TrackIt

1.0

Generated by Doxygen 1.8.2

Tue May 21 2013 14:05:14

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 BBox Struct Reference

Represents a bounding box with its properties like type, framenumber and geometry.

`#include <types.h>`

Public Types

- enum Type { NULLTYPE =0, SINGLE =1, KEYBOX =2, VIRTUAL =3 }

  Type of a bounding box.

Public Member Functions

- BBox ()

  Constructs a NULL bounding box.
- BBox (int framenumber, QRect const &rect, int objectID, Type type=SINGLE)

  Constructs a valid bounding box.
- bool operator== (BBox const &bbox) const

  Compares two BBoxes using only the framenumbers.

Public Attributes

- Type type

  Type of the bounding box.
- int framenumber

  Number of the frame the bounding box appears.
- QRect rect

  Geometry of the bounding box.
- int objectID

  ID of the object the bounding box belongs to.

Related Functions

(Note that these are not member functions.)

- BBox interpolate (int framenumber, BBox const &bboxA, BBox const &bboxB)

Retruns a box interpolated between two specified boxes using the framenumbers.

- QList< BBox > interpolate (int frameStart, int frameEnd, BBox const &bboxA, BBox const &bboxB)

  Returns a list of interpolated BBoxes between specified framenumbers..

3.1.1 Detailed Description

Represents a bounding box with its properties like type, framenumber and geometry.

This struct is the core element of the data structure to save the tracking data, where multiple BBoxes form one Object. It contains the bounding box' geometry in rect, the number of the frame it belongs to in framenumber, it's type in type and for convenience the id of its parent object in objectID.

3.1.2 Member Enumeration Documentation

3.1.2.1 enum BBox::Type

Type of a bounding box.

The type specifies whether the box really exists, got interpolated or is null.

Enumerator:

NULLTYPE Invalid bounding box (box isn't existing).

SINGLE Normal bounding box only present for one frame.

KEYBOX Like a keyframe; Not existing bounding boxes before this box will be interpolated.

VIRTUAL Interpolated bounding box.

3.1.3 Constructor & Destructor Documentation

3.1.3.1 BBox::BBox ( )

Constructs a NULL bounding box.

The type is initialised to NULLTYPE, the other members are set to alike values

3.1.3.2 BBox::BBox (int *framenumber,* QRect const & *rect,* int *objectID,* Type *type* = SINGLE)

Constructs a valid bounding box.

The members get initialised according to the same named parameters. No more, no less.

3.1.4 Member Function Documentation

3.1.4.1 bool BBox::operator== (BBox const & *bbox*) const

Compares two BBoxes using only the framenumbers.

This operator should be used to determine whether a bounding box allready exists for a specific frame or not, regardless it's geometry.

3.1.5 Friends And Related Function Documentation

3.1.5.1 BBox interpolate (int *framenumber,* BBox const & *bboxA,* BBox const & *bboxB*) [related]

Retruns a box interpolated between two specified boxes using the framenumbers.

The function interpolates linear, weightet with the differences from the target framenumber to the bording BBoxes framenumbers.

3.1.5.2 QList< BBox > interpolate (int *frameStart*, int *frameEnd*, BBox const & *bboxA*, BBox const & *bboxB*) [related]

Returns a list of interpolated BBoxes between specified framenumbers..

Convenience function returning a list of BBoxes interpolated between frameStart and frameEnd. (BBoxA and BBoxB are not part of the returned list!) (fixme: function not tested yet)

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

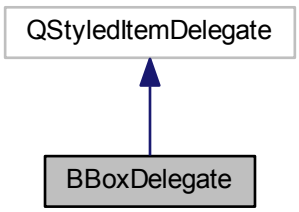
- types.h
- types.cpp

3.2 BBoxDelegate Class Reference

A delegate that renders integers as pixmaps.

`#include <bboxdelegate.h>`

Collaboration diagram for BBoxDelegate:



Public Member Functions

- BBoxDelegate (QObject *parent=0)
  Default constructor.
- void paint (QPainter *painter, QStyleOptionViewItem const &option, QModelIndex const &index) const
  Paints the data at the given index.
- QSize sizeHint (QStyleOptionViewItem const &option, QModelIndex const &index) const
  Virtual size hint method.

Private Member Functions

- void initPixmaps ()
  Initializes the pixmap cache.
- QColor color (BBox::Type type, bool light=false) const
  Returns the hard coded colors.
- QPixmap const & pixmap (int width, BBox::Type type, bool current) const

Returns the appropriate pixmap from the cache.

- QPixmap createPixmap (int width, BBox::Type type, bool current) const

  Creates a pixmap.

Private Attributes

- QList< QPixmap > pixmaps

  Internal cache of prerendered pixmaps.

3.2.1 Detailed Description

A delegate that renders integers as pixmaps.

The integers get interpreted as BBox::Type and a corresponding representation gets rendered in the view. The color mapping is hard coded in color and the pixmaps get cached on construction.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BBoxDelegate::BBoxDelegate (QObject * *parent* = 0) [explicit]

Default constructor.

Initializes the internal pixmap cache pixmaps.

See Also

void initPixmaps()

3.2.3 Member Function Documentation

3.2.3.1 QColor BBoxDelegate::color (BBox::Type *type*, bool *light* = false) const [private]

Returns the hard coded colors.

The color depends on the given *type* and if it is *light* or not.

See Also

colorbrewer2.org

3.2.3.2 QPixmap BBoxDelegate::createPixmap (int *width*, BBox::Type *type*, bool *current*) const [private]

Creates a pixmap.

The pixmap is colored according to the BBs *type* and whether it's *current* or not.

3.2.3.3 void BBoxDelegate::initPixmaps ( ) [private]

Initializes the pixmap cache.

A Pixmap for any type, selection type and size is created.

See Also

QPixmap createPixmap(int width, BBox::Type type, bool current) const

3.2.3.4 void BBoxDelegate::paint (QPainter * *painter,* QStyleOptionViewItem const & *option,* QModelIndex const & *index*) const

Paints the data at the given *index*.

If the data at the given *index* can be converted to integer it gets interpreted as BBox::Type and according to this type and the given *option* a representing pixmap gets painted with the *painter*.

Otherwise the default paint method from `QStyledItemDelegate` is called.

3.2.3.5 QPixmap const & BBoxDelegate::pixmap (int *width,* BBox::Type *type,* bool *current*) const [private]

Returns the appropriate pixmap from the cache.

The pixmap is chosen according to the given *width*, *type* and if it is *current* (selected) or not.

3.2.3.6 QSize BBoxDelegate::sizeHint (QStyleOptionViewItem const & *option,* QModelIndex const & *index*) const

Virtual size hint method.

Returns the size needed by the delegate to display the item specified by *index*, taking into account the style information provided by *option*.

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

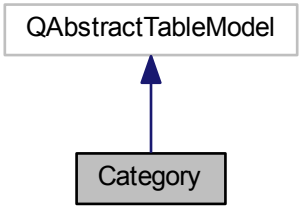
- bboxdelegate.h
- bboxdelegate.cpp

3.3 Category Class Reference

Represents a category of Objects like "Person" or "Car".

`#include <category.h>`

Collaboration diagram for Category:



Public Member Functions

- Category (QString const &name, QObject *parent=0)

  Constructs a category with the specified name.
- Category (QDataStream &in, QObject *parent=0)

  Reads a category from a given stream in.

- ~Category ()

  Deletes all objects managed by the category.
- int rowCount (QModelIndex const &parent=QModelIndex()) const

  Returns the number of objects in the list.
- int columnCount (QModelIndex const &parent=QModelIndex()) const

  Returns the width of the data.
- void setColumnCount (int n)

  Sets the width of the data.
- QVariant data (QModelIndex const &index, int role=Qt::DisplayRole) const

  Returns the data stored under the given role for the item referred to by the index.
- QVariant headerData (int section, Qt::Orientation orientation, int role=Qt::DisplayRole) const

  Returns the data for the given role and section in the header with the specified orientation.
- void setName (QString const &newName)

  Setter for the categorie's name.
- QString const & getName () const

  Getter for the categorie's name.
- QList< BBox > getBBoxes (int framenumber) const

  Returns the bounding boxes of all objects for the specified framenumber.
- Object * getObject (int id)

  Returns a pointer to the object with the specified ID.
- QList< Object * > const & getObjects () const

  Getter for the stored objects.
- int findObject (int id) const

  Returns the row of the object with the given id.
- bool isEmpty () const

  Returns whether the category contains no objects.
- void addObject (Object *object)

  Adds the object to the internal list.
- Category & operator<< (Object *object)

  Adds the object to the internal list.
- void newObject ()

  Creates a new object and adds it to the internal list.
- void deleteObjectAt (int row)

  Deletes the object in the specified row.
- Object * takeObjectAt (int row)

  Takes the object from the specified row.
- void deleteBBoxes (QModelIndexList const &selection)

  Deletes the BBoxes specified in the given selection.
- void sortByID ()

  Sorts the objects by their ID.
- void sortByFN ()

  Sorts the objects by their framenumber.
- void save (QDataStream &out) const

  Saves the category to a given stream out.
- void load (QDataStream &in)

  Loads the category from a given stream in.
- int getFramecount () const

  Returns the overall framecount of the category.

