void uploadErrorReport (const QString &report)

  *Upload the error report asynchronously.*

### 4.225.1 Detailed Description

Client for interacting with Engauge server.

Definition at line 16 of file NetworkClient.h.

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

- Network/NetworkClient.h
- Network/NetworkClient.cpp

## 4.226 NonPdf Class Reference

Wrapper around the QImage class for read and importing non-PDF files.

`#include <NonPdf.h>`

### Public Member Functions

- NonPdf ()

  *Single constructor.*
- NonPdfReturn load (const QString &fileName, QImage &image, ImportCropping importCropping, bool isErrorReportRegressionTest) const

  *Try to load the specified file. Success is indicated in the function return value.*

### 4.226.1 Detailed Description

Wrapper around the QImage class for read and importing non-PDF files.

Definition at line 26 of file NonPdf.h.

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

- NonPdf/NonPdf.h
- NonPdf/NonPdf.cpp

## 4.227 NonPdfCropping Class Reference

This class shows a frame around the selected portion of the import preview window.

`#include <NonPdfCropping.h>`

### Public Member Functions

- NonPdfCropping (QGraphicsScene &scene, ViewPreview &view)
  *Single constructor.*
- QRectF frameRect () const
  *Frame rectangle selected by user.*
- void moveBL (const QPointF &newPos, const QPointF &oldPos)
  *Bottom left corner handle was moved.*
- void moveBR (const QPointF &newPos, const QPointF &oldPos)
  *Bottom right corner handle was moved.*
- void moveTL (const QPointF &newPos, const QPointF &oldPos)
  *Top left corner handle was moved.*
- void moveTR (const QPointF &newPos, const QPointF &oldPos)
  *Top right corner handle was moved.*
- QSize windowSize () const
  *Size of window in scene coordinates.*

### Static Public Attributes

- static const int NON_PDF_CROPPING_BOTTOM = 1
  *Bit flag when handle is aligned with bottom edge at reference point.*
- static const int NON_PDF_CROPPING_LEFT = 2
  *Bit flag when handle is aligned with left edge at reference point.*
- static const int NON_PDF_CROPPING_RIGHT = 4
  *Bit flag when handle is aligned with right edge at reference point.*
- static const int NON_PDF_CROPPING_TOP = 8
  *Bit flag when handle is aligned with top edge at reference point.*

#### 4.227.1 Detailed Description

This class shows a frame around the selected portion of the import preview window.

This class was developed as a non-pdf equivalent to the PdfCropping class. See that class for more details

Definition at line 22 of file NonPdfCropping.h.

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

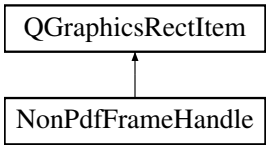
- NonPdf/NonPdfCropping.h
- NonPdf/NonPdfCropping.cpp

## 4.228 NonPdfFrameHandle Class Reference

This class acts as a single handle for the NonPdfCropping class.

```
#include <NonPdfFrameHandle.h>
```

Inheritance diagram for NonPdfFrameHandle:



### Public Member Functions

- NonPdfFrameHandle (QGraphicsScene &scene, QGraphicsView &view, const QPointF &pointReference, int orientationFlags, NonPdfCropping &nonPdfCropping, int zValue)

  *Single constructor.*
- virtual QVariant itemChange (GraphicsItemChange change, const QVariant &value)

  *Intercept the drags and process them, which is the whole point of handles.*
- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

  *Override the paint method so the dashed-border-when-selected can be removed.*
- void setDisableEventsWhileMovingAutomatically (bool disable)

  *Temporarily disable event handling so code can move this object without triggering a cascade of events.*

#### 4.228.1 Detailed Description

This class acts as a single handle for the NonPdfCropping class.

Definition at line 19 of file NonPdfFrameHandle.h.

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

- NonPdf/NonPdfFrameHandle.h
- NonPdf/NonPdfFrameHandle.cpp

## 4.229 OrdinalGenerator Class Reference

Utility class for generating ordinal numbers.

```
#include <OrdinalGenerator.h>
```

### Public Member Functions

- double generateAxisPointOrdinal (const Document &document)

  *Select ordinal just for uniqueness, since there is never a curve drawn through the axis points.*
- double generateCurvePointOrdinal (const Document &document, const Transformation &transformation, const QPointF &posScreen, const QString &curveName)

  *Select ordinal so new point curve passes smoothly through existing points.*

#### 4.229.1 Detailed Description

Utility class for generating ordinal numbers.

For axis points, the ordinals are arbitrary but must be unique. For curve points, point is inserted according to its position and the CurveStyle settings

Definition at line 18 of file OrdinalGenerator.h.

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

- Ordinal/OrdinalGenerator.h
- Ordinal/OrdinalGenerator.cpp

## 4.230 Pdf Class Reference

Wrapper around the Poppler library.

`#include <Pdf.h>`

### Public Member Functions

- Pdf ()
  *Single constructor.*
- PdfReturn load (const QString &fileName, QImage &image, int resolution, ImportCropping importCropping, bool isErrorReportRegressionTest) const
  *Try to load the specified file. Success is indicated in the function return value.*

### 4.230.1 Detailed Description

Wrapper around the Poppler library.

Engauge uses that library to read and import PDF files.

This class is only compiled and linked in when ENGAUGE_PDF is defined, since it links to the optional poppler library.

Definition at line 28 of file Pdf.h.

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

- Pdf/Pdf.h
- Pdf/Pdf.cpp

## 4.231 PdfCropping Class Reference

This class shows a frame around the selected portion of the pdf import preview window.

`#include <PdfCropping.h>`

### Public Member Functions

- PdfCropping (QGraphicsScene &scene, ViewPreview &view)
  *Single constructor.*
- QRectF frameRect () const
  *Frame rectangle selected by user.*
- void moveBL (const QPointF &newPos, const QPointF &oldPos)
  *Bottom left corner handle was moved.*
- void moveBR (const QPointF &newPos, const QPointF &oldPos)
  *Bottom right corner handle was moved.*
- void moveTL (const QPointF &newPos, const QPointF &oldPos)
  *Top left corner handle was moved.*
- void moveTR (const QPointF &newPos, const QPointF &oldPos)
  *Top right corner handle was moved.*
- QSize windowSize () const
  *Size of window in scene coordinates.*

### Static Public Attributes

- static const int PDF_CROPPING_BOTTOM = 1

  *Bit flag when handle is aligned with bottom edge at reference point.*
- static const int PDF_CROPPING_LEFT = 2

  *Bit flag when handle is aligned with left edge at reference point.*
- static const int PDF_CROPPING_RIGHT = 4

  *Bit flag when handle is aligned with right edge at reference point.*
- static const int PDF_CROPPING_TOP = 8

  *Bit flag when handle is aligned with top edge at reference point.*

#### 4.231.1 Detailed Description

This class shows a frame around the selected portion of the pdf import preview window.

Originally there were 4 handles at the corners and 4 handles at the middles of the sides, but dragging the corner handles did not result in 1/2 the movement at the middle handles. The middle handles were deemed not worth the effort

Definition at line 24 of file PdfCropping.h.

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

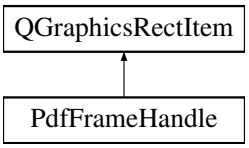
- Pdf/PdfCropping.h
- Pdf/PdfCropping.cpp

## 4.232 PdfFrameHandle Class Reference

This class acts as a single handle for the PdfCropping class.

`#include <PdfFrameHandle.h>`

Inheritance diagram for PdfFrameHandle:




### Public Member Functions

- PdfFrameHandle (QGraphicsScene &scene, QGraphicsView &view, const QPointF &pointReference, int orientationFlags, PdfCropping &pdfCropping, int zValue)

  *Single constructor.*
- virtual QVariant itemChange (GraphicsItemChange change, const QVariant &value)

  *Intercept the drags and process them, which is the whole point of handles.*
- virtual void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget)

  *Override the paint method so the dashed-border-when-selected can be removed.*
- void setDisableEventsWhileMovingAutomatically (bool disable)

  *Temporarily disable event handling so code can move this object without triggering a cascade of events.*

### 4.232.1 Detailed Description

This class acts as a single handle for the PdfCropping class.

Definition at line 19 of file PdfFrameHandle.h.

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

- Pdf/PdfFrameHandle.h
- Pdf/PdfFrameHandle.cpp

## 4.233 Point Class Reference

Class that represents one digitized point. The screen-to-graph coordinate transformation is always external to this class.

`#include <Point.h>`

### Public Member Functions

- Point ()

  *Default constructor so this class can be used inside a container.*
- Point (const QString &curveName, const QPointF &posScreen)

  *Constructor for Checker temporary points, before real point gets added.*
- Point (const QString &curveName, const QPointF &posScreen, const QPointF &posGraph, bool isXOnly)

  *Constructor for temporary point used to pre-check transformation points, before real point gets added.*
- Point (const QString &curveName, const QString &identifier, const QPointF &posScreen, const QPointF &posGraph, double ordinal, bool isXOnly)

  *Constructor for axis points with identifier (after redo). The position, in screen coordinates, applies to the center of the Point.*
- Point (const QString &curveName, const QPointF &posScreen, const QPointF &posGraph, double ordinal, bool isXOnly)

  *Constructor for axis points without identifier (after redo). The position, in screen coordinates, applies to the center of the Point.*
- Point (const QString &curveName, const QString &identifier, const QPointF &posScreen, double ordinal)

  *Constructor for graph points with identifier (after redo)*
- Point (const QString &curveName, const QPointF &posScreen, double ordinal)

  *Constructor for graph points without identifier (after redo)*
- Point (QXmlStreamReader &reader)

  *Constructor when loading from serialized xml.*
- Point & operator= (const Point &point)

  *Assignment constructor.*
- Point (const Point &point)

  *Copy constructor.*
- bool hasOrdinal () const

  *True if ordinal is defined.*
- bool hasPosGraph () const

  *True if graph position is defined.*
- QString identifier () const

*Unique identifier for a specific Point.*

- bool isXOnly () const

  *In DOCUMENT_AXES_POINTS_REQUIRED_4 modes, this is true/false if y/x coordinate is undefined.*
- bool isAxisPoint () const

  *True if point is an axis point. This is used only for sanity checks.*
- double ordinal (ApplyHasCheck applyHasCheck=KEEP_HAS_CHECK) const

  *Get method for ordinal. Skip check if copying one instance to another.*
- QPointF posGraph (ApplyHasCheck applyHasCheck=KEEP_HAS_CHECK) const

  *Accessor for graph position. Skip check if copying one instance to another.*
- QPointF posScreen () const

  *Accessor for screen position.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void saveXml (QXmlStreamWriter &writer) const

  *Serialize to stream.*
- void setCurveName (const QString &curveName)

  *Update the point identifer to match the specified curve name.*
- void setOrdinal (double ordinal)

  *Set the ordinal used for ordering Points.*
- void setPosGraph (const QPointF &posGraph)

  *Set method for position in graph coordinates.*
- void setPosScreen (const QPointF &posScreen)

  *Set method for position in screen coordinates.*

### Static Public Member Functions

- static QString curveNameFromPointIdentifier (const QString &pointIdentifier)

  *Parse the curve name from the specified point identifier. This does the opposite of uniqueIdentifierGenerator.*
- static unsigned int identifierIndex ()

  *Return the current index for storage in case we need to reset it later while performing a Redo.*
- static void setIdentifierIndex (unsigned int identifierIndex)

  *Reset the current index while performing a Redo.*
- static QString temporaryPointIdentifier ()

  *Point identifier for temporary point that is used by DigitzeStateAxis.*
- static double UNDEFINED_ORDINAL ()

  *Get method for undefined ordinal constant.*

### 4.233.1 Detailed Description

Class that represents one digitized point. The screen-to-graph coordinate transformation is always external to this class.

