CAVAPA-GUI

30.5.2014

Generated by Doxygen 1.8.7

Mon Jun 2 2014 00:07:04

Contents

Chapter 1

CAVAPA-GUI

CAVAPA-GUI is a graphical user interface for the CAVAPA application. The CAVAPA application is a computer vision based movement measurement application that can be used to measure the activity of a group of people. CAVAPA-GUI can be used to perform the calibration required by the CAVAPA application and to visualize the activity measured by the CAVAPA application.

License

Chapter 2

Namespace Index

2.1 Namespace List

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

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

Namespace Documentation

5.1 cavapa_gui Namespace Reference

Basic namespace of the CAVAPA calculation GUI.

Classes

- class ActivityGraph

  Main class for Cavapa graphic curve display widget.
- class AnalysisController

  Base class of the CAVAPA calculations.
- class CalibrationPoints

  CalibrationPoints stores a collection of calibration points that can be drawn on a QPainter.
- class Camera

  The class for a camera source.
- class CaptureInterface

  Interface for the capture devices.
- class Controller

  The Controller class handles the messages between UserInterface and AnalysisController.
- class CopyableListWidget

  A custom list widget class that allows items to be copied to clipboard.
- class ExportDialog

  Class for export dialog of graphwidget.
- struct ExportOptions

  The structure holds CSV export options.
- struct FrameCapture

  The structure is used for storing a single source frame.
- struct FrameStats

  Structure for holding frame statistics on activity.
- class GeneralSettings

  Represents the general settings of the application.
- struct GraphRegion

  Struct for one region of the graph.
- struct GraphSettings

  Graph settings type for font/colors/line widths.
- class GraphSettingsDialog

  Class for settings dialog of graphwidget.

- class GraphWidget

  The widget holds the ActivityGraph and all its controls.
- class GridEngine

  Draws a grid with OpenGL.
- class HTTPCapture

  The Capture device for HTTP images.
- class KeyValueCollection

  Represents a collection of key-value pairs.
- class MainWindow

  The main window for the CAVAPA-GUI application.
- struct Marker

  The structure holds the properties of a single marker on the graph.
- class MarkerDialog

  Class for marker creation dialog of the graphwidget.
- class Metadata

  Represents the metadata associated with a measurement.
- class MetaWidget

  Represents the widget containing all metadata related to a measurement.
- class OpenCVCapture

  The OpenCV capturing class.
- class OpenStreamDialog

  Represents a dialog for opening a stream.
- class Recorder

  The buffered recorder class creates the video file from the images.
- struct RecorderStats

  Statistics structure used for Recorder.
- class Results

  The class holds the calculation results for each frame.
- class SceneWidget

  A widget that holds the output and media controls of a video source.
- class Settings

  Provides methods for loading and saving the application settings.
- class SettingsDialog

  Represents the dialog for the general settings of the application.
- class Source

  The camera and video file source base class.
- struct SourceInfo

  Holds the information related to a source including id number, type, settings, a list of the recorded videos and a list of the calibration points.
- class SourceScene

  SourceScene draws a video stream and configures camera parameters.
- class SourceSettings

  Represents the settings for a video source.
- struct SourceStats

  The structure to hold statistics about the source performance.
- class UserInterface

  The UserInterface class defines the methods and signals that a user interface needs to have.
- struct VertexData

  Represents the information of a single vertex.
- class VideoEngine

  A class for drawing the video frame with OpenGL.

- class VideoFile

  The class for the video file source.

- class VideoFileSet

  The extension to the VideoFile class for handling multiple files.

Typedefs

- using FrameTime = std::uint64_t

  Used to store milliseconds interval in frame times.

- using GraphMarker = Marker

  Used to define the marker for graph.

- using SourceID = unsigned int

  Used to indicate unique source ID-numbers.

Enumerations

- enum SourceType {
  CAMERA, **NOTHING**, **STREAM**, **VIDEO**,
  VIDEOSET }

  Available source types are the following ones: CAMERA = hardware or network camera, NOTHING = not a working source, STREAM = network stream, VIDEO = video file and VIDEOSET = set of multiple files.

- enum ErrorLevel : int { **INFO** = 1, **WARNING** = 2, **CRITICAL** = 3 }

  Error level indicator.

- enum EXPORT_FLAGS : int { **NOTHING** = 0x00, **APPEND** = 0x01, **USE_COMMA** = 0x02 }

  The flags that are used for exporting the results.

- enum FrameFlags : std::uint32_t {
  FRAME_DEFAULT = 0, **FRAME_STOP** = 1, **FRAME_START** = 2, **FRAME_DUPLICATE** = 4,
  FRAME_WARNING = 8 }

  Parameter type to hold frame specific flags.

- enum State {
  Start, **NewMeasurement**, **CalculationInProgress**, **CalculationCancelled**,
  CalculationStopped, **CalculationCompleted**, **CalculationPaused**, **MeasurementOpened** }

  The State enum lists the possible application main states when performing a measurement.

Variables

- const int UNKNOWN_DEVICE = -2

  Used to indicate unknown hardware device.

- const SourceID UNDEFINED_SOURCE = 0

  Used to indicate unknown sources.

5.1.1 Detailed Description

Basic namespace of the CAVAPA calculation GUI.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum cavapa_gui::ErrorLevel : int [strong]

Error level indicator.

Values are comparable, higher integer values indicate more critical error. INFO = just a regular information without affecting execution, WARNING = user should take note of this error but execution continues, ERROR = critical error and the execution is interrupted.

5.1.2.2 enum cavapa_gui::EXPORT_FLAGS : int [strong]

The flags that are used for exporting the results.

Remarks

You can use bitwise OR operations (|) to combine several flags.

5.1.2.3 enum cavapa_gui::FrameFlags : std::uint32_t [strong]

Parameter type to hold frame specific flags.

These flags can indicate the starting and stopping -points of calculation, warnings during calculation and frames that contain no new information from the previous one. The duplicate method can be used to "freeze" video file recording when no movement is detected.

Chapter 6

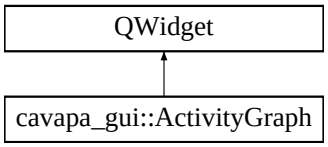
Class Documentation

6.1 cavapa_gui::ActivityGraph Class Reference

Main class for Cavapa graphic curve display widget.

`#include <activitygraph.h>`

Inheritance diagram for cavapa_gui::ActivityGraph:



Public Slots

- void zoomIn (QDateTime start, QDateTime end)

  Zooms graph in.
- void zoomOut ()

  Zooms graph out.
- void scrollLeft ()

  Scrolls graph left.
- void scrollRight ()

  Scrolls graph right.
- void scrollMouse (int position)

  Scrolls zoomed graph area with scrollbar/mouse.
- void setStartTime (QDateTime time)

  Gets the graph region starttime from the timeEdit of parent.
- void setEndTime (QDateTime time)

  Gets graph region endtime from timeEdit of parent.
- void setTimeRelative (bool relative)

  Changes time type between relative/absolute.
- void showActivity (bool activity)

  Changes between showing/hiding activity curve.
- void showCount (bool count)

  Changes between showing/hiding count curve.
- void showMarkerTexts (bool show)

  Changes between showing/hiding marker texts without mouseover or selecting.

- void addMarker (std::string text)

  Adds new marker to the selected position.
- void removeSelectedMarker ()

  Removes the selected time marker (if an active marker is selected).
- void setFollowLatest (bool latest)

  Sets the graph to follow only the latest data.
- void setLatestLength (QTime length)

  Sets the length of latest-data mode.
- void setNewSettings (GraphSettings settings)

  Gets new settings for font, colors and line widths.
- void setMarkers (std::vector< GraphMarker > markers)

  Sets the markers from parent.
- void setStartDateTime (QDateTime start)

  Sets start datetime for the graph.
- void setCurrentPosition (QDateTime point)

  Sets the current selection to time point.

Signals

- void statisticsRequested (FrameTime start, FrameTime stop, int points)

  Signal to be emitted when the graph wants data to be retrieved.
- void sendTimeArea (QDateTime start, QDateTime end)

  Sends Selection start and end time points.
- void sendFrameTimePosition (FrameTime position)

  Sends frametime of the selection position.
- void sendScrollPosition (int position)

  Sends correct scroll bar position when zooming in/out.
- void changeRTLabel (bool rt)

  Changes the realtime label of parent.
- void changeAbsoluteEnabled (bool ae)

  Sets absolute radio button enabled or disabled.
- void changeFixed ()

  Sets parent's fixed time window -checkbox unchecked.
- void sendSelectedPoint (QDateTime point)

  Sends QDateTime of selected position.
- void sendAddNewMarkerNotify ()

  Sends a notify to parent of marker adding (when doubleclickin)
- void setZoomInActive (bool active)

  Sets zoom in button enabled or disabled.
- void setZoomOutActive (bool active)

  Sets zoom out button enabled or disabled.
- void setScrollLeftActive (bool active)

  Sets left slide button enabled or disabled.
- void setScrollRightActive (bool active)

  Sets right slide button enabled or disabled.
- void setScrollBarSize (int size)

  Sets scroll bar size.
- void markerChanged ()

  Informs the parent of marker adding.