Private Slots

- void objectDataChanged (int objectID, int framenumber)

  Internal slot for change feedback from objects.

Private Attributes

- int columncount

  The column count read from a video file.
- QString name

  The name of the category.
- QList< Object * > objects

  The list of objects.

3.3.1 Detailed Description

Represents a category of Objects like "Person" or "Car".

The class as a named list of Objects is derived from `QAbstractListModel` so it can be easily interacted with via a `QListView` on the GUI.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Category::Category (QString const & *name*, QObject * *parent* = `0`) `[explicit]`

Constructs a category with the specified *name*.

Since the class derives from QAbstractListModel this ctor is called as well.

See Also

`QAbstractItemModel::QAbstractListModel(QObject * parent = 0)`

3.3.2.2 Category::Category (QDataStream & *in*, QObject * *parent* = `0`) `[explicit]`

Reads a category from a given stream *in*.

Since the class derives from QAbstractListModel this ctor is called as well.

See Also

`QAbstractItemModel::QAbstractListModel(QObject * parent = 0)`

3.3.3 Member Function Documentation

3.3.3.1 void Category::addObject (Object * *object*)

Adds the *object* to the internal list.

In fact the pointer to the *object* gets added, so the submitted pointer stays valid. The category takes ownership of the *object* and deletes it if necessary.

3.3.3.2 int Category::columnCount (QModelIndex const & *parent* = `QModelIndex()`) const

Returns the width of the data.

See Also

`int QAbstractItemModel::columnCount(const QModelIndex & parent) const`

3.3.3.3 QVariant Category::data (QModelIndex const & *index*, int *role* = `Qt::DisplayRole`) const

Returns the data stored under the given *role* for the item referred to by the *index*.

See Also

`QVariant QAbstractItemModel::data(const QModelIndex & index, int role) const`

3.3.3.4 void Category::deleteBBoxes (QModelIndexList const & *selection*)

Deletes the BBoxes specified in the given *selection*.

The deletion order gets forwarded to the corresponding objects and al connected views get order to update.

See Also

void Object::deleteBBoxAt(int framenumber)

3.3.3.5 void Category::deleteObjectAt (int *row*)

Deletes the object in the specified *row*.

It gets the object via takeObjectAt which disconnects all signals and deletes it then.

Note

with the object all the associated bounding boxes get lost as well.

See Also

Object * takeObjectAt(int row)

3.3.3.6 int Category::findObject (int *id*) const

Returns the row of the object with the given *id*.

If the object can't be found -1 is returned.

3.3.3.7 QList< BBox > Category::getBBoxes (int *framenumber*) const

Returns the bounding boxes of all objects for the specified *framenumber*.

The objects simply get looped through collecting all the currently visible bounding boxes which get grouped in a QList.

Note

Since the returned list contains copies of the boxes altering them will not affect the original data.

See Also

BBox Object::getBBox(int framenumber) const

3.3.3.8 int Category::getFramecount ( ) const

Returns the overall framecount of the category.

This framecount is the actual maximum of all the objects last framenumbers and therefor can be seen as a minimum width for views.

3.3.3.9 Object * Category::getObject (int *id*)

Returns a pointer to the object with the specified *ID*.

If no such object exists a NULL pointer is returned instead.

3.3.3.10 QVariant Category::headerData (int *section*, Qt::Orientation *orientation*, int *role* = `Qt::DisplayRole`) const

Returns the data for the given *role* and *section* in the header with the specified *orientation*.

See Also

```
QVariant QAbstractItemModel::headerData(int section, Qt::Orientation orientation,
int role) const
```

3.3.3.11 void Category::newObject ( )

Creates a new object and adds it to the internal list.

The object is empty (doesn't contain any bounding boxes) and gets itself a unique ID.

3.3.3.12 void Category::objectDataChanged (int *objectID*, int *framenumber*) [private], [slot]

Internal slot for change feedback from objects.

It simply is a adapter to the dataChanged signal which it emits.

3.3.3.13 Category & Category::operator<< (Object * *object*)

Adds the *object* to the internal list.

In fact the pointer to the *object* gets added, so the submitted pointer stays valid. The category takes ownership of the *object* and deletes it if necessary.

See Also

void addObject(Object * object)

3.3.3.14 int Category::rowCount (QModelIndex const & *parent* = `QModelIndex()`) const

Returns the number of objects in the list.

See Also

```
int QAbstractItemModel::rowCount(const QModelIndex & parent = QModel-
Index()) const
```

3.3.3.15 void Category::save (QDataStream & *out*) const

Saves the category to a given stream *out*.

Saves it's name and all objects.

See Also

void Object::save(QDataStream & out) const

3.3.3.16 void Category::setColumnCount (int *n*)

Sets the width of the data.

Used to widen the attached views to match a video so even frames without boxes can be selected.

3.3.3.17 void Category::sortByFN ( )

Sorts the objects by their framenumber.

The sort is stable and uses the corresponding sort function from class Object

See Also

bool lessThanByFN(Object const * const object1, Object const * const object2)

3.3.3.18 void Category::sortByID ( )

Sorts the objects by their ID.

The sort is stable and uses the corresponding sort function from class Object

See Also

bool lessThanByID(Object const * const object1, Object const * const object2)

3.3.3.19 Object * Category::takeObjectAt (int *row*)

Takes the object from the specified *row*.

The connected signals get disconnected and the category ends it's ownership

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

- category.h
- category.cpp

3.4 DataWidget::Cell Struct Reference

Structure to represent a cell in a specific tab.

Public Member Functions

- Cell ()
  Default c'tor.
- Cell (int index, QModelIndex const &modelIndex)
  c'tor that takes a index and a model index for convenience
- bool isNull ()
  Checks if the Cell is invalid.

Public Attributes

- int index
  Tab index.
- int row
  Table row.
- int column
  Table column.

3.4.1 Detailed Description

Structure to represent a cell in a specific tab.

The struct is mainly used as a return type.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DataWidget::Cell::Cell ( ) [inline]

Default c'tor.

Creates a invalid Cell (all attributes have the value -1)

3.4.2.2 DataWidget::Cell::Cell (int *index*, QModelIndex const & *modelIndex*) [inline]

c'tor that takes a index and a model index for convenience

row and column get extracted from the *modelIndex*

3.4.3 Member Function Documentation

3.4.3.1 bool DataWidget::Cell::isNull ( ) [inline]

Checks if the Cell is invalid.

A Cell is invalid if any of it's attributes are negative.

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

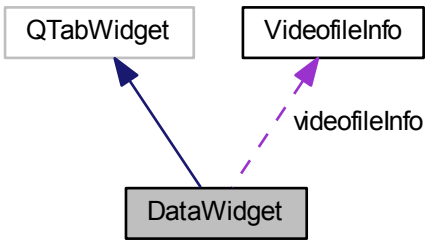
- datawidget.h

3.5 DataWidget Class Reference

Class managing the tracking data and it's representation as a QTabWidget.

`#include <datawidget.h>`

Collaboration diagram for DataWidget:



Classes

- struct Cell

  Structure to represent a cell in a specific tab.

Public Slots

- void openFile ()

  Opens a data file.
- void saveFile ()

  Saves the current data to a file.
- void saveFileAs ()

  Saves the current data to a file.
- void importFile ()

  Imports data from a file in a foreign format.
- void exportFile ()

  Exports data to a file in a foreign format.
- void newCategory ()

  Creates a new category and adds it to the internal list.
- void deleteCategory ()

  Deletes the currently shown category.
- void editCategory ()

  Enables the user to edit the current category.
- void sortByID ()

  Sorts the tracking data by ID.
- void sortByFN ()

  Sorts the tracking data by framenumber.
- void newObject ()

  Creates a new object and adds it to the currently shown category.
- void deleteObject ()

Deletes the currently selected object.