Definition at line 23 of file Point.h.

### 4.233.2 Constructor & Destructor Documentation

#### 4.233.2.1 Point() [1/2]

```
Point::Point (
            const QString & curveName,
            const QPointF & posScreen )
```

Constructor for Checker temporary points, before real point gets added.

The position, in screen coordinates, applies to the center of the Point

Definition at line 33 of file Point.cpp.

#### 4.233.2.2 Point() [2/2]

```
Point::Point (
            const QString & curveName,
            const QPointF & posScreen,
            const QPointF & posGraph,
            bool isXOnly )
```

Constructor for temporary point used to pre-check transformation points, before real point gets added.

The position, in screen coordinates, applies to the center of the Point

Definition at line 52 of file Point.cpp.

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

- Point/Point.h
- Point/Point.cpp

## 4.234 PointComparator Struct Reference

Comparator for sorting Point class.

`#include <PointComparator.h>`

### Public Member Functions

- bool operator() (const Point &a, const Point &b) const
  *Comparison function used by qSort.*

### 4.234.1 Detailed Description

Comparator for sorting Point class.

Definition at line 13 of file PointComparator.h.

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

- Point/PointComparator.h

## 4.235 PointIdentifiers Class Reference

Hash table class that tracks point identifiers as the key, with a corresponding boolean value.

`#include <PointIdentifiers.h>`

### Public Member Functions

- PointIdentifiers ()
  *Single constructor.*
- bool contains (const QString &pointIdentifier) const
  *True if specified entry exists in the table.*
- int count () const
  *Number of entries.*
- QString getKey (int i) const
  *Get key for index.*
- bool getValue (const QString &pointIdentifier) const
  *Get value for key.*
- void loadXml (QXmlStreamReader &reader)
  *Load from serialized xml.*
- void saveXml (QXmlStreamWriter &writer) const
  *Serialize table to xml.*
- void setKeyValue (const QString &pointIdentifier, bool value)
  *Set key/value pair.*

### 4.235.1 Detailed Description

Hash table class that tracks point identifiers as the key, with a corresponding boolean value.

Definition at line 19 of file PointIdentifiers.h.

### 4.235.2 Member Function Documentation

#### 4.235.2.1 getKey()

```
QString PointIdentifiers::getKey (
            int i ) const
```

Get key for index.

This involves copying of all the keys and is therefore slower than using key lookup, so should not be used for extremely numerous point sets

Definition at line 33 of file PointIdentifiers.cpp.

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

- Point/PointIdentifiers.h
- Point/PointIdentifiers.cpp

## 4.236 PointMatchAlgorithm Class Reference

Algorithm returning a list of points that match the specified point.

`#include <PointMatchAlgorithm.h>`

### Public Member Functions

- PointMatchAlgorithm (bool isGnuplot)

  *Single constructor.*
- QList< QPoint > findPoints (const QList< PointMatchPixel > &samplePointPixels, const QImage &image↩ Processed, const DocumentModelPointMatch &modelPointMatch, const Points &pointsExisting)

  *Find points that match the specified sample point pixels. They are sorted by best-to-worst match.*

### 4.236.1 Detailed Description

Algorithm returning a list of points that match the specified point.

This returns a list of matches, from best to worst. This is executed in a separate QThread so the gui thread is not blocked

Definition at line 26 of file PointMatchAlgorithm.h.

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

- Point/PointMatchAlgorithm.h
- Point/PointMatchAlgorithm.cpp

## 4.237 PointMatchPixel Class Reference

Single on or off pixel out of the pixels that define the point match mode's candidate point.

`#include <PointMatchPixel.h>`

### Public Member Functions

- PointMatchPixel (int xOffset, int yOffset, bool pixelIsOn)

  *Single basic constructor.*
- PointMatchPixel (const PointMatchPixel &other)

  *Copy constructor.*
- PointMatchPixel & operator= (const PointMatchPixel &other)

  *Assignment operator.*
- bool pixelIsOn () const

  *True/false if pixel is on/off.*
- int xOffset () const

  *X position relative to the center of the point.*
- int yOffset () const

  *Y position relative to the center of the point.*

### 4.237.1 Detailed Description

Single on or off pixel out of the pixels that define the point match mode's candidate point.

Definition at line 13 of file PointMatchPixel.h.

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

- Point/PointMatchPixel.h
- Point/PointMatchPixel.cpp

## 4.238 PointMatchTriplet Class Reference

Representation of one matched point as produced from the point match algorithm.

`#include <PointMatchTriplet.h>`

### Public Member Functions

- PointMatchTriplet (int x, int y, double correlation)
  *Single constructor.*
- bool operator< (const PointMatchTriplet &other) const
  *Comparison operator for sorting lists of this class using qSort.*
- double correlation () const
  *Get method for correlation.*
- QPoint point () const
  *Return (x,y) coordinates as a point.*
- int x () const
  *Get method for x coordinate.*
- int y () const
  *Get method for y coordinate.*

### 4.238.1 Detailed Description

Representation of one matched point as produced from the point match algorithm.

Definition at line 13 of file PointMatchTriplet.h.

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

- Point/PointMatchTriplet.h
- Point/PointMatchTriplet.cpp

## 4.239 PointStyle Class Reference

Details for a specific Point.

`#include <PointStyle.h>`

## Public Member Functions

- PointStyle ()

  *Default constructor only for use when this class is being stored by a container that requires the default constructor.*
- PointStyle (PointShape pointShape, unsigned int radius, int lineWidth, ColorPalette paletteColor)

  *Normal constructor. The style type and radius are determined by the currently selected Curve.*
- PointStyle (const PointStyle &other)

  *Copy constructor.*
- PointStyle & operator= (const PointStyle &other)

  *Assignment constructor.*
- bool isCircle () const

  *Return true if point is a circle, otherwise it is a polygon. For a circle, the radius is important and no polygon is needed from this class.*
- int lineWidth () const

  *Get method for line width.*
- void loadXml (QXmlStreamReader &reader)

  *Load model from serialized xml. Returns the curve name.*
- ColorPalette paletteColor () const

  *Get method for point color.*
- QPolygonF polygon () const

  *Return the polygon for creating a QGraphicsPolygonItem. The size is determined by the radius.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- int radius () const

  *Radius of point. For a circle this is all that is needed to draw a circle. For a polygon, the radius determines the size of the polygon.*
- void saveXml (QXmlStreamWriter &writer) const

  *Serialize to stream.*
- void setLineWidth (int width)

  *Set method for line width.*
- void setPaletteColor (ColorPalette paletteColor)

  *Set method for point color.*
- void setRadius (int radius)

  *Set method for point radius.*
- void setShape (PointShape shape)

  *Set method for point shape.*
- PointShape shape () const

  *Get method for point shape.*

## Static Public Member Functions

- static PointStyle defaultAxesCurve ()

  *Initial default for axes curve.*
- static PointStyle defaultGraphCurve (int index)

  *Initial default for index'th graph curve.*

### 4.239.1 Detailed Description

Details for a specific Point.

Definition at line 20 of file PointStyle.h.

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

- Point/PointStyle.h
- Point/PointStyle.cpp

## 4.240 ScaleBarAxisPointsUnite Class Reference

Given a set of point identifiers, if a map is in effect (with its two axis endpoints) then both axis points must be handled as a unit.

`#include <ScaleBarAxisPointsUnite.h>`

### Public Member Functions

- ScaleBarAxisPointsUnite ()

  *Single constructor.*
- QStringList unite (CmdMediator *cmdMediator, const QStringList &pointIdentifiersIn) const

  *Add.*

### 4.240.1 Detailed Description

Given a set of point identifiers, if a map is in effect (with its two axis endpoints) then both axis points must be handled as a unit.

For that to happen, for maps this class looks for one axis endpoint in a point identifier list, and if just one is in the list then the other is also added

This class has absolutely no effect (=noop) when a map is not in effect, or a map is in effect but zero or both axis endpoints are in the list

Definition at line 21 of file ScaleBarAxisPointsUnite.h.

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

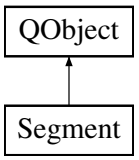
- ScaleBar/ScaleBarAxisPointsUnite.h
- ScaleBar/ScaleBarAxisPointsUnite.cpp

## 4.241 Segment Class Reference

Selectable piecewise-defined line that follows a filtered line in the image.

`#include <Segment.h>`

Inheritance diagram for Segment:




### Public Slots

- void slotHover (bool hover)

  *Slot for hover enter/leave events in the associated SegmentLines.*

### Signals

- void signalMouseClickOnSegment (QPointF posSegmentStart)

  *Pass mouse press event, with coordinates of first point in the Segment since that info uniquely identifies the owning Segment.*

### Public Member Functions

- Segment (QGraphicsScene &scene, int yLast, bool isGnuplot)

  *Single constructor.*
- void appendColumn (int x, int y, const DocumentModelSegments &modelSegments)

  *Add some more pixels in a new column to an active segment.*
- QList< QPoint > fillPoints (const DocumentModelSegments &modelSegments)

  *Create evenly spaced points along the segment.*
- QPointF firstPoint () const

  *Coordinates of first point in Segment.*
- void forwardMousePress ()

  *Forward mouse press event from a component SegmentLine that was just clicked on.*
- double length () const

  *Get method for length in pixels.*
- int lineCount () const

  *Get method for number of lines.*
- void removeUnneededLines (int *foldedLines)

  *Try to compress a segment that was just completed, by folding together line from point i to point i+1, with the line from i+1 to i+2, then the line from i+2 to i+3, until one of the points is more than a half pixel from the folded line.*
- void updateModelSegment (const DocumentModelSegments &modelSegments)

  *Update this segment given the new settings.*

### 4.241.1 Detailed Description

Selectable piecewise-defined line that follows a filtered line in the image.