Public Member Functions

- ActivityGraph (QWidget *parent=0)
  Creates and initializes the graph.
- GraphRegion sendGraphRegion ()
  Sends graph selection start and end positions.
- void setPrinting ()
  Sets render function ready to print graph into image file.
- void paint (QPainter &painter)
  Draws the graph.
- void updateStatistics (const std::vector< FrameStats > &stats)
  Updates the graph with new statistics.
- void resetGraph ()
  Resets the graph.
- FrameTime QDateTimeToFrameTime (QDateTime time)
  Changes QDateTime to FrameTime according the starting time of calculation.
- QDateTime FrameTimetoQDateTime (FrameTime ftime)
  Changes FrameTime into QDateTime according the beginframe.
- std::vector< GraphMarker > getMarkers ()
  Gets markers from the graph.
- QDateTime getStartDateTime ()
  Gets the start time of the graph.
- FrameTime getStartFrame ()
  Gets the first frame of the graph.
- QDateTime getSelectedTimePoint ()
  Gets the time point of currently selected frame.

Protected Member Functions

- void paintEvent (QPaintEvent *event)
  Qt paint event for drawing the graph.
- void mousePressEvent (QMouseEvent *event)
  Mouse Press event of the graph.
- void mouseReleaseEvent (QMouseEvent *event)
  Mouse release event of the graph.
- void mouseMoveEvent (QMouseEvent *event)
  Mouse move event of the graph.
- void mouseDoubleClickEvent (QMouseEvent *event)
  Double-clicking the graph notifies the parent to add new marker.
- void timerEvent (QTimerEvent *event)
  Graph's own timer event to request data from parent.
- void normalizedata ()
  Normalizes input data arrays into normalized arrays.

6.1.1 Detailed Description

Main class for Cavapa graphic curve display widget.

Qt widget that normalizes and presents the data, allows mouse actions for data interaction like selection zooming. It also presents markers for time events and timestamps for frame time position.

Author

Joel Kivelä

6.1.2 Constructor & Destructor Documentation

6.1.2.1 cavapa_gui::ActivityGraph::ActivityGraph (QWidget * *parent* = 0)

Creates and initializes the graph.

Parameters

parent	The parent widget.

6.1.3 Member Function Documentation

6.1.3.1 void cavapa_gui::ActivityGraph::addMarker (std::string *text*) [slot]

Adds new marker to the selected position.

Parameters

text	Text for new marker.

6.1.3.2 void cavapa_gui::ActivityGraph::changeAbsoluteEnabled (bool *ae*) [signal]

Sets absolute radio button enabled or disabled.

Parameters

ae	True/false for if enabled or not.

6.1.3.3 void cavapa_gui::ActivityGraph::changeRTLabel (bool *rt*) [signal]

Changes the realtime label of parent.

Parameters

rt	True/false for if drawing "Realtime" or not.

6.1.3.4 QDateTime cavapa_gui::ActivityGraph::FrameTimetoQDateTime (FrameTime *ftime*)

Changes FrameTime into QDateTime according the beginframe.

Parameters

ftime	FrameTime point.

Returns

QDateTime point.

6.1.3.5 std::vector< GraphMarker > cavapa_gui::ActivityGraph::getMarkers ( )

Gets markers from the graph.

Returns

Markers in vector array.

6.1.3.6 QDateTime cavapa_gui::ActivityGraph::getSelectedTimePoint ( )

Gets the time point of currently selected frame.

Returns

Time point as QDateTime.

6.1.3.7 QDateTime cavapa_gui::ActivityGraph::getStartDateTime ( )

Gets the start time of the graph.

Returns

The start time as QDateTime.

6.1.3.8 FrameTime cavapa_gui::ActivityGraph::getStartFrame ( )

Gets the first frame of the graph.

Returns

Frame Point in FrameTime.

6.1.3.9 void cavapa_gui::ActivityGraph::mouseDoubleClickEvent (QMouseEvent * *event*) [protected]

Double-clicking the graph notifies the parent to add new marker.

Parameters

event	Qt default mouse event.

6.1.3.10 void cavapa_gui::ActivityGraph::mouseMoveEvent (QMouseEvent * *event*) [protected]

Mouse move event of the graph.

Defines selected area.

Parameters

event	Qt default mouse event.

6.1.3.11 void cavapa_gui::ActivityGraph::mousePressEvent (QMouseEvent * *event*) [protected]

Mouse Press event of the graph.

Defines selected point.

Parameters

event	Qt default mouse event.

6.1.3.12 void cavapa_gui::ActivityGraph::mouseReleaseEvent (QMouseEvent * *event*) [protected]

Mouse release event of the graph.

Parameters

event	Qt default mouse event.

6.1.3.13 void cavapa_gui::ActivityGraph::paint (QPainter & *painter*)

Draws the graph.

Parameters

painter	QPainter to do the drawing.

6.1.3.14 void cavapa_gui::ActivityGraph::paintEvent (QPaintEvent * *event*) [protected]

Qt paint event for drawing the graph.

Parameters

event	Qt default paint event.

6.1.3.15 FrameTime cavapa_gui::ActivityGraph::QDateTimeToFrameTime (QDateTime *time*)

Changes QDateTime to FrameTime according the starting time of calculation.

Parameters

time	QDateTime point.

Returns

FrameTime point.

6.1.3.16 void cavapa_gui::ActivityGraph::scrollMouse (int *position*) [slot]

Scrolls zoomed graph area with scrollbar/mouse.

Parameters

position	The relative position of the view area, between 0 and 100.

6.1.3.17 void cavapa_gui::ActivityGraph::sendFrameTimePosition (FrameTime *position*) [signal]

Sends frametime of the selection position.

Parameters

position	Position in FrameTime.

6.1.3.18 GraphRegion cavapa_gui::ActivityGraph::sendGraphRegion ( )

Sends graph selection start and end positions.

Returns

End and start points as a GraphRegion type.

6.1.3.19 void cavapa_gui::ActivityGraph::sendScrollPosition (int *position*) [signal]

Sends correct scroll bar position when zooming in/out.

Parameters

position	Position as an integer from 0 to 100.

6.1.3.20 void cavapa_gui::ActivityGraph::sendSelectedPoint (QDateTime *point*) [signal]

Sends QDateTime of selected position.

Parameters

point	Position in QDateTime.

6.1.3.21 void cavapa_gui::ActivityGraph::sendTimeArea (QDateTime *start,* QDateTime *end*) [signal]

Sends Selection start and end time points.

Parameters

start	QDateTime of selection start.
end	QDataTime of selection end.

6.1.3.22 void cavapa_gui::ActivityGraph::setCurrentPosition (QDateTime *point*) [slot]

Sets the current selection to time point.

Parameters

point	Time point as QDateTime.

6.1.3.23 void cavapa_gui::ActivityGraph::setEndTime (QDateTime *time*) [slot]

Gets graph region endtime from timeEdit of parent.

Parameters

time	The end time.

6.1.3.24 void cavapa_gui::ActivityGraph::setFollowLatest (bool *latest*) [slot]

Sets the graph to follow only the latest data.

Parameters

latest	True/false for if showing only latest time window.

6.1.3.25 void cavapa_gui::ActivityGraph::setLatestLength (QTime *length*) [slot]

Sets the length of latest-data mode.

Parameters

length	Length in QTime.

6.1.3.26 void cavapa_gui::ActivityGraph::setMarkers (std::vector< GraphMarker > *markers*) [slot]

Sets the markers from parent.

Parameters

markers	Vector array of markers.

6.1.3.27 void cavapa_gui::ActivityGraph::setNewSettings (GraphSettings *settings*) [slot]

Gets new settings for font, colors and line widths.

Parameters

settings	The new settings as GraphSettings.

6.1.3.28 void cavapa_gui::ActivityGraph::setScrollBarSize (int *size*) [signal]

Sets scroll bar size.

Parameters

size	Size of bar as integer.

6.1.3.29 void cavapa_gui::ActivityGraph::setScrollLeftActive (bool *active*) [signal]

Sets left slide button enabled or disabled.

Parameters

active	True/false for if button active.

6.1.3.30 void cavapa_gui::ActivityGraph::setScrollRightActive (bool *active*) [signal]

Sets right slide button enabled or disabled.

Parameters

active	True if the right slide button should be active, false otherwise.

6.1.3.31 void cavapa_gui::ActivityGraph::setStartDateTime (QDateTime *start*) [slot]

Sets start datetime for the graph.

Parameters

start	Start time as QDateTime.

6.1.3.32 void cavapa_gui::ActivityGraph::setStartTime (QDateTime *time*) [slot]

Gets the graph region starttime from the timeEdit of parent.