- void editObject ()

  Enables the user to move objects to another category.
- void deleteBBox ()

  Deletes the selected bounding boxes.
- void clearData ()

  Deletes all tracking data.
- void setSelectedObject (int id)

  Changes the selection to the object with the specified ID.
- void setCurrentFrame (int framenumber)

  Sets the selection to the cell with the specified framenumber.
- void selectNextKeyframe ()

  Selects the next keyframe of the current object.
- void selectPreviousKeyframe ()

  Selects the previous keyframe of the current object.
- void selectNextObject ()

  Selects the next object.
- void selectPreviousObject ()

  Selects the previous object.
- void selectNextCategory ()

  Selects the next category.
- void selectPreviousCategory ()

  Selects the previous category.

Signals

- void dataDecreased ()

  Gets emitted when tracking data gets deleted.
- void selectedObjectChanged (int id)

  Gets emitted when another object gets selected and submits its ID.
- void currentFrameChanged (int framenumber)

  Gets emitted when the current framenumber changes.
- void requestVideo (QString filename)

  Gets emitted when a data file containing information about a associated video is opened.
- void categoryCountChanged (int count)

  Gets emitted when the number of categories changes in any way.
- void updateActions ()

  Gets emitted whenever the selection changes.

Public Member Functions

- DataWidget (QWidget *parent=0)

  Default c'tor, initializes everything.
- ~DataWidget ()

  Deletes all the tracking data.
- QList< BBox > getBBoxes (int framenumber) const

  Returns the bounding boxes of all objects of all categories for the specified framenumber.
- Object * getObject (int id) const

  Returns a pointer to the object with the specified ID.
- void setVFInfo (VideofileInfo const &newVideofileInfo)

Setter for the videofileInfo.

- virtual QSize minimumSizeHint () const

  Returns the recommended minimum size for the widget.
- bool isObjectSelected () const

  Returns if a object is selected.
- BBox::Type getCurrentBBoxType () const

  Returns the type of the current BBox.
- void setSelectedObjectByRow (int row)

  Sets the selection to the specified row.

Private Slots

- void selectionChanged (QModelIndex const &current, QModelIndex const &previous)

  Adapter from the selectionChanged Signal from the ListView to the one from this class.
- void deleteCategory (QAbstractButton *button)

  Slot to delete the category associated with the given button.
- void editCategory (QAbstractButton *button)

  Slot to edit the category associated with the given button.
- void onCurrentTabChanged (int index)

  Gets called when the current tab changes to index.
- void changeZoom (int newZoom)

  Called to change the zoomlevel of the TableView to newZoom.

Private Member Functions

- void clearDataImmediate ()

  Deletes all tracking data without further warning.
- void addCategory (Category *cat)

  Adds the given category cat to the internal list.
- Category * addCategory (QString const &name)

  Creates and adds a category with the specified name and returns a pointer to it.
- void deleteCategory (int index)

  Deletes the category with the given index.
- void editCategory (int index)

  Enables the user to edit the category with the given index.
- void addObject (Object *object, QString const &catName)

  Adds the object to the category specified by catName.
- void importViperFile (QFile &file)

  Imports tracking data from a ViPER file.
- void importBBFile (QFile &file)

  Imports tracking data from a BB file.
- void exportViperFile (QFile &file)

  Exports tracking data as a ViPER file.
- void exportBBFile (QFile &file)

  Exports tracking data as a BB file.
- void setSelection (int tab, int row, int column)

  Sets the selection.
- Cell getSelection () const

  Determines the current selection (single cell) and returns it.
- void createNewCategoryTab ()

Creates the special last tab that creates a new category when it gets focus.

- void createCornerWidget ()

  Creates the TableViews corner widget.
- QList< int > getSelectedRows () const

  Returns the rows contained in the current selection.
- void updateFramecount ()

  Determines and sets the maximum framecount of all objects.

Private Attributes

- int zoom

  The width of one box in the data view.
- int currentFrameNr

  Number of the currently selected frame.
- int selectedObjectID

  ID of the currently selected object.
- QString filename

  Filename of the current data file.
- VideofileInfo videofileInfo

  VideoFileInfo struct from the current video file.
- QList< Category * > categories

  The list of categories.
- QButtonGroup * closeBtnGroup

  Group to organize the close category buttons.
- QButtonGroup * editBtnGroup

  Group to organize the edit category buttons.

3.5.1 Detailed Description

Class managing the tracking data and it's representation as a QTabWidget.

The Tabwidget can load and save tracking data and adds a tab containing a QListView for each Category in this data. Since the Category class derives from QAbstractListModel the ListView is used to represent the Objects in the Category.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 DataWidget::DataWidget (QWidget * *parent* = 0) [explicit]

Default c'tor, initializes everything.

Also creates a default category and object for a swifter start.

See Also

```
QTabWidget::QTabWidget(QWidget * parent = 0)
```

3.5.3 Member Function Documentation

3.5.3.1 void DataWidget::addCategory (Category * *cat*) [private]

Adds the given category *cat* to the internal list.

The category is appended to the internal list of categories. Also a new QTableView with the category as model is created and added as a new tab.

Note

Category derives from QAbstractListModel to make this simple connection possible.

3.5.3.2 Category * DataWidget::addCategory (QString const & *name*) [private]

Creates and adds a category with the specified *name* and returns a pointer to it.

See Also

void DataWidget::addCategory(Category * cat)

3.5.3.3 void DataWidget::addObject (Object * *object*, QString const & *catName*) [private]

Adds the *object* to the category specified by *catName*.

If the specified category doesn't exist it gets created before inserting.

See Also

Category & Category::operator <<(Object * object)

3.5.3.4 void DataWidget::categoryCountChanged (int *count*) [signal]

Gets emitted when the number of categories changes in any way.

The *count* can be used to adapt the availability of actions that require at least one category.

See Also

void MainWindow::categoryCountChanged(int count)

3.5.3.5 void DataWidget::changeZoom (int *newZoom*) [private], [slot]

Called to change the zoomlevel of the TableView to *newZoom*.

The section size of all header views gets set to the new zoomlevel and the current view gets scrolled to keep the current selection centered.

3.5.3.6 void DataWidget::clearData ( ) [slot]

Deletes all tracking data.

The user gets warned and asked to confirm the process.

See Also

void clearDataImmediate()

3.5.3.7 void DataWidget::clearDataImmediate ( ) [private]

Deletes all tracking data without further warning.

This is called before new data is imported. Signal dataDecreased() gets emitted afterwards.

See Also

void clearData()

3.5.3.8 void DataWidget::createCornerWidget ( ) [private]

Creates the TableViews corner widget.

The corner widget contains a button for the clear action as well as the spinbox to adjust the zoomlevel.

3.5.3.9 void DataWidget::createNewCategoryTab ( ) [private]

Creates the special last tab that creates a new category when it gets focus.

The tab has a special '+' icon and contains only a button to create a new tab manually (only used when it's the only tab)

3.5.3.10 void DataWidget::currentFrameChanged (int *framenumber*) [signal]

Gets emitted when the current *framenumber* changes.

This is part of the mechanism to keep the selection in different views consistent.

See Also

void setCurrentFrame(int framenumber)
void VideoWidget::currentFrameChanged(int n);
void VideoWidget::seek(int frame)

3.5.3.11 void DataWidget::dataDecreased ( ) [signal]

Gets emitted when tracking data gets deleted.

Other classes holding pointers to any kind of tracking data should re-request them to prevent dead pointers.

3.5.3.12 void DataWidget::deleteBBox ( ) [slot]

Deletes the selected bounding boxes.

The user gets asked for confirmation and then deletion orders get sent to the current category

See Also

void Category::deleteBBoxes(QModelIndexList const & selection)

3.5.3.13 void DataWidget::deleteCategory ( ) [slot]

Deletes the currently shown category.

The current index gets determined and the corresponding function gets called

See Also

void deleteCategory(int index)
void deleteCategory(QAbstractButton * button)

3.5.3.14 void DataWidget::deleteCategory (int *index*) [private]

Deletes the category with the given *index*.

The user gets asked for confirmation and the corresponding signals get emitted

See Also

void deleteCategory()
void deleteCategory(QAbstractButton * button)

3.5.3.15 void DataWidget::deleteCategory (QAbstractButton * *button*) [private], [slot]

Slot to delete the category associated with the given *button*.

The coresponding category gets determined via the closeBtnGroup.

See Also

void deleteCategory(int index)
void deleteCategory()

3.5.3.16 void DataWidget::deleteObject ( ) [slot]

Deletes the currently selected object.

Actually the selection gets converted to a rownumber and the corresponding function from the currently shown category is called.