Clicking on a Segment results in the immediate creation of multiple Points along that Segment.

Definition at line 21 of file Segment.h.

### 4.241.2 Member Function Documentation

#### 4.241.2.1 firstPoint()

```
QPointF Segment::firstPoint ( ) const
```

Coordinates of first point in Segment.

This info can be used to uniquely identify a Segment. This method relies on SegmentFactory::removeEmpty↩ Segments to guarantee every Segment has at least one line

Definition at line 285 of file Segment.cpp.

#### 4.241.2.2 removeUnneededLines()

```
void Segment::removeUnneededLines (
            int * foldedLines )
```

Try to compress a segment that was just completed, by folding together line from point i to point i+1, with the line from i+1 to i+2, then the line from i+2 to i+3, until one of the points is more than a half pixel from the folded line.

this should save memory and improve user interface responsiveness

Definition at line 420 of file Segment.cpp.

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

- Segment/Segment.h
- Segment/Segment.cpp

## 4.242 SegmentFactory Class Reference

Factory class for Segment objects.

```
#include <SegmentFactory.h>
```

### Public Member Functions

- SegmentFactory (QGraphicsScene &scene, bool isGnuplot)

  *Single constructor.*
- void clearSegments (QList< Segment *> &segments)

  *Remove the segments created by makeSegments.*
- QList< QPoint > fillPoints (const DocumentModelSegments &modelSegments, QList< Segment *> segments)

  *Return segment fill points for all segments, for previewing.*
- void makeSegments (const QImage &imageFiltered, const DocumentModelSegments &modelSegments, QList< Segment *> &segments, bool useDlg=true)

  *Main entry point for creating all Segments for the filtered image.*

#### 4.242.1 Detailed Description

Factory class for Segment objects.

The input is the filtered image.

The strategy is to fill out the segments output array as each segment finishes. This makes it easy to keep too-short Segments out of the output array, versus adding every new Segment to the output array as soon as it is created

Definition at line 27 of file SegmentFactory.h.

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

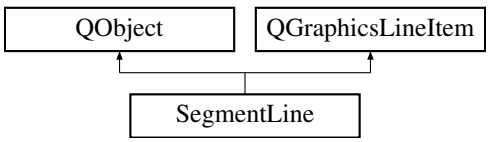
- Segment/SegmentFactory.h
- Segment/SegmentFactory.cpp

## 4.243 SegmentLine Class Reference

This class is a special case of the standard QGraphicsLineItem for segments.

`#include <SegmentLine.h>`

Inheritance diagram for SegmentLine:




### Signals

- void signalHover (bool)

  *Pass hover enter/leave events to Segment that owns this.*

### Public Member Functions

- SegmentLine (QGraphicsScene &scene, const DocumentModelSegments &modelSegments, Segment *segment)

  *Single constructor.*
- virtual void hoverEnterEvent (QGraphicsSceneHoverEvent *event)

  *Highlight this and all other SegmentLines belonging to the same Segment upon hover enter.*
- virtual void hoverLeaveEvent (QGraphicsSceneHoverEvent *event)

  *Unset highlighting triggered by hover enter.*
- virtual void mousePressEvent (QGraphicsSceneMouseEvent *event)

  *Create points along this curve.*
- Segment * segment () const

  *Segment that owns this line.*
- void setHover (bool hover)

  *Apply/remove highlighting triggered by hover enter/leave.*
- void updateModelSegment (const DocumentModelSegments &modelSegments)

  *Update this segment line with new settings.*

#### 4.243.1 Detailed Description

This class is a special case of the standard QGraphicsLineItem for segments.

Definition at line 17 of file SegmentLine.h.

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

- Segment/SegmentLine.h
- Segment/SegmentLine.cpp

## 4.244 SettingsForGraph Class Reference

Manage storage and retrieval of the settings for the curves.

`#include <SettingsForGraph.h>`

### Public Member Functions

- SettingsForGraph ()

  *Single constructor.*
- QString defaultCurveName (int indexOneBased, const QString &defaultName) const

  *Default graph name for the specified curve index.*
- QString groupNameForNthCurve (int indexOneBased) const

  *Return the group name, that appears in the settings file/registry, for the specified curve index.*
- int numberOfCurvesForImport () const

  *Return the number of curve names to be generated. Value is maximum of 1 and the number in the configuration file.*

### 4.244.1 Detailed Description

Manage storage and retrieval of the settings for the curves.

Definition at line 13 of file SettingsForGraph.h.

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

- Settings/SettingsForGraph.h
- Settings/SettingsForGraph.cpp

## 4.245 Spline Class Reference

Cubic interpolation given independent and dependent value vectors.

`#include <Spline.h>`

### Public Member Functions

- Spline (const std::vector< double > &t, const std::vector< SplinePair > &xy)
  *Initialize spline with independent (t) and dependent (x and y) value vectors.*
- SplinePair findSplinePairForFunctionX (double x, int numIterations) const
  *Use bisection algorithm to iteratively find the SplinePair interpolated to best match the specified x value.*
- SplinePair interpolateCoeff (double t) const
  *Return interpolated y for specified x.*
- SplinePair interpolateControlPoints (double t) const
  *Return interpolated y for specified x, for testing.*
- SplinePair p1 (unsigned int i) const
  *Bezier p1 control point for specified interval. P0 is m_xy[i] and P3 is m_xy[i+1].*
- SplinePair p2 (unsigned int i) const
  *Bezier p2 control point for specified interval. P0 is m_xy[i] and P3 is m_xy[i+1].*

### 4.245.1 Detailed Description

Cubic interpolation given independent and dependent value vectors.

X is handled as a dependent variable based on the unitless independent parameter t so curves are not restricted to x(i)!=x(i+1).

This class has two modes that can be run side by side:

1. Coefficient mode, where each interval has coefficients a,b,c,d in xy=ai+bi∗(t-ti)+ci∗(t-ti)^2+di∗(t-ti)^3
2. Bezier mode, where each interval has 2 endpoints and 2 control points in P=(1-s)^3∗P0+3∗(1-s)^2∗s∗↩ P1+3∗(1-s)∗s^2∗P2+s^3∗P3. Although interpolation can be performed in bezier mode, this interpolation was just to verify consistent operation between the two modes. The real purpose of this mode is to produce reliable control points, p1 and p2, for each interval. The control points can be used by external code that relies on control points to perform its own interpolation

Definition at line 21 of file Spline.h.

### 4.245.2 Constructor & Destructor Documentation

#### 4.245.2.1 Spline()

```
Spline::Spline (
            const std::vector< double > & t,
            const std::vector< SplinePair > & xy )
```

Initialize spline with independent (t) and dependent (x and y) value vectors.

Besides initializing the a,b,c,d coefficients for each interval, this constructor initializes bezier points (P1 and P2) for each interval, where P0 and P3 are the start and end points for each interval.

Definition at line 11 of file Spline.cpp.

### 4.245.3 Member Function Documentation

#### 4.245.3.1 findSplinePairForFunctionX()

```
SplinePair Spline::findSplinePairForFunctionX (
            double x,
            int numIterations ) const
```

Use bisection algorithm to iteratively find the SplinePair interpolated to best match the specified x value.

This assumes the curve is a function since otherwise there is the potential for multiple solutions

Definition at line 116 of file Spline.cpp.

#### 4.245.3.2 interpolateCoeff()

```
SplinePair Spline::interpolateCoeff (
            double t ) const
```

Return interpolated y for specified x.

The appropriate interval is selected from the entire set of piecewise-defined intervals, then the corresponding a,b,c,d coefficients are applied

Definition at line 166 of file Spline.cpp.

#### 4.245.3.3 interpolateControlPoints()

```
SplinePair Spline::interpolateControlPoints (
            double t ) const
```

Return interpolated y for specified x, for testing.

This uses the bezier points. If the t values are not separated by +1 consistently then this algorithm will probably need additional effort to work right

Definition at line 179 of file Spline.cpp.

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

- Spline/Spline.h
- Spline/Spline.cpp

## 4.246 SplineCoeff Class Reference

Four element vector of a,b,c,d coefficients and the associated x value, for one interval of a set of piecewise-defined intervals.

```
#include <SplineCoeff.h>
```

### Public Member Functions

- SplineCoeff (double t)
  *Partial constructor for use mostly by container classes.*
- SplineCoeff (double t, const SplinePair &a, const SplinePair &b, const SplinePair &c, const SplinePair &d)
  *Full constructor.*
- bool operator< (const SplineCoeff &e) const
  *Comparison operator for collection.*
- bool operator< (double t) const
  *Comparison operator for collection.*
- SplinePair a () const
  *Get method for a.*
- SplinePair b () const
  *Get method for b.*
- SplinePair c () const
  *Get method for c.*
- SplinePair d () const
  *Get method for d.*
- SplinePair eval (double t) const
  *Evaluate the value using the a,b,c,d coefficients, over this interval.*
- double t () const
  *T value associated with these a,b,c,d coefficients.*

### 4.246.1 Detailed Description

Four element vector of a,b,c,d coefficients and the associated x value, for one interval of a set of piecewise-defined intervals.

Definition at line 14 of file SplineCoeff.h.

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

- Spline/SplineCoeff.h
- Spline/SplineCoeff.cpp

## 4.247 SplinePair Class Reference

Single X/Y pair for cubic spline interpolation initialization and calculations.

`#include <SplinePair.h>`

### Public Member Functions

- SplinePair ()
  *Default constructor. Normally used only by generic container classes.*
- SplinePair (double scalar)
  *Constructor for filling vector with a single scalar. Provided for convenience over preferred constructor.*
- SplinePair (double x, double y)
  *Preferred constructor. Used when default constructor is not being used by generic container classes.*
- SplinePair (const SplinePair &other)
  *Assigment constructor.*
- SplinePair operator+ (const SplinePair &other) const
  *Addition operator.*
- SplinePair operator- (const SplinePair &other) const
  *Subtraction operator.*
- SplinePair operator∗ (const SplinePair &other) const
  *Multiplication operator.*
- SplinePair operator/ (const SplinePair &other) const
  *Division operator.*
- double x () const
  *Get method for x.*
- double y () const
  *Get method for y.*

### 4.247.1 Detailed Description

Single X/Y pair for cubic spline interpolation initialization and calculations.

Definition at line 11 of file SplinePair.h.

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

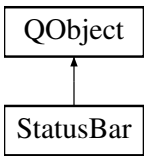
- Spline/SplinePair.h
- Spline/SplinePair.cpp

## 4.248 StatusBar Class Reference

Wrapper around QStatusBar to manage permanent widgets.