Parameters

time	The start time.

6.1.3.33 void cavapa_gui::ActivityGraph::setTimeRelative (bool *relative*) [slot]

Changes time type between relative/absolute.

Parameters

relative	True/false for if time mode is relative.

6.1.3.34 void cavapa_gui::ActivityGraph::setZoomInActive (bool *active*) [signal]

Sets zoom in button enabled or disabled.

Parameters

active	True/false for if button active.

6.1.3.35 void cavapa_gui::ActivityGraph::setZoomOutActive (bool *active*) [signal]

Sets zoom out button enabled or disabled.

Parameters

active	True/false for if button active.

6.1.3.36 void cavapa_gui::ActivityGraph::showActivity (bool *activity*) [slot]

Changes between showing/hiding activity curve.

Parameters

activity	True/false for if showing activity curve.

6.1.3.37 void cavapa_gui::ActivityGraph::showCount (bool *count*) [slot]

Changes between showing/hiding count curve.

Parameters

count	True/false for if showing count curve.

6.1.3.38 void cavapa_gui::ActivityGraph::showMarkerTexts (bool *show*) [slot]

Changes between showing/hiding marker texts without mouseover or selecting.

Parameters

show	True/false for if showing text markers continously.

6.1.3.39 void cavapa_gui::ActivityGraph::statisticsRequested (FrameTime *start*, FrameTime *stop*, int *points*) [signal]

Signal to be emitted when the graph wants data to be retrieved.

Parameters

start	Period start time. If 0, the period will start from the first possible frame.
stop	Period end time. If 0, the period will end at the last possible frame.
points	Number of points desired to be returned.

6.1.3.40 void cavapa_gui::ActivityGraph::timerEvent (QTimerEvent * *event*) [protected]

Graph's own timer event to request data from parent.

Parameters

event	Qt default timer event.

6.1.3.41 void cavapa_gui::ActivityGraph::updateStatistics (const std::vector< FrameStats > & *stats*)

Updates the graph with new statistics.

Either the graph requested new statistics with requestData()-signal or the graph is being initialized with new statistical information.

Parameters

stats	Statistical information from a certain period.
stats	New statistics in a vector array.

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

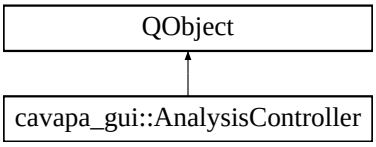
- gui/activitygraph.h
- gui/activitygraph.cpp

6.2 cavapa_gui::AnalysisController Class Reference

Base class of the CAVAPA calculations.

`#include <analysiscontroller.h>`

Inheritance diagram for cavapa_gui::AnalysisController:



Signals

- void calculationComplete ()

  Signals that calculation has ended.
- void error (const std::string &message, ErrorLevel level)

  Signals that something went wrong during the calculation.
- void updated (const std::vector< FrameCapture > &frames, const FrameStats &statistics)

  It signals frame updates or calculation updates.

Public Member Functions

- AnalysisController ()

  Creates the CAVAPA object.
- DISALLOW_COPY_AND_ASSIGN (AnalysisController)

  Copy and assign of the class is not allowed.
- SourceID addSource (const std::string &path, SourceID id=UNDEFINED_SOURCE, FrameTime start_point=0)

  Adds a custom video source.
- bool addSource (const std::vector< std::string > &paths, SourceID id=UNDEFINED_SOURCE, FrameTime start_point=0)

  Adds a custom source with multiple files and a fixed ID number.
- bool canChangeSourceResolution (SourceID id) const

  Checks if the source resolution can be changed.
- bool canPause () const

  Indicates if the running calculation can be paused.
- int countAvailableCameras () const

  Counts the available connected hardware cameras.
- bool exportResults (const ExportOptions &options) const

  Exports the current Results to the CSV File.
- double getCalculationFPS () const

  Retrieves the calculation framerate.
- std::string getLastError () const

  Retrieves the last error that occured.
- std::vector< FrameStats > getResults (FrameTime start, FrameTime stop, int points) const

  Returns calculation results from a given period.
- int getSourceCount () const

  Retrieves the total count of sources.
- std::string getSourceDescription (SourceID id) const

  Returns the description of the source.
- std::vector< cv::Rect > getSourceExcludes (SourceID id) const

  Returns the currently excluded rectangles from the source.
- std::vector< std::string > getSourceFilenames (SourceID id) const

  Returns the list of the files associated with the source.
- std::vector< SourceID > getSourceListing () const

  Returns a list of all sources in the class.
- FrameTime getSourcePosition (SourceID id) const

  Retrieves the current time position of a source.
- cv::Size getSourceResolution (SourceID id) const

  Retrieves the source's frame resolution.
- SourceStats getSourceStats (SourceID id) const

  Retrieves the source's current statistics.
- SourceType getSourceType (SourceID id) const

  Returns the type of the selected source.
- FrameTime getVideoLength (SourceID id) const

  Retrieves the length of the video source.
- int initCameras (int max_loops=10)

  Initializes the hardware cameras.
- bool isPaused () const

  Retrieves the current pause status.
- bool isRunning () const

Retrieves the current status of the calculation.

- bool load (const std::string &path="results.cpa")

  Loads the calculation results and recorded file information.
- bool pause ()

  Pauses or unpauses the calculation.
- bool removeSource (SourceID id)

  Removes the source.
- bool reset ()

  Resets the calculation results.
- bool save (const std::string &path="results.cpa")

  Saves the calculation results and sightings.
- bool setCameraFPS (double fps)

  Sets the default FPS for camera sources.
- void setHighlight (bool highlight)

  Sets the individual sighting rectangle highlights.
- void setHighlightColor (const cv::Scalar &color)

  Sets the sighting's highlighting color.
- bool setPosition (FrameTime pos)

  Sets the current time position of all the sources.
- bool setRecorderCodec (const std::string &codec, const std::string &extension="avi")

  Sets the default camera recorder codec.
- void setRecordTimeLimit (unsigned int length)

  Sets the maximum length of one recording.
- bool setSourceBarrelCorrection (SourceID id, double amount)

  Sets the barrel correction on a source.
- void setSourceExcludes (SourceID id, const std::vector< cv::Rect > &rects)

  Sets the excluded rectangles for a source.
- bool setSourceResolution (SourceID id, const cv::Size &new_size)

  Sets the source resolution.
- bool setVideoSpeed (double ratio=1.0)

  Sets the video source speed.
- void setWorkDirectory (const std::string &path)

  Sets the path for the calculation directory.
- bool sourceCanRecord (SourceID id) const

  Checks if the source is capable of recording.
- bool sourcePlay (SourceID id)

  Starts playing a video source.
- bool sourceSeek (SourceID id, FrameTime position)

  Seeks the video source to a specific time.
- bool sourceStepBackward (SourceID id)

  Steps the source one frame back.
- bool sourceStepForward (SourceID id)

  Steps the source one frame forward.
- bool sourceStop (SourceID id)

  Stops playing the video source.
- bool start (const std::vector< SourceID > &calc_sources, const std::vector< cavapa::CameraSettings > &settings, const std::vector< cavapa::CalibrationPoint > &points, bool record=false, int calc_time=0)

  Starts the CAVAPA calculation.
- void stop ()

  Stops the current CAVAPA calculation.

6.2.1 Detailed Description

Base class of the CAVAPA calculations.

The class is the heart of the CAVAPA calculation handling. Its purpose is to retrieve frames from the sources and pass them to the CAVAPA algorithm. It also stores the calculation results and includes the frame update timer.

The class can detect connected hardware cameras with initCameras() but any other video source must be manually added with addSource(). Once the sources are selected the calculation can be started with start().

During the calculation the class sends signals updated() on each frame. The calculation can then be stopped with stop() or if the calculation terminates on its own signal calculationComplete() is sent. Any errors during the calculation can be catched from the signal error().

User can select the calculation framerate with setCameraFPS when cameras are used as a source. When video files are used the framerate cannot be selected. The individual sources can be played and altered when the calculation is not running.

Author

Petri Partanen

6.2.2 Member Function Documentation

6.2.2.1 SourceID cavapa_gui::AnalysisController::addSource (const std::string & *path*, SourceID *id* = UNDEFINED_SOURCE, FrameTime *start_point* = 0)

Adds a custom video source.

Network cameras or video files need to be added with the method. The base recognizes automatically only hardware cameras.

Parameters

path	The video source path can be url or video file.
id	The desired ID number of the new source. This source ID must not exist. Use UNDEFINED_SOURCE to get a new unique ID.
start_point	Source start point will be used as the initial position of the source. All calls to setPosition() will use this as base. This is not used for sourceSeek() or other methods. It also has no effect on any other than video files.

Returns

ID number that was assigned to the new source or UNDEFINED_SOURCE if failed. Use getLastError() to retrieve reason for failure.