See Also

void Category::deleteObjectAt(int row)

3.5.3.17 void DataWidget::editCategory ( ) [slot]

Enables the user to edit the current category.

Determines the current index and calls the appropriate function

See Also

void editCategory(int index)
void editCategory(QAbstractButton * button)

3.5.3.18 void DataWidget::editCategory (int *index*) [private]

Enables the user to edit the category with the given *index*.

A QInputDialog is shown to get the new category name from the user.

See Also

void editCategory()
void editCategory(QAbstractButton * button)

3.5.3.19 void DataWidget::editCategory (QAbstractButton * *button*) [private], [slot]

Slot to edit the category associated with the given *button*.

Determines the index of the button using the editBtnGroup and calls the appropriate function

See Also

void editCategory(int index)
void editCategory()

3.5.3.20 void DataWidget::editObject ( ) [slot]

Enables the user to move objects to another category.

The selection of the new category happens via a dropdown containing all existing categories.

3.5.3.21 void DataWidget::exportBBFile (QFile & *file*) [private]

Exports tracking data as a BB file.

Note

Since the BB file format doesn't support categories, interpolated boxes and gaps between frames this information will be lost.

See Also

void exportFile()
void exportViperFile(QFile & file)

3.5.3.22 void DataWidget::exportFile ( ) [slot]

Exports data to a file in a foreign format.

Asks the user for a filename and type, creates a file and calls the appropriate export function.

See Also

void exportBBFile(QFile & file)
void exportViperFile(QFile & file)

3.5.3.23 void DataWidget::exportViperFile (QFile & *file*) [private]

Exports tracking data as a ViPER file.

Note

Since the ViPER file format doesnt support interpolated boxes they will be transformed to single boxes.
The official definition of the ViPER file format can be viewed here: ViPER XML

See Also

void exportFile()
void exportBBFile(QFile & file)

3.5.3.24 QList< BBox > DataWidget::getBBoxes (int *framenumber*) const

Returns the bounding boxes of all objects of all categories for the specified *framenumber*.

Internally all categories get asked for their matching bounding boxes, which in turn ask all their objects for their matching bounding boxes. So this kinda ripples through until the actual boxes are reached.

Note

Since the returned list contains copies of the boxes altering them will not affect the original data.

See Also

QList<BBox> Category::getBBoxes(int framenumber) const
BBox Object::getBBox(int framenumber) const

3.5.3.25 BBox::Type DataWidget::getCurrentBBoxType ( ) const

Returns the type of the current BBox.

This function is used to keep selection dependent actions in sync

3.5.3.26 Object * DataWidget::getObject (int *id*) const

Returns a pointer to the object with the specified *ID*.

If no such object exists a NULL pointer is returned instead.

3.5.3.27 QList< int > DataWidget::getSelectedRows ( ) const [private]

Returns the rows contained in the current selection.

Note

The list of rownumbers doesn't contain duplicates and is sorted in descending order

3.5.3.28 DataWidget::Cell DataWidget::getSelection ( ) const [inline], [private]

Determines the current selection (single cell) and returns it.

The selection consists of the corresponding tab index and the table row and column.

3.5.3.29 void DataWidget::importBBFile (QFile & *file*) [private]

Imports tracking data from a BB file.

If the reading or parsing fails at any stage before tracking data could be read the function simply aborts.

See Also

void importFile()
void importViperFile(QFile & file)

3.5.3.30 void DataWidget::importFile ( ) [slot]

Imports data from a file in a foreign format.

The corresponding filename gets asked from the user via a QFileDialog and the file is opened using the apropriate function.

See Also

void importBBFile(QFile & file)
void importViperFile(QFile & file)

3.5.3.31 void DataWidget::importViperFile (QFile & *file*) [private]

Imports tracking data from a ViPER file.

If the reading or parsing fails at any stage before tracking data could be read the function simply aborts.

Note

Since the ViPER format has far more potential than we need some informations simply get omitted. This inflicts the whole config node, all but one sourcefile nodes, file and content nodes, others than the first attribute node of an object and finally all other data nodes but data::bbox nodes.
The official definition of the ViPER file format can be viewed here: ViPER XML

See Also

void importFile()
void importBBFile(QFile & file)

3.5.3.32 bool DataWidget::isObjectSelected ( ) const

Returns if a object is selected.

This function is used to keep selection dependent actions in sync

3.5.3.33 void DataWidget::newCategory ( ) [slot]

Creates a new category and adds it to the internal list.

The categorie's name gets asked from the user via a QInputDialog. If a category with the same name allready exists or the user aborts the dialog nothing is done.

See Also

void deleteCategory()

3.5.3.34 void DataWidget::newObject ( ) [slot]

Creates a new object and adds it to the currently shown category.

Internally simply the corresponding function of the shown categorie is called.

See Also

void Category::newObject()

3.5.3.35 void DataWidget::onCurrentTabChanged (int *index*) [private], [slot]

Gets called when the current tab changes to *index*.

This is used to catch the case that the last tab gets focus. In this case the focus is set to the first index and a new category is created. ("new tab" tab behavior)

See Also

void newCategory()

3.5.3.36 void DataWidget::openFile ( ) [slot]

Opens a data file.

If anything bad happens before any data can be read the function aborts with a warning, else the old data is cleared an the new is read from th file.

Note

The filename is saved in filename for saveFile()

3.5.3.37 void DataWidget::requestVideo (QString *filename*) [signal]

Gets emitted when a data file containing information about a associated video is opened.

The *filename* of the video gets submitted so the VideoWidget can try to open it.

See Also

void VideoWidget::openRequest(QString openFilename)

3.5.3.38 void DataWidget::saveFile ( ) [slot]

Saves the current data to a file.

The filename is taken from filename. If this is empty saveFileAs() is called

See Also

void saveFileAs()

3.5.3.39 void DataWidget::saveFileAs ( ) [slot]

Saves the current data to a file.

The filename is asked from the user and saved in filename

3.5.3.40 void DataWidget::selectedObjectChanged (int *id*) [signal]

Gets emitted when another object gets selected and submits its *ID*.

This is part of the mechanism to keep the selection in different views consistent.

See Also

void setSelectedObject(int id)
void VideoWidget::selectionChanged(int id)
void VideoWidget::changeSelection(int id)

3.5.3.41 void DataWidget::selectionChanged (QModelIndex const & *current*, QModelIndex const & *previous*) [private], [slot]

Adapter from the selectionChanged Signal from the ListView to the one from this class.

The QModelIndex from the signal gets converted to the corresponding objects unique ID. The signal selectionChanged(int id) then gets emitted with said ID.

3.5.3.42 void DataWidget::selectNextKeyframe ( ) [slot]

Selects the next keyframe of the current object.

Note

Keyframe refers to a frame containing a key box for the current object

3.5.3.43 void DataWidget::selectPreviousKeyframe ( ) [slot]

Selects the previous keyframe of the current object.

Note

Keyframe refers to a frame containing a key box for the current object

3.5.3.44 void DataWidget::setCurrentFrame (int *framenumber*) [slot]

Sets the selection to the cell with the specified *framenumber*.

See Also

void currentFrameChanged(int framenumber)
void VideoWidget::seek(int frame)
void VideoWidget::currentFrameChanged(int n);

3.5.3.45 void DataWidget::setSelectedObject (int *id*) [slot]

Changes the selection to the object with the specified *ID*.

See Also

void selectionChanged(int id);
void GLWidget::changeSelection(int id)
void GLWidget::selectionChanged(int id)

3.5.3.46 void DataWidget::setSelectedObjectByRow (int *row*)

Sets the selection to the specified *row*.

Note

This doesn't use setSelection and therefore doesn't scroll to the new selection

3.5.3.47 void DataWidget::setSelection (int *tab,* int *row,* int *column*) [private]

Sets the selection.

This is used internally to change the selection and scrolls the view to center the new selection.

3.5.3.48 void DataWidget::setVFInfo (VideofileInfo const & *newVideofileInfo*)

Setter for the videofileInfo.

This is used so the VideoWidget can tell the DataWidget about a newly loaded videos properties.

3.5.3.49 void DataWidget::sortByFN ( ) [slot]

Sorts the tracking data by framenumber.

Actually every category is told to sort itself

See Also

void Category::sortByFN()

3.5.3.50 void DataWidget::sortByID ( ) [slot]

Sorts the tracking data by ID.

Actually every category is told to sort itself

See Also

void Category::sortByID()

3.5.3.51 void DataWidget::updateActions ( ) [signal]

Gets emitted whenever the selection changes.

This is used so the MainWindow can adapt the availability of selection dependent actions.