`#include <StatusBar.h>`

Inheritance diagram for StatusBar:




### Public Slots

- void slotZoom (int)

  *Receive zoom selection from MainWindow.*

### Signals

- void signalZoom (int)

  *Send zoom factor, that was just selected in the status bar, to MainWindow.*

### Public Member Functions

- StatusBar (QStatusBar &statusBar)

  *Single constructor that accepts the previously-constructed standard QStatusBar.*
- void setCoordinates (const QString &coordsScreen, const QString &coordsGraph, const QString &resolutionGraph)

  *Populate the coordinates fields. Unavailable values are empty. Html-encoding to highlight with colors is supported.*
- void setStatusBarMode (StatusBarMode statusBarMode)

  *Set the status bar visibility mode.*
- void showTemporaryMessage (const QString &message)

  *Show temporary message in status bar. After a short interval the message will disappear.*
- StatusBarMode statusBarMode () const

  *Current mode for status bar visibility. This is tracked locally so this class knows when to hide/show the status bar.*
- void wakeUp ()

  *Enable all widgets in the status bar. This is called just after a Document becomes active.*

### 4.248.1 Detailed Description

Wrapper around QStatusBar to manage permanent widgets.

This class does not inherit from QStatusBar since QApplication automatically sets up its own QStatusBar

Definition at line 24 of file StatusBar.h.

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

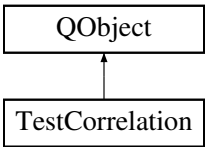
- StatusBar/StatusBar.h
- StatusBar/StatusBar.cpp

## 4.249 TestCorrelation Class Reference

Unit tests of fast correlation algorithm.

`#include <TestCorrelation.h>`

Inheritance diagram for TestCorrelation:




### Public Member Functions

- TestCorrelation (QObject *parent=0)
  *Single constructor.*

### 4.249.1 Detailed Description

Unit tests of fast correlation algorithm.

Definition at line 7 of file TestCorrelation.h.

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

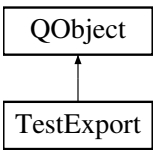
- Test/TestCorrelation.h
- Test/TestCorrelation.cpp

## 4.250 TestExport Class Reference

Unit test of Export classes.

`#include <TestExport.h>`

Inheritance diagram for TestExport:




### Public Member Functions

- TestExport (QObject *parent=0)
  *Single constructor.*

### 4.250.1 Detailed Description

Unit test of Export classes.

Definition at line 15 of file TestExport.h.

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

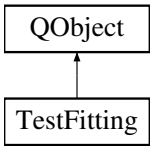
- Test/TestExport.h
- Test/TestExport.cpp

## 4.251 TestFitting Class Reference

Unit test of Fitting classes.

```
#include <TestFitting.h>
```

Inheritance diagram for TestFitting:




### Public Member Functions

- TestFitting (QObject *parent=0)

  *Single constructor.*

### 4.251.1 Detailed Description

Unit test of Fitting classes.

Definition at line 7 of file TestFitting.h.

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

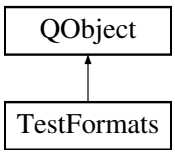
- Test/TestFitting.h
- Test/TestFitting.cpp

## 4.252 TestFormats Class Reference

Unit tests of formats.

`#include <TestFormats.h>`

Inheritance diagram for TestFormats:




### Public Member Functions

- TestFormats (QObject ∗parent=0)
  *Single constructor.*

### 4.252.1 Detailed Description

Unit tests of formats.

Definition at line 8 of file TestFormats.h.

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

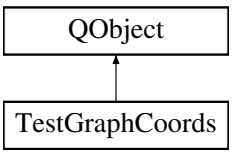
- Test/TestFormats.h
- Test/TestFormats.cpp

## 4.253 TestGraphCoords Class Reference

Unit tests of graph coordinate sanity checking.

`#include <TestGraphCoords.h>`

Inheritance diagram for TestGraphCoords:




### Public Member Functions

- TestGraphCoords (QObject ∗parent=0)
  *Single constructor.*

### 4.253.1 Detailed Description

Unit tests of graph coordinate sanity checking.

Definition at line 10 of file TestGraphCoords.h.

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

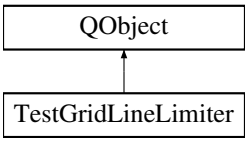
- Test/TestGraphCoords.h
- Test/TestGraphCoords.cpp

## 4.254 TestGridLineLimiter Class Reference

Unit test of GridLineLimiter class.

`#include <TestGridLineLimiter.h>`

Inheritance diagram for TestGridLineLimiter:




### Public Member Functions

- TestGridLineLimiter (QObject ∗parent=0)

  *Single constructor.*

### 4.254.1 Detailed Description

Unit test of GridLineLimiter class.

Definition at line 7 of file TestGridLineLimiter.h.

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

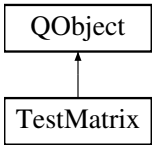
- Test/TestGridLineLimiter.h
- Test/TestGridLineLimiter.cpp

## 4.255 TestMatrix Class Reference

Unit tests of matrix.

`#include <TestMatrix.h>`

Inheritance diagram for TestMatrix:




### Public Member Functions

- TestMatrix (QObject ∗parent=0)
  *Single constructor.*

### 4.255.1 Detailed Description

Unit tests of matrix.

Definition at line 8 of file TestMatrix.h.

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

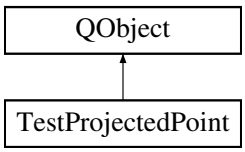
- Test/TestMatrix.h
- Test/TestMatrix.cpp

## 4.256 TestProjectedPoint Class Reference

Unit test of spline library.

`#include <TestProjectedPoint.h>`

Inheritance diagram for TestProjectedPoint:




### Public Member Functions

- TestProjectedPoint (QObject ∗parent=0)
  *Single constructor.*

### 4.256.1 Detailed Description

Unit test of spline library.

Definition at line 7 of file TestProjectedPoint.h.

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

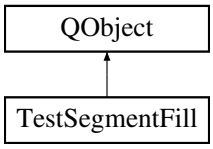
- Test/TestProjectedPoint.h
- Test/TestProjectedPoint.cpp

## 4.257 TestSegmentFill Class Reference

Unit test of segment fill feature.

`#include <TestSegmentFill.h>`

Inheritance diagram for TestSegmentFill:




### Public Member Functions

- TestSegmentFill (QObject *parent=0)

  *Single constructor.*

### 4.257.1 Detailed Description

Unit test of segment fill feature.

Definition at line 7 of file TestSegmentFill.h.

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

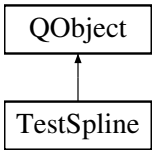
- Test/TestSegmentFill.h
- Test/TestSegmentFill.cpp

## 4.258 TestSpline Class Reference

Unit test of spline library.

`#include <TestSpline.h>`

Inheritance diagram for TestSpline:




### Public Member Functions

- TestSpline (QObject *parent=0)
  *Single constructor.*

### 4.258.1 Detailed Description

Unit test of spline library.

Definition at line 7 of file TestSpline.h.

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

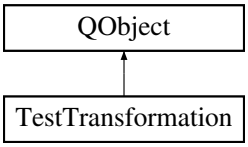
- Test/TestSpline.h
- Test/TestSpline.cpp

## 4.259 TestTransformation Class Reference

Unit test of transformation class. Checking mostly involves verifying forward/reverse are inverses of each other.

`#include <TestTransformation.h>`

Inheritance diagram for TestTransformation:




### Public Member Functions

- TestTransformation (QObject *parent=0)
  *Single constructor.*

### 4.259.1 Detailed Description

Unit test of transformation class. Checking mostly involves verifying forward/reverse are inverses of each other.

Definition at line 10 of file TestTransformation.h.

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

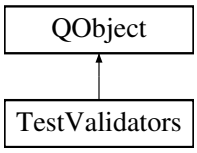
- Test/TestTransformation.h
- Test/TestTransformation.cpp

## 4.260 TestValidators Class Reference

Unit tests of validators.

```
#include <TestValidators.h>
```

Inheritance diagram for TestValidators:




### Public Member Functions

- TestValidators (QObject *parent=0)

  *Single constructor.*

### 4.260.1 Detailed Description

Unit tests of validators.

Definition at line 10 of file TestValidators.h.

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

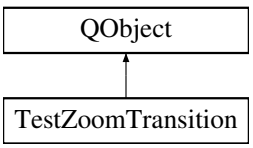
- Test/TestValidators.h
- Test/TestValidators.cpp

## 4.261 TestZoomTransition Class Reference

Unit test of ZoomTransition class.

`#include <TestZoomTransition.h>`

Inheritance diagram for TestZoomTransition:




### Public Member Functions

- TestZoomTransition (QObject *parent=0)

  *Single constructor.*

### 4.261.1 Detailed Description

Unit test of ZoomTransition class.

Definition at line 7 of file TestZoomTransition.h.

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

- Test/TestZoomTransition.h
- Test/TestZoomTransition.cpp

## 4.262 Transformation Class Reference

Affine transformation between screen and graph coordinates, based on digitized axis points.

`#include <Transformation.h>`

## Public Member Functions

- Transformation ()

  *Default constructor. This is marked as undefined until the proper number of axis points are added.*
- Transformation (const Transformation &other)

  *Copy constructor.*
- Transformation & operator= (const Transformation &other)

  *Assignment operator.*
- void identity ()

  *Identity transformation.*
- bool operator!= (const Transformation &other)

  *Inequality operator. This is marked as defined.*
- void coordTextForStatusBar (QPointF cursorScreen, QString &coordsScreen, QString &coordsGraph, QString &resolutionGraph, const QString &needMoreText)

  *Return string descriptions of cursor coordinates for status bar.*
- DocumentModelCoords modelCoords () const

  *Get method for DocumentModelCoords.*
- DocumentModelGeneral modelGeneral () const

  *Get method for DocumentModelGeneral.*
- MainWindowModel modelMainWindow () const

  *Get method for MainWindowModel.*
- void printStream (QString indentation, QTextStream &str) const

  *Debugging method that supports print method of this class and printStream method of some other class(es)*
- void resetOnLoad ()

  *Reset, when loading a document after the first, to same state that first document was at when loaded.*
- bool transformIsDefined () const

  *Transform is defined when at least three axis points have been digitized.*
- void transformLinearCartesianGraphToRawGraph (const QPointF &coordGraph, QPointF &coordScreen) const

  *Transform from linear cartesian graph coordinates to cartesian, polar, linear, log coordinates.*
- void transformLinearCartesianGraphToScreen (const QPointF &coordGraph, QPointF &coordScreen) const

  *Transform from linear cartesian graph coordinates to cartesian pixel screen coordinates.*
- QTransform transformMatrix () const

  *Get method for copying only, for the transform matrix.*
- void transformRawGraphToLinearCartesianGraph (const QPointF &pointRaw, QPointF &pointLinearCartesian) const