6.2.2.2 bool cavapa_gui::AnalysisController::addSource (const std::vector< std::string > & *paths*, SourceID *id* = UNDEFINED_SOURCE, FrameTime *start_point* = 0)

Adds a custom source with multiple files and a fixed ID number.

It can be used to play existing results from video files.

Parameters

paths	The ordered list of the video filenames.
id	The desired ID number of the new source. The source ID must not exist. Use UNDEFINED_SOURCE to get a new unique ID.

start_point	The source start point. Will be used as the initial position of the source. All calls to setPosition() will use this as base. This is not used for sourceSeek() or other methods. It also has no effect on any other than video files.

Returns

True if all files were successfully opened or false otherwise. Use getLastError() to retrieve reason for failure.

Remarks

All video files must match in resolution and framerate.

6.2.2.3 void cavapa_gui::AnalysisController::calculationComplete ( ) [signal]

Signals that calculation has ended.

It is sent for example when the video file source has reached its last frame. It could also be sent when the disk has ran out of space.

Remarks

The last signal of updated() or error() is sent just before this.

6.2.2.4 bool cavapa_gui::AnalysisController::canChangeSourceResolution (SourceID *id*) const

Checks if the source resolution can be changed.

Parameters

id	The source ID number.

Returns

True if the resolution can be changed or false otherwise.

Remarks

You can expect that most likely only hardware cameras will return true.

6.2.2.5 bool cavapa_gui::AnalysisController::canPause ( ) const

Indicates if the running calculation can be paused.

Returns

True if pause is possible or false otherwise.

Remarks

Most likely will be true only if all sources are video files.

6.2.2.6 int cavapa_gui::AnalysisController::countAvailableCameras ( ) const

Counts the available connected hardware cameras.

The method only recognizes USB and other cameras. No network streams or other sources are counted. This will not re-attempt to open already open HW-devices.

Returns

The total amount of hardware cameras that are currently initialized.

6.2.2.7 void cavapa_gui::AnalysisController::error (const std::string & *message,* ErrorLevel *level*) [signal]

Signals that something went wrong during the calculation.

It is used when there is no other way to indicate an error.

Parameters

message	The error message.
level	The error level.

Remarks

When an error happens during a call to a public method, they return failure value and that information can then be retrieved with getLastError(). Those errors do not raise this signal separately.

6.2.2.8 bool cavapa_gui::AnalysisController::exportResults (const ExportOptions & *options*) const [inline]

Exports the current Results to the CSV File.

Parameters

options	The parameters for export.

Returns

True if the export was successful or false otherwise.

6.2.2.9 double cavapa_gui::AnalysisController::getCalculationFPS ( ) const [inline]

Retrieves the calculation framerate.

It retrieves only the current running calculation framerate. It may not be what was given with setFPS() as it is adjusted to match the selected source framerates.

Returns

The current framerate or 0.0 if no calculation is running.

6.2.2.10 std::string cavapa_gui::AnalysisController::getLastError ( ) const [inline]

Retrieves the last error that occured.

Some of the class methods return failure values and this method can be used to retrieve more detailed information on what happened.

Returns

The error description.

6.2.2.11 std::vector<FrameStats> cavapa_gui::AnalysisController::getResults (FrameTime *start*, FrameTime *stop*, int *points*) const [inline]

Returns calculation results from a given period.

It can be used to return frame results from a certain period. Average values are calculated automatically from the returned period points.

Parameters

start	The period start time. If 0, the period will start from the first possible frame.
stop	The period end time. If 0, the period will end at the last possible frame.
points	The number of points to return from the specified period.

Returns

The calculation results from a given period. There will be [0, 'points'] number of FrameStats returned.

6.2.2.12 int cavapa_gui::AnalysisController::getSourceCount ( ) const [inline]

Retrieves the total count of sources.

It just sums up all the sources in the class, similarly as retrieving the source listing with getSourceListing() and checking its size.

Returns

The total available source count, including hardware cameras and other sources.

6.2.2.13 string cavapa_gui::AnalysisController::getSourceDescription (SourceID *id*) const

Returns the description of the source.

Parameters

id	The source ID number.

Returns

The camera name for the hardware cameras, the path for video files and the network streams.

Remarks

Similar hardware cameras will have the same description due to the model name. Some cameras may return just an empty string.

6.2.2.14 std::vector<cv::Rect> cavapa_gui::AnalysisController::getSourceExcludes (SourceID *id*) const

Returns the currently excluded rectangles from the source.

Parameters

id	The source ID number.

Returns

The list of rectangles that are currently drawn on the top of each frame before they are passed to the algorithm.

6.2.2.15 vector< string > cavapa_gui::AnalysisController::getSourceFilenames (SourceID *id*) const

Returns the list of the files associated with the source.

With the cameras these will be the names of the files that were recorded.

Parameters

id	The source ID number.

Returns

The list of the files.

6.2.2.16 vector< SourceID > cavapa_gui::AnalysisController::getSourceListing ( ) const

Returns a list of all sources in the class.

Returns

List of the source ID numbers.

6.2.2.17 FrameTime cavapa_gui::AnalysisController::getSourcePosition (SourceID *id*) const

Retrieves the current time position of a source.

This will be the current time on the video files. Cameras and other sources will always return 0.

Parameters

id	The source ID number.

Returns

FrameTime of the source's current position. 0 for cameras or sources that do not exist.

6.2.2.18 Size cavapa_gui::AnalysisController::getSourceResolution (SourceID *id*) const

Retrieves the source's frame resolution.

Parameters

id	The source ID number.

Returns

The frame size or (0, 0) if it is not found or error is encountered.

6.2.2.19 SourceStats cavapa_gui::AnalysisController::getSourceStats (SourceID *id*) const

Retrieves the source's current statistics.

The statistics can be used to get information on the missed, recorded etc. frames, and can provide insight on possible source lag or other problems. This is also the way to know how many frames a video file has.

Parameters

id	The source ID number.

Returns

The statistics of the source.

Remarks

Some frames may be buffered in which case they might not appear on the statistics.

6.2.2.20 SourceType cavapa_gui::AnalysisController::getSourceType (SourceID *id*) const

Returns the type of the selected source.

Parameters

id	The source ID number.

Returns

The type of the source.

6.2.2.21 FrameTime cavapa_gui::AnalysisController::getVideoLength (SourceID *id*) const

Retrieves the length of the video source.

Parameters

id	The source ID number.

Returns

The length of the video in milliseconds or 0 if failed.

6.2.2.22 int cavapa_gui::AnalysisController::initCameras (int *max_loops* = `10`)

Initializes the hardware cameras.

Parameters

max_loops	The maximum number of the devices to be checked for HW cameras. This should always be more than the actual number of cameras that are expected to be connected.

Returns

The number of new hardware cameras discovered.

Remarks

This is of course not necessary to call if the calculation is only going to consist of video files.

6.2.2.23 bool cavapa_gui::AnalysisController::isPaused ( ) const [inline]

Retrieves the current pause status.

Returns

True if paused or false otherwise.

6.2.2.24 bool cavapa_gui::AnalysisController::isRunning ( ) const [inline]

Retrieves the current status of the calculation.

Returns

True if calculation is running or false otherwise.

6.2.2.25 bool cavapa_gui::AnalysisController::load (const std::string & *path* = "results.cpa")

Loads the calculation results and recorded file information.

It loads existing results from a file. Once a result file has been opened, you cannot modify the sources. You can only play the recorded calculation and observe the results. You need to call reset() to be able to start a new calculation or add sources.

Parameters

path	The path of the results file.

Returns

True if load was successful or false otherwise. Use getLastError() to retrieve reason for failure.

6.2.2.26 bool cavapa_gui::AnalysisController::pause ( )

Pauses or unpauses the calculation.

Returns

True if pause/unpause was successful or false otherwise. Use getLastError() to retrieve reason for failure.

Remarks

Use canPause() to see if the calculation can be paused.

6.2.2.27 bool cavapa_gui::AnalysisController::removeSource (SourceID *id*)

Removes the source.

Parameters

id	The source ID number.

Returns

True if the source was removed or false otherwise. use getLastError() to retrieve reason for failure.

6.2.2.28 bool cavapa_gui::AnalysisController::reset ( )

Resets the calculation results.

After this call, all the saved results are entirely gone.

Returns

True if the reset was successful or false otherwise. Use getLastError() to retrieve reason for failure.

Remarks

The calculation must be stopped to reset.

6.2.2.29 bool cavapa_gui::AnalysisController::save (const std::string & *path* = "results.cpa")

Saves the calculation results and sightings.

It saves the current CAVAPA calculation into a file. It does not save recorded filenames or any information on the sources.

Parameters

path	The path of the results file.

Returns

True if save was successful or false otherwise. Use getLastError() to retrieve reason for failure.

6.2.2.30 bool cavapa_gui::AnalysisController::setCameraFPS (double *fps*)

Sets the default FPS for camera sources.