See Also

void VideoWidget::updateActions()
void MainWindow::updateActions()

3.5.3.52 void DataWidget::updateFramecount ( ) [private]

Determines and sets the maximum framecount of all objects.

This is used to get a default width for the views if no video file is opened.

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

- datawidget.h
- datawidget.cpp

3.6 IDCounter Class Reference

Class managing the unique object IDs.

`#include <idcounter.h>`

Public Member Functions

- IDCounter ()
  Default c'tor.
- int getID ()
  Returns the next free ID.
- void reset (int id=0)
  Resets the counter to the given ID.

Static Public Member Functions

- static IDCounter * getGlobalInstance ()
  returns a pointer to the global instance

Private Attributes

- int nextID

  the next ID that can be requested

Related Functions

(Note that these are not member functions.)

- #define idCounter IDCounter::getGlobalInstance()

  A simple define to make calling the global instance easier.

3.6.1 Detailed Description

Class managing the unique object IDs.

The class has a global instance and everytime you request a new ID the counter gets increased. You can also reset the counter or set it to a given ID.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 IDCounter::IDCounter ( )

Default c'tor.

The nextID gets initialized to 0.

3.6.3 Member Function Documentation

3.6.3.1 IDCounter * IDCounter::getGlobalInstance ( ) [static]

returns a pointer to the global instance

The global instance is not really global but a function static one. Bazinga!

3.6.3.2 void IDCounter::reset (int *id* = 0)

Resets the counter to the given *ID*.

If no ID is given it defaults to 0.

3.6.4 Friends And Related Function Documentation

3.6.4.1 #define idCounter IDCounter::getGlobalInstance() [related]

A simple define to make calling the global instance easier.

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

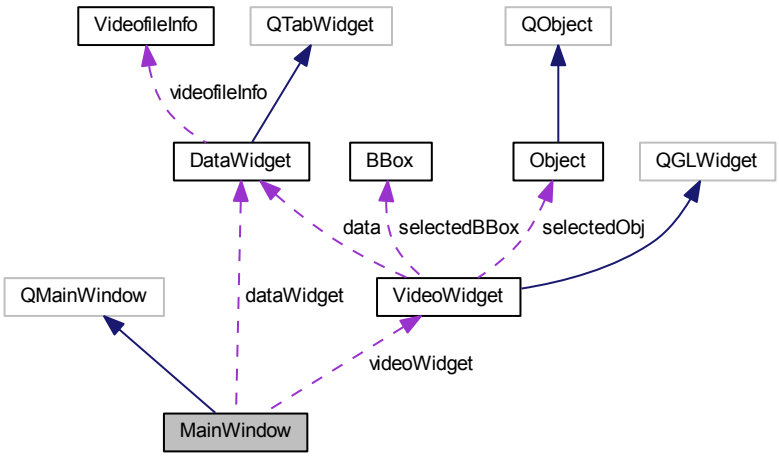
- idcounter.h
- idcounter.cpp

3.7 MainWindow Class Reference

Class managing the GUI and connecting the other classes.

`#include <mainwindow.h>`

Collaboration diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget *parent=0)

  Default ctor.
- ~MainWindow ()

  Default dtor.

Private Slots

- void zoomChanged (float zoom)

  Updates the zoomLabel.
- void updateSeekLabel (int n)

  Updates the seekLabel.
- void changeMaxFrames (int n)

  Updates the seek slider range.
- void dataContextMenu (QPoint const &pos)

  Creates and shows the data widgets context menu.
- void categoryCountChanged (int count)

  Updates selection dependent actions.
- void updateActions ()

  Updates selection dependent actions.

Private Member Functions

- void initGUI ()

  Inits the GUI.
- void createActions ()

  Creates the Actions.
- void createMenus ()

  Creates the Menus.
- void createToolbars ()

  Creates the Toolbars.
- void createDockWidgets ()

  Creates the DockWidgets.
- void createCentralWidget ()

  Creates the central widget (VideoWidget)
- void createStatusBar ()

  Creates the status bar.
- void createAboutDialog ()

  Creates the about dialog.
- QLayout * createZoomLayout ()

  Creates the layout for the zoom actions.
- void createIcons ()

  Creates the 'new' and 'convert' icons.

Private Attributes

- VideoWidget * videoWidget

  The GLWidget instance.
- DataWidget * dataWidget

  The TrackingdataWidget instance.
- QSlider * slider

  The Slider for seeking in the video.
- QAction * openVideoAct

  Action to open a video file.
- QAction * openDataAct

  Action to open a data file.
- QAction * saveDataAct

  Action to save a data file.
- QAction * saveDataAsAct

  Action to save a data file under a specific filename.
- QAction * importDataAct

  Action to import a data file.
- QAction * exportDataAct

  Action to export a data file.
- QAction * quitAct

  Action to quit the application.
- QAction * playPauseAct

  Action to start/pause video playback.
- QAction * nextFrameAct

  Action to seek forward.
- QAction * previousFrameAct

  Action to seek backward.

- QAction * nextFrameKeyAct

  Action to seek forward until keybox.
- QAction * previousFrameKeyAct

  Action to seek backward until keybox.
- QAction * nextCategoryAct

  Action to select the next category.
- QAction * previousCategoryAct

  Action to select the previous category.
- QAction * nextObjectAct

  Action to select the next object.
- QAction * previousObjectAct

  Action to select the previous object.
- QAction * clearDataAct

  Action to clear all the tracking data.
- QAction * newBoxAct

  Action to create a new bounding box.
- QAction * newKeyboxAct

  Action to create a new key bounding box.
- QAction * deleteBoxAct

  Action to delete a bounding box.
- QAction * zoomInAct

  Action to zoom in.
- QAction * zoomOutAct

  Action to zoom out.
- QAction * zoomResetAct

  Action to reset zoom.
- QAction * zoomFitAct

  Action to activate auto zoom.
- QAction * newCatAct

  Action to create a new category.
- QAction * deleteCatAct

  Action to delete a category.
- QAction * editCatAct

  Action to rename a category.
- QAction * sortByIDAct

  Action to sort all data by object ID.
- QAction * sortByFNAct

  Action to sort all data by appearence.
- QAction * newObjAct

  Action to create a new object.
- QAction * deleteObjAct

  Action to delete a object.
- QAction * editObjAct

  Action to edit a object.
- QAction * toggleCacheAct

  Action to switch cache on or off.
- QAction * toggleCenterlineAct

  Action to switch centerline visibility on or off.
- QAction * setCacheProperties

  Action to manipulate the cache.
- QAction * aboutAction

Action to show a about dialog.

- QLabel * seekLabel

  The label shows the current framenumber.

- QLabel * zoomLabel

  The label shows the current zoom factor.

- QLabel * timeLabel

  The label shows the elapsed time.

- QIcon newSingleBoxIcon

  Icon for a the new single box action.

- QIcon newKeyBoxIcon

  Icon for a the new key box action.

- QIcon convertSingleBoxIcon

  Icon for a the convert to single box action.

- QIcon convertKeyBoxIcon

  Icon for a the convert to key box action.

- QDialog * aboutDialog

3.7.1 Detailed Description

Class managing the GUI and connecting the other classes.

The MainWindow holds all the QActions used for user interaction and instances and connects the Trackingdata-Widget and GLWidget.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

Default ctor.

See Also

void initGUI()

3.7.3 Member Function Documentation

3.7.3.1 void MainWindow::categoryCountChanged (int *count*) [private], [slot]

Updates selection dependent actions.

This only concerns actions that need one or more category present.

3.7.3.2 void MainWindow::createDockWidgets ( ) [private]

Creates the DockWidgets.

Apparently there is only the data dock widget.

3.7.3.3 void MainWindow::createIcons ( ) [private]

Creates the 'new' and 'convert' icons.

These icons are used in different places so they are defined classwide.

3.7.3.4 void MainWindow::createStatusBar ( ) [private]

Creates the status bar.

Actually the possible status bar isn't used, but if you plan to do so here is the place to set it up.

3.7.3.5 QLayout * MainWindow::createZoomLayout ( ) [private]

Creates the layout for the zoom actions.

This layout is part of the central widget

3.7.3.6 void MainWindow::dataContextMenu (QPoint const & *pos*) [private], [slot]

Creates and shows the data widgets context menu.

The context menu contains all the data actions but all inactive (because currently useless) actions are removed.

3.7.3.7 void MainWindow::initGUI ( ) [private]

Inits the GUI.

This involves creating the data and video widgets and all actions, menus, toolbars etc...