  *Convert graph coordinates (linear or log, cartesian or polar) to linear cartesian coordinates.*
- void transformRawGraphToScreen (const QPointF &pointRaw, QPointF &pointScreen) const

  *Transform from raw graph coordinates to linear cartesian graph coordinates, then to screen coordinates.*
- void transformScreenToLinearCartesianGraph (const QPointF &pointScreen, QPointF &pointLinearCartesian) const

  *Transform screen coordinates to linear cartesian coordinates.*
- void transformScreenToRawGraph (const QPointF &coordScreen, QPointF &coordGraph) const

  *Transform from cartesian pixel screen coordinates to cartesian/polar graph coordinates.*
- void update (bool fileIsLoaded, const CmdMediator &cmdMediator, const MainWindowModel &modelMainWindow)

  *Update transform by iterating through the axis points.*

### Static Public Member Functions

- static QTransform calculateTransformFromLinearCartesianPoints (const QPointF &posFrom0, const QPointF &posFrom1, const QPointF &posFrom2, const QPointF &posTo0, const QPointF &posTo1, const QPointF &posTo2)

  *Calculate QTransform using from/to points that have already been adjusted for, when applicable, log scaling and polar coordinates.*
- static QPointF cartesianFromCartesianOrPolar (const DocumentModelCoords &modelCoords, const QPointF &posGraphIn)

  *Output cartesian coordinates from input cartesian or polar coordinates. This is static for easier use externally.*
- static QPointF cartesianOrPolarFromCartesian (const DocumentModelCoords &modelCoords, const QPointF &posGraphIn)

  *Output cartesian or polar coordinates from input cartesian coordinates. This is static for easier use externally.*
- static double logToLinearCartesian (double xy)

  *Convert cartesian scaling from log to linear. Calling code is responsible for determining if this is necessary.*
- static double logToLinearRadius (double r, double rCenter)

  *Convert radius scaling from log to linear. Calling code is responsible for determining if this is necessary.*

### Friends

- class **TestExport**
- class **TestTransformation**

#### 4.262.1 Detailed Description

Affine transformation between screen and graph coordinates, based on digitized axis points.

Transformation from screen pixels to graph coordinates involves two steps:

1. Transform from screen pixels (which are linear and cartesian) to linear cartesian graph coordinates
2. Transform from linear cartesian graph coordinates to the final graph coordinates, which are linear or log scaled, and cartesian or polar

Transformation from graph coordinates to screen pixels reverses the steps involved in the transformation from screen pixels to graph coordinates

Log scaling is calculated as (xLinear - xLogMin) / (xLogMax - xLogMin) = (ln(xLog) - ln(xLogMin)) / (ln(xLogMax) - ln(xLogMin)), which leaves the points (xLogMin,yLonMin) and (xLogMax,yLogMax) unaffected but gives log growth on all other points

Definition at line 31 of file Transformation.h.

#### 4.262.2 Member Function Documentation

#### 4.262.2.1 calculateTransformFromLinearCartesianPoints()

```
QTransform Transformation::calculateTransformFromLinearCartesianPoints (
            const QPointF & posFrom0,
            const QPointF & posFrom1,
            const QPointF & posFrom2,
            const QPointF & posTo0,
            const QPointF & posTo1,
            const QPointF & posTo2 )  [static]
```

Calculate QTransform using from/to points that have already been adjusted for, when applicable, log scaling and polar coordinates.

The points are linear and cartesian.

This method is kept very generic since it used for different types of transformations:

1. In many place to calculate screen-to/from-graph
2. By Checker to calculate unaligned-to/from-aligned

Definition at line 59 of file Transformation.cpp.

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

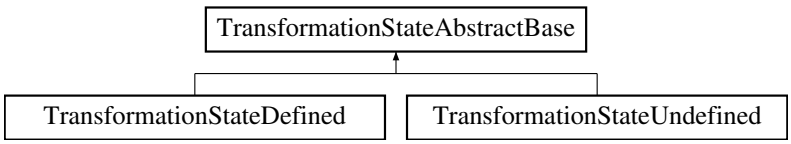
- Transformation/Transformation.h
- Transformation/Transformation.cpp

## 4.263 TransformationStateAbstractBase Class Reference

Base class for all transformation states. This serves as an interface to TransformationStateContext.

`#include <TransformationStateAbstractBase.h>`

Inheritance diagram for TransformationStateAbstractBase:




### Public Member Functions

- TransformationStateAbstractBase (TransformationStateContext &context)

  *Single constructor.*
- virtual void begin (CmdMediator &cmdMediator, const Transformation &transformation, const QString &selectedGraphCurve)=0

  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- virtual void end (CmdMediator &cmdMediator, const Transformation &transformation)=0

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void updateAxesChecker (CmdMediator &cmdMediator, const Transformation &transformation)=0

  *Apply the new DocumentModelAxesChecker.*

### Protected Member Functions

- TransformationStateContext & context ()

  *Reference to the TransformationStateContext that contains all the TransformationStateAbstractBase subclasses, without const.*

#### 4.263.1 Detailed Description

Base class for all transformation states. This serves as an interface to TransformationStateContext.

Definition at line 25 of file TransformationStateAbstractBase.h.

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

- Transformation/TransformationStateAbstractBase.h
- Transformation/TransformationStateAbstractBase.cpp

## 4.264 TransformationStateContext Class Reference

Context class for transformation state machine.

`#include <TransformationStateContext.h>`

### Public Member Functions

- TransformationStateContext (QGraphicsScene &scene, bool isGnuplot)

  *Single constructor.*
- bool isGnuplot () const

  *Flag for gnuplot debug files.*
- void resetOnLoad ()

  *Reset, when loading a document after the first, to same state that first document was at when loaded.*
- void triggerStateTransition (TransformationState transformationState, CmdMediator &cmdMediator, const Transformation &transformation, const QString &selectedGraphCurve)

  *Trigger a state transition to be performed immediately.*
- void updateAxesChecker (CmdMediator &cmdMediator, const Transformation &transformation)

  *Apply the new DocumentModelAxesChecker.*

#### 4.264.1 Detailed Description

Context class for transformation state machine.

This removes some tricky state processing from MainWindow. Unlike typical state machines, the transitions are driven directly from the outside rather than indirectly by events that are processed by the states (this has triggerStateTransition rather than requestStateTransition)

Definition at line 21 of file TransformationStateContext.h.

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

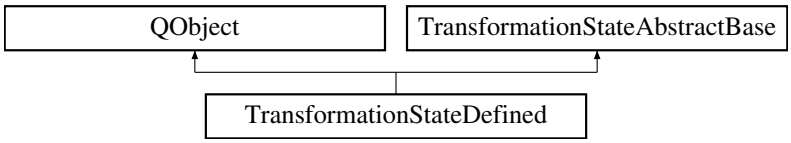
- Transformation/TransformationStateContext.h
- Transformation/TransformationStateContext.cpp

## 4.265 TransformationStateDefined Class Reference

Class to show transformation since transformation is defined.

`#include <TransformationStateDefined.h>`

Inheritance diagram for TransformationStateDefined:




### Public Member Functions

- TransformationStateDefined (TransformationStateContext &context, QGraphicsScene &scene)
  *Single constructor.*
- virtual void begin (CmdMediator &cmdMediator, const Transformation &transformation, const QString &selectedGraphCurve)
  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- virtual void end (CmdMediator &cmdMediator, const Transformation &transformation)
  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void updateAxesChecker (CmdMediator &cmdMediator, const Transformation &transformation)
  *Apply the new DocumentModelAxesChecker.*

### Additional Inherited Members

### 4.265.1 Detailed Description

Class to show transformation since transformation is defined.

Definition at line 18 of file TransformationStateDefined.h.

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

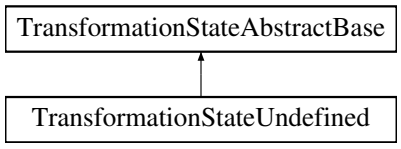
- Transformation/TransformationStateDefined.h
- Transformation/TransformationStateDefined.cpp

## 4.266 TransformationStateUndefined Class Reference

Class to not show transformation since transformation is undefined.

`#include <TransformationStateUndefined.h>`

Inheritance diagram for TransformationStateUndefined:



### Public Member Functions

- TransformationStateUndefined (TransformationStateContext &context, QGraphicsScene &scene)

  *Single constructor.*
- virtual void begin (CmdMediator &cmdMediator, const Transformation &transformation, const QString &selectedGraphCurve)

  *Method that is called at the exact moment a state is entered. Typically called just after end for the previous state.*
- virtual void end (CmdMediator &cmdMediator, const Transformation &transformation)

  *Method that is called at the exact moment a state is exited. Typically called just before begin for the next state.*
- virtual void updateAxesChecker (CmdMediator &cmdMediator, const Transformation &transformation)

  *Apply the new DocumentModelAxesChecker.*

### Additional Inherited Members

#### 4.266.1 Detailed Description

Class to not show transformation since transformation is undefined.

Definition at line 13 of file TransformationStateUndefined.h.

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

- Transformation/TransformationStateUndefined.h
- Transformation/TransformationStateUndefined.cpp

## 4.267 TranslatorContainer Class Reference

Class that stores QTranslator objects for the duration of application execution.

`#include <TranslatorContainer.h>`

### Public Member Functions

- TranslatorContainer (QApplication &app)

  *Single constructor. Argument is needed so object is not optimized away in main() in Windows.*

#### 4.267.1 Detailed Description

Class that stores QTranslator objects for the duration of application execution.

Definition at line 8 of file TranslatorContainer.h.

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

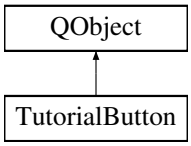
- Translator/TranslatorContainer.h
- Translator/TranslatorContainer.cpp

## 4.268 TutorialButton Class Reference

Show a button with text for clicking ion. The button is implemented using layering of two graphics items (text and rectangle)

`#include <TutorialButton.h>`

Inheritance diagram for TutorialButton:




### Signals

- void signalTriggered ()

  *Signal that button was triggered.*

### Public Member Functions

- TutorialButton (const QString &text, QGraphicsScene &scene)

  *Single constructor. Position is set after creation using setGeometry.*
- void handleTriggered ()

  *Callback to be called when button was triggered by mouse event.*
- void setGeometry (const QPoint &pos)

  *Set the position. This is called after creation so screen extent is available for positioning calculations.*
- QSize size () const

  *Size of this button.*

### 4.268.1 Detailed Description

Show a button with text for clicking ion. The button is implemented using layering of two graphics items (text and rectangle)

Definition at line 20 of file TutorialButton.h.

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

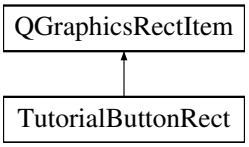
- Tutorial/TutorialButton.h
- Tutorial/TutorialButton.cpp

## 4.269 TutorialButtonRect Class Reference

This class customizes QGraphicsRectItem so it performs a callback after a mouse event.