It will not have any effect if there are any cameras in the calculation that report a lower framerate than what has been set manually. Only HW cameras are able to report framerates. FPS can only be changed when calculation is not running.

Parameters

fps	The framerate to be set.

Returns

True if the framerate set was succesful or false otherwise. Use getLastError() to retrieve the reason for failure.

Remarks

It is important to understand that when setting the recording framerate for hardware cameras, it is MUCH better to have a lower framerate than a very high. This is due to the fact that if too many retrieval calls are send to the camera it starts to lag, and retrieval times become extremely high. 10 ms retrieval times can become 100 ms if the previous retrieval was not finished in time.

6.2.2.31 void cavapa_gui::AnalysisController::setHighlight (bool *highlight*) [inline]

Sets the individual sighting rectangle highlights.

This will affect all incoming source frames. The detected sightings will be highlighted with a rectangle.

Parameters

highlight	True if the sightings should be highlighted or false if they should not.

6.2.2.32 void cavapa_gui::AnalysisController::setHighlightColor (const cv::Scalar & *color*) [inline]

Sets the sighting's highlighting color.

The color is used in the rectangles that are drawn around detected individuals on source frames.

Parameters

color	The color in BGR value (*not* RGB!).

6.2.2.33 bool cavapa_gui::AnalysisController::setPosition (FrameTime *pos*)

Sets the current time position of all the sources.

The purpose of the method is to seek the calculation result point for every video source. Video sources will use their initialized start point as a base for the time seek. The position can only be changed when the calculation is not running.

Parameters

pos	Desired time position.

Returns

True if successful or false if any of the videos files failed to seek to the new position.

6.2.2.34 bool cavapa_gui::AnalysisController::setRecorderCodec (const std::string & *codec,* const std::string & *extension* = "avi")

Sets the default camera recorder codec.

It only affects new recordings, not the ones that are already running.

Parameters

codec	The 4 character code for the video (FourCC). See http://www.fourcc.org/codecs.php
extension	The file extension that is used for new recordings. It MUST match the codec, otherwise files will fail to load.

Returns

True if the codec was set or false otherwise.

Remarks

For linux and Windows compatibility it seems that "DIVX" is a safe codec to use. OS X support is unknown.

6.2.2.35 void cavapa_gui::AnalysisController::setRecordTimeLimit (unsigned int *length*) [inline]

Sets the maximum length of one recording.

Parameters

length	The length of the recording in seconds or 0 if no limit.

6.2.2.36 bool cavapa_gui::AnalysisController::setSourceBarrelCorrection (SourceID *id*, double *amount*)

Sets the barrel correction on a source.

It will add the given effect on all frames returned from the source from now on. These frames will also be passed on the calculation.

Parameters

id	The source ID number.
amount	The amount of barrel correction to be applied. 0.0 means no effect. Positive values increase the effect. Amount can't be negative.

Returns

True if the new barrel correction value was set or false otherwise. Use getLastError() to retrieve reason for failure.

Remarks

This can be changed during the calculation, but it is highly not recommended as it might produce lag due to multithreaded image retrievals.

6.2.2.37 void cavapa_gui::AnalysisController::setSourceExcludes (SourceID *id*, const std::vector< cv::Rect > & *rects*)

Sets the excluded rectangles for a source.

The given rectangles will be drawn black on all frames that the source produces. These are used to exclude specific areas from the source frame. The rectangles are always drawn on the frame before analysing them, but after they are written to a file.

Parameters

id	The source ID number.
rects	The rectangles to be excluded.

6.2.2.38 bool cavapa_gui::AnalysisController::setSourceResolution (SourceID *id*, const cv::Size & *new_size*)

Sets the source resolution.

Parameters

id	The source ID number.
new_size	The desired size.

Returns

True if the new size was set, false otherwise. Use getLastError() to retrieve reason for failure.

Remarks

Use canChangeSourceResolution() to check if a change is even possible. Mainly hardware cameras can only have their resolution changed.

6.2.2.39 bool cavapa_gui::AnalysisController::setVideoSpeed (double *ratio* = `1.0`)

Sets the video source speed.

If all the sources are video files this will multiply the framerate, otherwise this has no effect.

Parameters

ratio	The desired video speed ratio. Negative or 0.0 will indicate as fast as possible.

Returns

True if the speed was changed successfully or false otherwise. Use getLastError() to get more information on failure.

6.2.2.40 void cavapa_gui::AnalysisController::setWorkDirectory (const std::string & *path*)

Sets the path for the calculation directory.

The directory will be used to store recordings, calculation results etc.

Parameters

path	The directory path.

6.2.2.41 bool cavapa_gui::AnalysisController::sourceCanRecord (SourceID *id*) const

Checks if the source is capable of recording.

Parameters

id	The source ID number.

Returns

True if the recording is supported or false otherwise.

6.2.2.42 bool cavapa_gui::AnalysisController::sourcePlay (SourceID *id*)

Starts playing a video source.

Parameters

id	The source ID number.

Returns

True if successfull or false otherwise. Use getLastError() to retrieve failure message.

6.2.2.43 bool cavapa_gui::AnalysisController::sourceSeek (SourceID *id*, FrameTime *position*)

Seeks the video source to a specific time.

Parameters

id	The source ID number.
position	The time position of the video.

Returns

True if seeking was successul or false otherwise. Use getLastError() to retrieve failure message.

Remarks

Video can't be seeked during the calculation.

6.2.2.44 bool cavapa_gui::AnalysisController::sourceStepBackward (SourceID *id*) [inline]

Steps the source one frame back.

Parameters

id	Source ID number.

Returns

True if the step was successful or false otherwise. Use getLastError() to retrieve failure message.

Remarks

The video files can be stepped only while the calculation is not running or the video is not playing.

6.2.2.45 bool cavapa_gui::AnalysisController::sourceStepForward (SourceID *id*) [inline]

Steps the source one frame forward.

Parameters

id	The source ID number.

Returns

True if the step was successful or false otherwise. Use getLastError() to retrieve failure message.

Remarks

The video files can be stepped only while the calculation is not running or the video is not playing.

6.2.2.46 bool cavapa_gui::AnalysisController::sourceStop (SourceID *id*)

Stops playing the video source.

Parameters

id	The source ID number.

Returns

True if successfull or false otherwise. Use getLastError() to retrieve failure message.

6.2.2.47 bool cavapa_gui::AnalysisController::start (const std::vector< SourceID > & *calc_sources*, const std::vector< cavapa::CameraSettings > & *settings*, const std::vector< cavapa::CalibrationPoint > & *points*, bool *record* = `false`, int *calc_time* = `0`)

Starts the CAVAPA calculation.

Parameters

calc_sources	The sources to be included in the calculation.
settings	The source settings. The resolution and the barrel values have no effect as AnalysisController will fill them in automatically.
points	Calibration points.
record	If the video sources should be recorded or not.
calc_time	Maximum runtime in seconds. Infinite if 0.

Returns

True if calculation was started or false otherwise. Use getLastError() to retrieve reason for failure.

6.2.2.48 void cavapa_gui::AnalysisController::stop ( )

Stops the current CAVAPA calculation.

Remarks

It does not destroy the results.

6.2.2.49 void cavapa_gui::AnalysisController::updated (const std::vector< FrameCapture > & *frames*, const FrameStats & *statistics*) `[signal]`

It signals frame updates or calculation updates.

When the calculation is in progress, only the sources selected to the calculation will perform frame updates. When the calculation is not in progress, all the sources will return the current frame.

Parameters

frames	The current frames of the video sources.
statistics	Holds the frame statistics.

Remarks

The frame statistics will be empty when the calculation is not running.

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

- analysiscontroller.h
- analysiscontroller.cpp

6.3 cavapa_gui::CalibrationPoints Class Reference

CalibrationPoints stores a collection of calibration points that can be drawn on a QPainter.

`#include <calibrationpoints.h>`

Public Member Functions

- void push_back (QPointF newPoint)

  Adds a new calibration point.
- void draw (QPainter &painter) const

  Draws all the points on a QPainter.
- unsigned int size () const

  Returns the number of control points in CalibrationPoints.
- void setHighlightedPoint (const int i)

  Sets the highlighted calibration point.
- QPointF at (const int pos)

  Returns the calibration point at specified location.
- QPointF & operator[] (const int pos)

  Returns the calibration point at specified location.
- void clear ()

  Removes all the calibration points.
- void removeHighlighted ()

  Removes the highlighted calibration point.

Static Public Member Functions

- static QColor getComplementaryColor (const QColor color)

  Returns the complementary color of the specified color.

6.3.1 Detailed Description

CalibrationPoints stores a collection of calibration points that can be drawn on a QPainter.

A point can be highlighted and the highlighted point or all points can be removed.

The points are stored as relative values so that the coordinates are values between 0 and 1.

Author

Oskari Leppäaho

6.3.2 Member Function Documentation

6.3.2.1 QPointF cavapa_gui::CalibrationPoints::at (const int *pos*) [inline]

Returns the calibration point at specified location.