3.7.3.8 void MainWindow::updateActions ( ) [private], [slot]

Updates selection dependent actions.

The type of the currnetly selected bounding box and whether or not a object is selected get asked from the user and the actions get updated accordingly.

3.7.3.9 void MainWindow::updateSeekLabel (int *n*) [private], [slot]

Updates the seekLabel.

The timeLabel gets updated, too.

3.7.4 Member Data Documentation

3.7.4.1 QDialog* MainWindow::aboutDialog [private]

The about dialog

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

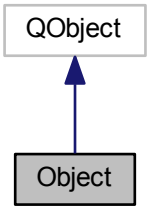
- mainwindow.h
- mainwindow.cpp

3.8 Object Class Reference

Represents a object in the video consisting of several BBoxes.

`#include <object.h>`

Collaboration diagram for Object:



Signals

- void dataChanged (int objectID, int framenumber)

  Gets emitted when a bbox gets added directly.

Public Member Functions

- Object ()

  Creates an empty object.
- Object (QDataStream &in)

  Creates an object from a stream.
- Object (QDomElement const &objectElem)

  Creates an object from a `QDomElement` *containing an object node from a viper file.*
- void addBBox (BBox const &bbox)

  Adds the bounding box bbox to the internal list.
- void deleteBBoxAt (int framenumber)

  Removes the bounding box with the given framenumber from the internal list.
- int getID () const

  Getter for id.
- BBox getBBox (int framenumber) const

  Returns a bounding box for the specified framenumber.
- BBox * getBBoxPointer (int framenumber)

  Returns a pointer to the bounding box for the specified framenumber.
- BBox * getPrecedingBBoxPointer (int framenumber)

  Returns a pointer to the bounding box preceding the box with the given framenumber.
- QMap< int, BBox > const & getBBoxes () const

  Getter for bboxes.
- bool isEmpty () const

  Returns true if the object doesn't contain any bounding boxes.
- QDomElement toViperNode (QDomDocument &doc, QString const &catName) const

  Returns a `QDomElement` *containing the object in the viper format.*
- BBox const & firstBBox () const

  Returns a reference to the first existing bounding box.
- BBox const & lastBBox () const

  Returns a reference to the last existing bounding box.
- void save (QDataStream &out) const

  Saves the data to a stream.

Private Member Functions

- QString getViperFramespan () const

  Returns the framespan of the object as a string.

Private Attributes

- int id

  The unique ID of the object.
- QMap< int, BBox > bboxes

  The list of bounding boxes.

Related Functions

(Note that these are not member functions.)

- bool lessThanByID (Object const *const object1, Object const *const object2)
- bool lessThanByFN (Object const *const object1, Object const *const object2)

3.8.1 Detailed Description

Represents a object in the video consisting of several BBoxes.

In detail the class only consists of a unique ID given at creation and a QMap of BBox instances.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Object::Object ( )

Creates an empty object.

The object gets assigned a unique ID so it can be identified.

3.8.2.2 Object::Object (QDataStream & *in*) [explicit]

Creates an object from a stream.

The stream data is interpreted as an object in the BTD file format.

3.8.2.3 Object::Object (QDomElement const & *objectElem*) [explicit]

Creates an object from a QDomElement containing an object node from a viper file.

The ctor parses the viper object node for bounding boxes and adds them as BBox instances to the internal list. Boxes that last for more than one frame get converted to a BBox::SINGLE and a BBox::KEYBOX typed BBox marking the beginning and the end of the box from the viper file.

Note

The object gets assigned a new unique ID, whereas the ID from the viper file gets omitted to keep the integrity of the internal ID generator.

See Also

QDomElement toViperNode(QDomDocument & doc, QString const & catName) const

3.8.3 Member Function Documentation

3.8.3.1 void Object::addBBox (BBox const & *bbox*)

Adds the bounding box *bbox* to the internal list.

The box gets inserted so that the list of boxes stays sorted by framenumber. If a box with the given framenumber already exists it will be replaced by the new box.

3.8.3.2 void Object::dataChanged (int *objectID*, int *framenumber*) [signal]

Gets emitted when a bbox gets added directly.

This is used to inform the wrapping category about changes made directly to this class.

3.8.3.3 void Object::deleteBBoxAt (int *framenumber*)

Removes the bounding box with the given *framenumber* from the internal list.

If no such box exists nothing happens.

3.8.3.4 BBox const & Object::firstBBox ( ) const

Returns a reference to the first existing bounding box.

This is just a convenience function.

See Also

BBox const & lastBBox() const

3.8.3.5 BBox Object::getBBox (int *framenumber*) const

Returns a bounding box for the specified *framenumber*.

If there is no box defined for this frame either a interpolated or a NULL bounding box is constructed and returned, according to the surrounding boxes.

See Also

BBox * getBBox(int framenumber)

3.8.3.6 BBox * Object::getBBoxPointer (int *framenumber*)

Returns a pointer to the bounding box for the specified *framenumber*.

The box is a existing, modifiable one; instead of interpolated or NULL boxes a NULL pointer is returned.

See Also

BBox getBBox(int framenumber) const

3.8.3.7 BBox * Object::getPrecedingBBoxPointer (int *framenumber*)

Returns a pointer to the bounding box preceding the box with the given *framenumber*.

If no such box exists a NULL pointer is returned.

3.8.3.8 QString Object::getViperFramespan ( ) const [private]

Returns the framespan of the object as a string.

The string is formatted according to the ViPER format and therefore also contains all holes.

Note

ViPER has 1-based framenumbers, default is 0-based!

3.8.3.9 bool Object::isEmpty ( ) const

Returns true if the object doesn't contain any bounding boxes.

Internal simply the corresponding `isEmpty` function of the `QMap` containing the bounding boxes is forwarded.

3.8.3.10 BBox const & Object::lastBBox ( ) const

Returns a reference to the last existing bounding box.

This is just a convenience function.

See Also

BBox const & firstBBox() const

3.8.3.11 void Object::save (QDataStream & *out*) const

Saves the data to a stream.

The data is saved in the BTD file format.

3.8.3.12 QDomElement Object::toViperNode (QDomDocument & *doc*, QString const & *catName*) const

Returns a `QDomElement` containing the object in the viper format.

Note

If there are virtual boxes they get saved as single boxes except for stationary ones (where the two spanning boxes have the same geometry) which get saved using ViPER's run length encoding.

3.8.4 Friends And Related Function Documentation

3.8.4.1 bool lessThanByFN (Object const *const *object1*, Object const *const *object2*) [related]

A object counts as less than another if the framenumber of its first bounding box is less than the one from the other object. If they are the same the framenumbers of the last bounding box get compared.

3.8.4.2 bool lessThanByID (Object const *const *object1*, Object const *const *object2*) [related]

A object counts as less than another if its ID is less than the others

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

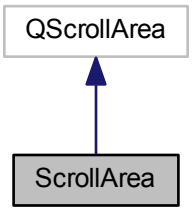
- object.h
- object.cpp

3.9 ScrollArea Class Reference

An extended QScrollArea.

`#include <scrollarea.h>`

Collaboration diagram for ScrollArea:



Signals

- void sizeChanged (QSize size)

  Gets emitted when the size of the scrollare changes.

Public Member Functions

- ScrollArea (QWidget ∗parent=0)

  Default c'tor.

Protected Member Functions

- virtual void mouseMoveEvent (QMouseEvent ∗event)

  Reimplemented mouse move event.
- virtual void mousePressEvent (QMouseEvent ∗event)

  Updates lastPos with the current mouse position.
- virtual void resizeEvent (QResizeEvent ∗event)

  Emits sizeChanged.

Private Attributes

- QPoint lastPos

  Last position of the cursor.

3.9.1 Detailed Description

An extended QScrollArea.

This scrollarea is used to make the VideoWidget scrollable

3.9.2 Member Function Documentation

3.9.2.1 void ScrollArea::mouseMoveEvent (QMouseEvent * *event*) [protected], [virtual]

Reimplemented mouse move event.

If the alt key and left mouse button is pressed the scrollarea's scrollbars get moved with the mouse.

3.9.2.2 void ScrollArea::sizeChanged (QSize *size*) [signal]

Gets emitted when the size of the scrollare changes.

This is used to adjust the zoom of the contained VideoWidget on autozoom

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

- scrollarea.h
- scrollarea.cpp

3.10 VideofileInfo Struct Reference

Header data of a video file bundled for interchange.

`#include <types.h>`

Public Member Functions

- VideofileInfo ()

  Default c'tor.
- VideofileInfo (QString const &name, int n, QSize size)

  Simple constructor to initialise the struct.