`#include <TutorialButtonRect.h>`

Inheritance diagram for TutorialButtonRect:




### Public Member Functions

- TutorialButtonRect (TutorialButton &tutorialButton)
  *Single constructor.*
- virtual void mouseReleaseEvent (QGraphicsSceneMouseEvent *event)
  *Forward mouse event to TutorialButton.*

### 4.269.1 Detailed Description

This class customizes QGraphicsRectItem so it performs a callback after a mouse event.

Definition at line 15 of file TutorialButtonRect.h.

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

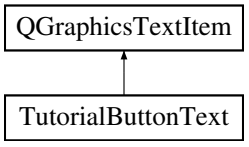
- Tutorial/TutorialButtonRect.h
- Tutorial/TutorialButtonRect.cpp

## 4.270 TutorialButtonText Class Reference

This class customizes QGraphicsTextItem so it performs a callback after a mouse event.

`#include <TutorialButtonText.h>`

Inheritance diagram for TutorialButtonText:



### Public Member Functions

- TutorialButtonText (TutorialButton &tutorialButton, const QString &text, TutorialButtonRect *rect)

  *Single constructor.*
- virtual void mouseReleaseEvent (QGraphicsSceneMouseEvent *event)

  *Forward mouse event to TutorialButton.*

#### 4.270.1 Detailed Description

This class customizes QGraphicsTextItem so it performs a callback after a mouse event.

Definition at line 15 of file TutorialButtonText.h.

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

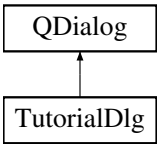
- Tutorial/TutorialButtonText.h
- Tutorial/TutorialButtonText.cpp

## 4.271 TutorialDlg Class Reference

Tutorial using a strategy like a comic strip with decision points deciding which panels appear.

```
#include <TutorialDlg.h>
```

Inheritance diagram for TutorialDlg:




### Public Member Functions

- TutorialDlg (MainWindow *mainWindow)

  *Single constructor.*
- QSize backgroundSize () const

  *Make geometry available for layout.*
- QGraphicsScene & scene ()

  *Single scene the covers the entire tutorial dialog.*
- QGraphicsView & view ()

  *Single view that displays the single scene.*

### 4.271.1 Detailed Description

Tutorial using a strategy like a comic strip with decision points deciding which panels appear.

This is implemented as a QGraphicsScene with states in charge of managing the contents of the scene

Definition at line 19 of file TutorialDlg.h.

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

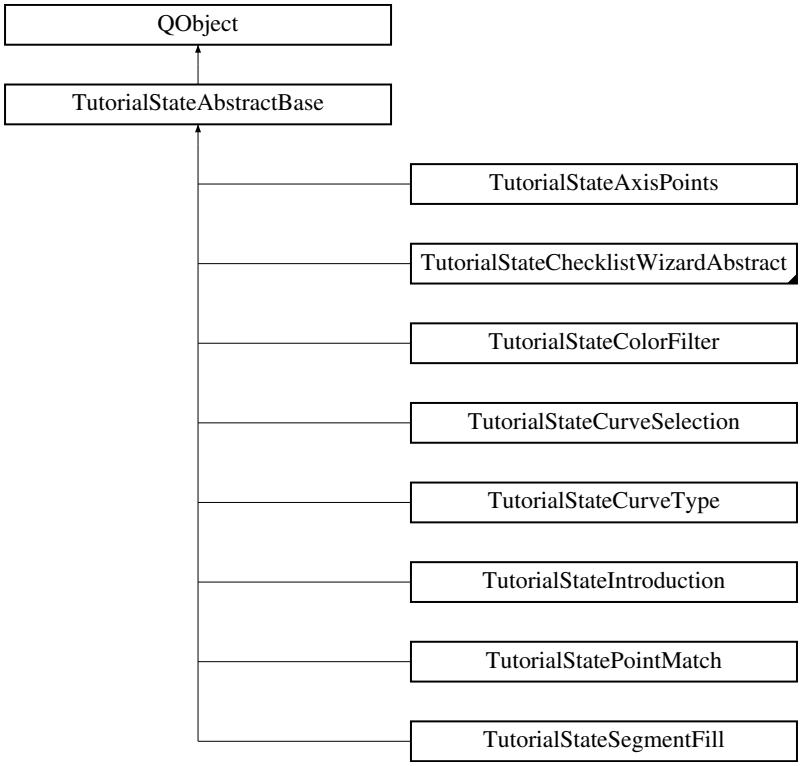
- Tutorial/TutorialDlg.h
- Tutorial/TutorialDlg.cpp

## 4.272 TutorialStateAbstractBase Class Reference

One state manages one panel of the tutorial.

`#include <TutorialStateAbstractBase.h>`

Inheritance diagram for TutorialStateAbstractBase:




### Public Member Functions

- TutorialStateAbstractBase (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()=0

  *Transition into this state.*
- virtual void end ()=0

  *Transition out of this state.*

### Protected Member Functions

- int buttonMargin () const

  *Buttons are placed up against bottom side, and left or right side, separated by this margin.*
- TutorialStateContext & context ()

  *Context class for the tutorial state machine.*
- QGraphicsPixmapItem * createPixmapItem (const QString &resource, const QPoint &pos)

  *Factory method for pixmap items.*
- QGraphicsTextItem * createTextItem (const QString &text, const QPoint &pos)

  *Factory method for text items.*
- QGraphicsTextItem * createTitle (const QString &text)

  *Factory method for title items.*

#### 4.272.1 Detailed Description

One state manages one panel of the tutorial.

Definition at line 30 of file TutorialStateAbstractBase.h.

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

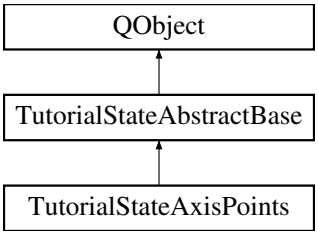
- Tutorial/TutorialStateAbstractBase.h
- Tutorial/TutorialStateAbstractBase.cpp

## 4.273 TutorialStateAxisPoints Class Reference

Axis points panel discusses axis point digitization.

`#include <TutorialStateAxisPoints.h>`

Inheritance diagram for TutorialStateAxisPoints:




### Public Slots

- void slotNext ()

  *Slot called when next button is triggered.*
- void slotPrevious ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStateAxisPoints (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Transition into this state.*
- virtual void end ()

  *Transition out of this state.*

### Additional Inherited Members

#### 4.273.1 Detailed Description

Axis points panel discusses axis point digitization.

Definition at line 18 of file TutorialStateAxisPoints.h.

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

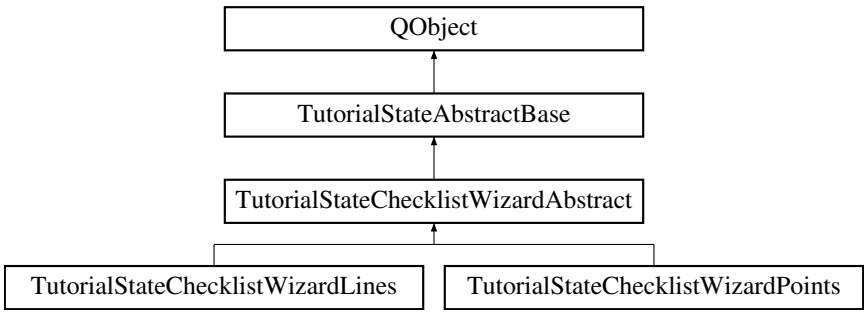
- Tutorial/TutorialStateAxisPoints.h
- Tutorial/TutorialStateAxisPoints.cpp

## 4.274 TutorialStateChecklistWizardAbstract Class Reference

Abstract class that supports checklist wizard panels.

`#include <TutorialStateChecklistWizardAbstract.h>`

Inheritance diagram for TutorialStateChecklistWizardAbstract:




### Public Member Functions

- TutorialStateChecklistWizardAbstract (TutorialStateContext &context)

  *Single constructor.*

### Protected Member Functions

- void begin ()

  *Common begin processing.*
- void end ()

  *Common end processing.*
- TutorialButton * previous ()

  *Previous button for hooking up button to slot.*

### 4.274.1 Detailed Description

Abstract class that supports checklist wizard panels.

Definition at line 18 of file TutorialStateChecklistWizardAbstract.h.

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

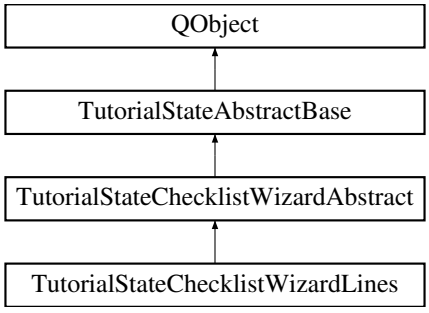
- Tutorial/TutorialStateChecklistWizardAbstract.h
- Tutorial/TutorialStateChecklistWizardAbstract.cpp

## 4.275 TutorialStateChecklistWizardLines Class Reference

Checklist wizard panel for lines discusses the checklist wizard, and returns to TRANSITION_STATE_SEGMENT↩ _FILL.

```
#include <TutorialStateChecklistWizardLines.h>
```

Inheritance diagram for TutorialStateChecklistWizardLines:




### Public Slots

- void slotPrevious ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStateChecklistWizardLines (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Common begin processing.*
- virtual void end ()

  *Common end processing.*

### Additional Inherited Members

#### 4.275.1 Detailed Description

Checklist wizard panel for lines discusses the checklist wizard, and returns to TRANSITION_STATE_SEGMENT↩ _FILL.

Definition at line 18 of file TutorialStateChecklistWizardLines.h.

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

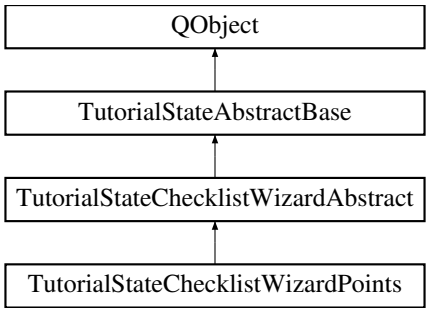
- Tutorial/TutorialStateChecklistWizardLines.h
- Tutorial/TutorialStateChecklistWizardLines.cpp

## 4.276 TutorialStateChecklistWizardPoints Class Reference

Checklist wizard panel for points discusses the checklist wizard, and returns to TRANSITION_STATE_POINT_M↩ ATCH.

```
#include <TutorialStateChecklistWizardPoints.h>
```

Inheritance diagram for TutorialStateChecklistWizardPoints:




### Public Slots

- void slotPrevious ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStateChecklistWizardPoints (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Common begin processing.*
- virtual void end ()

  *Common end processing.*

### Additional Inherited Members

#### 4.276.1 Detailed Description

Checklist wizard panel for points discusses the checklist wizard, and returns to TRANSITION_STATE_POINT_M↩ ATCH.