Parameters

pos	Position of the calibration point to return.

Returns

The calibration point at specified location.

6.3.2.2 void cavapa_gui::CalibrationPoints::draw (QPainter & *painter*) const

Draws all the points on a QPainter.

Draws the points as filled circles and the index (+1) of the point next to the point

Parameters

painter	The QPainter to draw on.

6.3.2.3 QColor cavapa_gui::CalibrationPoints::getComplementaryColor (const QColor *color*) [static]

Returns the complementary color of the specified color.

Parameters

color	The color whose complementary color is to be returned.

Returns

The complementary color.

6.3.2.4 QPointF& cavapa_gui::CalibrationPoints::operator[] (const int *pos*) [inline]

Returns the calibration point at specified location.

Parameters

pos	Position of the calibration point to return.

Returns

The calibration point at specified location.

6.3.2.5 void cavapa_gui::CalibrationPoints::push_back (QPointF *newPoint*)

Adds a new calibration point.

Parameters

newPoint	The new point. The values of the coordinates should be between 0 and 1.

6.3.2.6 void cavapa_gui::CalibrationPoints::removeHighlighted ( )

Removes the highlighted calibration point.

Points with an index bigger than the highlighted index will have their index reduced by one. The point next to the removed point will be the new highlighted point. If the last calibration point was removed, the point before that will be the new highlighted point.

6.3.2.7 void cavapa_gui::CalibrationPoints::setHighlightedPoint (const int *i*) [inline]

Sets the highlighted calibration point.

The highlighted calibration point will be drawn with the complementary color of the current drawing color.

Parameters

i	The index of the point that will be highlighted.

6.3.2.8 unsigned int cavapa_gui::CalibrationPoints::size ( ) const [inline]

Returns the number of control points in CalibrationPoints.

Returns

Number of control points.

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

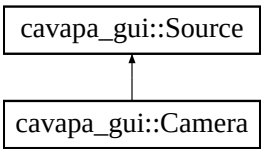
- gui/calibrationpoints.h
- gui/calibrationpoints.cpp

6.4 cavapa_gui::Camera Class Reference

The class for a camera source.

`#include <camera.h>`

Inheritance diagram for cavapa_gui::Camera:



Public Member Functions

- Camera (SourceID desired_id=UNDEFINED_SOURCE)
  Creates a new source.
- DISALLOW_COPY_AND_ASSIGN (Camera)
  Copy and assign of the class is not allowed.
- bool canRecord () const override
  Returns information on the source recording abilities.
- FrameCapture get (FrameTime passed_time) override
  Retrieves the next frame from the camera.
- double getBarrelCorrection () override
  Returns the barrel correction applied to the source frames.
- std::string getDescription () const override
  Retrieves the description of the source.
- int getDeviceID () const
  Retrieves the device number used to initialize the device.
- double getFramerate () const override

Retrieves the framerate.

- FrameTime getPosition () const overridefinal

  Retrieves the current time position of the source.

- std::vector< std::string > getRecordings () const

  Returns the names of the video files that were recorded.

- cv::Size getResolution () const override

  Retrieves the resolution of the source.

- SourceType getSourceType () const override

  Returns the type of the source.

- SourceStats getStats () override

  Retrieves the source statistical information.

- bool hasNew () override

  Checks whether the device has a new image to be retrieved or not.

- bool open (int index)

  Opens a new camera object.

- bool open (const std::string &url)

  Opens a new camera object.

- bool isOpen () const override

  Checks if the source has been initialized.

- bool isPlaying () const override

  Checks whether the source is playing or not.

- bool record (const std::string &filename, const std::string &codec="") override

  Starts the recording of the source to the file.

- void setBarrelCorrection (double amount) override

  Sets the new barrel correction value.

- bool setResolution (const cv::Size &new_size) override

  Sets the source resolution.

- void stop () override

  Stops the camera from recording.

Static Public Member Functions

- static std::string getCodec ()

  Retrieves the current codec used.

- static double getDefaultFPS ()

  Retrieves the default framerate that has been set.

- static bool setCodec (const std::string &codec)

  Sets recorders codec for new videos.

- static void setDefaultFPS (double fps)

  Sets the cameras default framerate.

Additional Inherited Members

6.4.1 Detailed Description

The class for a camera source.

The class is used to retrieve frames from HW cameras and network streams. The class provides a Recorder that can save frames to disk into video file.

HW camera frame retrievals take about 2ms. Recording the frame to a file at the same time takes about 3ms per camera. However, if the FPS retrieval is too high for that particular camera, it starts lag and can take up to 30-90 ms per image retrieval. This is solved in the OpenCVCapture by multithreading the retrieval process.

By multithreading the retrieval process, the programs base application can be sure that it get's either a new image from the camera or a duplicate of the last image it already showed. Showing the last image twice in a row is not a problem at all, because that's exactly what happens when you cut your frame rate in half, you just see the same image a little bit longer.

Author

Petri Partanen

6.4.2 Constructor & Destructor Documentation

6.4.2.1 cavapa_gui::Camera::Camera (SourceID *desired_id* = UNDEFINED_SOURCE) [inline]

Creates a new source.

Parameters

desired_id	The desired ID number. Use UNDEFINED_SOURCE if you do not want to define it to a specific ID. This does not check if the source with the given ID already exists. The ID number will not be given to another new sources with UNDEFINED_SOURCE ID number.

6.4.3 Member Function Documentation

6.4.3.1 bool cavapa_gui::Camera::canRecord ( ) const [inline], [override], [virtual]

Returns information on the source recording abilities.

Returns

True if the source can record, false otherwise.

Implements cavapa_gui::Source.

6.4.3.2 FrameCapture cavapa_gui::Camera::get (FrameTime *passed_time*) [override], [virtual]

Retrieves the next frame from the camera.

The time supplied to the function informs the source about how much time has passed since the last image retrieval. The source will depend on its internal clock, either retrieving a new image or returning the buffered image that it retrieved previously.

Parameters

passed_time	The time passed since the last call (in milliseconds).

Returns

The latest frame capture. This will actually return a reference to the original image, so you are NOT allowed to edit it! Otherwise the source buffer will be altered too. This is just the way OpenCV handles Mat memory.

Remarks

Forcing a camera to not skip frames will produce lag. Also, please note that stereo cameras are not (yet) supported. The first capture index (0) is retrieved from the OpenCV capturing device. Several buffers will need to be implemented for multichannel support.

TODO: Handle error flag!

Implements cavapa_gui::Source.

6.4.3.3 double cavapa_gui::Camera::getBarrelCorrection ( ) [inline], [override], [virtual]

Returns the barrel correction applied to the source frames.

Returns

The barrel effect value.

Implements cavapa_gui::Source.

6.4.3.4 static std::string cavapa_gui::Camera::getCodec ( ) [inline], [static]

Retrieves the current codec used.

Returns

The codec name.

Remarks

The same codec is used for all recordings. Individual cameras cannot have unique codecs.

6.4.3.5 static double cavapa_gui::Camera::getDefaultFPS ( ) [inline], [static]

Retrieves the default framerate that has been set.

The framerate used for the cameras do not give any information on their current framerate capabilities.

Returns

The default framerate.

6.4.3.6 std::string cavapa_gui::Camera::getDescription ( ) const [inline], [override], [virtual]

Retrieves the description of the source.

Returns

The source description.

Remarks

You cannot trust it 100%. OpenCV does not provide any API to recognize camera hardware. See: https://code.ros.org/trac/opencv/ticket/935 Qt is used get the camera description. It is however unclear if the string returned by QCamera does match with the actual device used by OpenCV.

Implements cavapa_gui::Source.

6.4.3.7 int cavapa_gui::Camera::getDeviceID ( ) const [inline]

Retrieves the device number used to initialize the device.

Returns

Device ID-number or UNKNOWN_DEVICE if not available.

6.4.3.8 double cavapa_gui::Camera::getFramerate ( ) const [override], [virtual]

Retrieves the framerate.

Returns

The framerate or 0.0 if not supported.

Implements cavapa_gui::Source.

6.4.3.9 FrameTime cavapa_gui::Camera::getPosition ( ) const [inline], [final], [override], [virtual]

Retrieves the current time position of the source.

Returns

The source time position.
0. Cameras do not support this method.

Implements cavapa_gui::Source.

6.4.3.10 std::vector< string > cavapa_gui::Camera::getRecordings ( ) const

Returns the names of the video files that were recorded.

Returns

The list of recorded files.

6.4.3.11 cv::Size cavapa_gui::Camera::getResolution ( ) const [inline], [override], [virtual]

Retrieves the resolution of the source.

Returns

The source resolution.

Implements cavapa_gui::Source.

6.4.3.12 SourceType cavapa_gui::Camera::getSourceType ( ) const [override], [virtual]

Returns the type of the source.

Returns

The source type.

Reimplemented from cavapa_gui::Source.