Public Attributes

- QString filename

  Name of the file.
- int framecount

  Number of frames contained in the video.
- QSize size

  Resolution of the video.

3.10.1 Detailed Description

Header data of a video file bundled for interchange.

This struct exists to simply get the video information needed to export viper files from the class holding the video data (GLWidget) to the class holding the tracking data (TrackingdataWidget).

3.10.2 Constructor & Destructor Documentation

3.10.2.1 VideofileInfo::VideofileInfo ( )

Default c'tor.

Creates an empty VideofileInfo

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

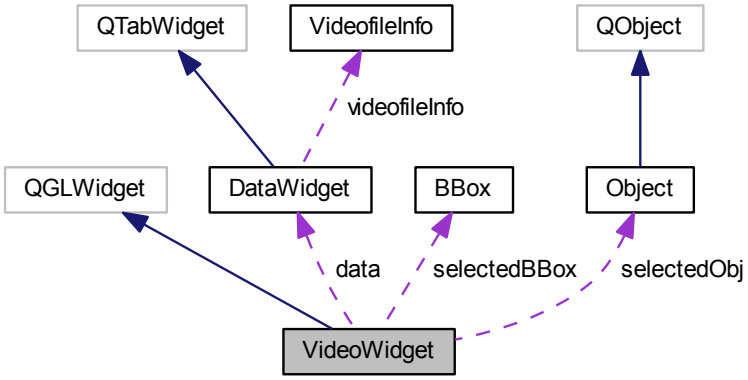
- types.h
- types.cpp

3.11 VideoWidget Class Reference

Class managing the video data and doing all the rendering.

`#include <videowidget.h>`

Collaboration diagram for VideoWidget:



Public Slots

- void openVideoFile ()

  Opens a video file using OpenCV.
- void openRequest (QString openFilename)

  Opens the video file with the given openFilename using openCV.
- void showNextFrame ()

  Shows the next frame of the video.
- void showPreviousFrame ()

  Shows the previous frame of the video.
- void play (bool play=true)

  Toggles Play/Pause.
- void seek (int frame)

  shows a frame specified by its framenumber frame
- void createBBox ()

  Initiates the creation of a new bounding box.
- void createKeyBBox ()

  Initiates the creation of a new key bounding box.
- void zoomIn ()

  Zooms in a bit.

- void zoomOut ()

  Zooms out a bit.
- void zoomFit (bool checked)

  Sets autoZoom to checked.
- void zoomReset ()

  Resets the zoomfactor.
- void changeSelection (int id)

  Changes the selected object to the object with the specified ID.
- void updateData ()

  Rerequests the cached data from the DataWidget.
- void setAvailableSize (QSize newSize)

  Sets the availableSize to newSize.
- void toggleCache ()

  Toggles frame caching.
- void setCacheProperties ()

  Shows a dialog to edit the cache settings.
- void setCacheSize (int newSize)

  Sets the frame cahce size to newSize.
- void setCacheSizeText (int size)

  Updates the frame cache size label of the properties dialog.
- void clearFrameCache ()

  Clears the frame cache.
- void toggleCenterlines ()

  Toggles the visibilit of the centerlines used to trace objects.

Signals

- void playToggled (bool play)

  Gets emitted when the video is started/paused.
- void currentFrameChanged (int n)

  Gets emitted each time a new frame is shown.
- void selectionChanged (int id)

  Gets emitted when an other box gets selected and submits its objects ID.
- void zoomChanged (float zoom)

  Gets emitted whenever the zoomfactor zoom changes.
- void zoomChangedManually (bool manually)

  Gets emitted whenever the zoom is changed actively by the user.
- void maxFramesChanged (int n)

  Emitted whenever a new video is opened.
- void cacheSizeChanged (QString newCacheSizeText)

  Emitted whenever the size of the framecache changes.
- void boxCreationStarted (bool started)

  Emitted with false whenever a initiated box creation gets aborted.
- void updateActions ()

  Emitted whenever the actions of the MainWindow should update.

Public Member Functions

- VideoWidget (DataWidget *data, QWidget *parent=0)

  Ctor which takes a pointer to a DataWidget.
- double getFramerate ()

  Returns the Framerate of the current video.

Protected Member Functions

- void wheelEvent (QWheelEvent ∗event)

  Mouse wheel event handler.
- void mouseMoveEvent (QMouseEvent ∗event)

  Mouse move event handler.
- void mousePressEvent (QMouseEvent ∗event)

  Mouse button press event handler.
- void mouseReleaseEvent (QMouseEvent ∗event)

  Mouse button release event handler.

Private Types

- enum Hitarea {
  NONE =0, TOPLEFT, TOP, TOPRIGHT,
  LEFT, CENTER, RIGHT, BOTTOMLEFT,
  BOTTOM, BOTTOMRIGHT }

  Denotes the hitarea of a bounding box.

Private Member Functions

- void initializeGL ()

  Initialization after context creation.
- void paintGL ()

  Rendering happens here.
- void resizeGL (int width, int height)

  Resize event handler.
- void renderCurrentFrame () const

  Renders the current video frame.
- void renderBBox (BBox const &bbox, bool active=false) const

  Renders a bounding box.
- void renderCenterline () const

  Renders the centerline for the active object.
- void renderSelectedObject () const

  Renders the currently selected object.
- void resizeTexture ()

  Recreates the frame texture with the specified size.
- void updateTexture (cv::Mat const &mat) const

  Uploads the frame in mat to the frame texture.
- void updateCursor ()

  Updates the cursor according to its context.
- Hitarea isHit (QRect const &rect, QPoint const &pos) const

  Determines which Hitarea (handle) of the rect was hitten by the cursors pos.
- void setZoom (qreal newZoom)

  Sets the zoom level to newZoom.

Private Attributes

- bool autoZoom

  Indicates whether or not the widget should adjust its size to the available space.
- bool showCenterLines

  Indicates whether or not the centerline of the selected object should be shown.
- bool createBoxFlag

  Indicates that a bounding box should be created.
- bool createKeyboxFlag

  Indicates that a key bounding box should be created.
- QSize videoSize

  The video's resolution.
- QSize availableSize

  The available size for the widget.
- qreal zoom

  The current zoomfactor.
- QSizeF texCoords

  The boundary of the texture coordinates to render the video data.
- GLuint currentTexture

  OpenGL name of the texture holding the current frame.
- int currentFrame

  The number of the currently shown frame.
- Object * selectedObj

  The currently selected object.
- BBox * selectedBBox

  The currently selected bounding box.
- Hitarea hitArea

  The area hitten by a click on a bounding box.
- QPoint hitPos

  The point hitten by the mouse on the video.
- QList< BBox > bboxes

  List of currently visible bounding boxes.
- cv::VideoCapture capture

  The OpenCV capture holding the video data.
- DataWidget * data

  Pointer to the tracking data.
- QTimer * timer

  Timer for video playback.
- QMap< int, cv::Mat > fCache

  FrameCache for faster scrollback!
- bool cacheEnabled

  Indicates whether or not the framecaching should be active.
- int cacheSize

  The cache size.
- int cacheSizeFactor

  The cache size factor.

3.11.1 Detailed Description

Class managing the video data and doing all the rendering.

The GL widget can load video files using OpenCV and can render it and the corresponding bounding boxes from the TrackingdataWidget via OpenGL. It also allows direct mouse interaction with the boxes.

3.11.2 Member Enumeration Documentation

3.11.2.1 enum VideoWidget::Hitarea [private]

Denotes the hitarea of a bounding box.

It is used to return which area was hit by a mouse event.

Enumerator:

NONE Box wasn't hit.

TOPLEFT Top left handle was hit.

TOP Top handle was hit.

TOPRIGHT Top right handle was hit.

LEFT Left handle was hit.

CENTER Area of the box which is not a handle was hit.

RIGHT Right handle was hit.

BOTTOMLEFT Bottom left handle was hit.

BOTTOM Bottom handle was hit.

BOTTOMRIGHT Bottom right handle was hit.

3.11.3 Member Function Documentation

3.11.3.1 void VideoWidget::boxCreationStarted (bool *started*) [signal]

Emitted with false whenever a initiated box creation gets aborted.

This notation is used so it can connect directly to a setChecked(bool) slot of a QAction.

3.11.3.2 void VideoWidget::changeSelection (int *id*) [slot]

Changes the selected object to the object with the specified *ID*.

The corresponding object is retrieved from the DataWidget. The selectedBBox is updated as well.

See Also

void selectionChanged(int id)

3.11.3.3 void VideoWidget::createBBox ( ) [slot]

Initiates the creation of a new bounding box.