Definition at line 18 of file TutorialStateChecklistWizardPoints.h.

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

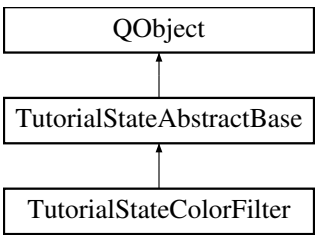
- Tutorial/TutorialStateChecklistWizardPoints.h
- Tutorial/TutorialStateChecklistWizardPoints.cpp

## 4.277 TutorialStateColorFilter Class Reference

Color filter panel discusses the curve-specific color filtering.

```
#include <TutorialStateColorFilter.h>
```

Inheritance diagram for TutorialStateColorFilter:




### Public Slots

- void slotBack ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStateColorFilter (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Transition into this state.*
- virtual void end ()

  *Transition out of this state.*

### Additional Inherited Members

#### 4.277.1 Detailed Description

Color filter panel discusses the curve-specific color filtering.

Definition at line 18 of file TutorialStateColorFilter.h.

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

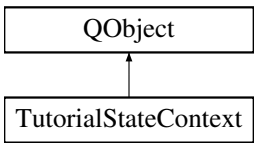
- Tutorial/TutorialStateColorFilter.h
- Tutorial/TutorialStateColorFilter.cpp

## 4.278 TutorialStateContext Class Reference

Context class for tutorial state machine.

`#include <TutorialStateContext.h>`

Inheritance diagram for TutorialStateContext:




### Public Member Functions

- TutorialStateContext (TutorialDlg &tutorialDlg)

  *Single constructor.*
- void requestDelayedStateTransition (TutorialState tutorialState)

  *Request a transition to the specified state from the current state.*
- void requestImmediateStateTransition (TutorialState tutorialState)

  *Request a transition to the specified state from the current state.*
- TutorialDlg & tutorialDlg ()

  *Access to tutorial dialogs and its scene.*

### 4.278.1 Detailed Description

Context class for tutorial state machine.

Each state represents one panel in the tutorial Tutorial assumptions:

1. Dealing with multiple curves is postponed until the end of the tutorial.

Definition at line 20 of file TutorialStateContext.h.

### 4.278.2 Member Function Documentation

#### 4.278.2.1 requestDelayedStateTransition()

```
void TutorialStateContext::requestDelayedStateTransition (
            TutorialState tutorialState )
```

Request a transition to the specified state from the current state.

A timer is used. This assumes TutorialStateContext is NOT on the stack - probably since an external event (mouse click, ...) resulted in a callback to the current state

Definition at line 81 of file TutorialStateContext.cpp.

#### 4.278.2.2 requestImmediateStateTransition()

```
void TutorialStateContext::requestImmediateStateTransition (
            TutorialState tutorialState )
```

Request a transition to the specified state from the current state.

The transition is delayed until the current state is off the stack to prevent stack corruption errors. This assumes TutorialStateContext is on the stack to finish the transition after execution returns from the state

Definition at line 90 of file TutorialStateContext.cpp.

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

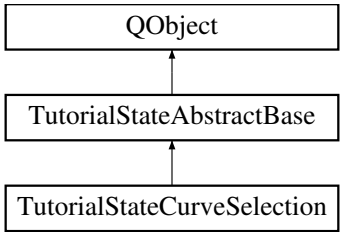
- Tutorial/TutorialStateContext.h
- Tutorial/TutorialStateContext.cpp

## 4.279 TutorialStateCurveSelection Class Reference

Curve selection panel discusses how to select a curve, and perform setup on the selected curve.

`#include <TutorialStateCurveSelection.h>`

Inheritance diagram for TutorialStateCurveSelection:




### Public Slots

- void slotColorFilter ()

  *Slot called when settings button is triggered.*
- void slotNext ()

  *Slot called when next button is triggered.*
- void slotPrevious ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStateCurveSelection (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Transition into this state.*
- virtual void end ()

  *Transition out of this state.*

### Additional Inherited Members

### 4.279.1 Detailed Description

Curve selection panel discusses how to select a curve, and perform setup on the selected curve.

Definition at line 18 of file TutorialStateCurveSelection.h.

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

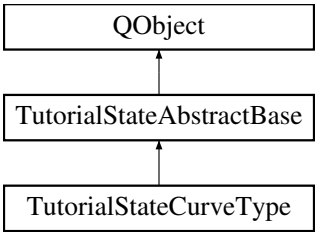
- Tutorial/TutorialStateCurveSelection.h
- Tutorial/TutorialStateCurveSelection.cpp

## 4.280 TutorialStateCurveType Class Reference

Curve type state/panel lets user select the curve type (lines or points)

`#include <TutorialStateCurveType.h>`

Inheritance diagram for TutorialStateCurveType:




### Public Slots

- void slotNextCurves ()

  *Slot called when next button for curves is triggered.*
- void slotNextLines ()

  *Slot called when next button for lines is triggered.*
- void slotPrevious ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStateCurveType (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Transition into this state.*
- virtual void end ()

  *Transition out of this state.*

### Additional Inherited Members

### 4.280.1 Detailed Description

Curve type state/panel lets user select the curve type (lines or points)

Definition at line 18 of file TutorialStateCurveType.h.

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

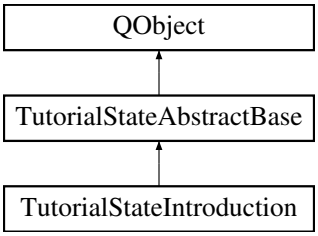
- Tutorial/TutorialStateCurveType.h
- Tutorial/TutorialStateCurveType.cpp

## 4.281 TutorialStateIntroduction Class Reference

Introduction state/panel is the first panel the user sees.

`#include <TutorialStateIntroduction.h>`

Inheritance diagram for TutorialStateIntroduction:




### Public Slots

- void slotNext ()

  *Slot called when next button is triggered.*

### Public Member Functions

- TutorialStateIntroduction (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Transition into this state.*
- virtual void end ()

  *Transition out of this state.*

### Additional Inherited Members

### 4.281.1 Detailed Description

Introduction state/panel is the first panel the user sees.

Definition at line 18 of file TutorialStateIntroduction.h.

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

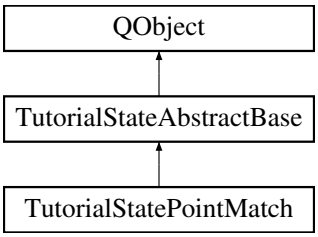
- Tutorial/TutorialStateIntroduction.h
- Tutorial/TutorialStateIntroduction.cpp

## 4.282 TutorialStatePointMatch Class Reference

Point match panel discusses the matching of points in curves without lines.

`#include <TutorialStatePointMatch.h>`

Inheritance diagram for TutorialStatePointMatch:




### Public Slots

- void slotNext ()

  *Slot called when next button is triggered.*
- void slotPrevious ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStatePointMatch (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Transition into this state.*
- virtual void end ()

  *Transition out of this state.*

### Additional Inherited Members

### 4.282.1 Detailed Description

Point match panel discusses the matching of points in curves without lines.

Definition at line 18 of file TutorialStatePointMatch.h.

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

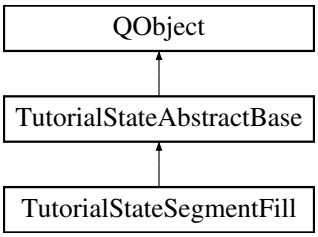
- Tutorial/TutorialStatePointMatch.h
- Tutorial/TutorialStatePointMatch.cpp

## 4.283 TutorialStateSegmentFill Class Reference

Segment fill panel discusses the digitization of points along curve lines.

`#include <TutorialStateSegmentFill.h>`

Inheritance diagram for TutorialStateSegmentFill:




### Public Slots

- void slotNext ()

  *Slot called when next button is triggered.*
- void slotPrevious ()

  *Slot called to return to previous panel.*

### Public Member Functions

- TutorialStateSegmentFill (TutorialStateContext &context)

  *Single constructor.*
- virtual void begin ()

  *Transition into this state.*
- virtual void end ()

  *Transition out of this state.*

### Additional Inherited Members

### 4.283.1 Detailed Description

Segment fill panel discusses the digitization of points along curve lines.

Definition at line 18 of file TutorialStateSegmentFill.h.

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

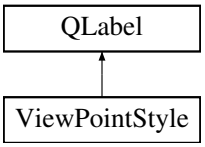
- Tutorial/TutorialStateSegmentFill.h
- Tutorial/TutorialStateSegmentFill.cpp

## 4.284 ViewPointStyle Class Reference

Class that displays a view of the current Curve's point style.

`#include <ViewPointStyle.h>`

Inheritance diagram for ViewPointStyle:




### Public Member Functions

- ViewPointStyle (QWidget *parent=0)
  *Single constructor.*
- void setEnabled (bool enabled)
  *Show the style with semi-transparency or full-transparency to indicate if associated Curve is active or not.*
- void setPointStyle (const PointStyle &pointStyle)
  *Apply the PointStyle of the currently selected curve.*
- void unsetPointStyle ()
  *Apply no PointStyle.*

### 4.284.1 Detailed Description

Class that displays a view of the current Curve's point style.

Do NOT apply a visible border since that would hide a square drawn just inside the four sides.

Definition at line 16 of file ViewPointStyle.h.

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

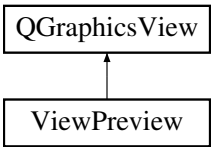
- View/ViewPointStyle.h
- View/ViewPointStyle.cpp

## 4.285 ViewPreview Class Reference

Class that modifies QGraphicsView to automatically expand/shrink the view to fit the window, after resize events.

`#include <ViewPreview.h>`

Inheritance diagram for ViewPreview:



### Public Types

- enum ViewAspectRatio { **VIEW_ASPECT_RATIO_VARIABLE**, **VIEW_ASPECT_RATIO_ONE_TO_ONE** }

  *Prevent aspect ratio distortion in certain previews by providing fixed 1:1 aspect ratio option.*

### Signals

- void signalMouseMove (QPointF pos)

  *Forward the mouse move events.*

### Public Member Functions

- ViewPreview (QGraphicsScene ∗scene, ViewAspectRatio viewAspectRatio, QWidget ∗parent=0)

  *Single constructor.*
- virtual void mouseMoveEvent (QMouseEvent ∗event)

  *Intercept cursor move events and forward them.*
- virtual void resizeEvent (QResizeEvent ∗event)

  *Intercept resize events so we can rescale to the graphics items just fit into the resized window.*
- virtual void wheelEvent (QWheelEvent ∗event)

  *Intercept wheel event and discard it so accidentally moving the wheel does not move drawn items out of view.*

#### 4.285.1 Detailed Description

Class that modifies QGraphicsView to automatically expand/shrink the view to fit the window, after resize events.