6.4.3.13 SourceStats cavapa_gui::Camera::getStats ( ) [override], [virtual]

Retrieves the source statistical information.

Returns

The source statistics.

Implements cavapa_gui::Source.

6.4.3.14 bool cavapa_gui::Camera::hasNew ( ) [inline], [override], [virtual]

Checks whether the device has a new image to be retrieved or not.

This checks if the FrameTime on the buffered image differs from what was retrieved on the last get().

Returns

True if a new image available or false otherwise.
True. Cameras always report new images despite of what is in the buffer.

Implements cavapa_gui::Source.

6.4.3.15 bool cavapa_gui::Camera::isOpen ( ) const [inline], [override], [virtual]

Checks if the source has been initialized.

Returns

True if initialized, false otherwise.

Implements cavapa_gui::Source.

6.4.3.16 bool cavapa_gui::Camera::isPlaying ( ) const [inline], [override], [virtual]

Checks whether the source is playing or not.

Returns

True if the source is playing or false otherwise.
True. Cameras are always playing.

Implements cavapa_gui::Source.

6.4.3.17 bool cavapa_gui::Camera::open (int *index*)

Opens a new camera object.

Parameters

index	The device index number.

Returns

True if opened succesfully or false otherwise.

6.4.3.18 bool cavapa_gui::Camera::open (const std::string & *url*)

Opens a new camera object.

Parameters

url	The device url.

Returns

True if opened succesfully or false otherwise.

6.4.3.19 bool cavapa_gui::Camera::record (const std::string & *filename*, const std::string & *codec* = " ") [override], [virtual]

Starts the recording of the source to the file.

This can also be used to start an entirely new video file during the recording. The recording file will change immediately.

Parameters

filename	The path of the recording file.
codec	The codec to be used for recording.

Returns

True if the recording started or false otherwise.

Implements cavapa_gui::Source.

6.4.3.20 void cavapa_gui::Camera::setBarrelCorrection (double *amount*) [override], [virtual]

Sets the new barrel correction value.

The value is used to correct barrel effect (lens correction) on each image that the source will return.

Parameters

amount	The barrel value.

Implements cavapa_gui::Source.

6.4.3.21 bool cavapa_gui::Camera::setCodec (const std::string & *codec*) [static]

Sets recorders codec for new videos.

See the list of codecs at http://www.fourcc.org/codecs.php

Parameters

codec	The new Codec.

Returns

True if the new codec was set or false otherwise.

Remarks

The same codec is used globally for all camera recordings. You cannot set individual codecs.

6.4.3.22 static void cavapa_gui::Camera::setDefaultFPS (double *fps*) [inline], [static]

Sets the cameras default framerate.

This is applied to the cameras that do not give out any other framerate values. This will affect most USB cameras for example.

Parameters

fps	The new framerate to be set.

6.4.3.23 bool cavapa_gui::Camera::setResolution (const cv::Size & *new_size*) [override], [virtual]

Sets the source resolution.

Parameters

new_size	The new resolution for the source.

Returns

True if the resolution was set, false otherwise.

Implements cavapa_gui::Source.

6.4.3.24 void cavapa_gui::Camera::stop ( ) [override], [virtual]

Stops the camera from recording.

This stops the recorder and let's it write the rest of the buffer to the file.

Implements cavapa_gui::Source.

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

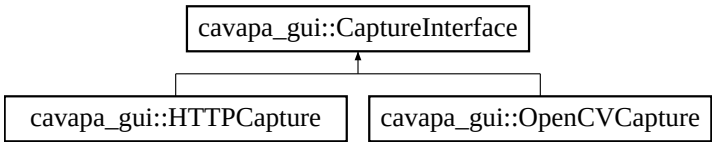
- source/camera.h
- source/camera.cpp

6.5 cavapa_gui::CaptureInterface Class Reference

Interface for the capture devices.

`#include <captureinterface.h>`

Inheritance diagram for cavapa_gui::CaptureInterface:



Public Types

- enum INTERFACE_TYPE { **HTTP**, **OPENCV** }

  Used to define the type of source interface.

Public Member Functions

- DISALLOW_COPY_AND_ASSIGN (CaptureInterface)

  Copy and assign of the class is not allowed.
- virtual double getBarrelCorrection ()=0

Retrieves the barrel correction applied to the source.

- virtual FrameCapture getBuffered ()=0

  Retrieves the currently buffered frame.
- virtual std::string getDescription () const =0

  Retrieves the description of the device.
- virtual int getDeviceID () const =0

  Returns the ID number the device was initialized with.
- virtual double getFramerate () const =0

  Retrieves the capture device framerate.
- virtual int getMissedFrames () const =0

  Retrieves the logged frame misses.
- virtual FrameCapture getNext (bool skip, bool *error)=0

  Retrieves the next frame.
- virtual std::string getPath () const =0

  Retrieves the path given during the initialization.
- virtual cv::Size getResolution () const =0

  Retrieves the capture device resolution.
- virtual int getRetrievedFrames () const =0

  Retrieves the total logged frames.
- virtual INTERFACE_TYPE getType () const =0

  Retrieves the type of the device.
- virtual bool hasNew ()=0

  Checks whether the device has a new image to retrieve or not.
- virtual bool isOpen () const =0

  Checks whether the capturing device is open or not.
- virtual bool open (const std::string &path)=0

  Opens the video file or the stream.
- virtual void resetStats ()=0

  Resets the statistical counters.
- virtual void setBarrelCorrection (double amount)=0

  Sets the amount of the barrel correction to be applied.
- virtual bool setResolution (const cv::Size &new_size)=0

  Sets the capturing device resolution.

6.5.1 Detailed Description

Interface for the capture devices.

The inherits from this class are used by the actual video frame sources. The class represents the core functionality of device image retrieval.

The key functions of this abstract class are getNext() and open().

Author

Petri Partanen

6.5.2 Member Function Documentation

6.5.2.1 virtual double cavapa_gui::CaptureInterface::getBarrelCorrection ( ) [pure virtual]

Retrieves the barrel correction applied to the source.

Returns

The amount of the barrel correction.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.2 virtual FrameCapture cavapa_gui::CaptureInterface::getBuffered ( ) [pure virtual]

Retrieves the currently buffered frame.

Returns

The buffered frame.

Remarks

This does not signal for a new image fetch from the device.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.3 virtual std::string cavapa_gui::CaptureInterface::getDescription ( ) const [pure virtual]

Retrieves the description of the device.

Returns

For the devices initialized with HW index, this will be the name of the HW device. For other devices this is the path given during initialization.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.4 virtual int cavapa_gui::CaptureInterface::getDeviceID ( ) const [pure virtual]

Returns the ID number the device was initialized with.

Returns

The device ID used in the initialization or UNKNOWN_DEVICE if not specified.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.5 virtual double cavapa_gui::CaptureInterface::getFramerate ( ) const [pure virtual]

Retrieves the capture device framerate.

Returns

The current framerate.

Implemented in cavapa_gui::HTTPCapture, and cavapa_gui::OpenCVCapture.

6.5.2.6 virtual int cavapa_gui::CaptureInterface::getMissedFrames ( ) const [pure virtual]

Retrieves the logged frame misses.

A frame miss occurs everytime when getNext() is called and the previous retrieval was still ongoing.

Returns

The count of the missed frames.

Implemented in cavapa_gui::HTTPCapture, and cavapa_gui::OpenCVCapture.

6.5.2.7 virtual FrameCapture cavapa_gui::CaptureInterface::getNext (bool *skip*, bool * *error*) [pure virtual]

Retrieves the next frame.

Basically it returns the currently buffered frame and signals the device to grab a new image for next call to the function. If the device has not returned from previous image grab the call is recorded as a missed frame.

Parameters

skip	Specifies if the frame can be skipped. When set to true, this will force the main thread to wait for a new image from the device. This can result in a huge lag if the device is not capable of retrieving images fast enough.
error	Will receive an error message if something went wrong.

Returns

The next frame with the reference to the image! You must make a clone if you wish to alter it!

Implemented in cavapa_gui::HTTPCapture, and cavapa_gui::OpenCVCapture.

6.5.2.8 virtual std::string cavapa_gui::CaptureInterface::getPath ( ) const [pure virtual]

Retrieves the path given during the initialization.

Returns

The path of the device or an empty string if not found.

Implemented in cavapa_gui::HTTPCapture, and cavapa_gui::OpenCVCapture.

6.5.2.9 virtual cv::Size cavapa_gui::CaptureInterface::getResolution ( ) const [pure virtual]

Retrieves the capture device resolution.

Returns

The resolution of the device.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.10 virtual int cavapa_gui::CaptureInterface::getRetrievedFrames ( ) const [pure virtual]

Retrieves the total logged frames.

Basically, this is a counter on how many times getNext() has been called.

Returns

The count of the total frames returned.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.11 virtual INTERFACE_TYPE cavapa_gui::CaptureInterface::getType ( ) const [pure virtual]

Retrieves the type of the device.