Internally the createBoxFlag is set and the cursor is set to a crosshair. If a virtual box exists it gets converted to a single box instead and if a box exists for the last frame it gets copied instead, so it should be easier to create consistent boxes.

See Also

void createKeyBBox()

3.11.3.4 void VideoWidget::createKeyBBox ( ) [slot]

Initiates the creation of a new key bounding box.

Internally the createKeyboxFlag is set and the cursor is set to a crosshair. If a virtual box exists it gets converted to a key box instead and if a box exists before it gets copied instead.

See Also

void createBBox()

3.11.3.5 void VideoWidget::currentFrameChanged (int *n*) [signal]

Gets emitted each time a new frame is shown.

the new frames number is submitted in *n*.

See Also

void seek(int frame)
DataWidget::setCurrentFrame(int framenumber)
DataWidget::currentFrameChanged(int framenumber)

3.11.3.6 double VideoWidget::getFramerate ( )

Returns the Framerate of the current video.

The framerate is obtained from the openCV capture

3.11.3.7 void VideoWidget::initializeGL ( ) [private]

Initialization after context creation.

The GL states are set and a default projection is defined.

3.11.3.8 VideoWidget::Hitarea VideoWidget::isHit (QRect const & *rect*, QPoint const & *pos*) const [private]

Determines which Hitarea (handle) of the *rect* was hitten by the cursors *pos*.

Internally a small QRect is created, moved to every handles position and checked for containment of the *pos*

Note

The *pos* has to be in video coordinates!

3.11.3.9 void VideoWidget::maxFramesChanged (int *n*) [signal]

Emitted whenever a new video is opened.

This is used to keep the seek sliders range in sync.

3.11.3.10 void VideoWidget::mouseMoveEvent (QMouseEvent * *event*) [protected]

Mouse move event handler.

If hitArea is defined the according edge/corner of the box gets moved. Else the cursor gets updated via updateCursor().

3.11.3.11 void VideoWidget::mousePressEvent (QMouseEvent * *event*) [protected]

Mouse button press event handler.

If createBoxFlag or createKeyboxFlag is set a new zero sized bounding box gets created at the *events* position and hitArea is set to BOTTOMRIGHT so a following move event will resize the box.

Else, if a BBox is currently selected the corresponding Hitarea is determined.

If a unselected BBox or nothing gets hit the selectedObj and selectedBBox are updated.

3.11.3.12 void VideoWidget::mouseReleaseEvent (QMouseEvent * *event*) [protected]

Mouse button release event handler.

hitArea is set to NONE and created boxes get normalized

3.11.3.13 void VideoWidget::openRequest (QString *openFilename*) [slot]

Opens the video file with the given *openFilename* using openCV.

If the video doesn't exists nothing happens, else it is opened without any further prompt into the openCV capture and a VideofileInfo gets screated and sent to the DataWidget.

3.11.3.14 void VideoWidget::openVideoFile ( ) [slot]

Opens a video file using OpenCV.

The filename is asked from the user via a QFileDialog and the video is opened using openRequest

3.11.3.15 void VideoWidget::paintGL ( ) [private]

Rendering happens here.

First the current video frame gets rendered, then the currently visible bounding boxes and the selected object.

See Also

void renderCurrentFrame() const
void renderBBox(BBox const & bbox, bool active) const
void renderSelectedObject() const

3.11.3.16 void VideoWidget::play (bool *play* = true) [slot]

Toggles Play/Pause.

The timer gets started with a interval according to the videos FPS and playToggled gets emitted with true.

Note

the timer timeout signal is connected to showNextFrame()

3.11.3.17 void VideoWidget::playToggled (bool *play*) [signal]

Gets emitted when the video is started/paused.

This signal is used to keep the play/pause button on the GUI in sync, where *play* holds whether the video is now playing or paused.

See Also

void MainWindow::togglePlayPause(bool play)

3.11.3.18 void VideoWidget::renderBBox (BBox const & *bbox*, bool *active* = `false`) const [private]

Renders a bounding box.

The color gets chosen according to *active*

3.11.3.19 void VideoWidget::renderCenterline ( ) const [private]

Renders the centerline for the active object.

The centerline connects the centers of all bounding boxes to represent an object

3.11.3.20 void VideoWidget::renderCurrentFrame ( ) const [private]

Renders the current video frame.

Since the frame is saved in a GL texture simply a view filling quad with the said texture is rendered.

Note

Since the texture is ensured to have power of two dimensions the texture coordinates don't reach to 1.0 but to texCoords.

3.11.3.21 void VideoWidget::renderSelectedObject ( ) const [private]

Renders the currently selected object.

The object is represented by a white line connecting the center of all of its BBoxes. The selected box is also rendered as a fat box with visible handles and a white point at its center to clarify its position on the white line.

See Also

void renderCenterline() const
void renderBBox(BBox const & bbox, bool active = false) const

3.11.3.22 void VideoWidget::resizeGL (int *width*, int *height*) [private]

Resize event handler.

Simply updates the viewport to the new *width* and *height*.

3.11.3.23 void VideoWidget::resizeTexture ( ) [private]

Recreates the frame texture with the specified size.

The width and height are powers of two to support legacy video systems. The Coordinate of the loose edge of the actual image data is saved in texCoords for rendering.

3.11.3.24 void VideoWidget::seek (int *frame*) [slot]

shows a frame specified by its framenumber *frame*

seek: loads frames to show into texture. cacheSize frames before actual frame are cached to achieve better scrollback performance This is where framecaching is done: it consists of 6 cases: 1) cache is empty -> refill. 2) the frame requested is in the cache -> just show. 3) the frame is consecutively following the one in the cache (happens often) -> cache new one, delete smallest one. 4) closer to the start than cache is big -> clear cache and refill from start. 5) from new frame backward towards cache is the stuff.. 6) not in cache-> empty cache and refill

3.11.3.25 void VideoWidget::selectionChanged (int *id*) [signal]

Gets emitted when an other box gets selected and submits its objects *ID*.

This is part of the mechanism to keep the selection in different views consistent.

See Also

void changeSelection(int id)
void DataWidget::setSelectedObject(int id)
void DataWidget::selectedObjectChanged(int id)

3.11.3.26 void VideoWidget::setAvailableSize (QSize *newSize*) [slot]

Sets the availableSize to *newSize*.

If autoZoom is set the zoom is adjusted, too.

See Also

void setZoom(qreal newZoom)

3.11.3.27 void VideoWidget::setCacheSize (int *newSize*) [slot]

Sets the frame cahce size to *newSize*.

The cahce also gets cleared

3.11.3.28 void VideoWidget::setZoom (qreal *newZoom*) [private]

Sets the zoom level to *newZoom*.

Also emits signal zoomChanged(float zoom)

3.11.3.29 void VideoWidget::showNextFrame ( ) [slot]

Shows the next frame of the video.

Simply calls seek(currentFrame+1)

See Also

void seek(int frame)

3.11.3.30 void VideoWidget::showPreviousFrame ( ) [slot]

Shows the previous frame of the video.

Simply calls seek(currentFrame-1)

See Also

void seek(int frame)

3.11.3.31 void VideoWidget::updateCursor ( ) [private]

Updates the cursor according to its context.

If it's over a handle of the selected box it gets changed to a resize cursor else it's reset to the default cursor.

3.11.3.32 void VideoWidget::updateData ( ) [slot]

Rerequests the cached data from the DataWidget.

This should be done whenever the data changes and the cached pointers could be invalid.

3.11.3.33 void VideoWidget::updateTexture (cv::Mat const & *mat*) const [private]

Uploads the frame in *mat* to the frame texture.

Note

OpenCV uses BGR, OpenGL uses RGB.

3.11.3.34 void VideoWidget::wheelEvent (QWheelEvent * *event*) [protected]

Mouse wheel event handler.

Scroll up: Zoom in - Scroll down: Zoom out

Note

The zoom factor is saved in zoom and is clamped to [0.1, 2.0]

3.11.3.35 void VideoWidget::zoomChanged (float *zoom*) [signal]

Gets emitted whenever the zoomfactor zoom changes.

This is used to keep the zoomLabel of the MainWindow in sync.

3.11.3.36 void VideoWidget::zoomChangedManually (bool *manually*) [signal]

Gets emitted whenever the zoom is changed actively by the user.

This is used to deactivate autozoom on user interaction

3.11.3.37 void VideoWidget::zoomFit (bool *checked*) [slot]

Sets autoZoom to *checked*.

The video also gets resized to fit the available size

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

- videowidget.h
- videowidget.cpp

Index