Definition at line 14 of file ViewPreview.h.

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

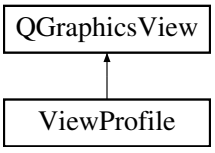
- View/ViewPreview.h
- View/ViewPreview.cpp

## 4.286 ViewProfile Class Reference

Class that modifies QGraphicsView to present a two-dimensional profile, with movable dividers for selecting a range.

```
#include <ViewProfile.h>
```

Inheritance diagram for ViewProfile:



### Public Member Functions

- ViewProfile (QGraphicsScene ∗scene, int minimumWidth, QWidget ∗parent=0)
  *Single constructor.*
- virtual void resizeEvent (QResizeEvent ∗event)
  *Intercept resize events so the geometry can be scaled to perfectly fit into the window.*

#### 4.286.1 Detailed Description

Class that modifies QGraphicsView to present a two-dimensional profile, with movable dividers for selecting a range.

Definition at line 15 of file ViewProfile.h.

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

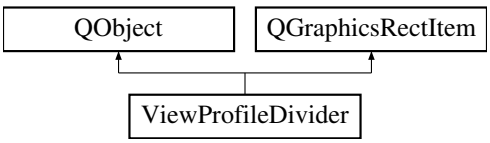
- View/ViewProfile.h
- View/ViewProfile.cpp

## 4.287 ViewProfileDivider Class Reference

Divider that can be dragged, in a dialog QGraphicsView.

`#include <ViewProfileDivider.h>`

Inheritance diagram for ViewProfileDivider:




### Signals

- void signalMovedLow (double xSceneOther)
  *Signal used when divider is dragged and m_isLowerBoundary is true.*
- void signalMovedHigh (double xSceneOther)
  *Signal used when divider is dragged and m_isLowerBoundary is false.*

### Public Member Functions

- ViewProfileDivider (QGraphicsScene &scene, QGraphicsView &view, int sceneWidth, int sceneHeight, int yCenter, bool isLowerBoundary)
  *Single constructor.*
- virtual QVariant itemChange (GraphicsItemChange change, const QVariant &value)
  *Intercept changes so divider movement can be restricted to horizontal direction only.*
- virtual void mousePressEvent (QGraphicsSceneMouseEvent ∗event)
  *Save paddle position at start of click-and-drag.*
- void setX (double x, double xLow, double xHigh)
  *Set the position by specifying the new x coordinate.*

### 4.287.1 Detailed Description

Divider that can be dragged, in a dialog QGraphicsView.

Click on the paddle to drag. There are three parts:

1. Paddle which is the superclass of this class, since we catch its events so dragging works
2. Divider which is a vertical line
3. Shaded area that extends from xAnchor to the divider

Definition at line 23 of file ViewProfileDivider.h.

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

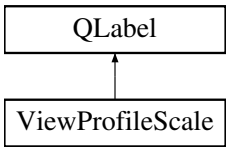
- View/ViewProfileDivider.h
- View/ViewProfileDivider.cpp

## 4.288 ViewProfileScale Class Reference

Linear horizontal scale, with the spectrum reflecting the active filter parameter.

`#include <ViewProfileScale.h>`

Inheritance diagram for ViewProfileScale:




### Public Member Functions

- ViewProfileScale (int minimumWidth, QWidget *parent=0)
  *Single constructor.*
- virtual void paintEvent (QPaintEvent *)
  *Draw the gradient.*
- void setBackgroundColor (QRgb rgbBackground)
  *Save the background color for foreground calculations.*
- void setColorFilterMode (ColorFilterMode colorFilterMode)
  *Change the gradient type.*

### 4.288.1 Detailed Description

Linear horizontal scale, with the spectrum reflecting the active filter parameter.

Definition at line 16 of file ViewProfileScale.h.

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

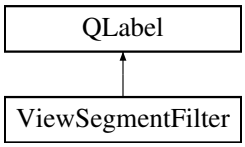
- View/ViewProfileScale.h
- View/ViewProfileScale.cpp

## 4.289 ViewSegmentFilter Class Reference

Class that displays the current Segment Filter in a MainWindow toolbar.

`#include <ViewSegmentFilter.h>`

Inheritance diagram for ViewSegmentFilter:




### Public Member Functions

- ViewSegmentFilter (QWidget *parent=0)
  *Single constructor.*
- virtual void paintEvent (QPaintEvent *event)
  *Paint with a horizontal linear gradient.*
- void setColorFilterSettings (const ColorFilterSettings &colorFilterSettings, const QPixmap &pixmap)
  *Apply the color filter of the currently selected curve. The pixmap is included so the background color can be computed.*
- void setEnabled (bool enabled)
  *Show the style with semi-transparency or full-transparency to indicate if associated Curve is active or not.*
- void unsetColorFilterSettings ()
  *Apply no color filter.*

### 4.289.1 Detailed Description

Class that displays the current Segment Filter in a MainWindow toolbar.

A gradient is displayed. No border is drawn so the appearance is consistent with ViewPointStyle which would not work with a border.

Definition at line 18 of file ViewSegmentFilter.h.

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

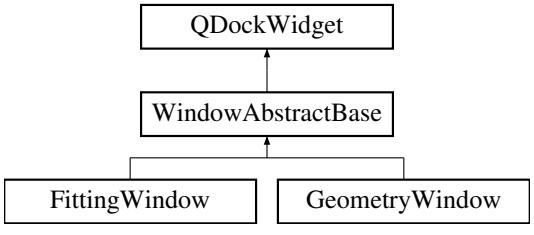
- View/ViewSegmentFilter.h
- View/ViewSegmentFilter.cpp

## 4.290 WindowAbstractBase Class Reference

Dockable widget abstract base class.

`#include <WindowAbstractBase.h>`

Inheritance diagram for WindowAbstractBase:



### Public Member Functions

- WindowAbstractBase (QWidget *parent)

  *Single constructor. Parent is needed or else this widget cannot be redocked after being undocked.*
- virtual void clear ()=0

  *Clear stale information.*
- virtual void closeEvent (QCloseEvent *event)=0

  *Catch close event so corresponding menu item in MainWindow can be updated accordingly.*
- virtual void doCopy ()=0

  *Copy the current selection to the clipboard.*
- void getTableStatus (bool &tableIsActive, bool &tableIsCopyable) const

  *Give table status so MainWindow can determine if table can be copied.*
- virtual void update (const CmdMediator &cmdMediator, const MainWindowModel &modelMainWindow, const QString &curveSelected, const Transformation &transformation)=0

  *Populate the table with the specified Curve.*

### Protected Member Functions

- virtual QTableView * view () const =0

  *QTableView-based class used by child class.*

#### 4.290.1 Detailed Description

Dockable widget abstract base class.

This class enforces support for the MainWindow class, in terms of copying selected stuff, and also for performing clearing and updates

Definition at line 20 of file WindowAbstractBase.h.

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

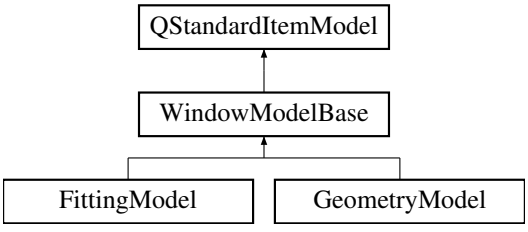
- Window/WindowAbstractBase.h
- Window/WindowAbstractBase.cpp

## 4.291 WindowModelBase Class Reference

Model for WindowTable.

`#include <WindowModelBase.h>`

Inheritance diagram for WindowModelBase:



### Public Member Functions

- WindowModelBase ()
  *Single constructor.*
- QMimeData * mimeData (const QModelIndexList &indexes) const
  *Support dragging of multiple cells.*
- QString selectionAsHtml () const
  *Convert the selection into exportable html which is good for spreadsheets.*
- QString selectionAsText (ExportDelimiter delimiter) const
  *Convert the selection into exportable text which is good for text editors.*
- void setDelimiter (ExportDelimiter delimiter)
  *Save output delimiter.*
- void setView (WindowTable &view)
  *Save the view so this class can access the current selection.*

#### 4.291.1 Detailed Description

Model for WindowTable.

Definition at line 18 of file WindowModelBase.h.

#### 4.291.2 Member Function Documentation

##### 4.291.2.1 mimeData()

```
QMimeData * WindowModelBase::mimeData (
            const QModelIndexList & indexes ) const
```

Support dragging of multiple cells.

Without this only one cell can be copied by dragging. Clipboard copying is handled elsewhere in the window class

Definition at line 34 of file WindowModelBase.cpp.

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

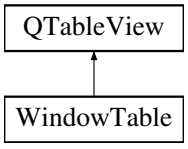
- Window/WindowModelBase.h
- Window/WindowModelBase.cpp

## 4.292 WindowTable Class Reference

Table view class with support for both drag-and-drop and copy-and-paste.

```
#include <WindowTable.h>
```

Inheritance diagram for WindowTable:



### Signals

- void signalTableStatusChange ()
  *Sent when a change occurs that should affect the Copy menu item.*

### Public Member Functions

- WindowTable (WindowModelBase &model)
  *Single constructor.*
- virtual void focusInEvent (QFocusEvent *)
  *Catch this table status change.*
- virtual void focusOutEvent (QFocusEvent *)
  *Catch this table status change.*
- virtual void selectionChanged (const QItemSelection &selected, const QItemSelection &deselected)
  *Catch this table status change.*

#### 4.292.1 Detailed Description

Table view class with support for both drag-and-drop and copy-and-paste.

Definition at line 17 of file WindowTable.h.

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

- Window/WindowTable.h
- Window/WindowTable.cpp

## 4.293 ZoomTransition Class Reference

Perform calculations to determine the next zoom setting given the current zoom setting, when zooming in or out.

`#include <ZoomTransition.h>`

### Public Member Functions

- ZoomTransition ()
  *Single constructor.*
- double mapToFactor (ZoomFactor zoomFactor) const
  *Return the floating precision zoom factor given the enum value.*
- ZoomFactor zoomIn (ZoomFactor currentZoomFactor, double m11, double m22, bool actionZoomFillIs↩ Checked) const
  *Zoom in.*
- ZoomFactor zoomOut (ZoomFactor currentZoomFactor, double m11, double m22, bool actionZoomFillIs↩ Checked) const
  *Zoom out.*

#### 4.293.1 Detailed Description

Perform calculations to determine the next zoom setting given the current zoom setting, when zooming in or out.

Definition at line 14 of file ZoomTransition.h.

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

- Zoom/ZoomTransition.h
- Zoom/ZoomTransition.cpp

# Index