Returns

The interface type.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.12 virtual bool cavapa_gui::CaptureInterface::hasNew ( ) [pure virtual]

Checks whether the device has a new image to retrieve or not.

This checks if the FrameTime on the buffered image differs from what was retrieved on the last getNext().

Returns

True if new image or false otherwise.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.13 virtual bool cavapa_gui::CaptureInterface::isOpen ( ) const [pure virtual]

Checks whether the capturing device is open or not.

Returns

True if the device is open or false otherwise.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.14 virtual bool cavapa_gui::CaptureInterface::open (const std::string & *path*) [pure virtual]

Opens the video file or the stream.

Parameters

path	The path of the video file or the network stream.

Returns

True if it was opened successfully or false otherwise.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.15 virtual void cavapa_gui::CaptureInterface::resetStats ( ) [pure virtual]

Resets the statistical counters.

This affects the missed and total frame counters.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.16 virtual void cavapa_gui::CaptureInterface::setBarrelCorrection (double *amount*) [pure virtual]

Sets the amount of the barrel correction to be applied.

Parameters

amount	The barrel effect. 0.0 means none.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

6.5.2.17 virtual bool cavapa_gui::CaptureInterface::setResolution (const cv::Size & *new_size*) [pure virtual]

Sets the capturing device resolution.

Parameters

new_size	The size to be set.

Returns

True if at least one of the dimension was set or false otherwise.

Implemented in cavapa_gui::OpenCVCapture, and cavapa_gui::HTTPCapture.

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

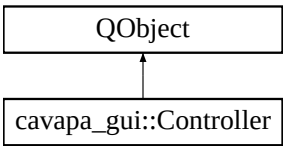
- source/captureinterface.h

6.6 cavapa_gui::Controller Class Reference

The Controller class handles the messages between UserInterface and AnalysisController.

`#include <controller.h>`

Inheritance diagram for cavapa_gui::Controller:



Public Slots

- void aboutToQuitApp ()

  Warns the object for application termination.
- void run ()

  Launches all the registered interfaces.

Signals

- void finished ()

  Emitted when all the interfaces have been closed.

Public Member Functions

- Controller ()

  Constructs the Controller object.

- void addInterface (UserInterface *cInterface)

  Adds a new interface to the controller.
- void startInterfaces ()

  Starts all the user interfaces.

6.6.1 Detailed Description

The Controller class handles the messages between UserInterface and AnalysisController.

It also handles application settings.

Author

Mika Lehtinen

6.6.2 Member Function Documentation

6.6.2.1 void cavapa_gui::Controller::aboutToQuitApp ( ) [slot]

Warns the object for application termination.

The purpose of the method is to shut off cameras and other hooks so they have time to terminate.

6.6.2.2 void cavapa_gui::Controller::addInterface (UserInterface * *cInterface*)

Adds a new interface to the controller.

Parameters

cInterface	The interface to be added.

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

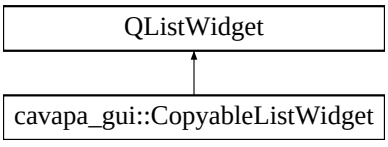
- controller.h
- controller.cpp

6.7 cavapa_gui::CopyableListWidget Class Reference

A custom list widget class that allows items to be copied to clipboard.

`#include <copyablelistwidget.h>`

Inheritance diagram for cavapa_gui::CopyableListWidget:



Public Member Functions

- CopyableListWidget (QWidget *parent=0)

  Constructs a new CopyableListWidget having the specified parent.

Protected Member Functions

- virtual void keyPressEvent (QKeyEvent *event)

  Reimplemented for handling a key press event.
- virtual void contextMenuEvent (QContextMenuEvent *event)

  Reimplemented for handling a context menu event.

6.7.1 Detailed Description

A custom list widget class that allows items to be copied to clipboard.

Author

Mika Lehtinen

6.7.2 Constructor & Destructor Documentation

6.7.2.1 cavapa_gui::CopyableListWidget::CopyableListWidget (QWidget * *parent* = 0)

Constructs a new CopyableListWidget having the specified parent.

Parameters

parent	The parent widget.

6.7.3 Member Function Documentation

6.7.3.1 void cavapa_gui::CopyableListWidget::contextMenuEvent (QContextMenuEvent * *event*) [protected], [virtual]

Reimplemented for handling a context menu event.

Parameters

event	The context menu event object.

6.7.3.2 void cavapa_gui::CopyableListWidget::keyPressEvent (QKeyEvent * *event*) [protected], [virtual]

Reimplemented for handling a key press event.

Parameters

event	The key event object.

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

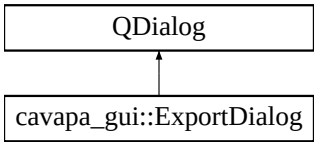
- gui/copyablelistwidget.h
- gui/copyablelistwidget.cpp

6.8 cavapa_gui::ExportDialog Class Reference

Class for export dialog of graphwidget.

`#include <exportdialog.h>`

Inheritance diagram for cavapa_gui::ExportDialog:



Signals

- void sendTimeStart (QDateTime start)

  Send edited time selection start point to the parent.
- void sendTimeEnd (QDateTime end)

  Send edited time selection end point to the parent.
- bool exportCommand (const ExportOptions &options)

  Passes export command to parent.

Public Member Functions

- ExportDialog (QWidget ∗parent=0)

  Constructs exporting dialog for the graph.
- void setTimeEdits (QDateTime start, QDateTime end)

  Sets export dialog's time points (start and end)
- void setMarkers (const std::vector< GraphMarker > markersfrom)

  Sets marker data into export dialog's comboboxes.
- void setBeginTimeFromMarker (int i)

  Sets the begin time of exporting according to a selected marker from the combobox.
- void setEndTimeFromMarker (int i)

  Sets the end time of exporting according to a selected marker from combobox.
- void setBeginDateTime (QDateTime time)

  Gets the starting time of the calculation from the parent that is used in marker position to QDateTime conversion.
- void setBeginFrameTime (FrameTime frame)

  Gets the very first frame of the calculation from the parent that is used in marker to QDateTime conversion.
- void setRelativeTime (bool relative)

  Sets time mode between relative and absolute.

6.8.1 Detailed Description

Class for export dialog of graphwidget.

Offers options for export types, file path and region selection from markers of graph

Author

Joel Kivelä

6.8.2 Constructor & Destructor Documentation

6.8.2.1 cavapa_gui::ExportDialog::ExportDialog (QWidget ∗ *parent* = 0) [explicit]

Constructs exporting dialog for the graph.

Parameters

parent	QWidget default parent.

6.8.3 Member Function Documentation

6.8.3.1 bool cavapa_gui::ExportDialog::exportCommand (const ExportOptions & *options*) [signal]

Passes export command to parent.

Parameters

options	Export Options.

Returns

Success or fail.

6.8.3.2 void cavapa_gui::ExportDialog::sendTimeEnd (QDateTime *end*) [signal]

Send edited time selection end point to the parent.

Parameters

end	End time point as QDateTime.

6.8.3.3 void cavapa_gui::ExportDialog::sendTimeStart (QDateTime *start*) [signal]

Send edited time selection start point to the parent.

Parameters

start	Start time point as QDateTime.

6.8.3.4 void cavapa_gui::ExportDialog::setBeginDateTime (QDateTime *time*)

Gets the starting time of the calculation from the parent that is used in marker position to QDateTime conversion.

Parameters

time	Starting time as QDateTime.

6.8.3.5 void cavapa_gui::ExportDialog::setBeginFrameTime (FrameTime *frame*)

Gets the very first frame of the calculation from the parent that is used in marker to QDateTime conversion.

Parameters

frame	The first frame.

6.8.3.6 void cavapa_gui::ExportDialog::setBeginTimeFromMarker (int *i*)

Sets the begin time of exporting according to a selected marker from the combobox.

Parameters

i	Index of marker in vector array.

6.8.3.7 void cavapa_gui::ExportDialog::setEndTimeFromMarker (int *i*)

Sets the end time of exporting according to a selected marker from combobox.

Parameters

i	Index of marker in vector array.

6.8.3.8 void cavapa_gui::ExportDialog::setMarkers (const std::vector< GraphMarker > *markersfrom*)

Sets marker data into export dialog's comboboxes.

Parameters

markersfrom	Markers as a vector array of GraphMarkers.

6.8.3.9 void cavapa_gui::ExportDialog::setRelativeTime (bool *relative*)

Sets time mode between relative and absolute.

Parameters

relative	True if relative time, else absolute.

6.8.3.10 void cavapa_gui::ExportDialog::setTimeEdits (QDateTime *start*, QDateTime *end*)

Sets export dialog's time points (start and end)

Parameters

start	Starting time as QDateTime.
end	Ending time as QDateTime.

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

- gui/exportdialog.h
- gui/exportdialog.cpp

6.9 cavapa_gui::ExportOptions Struct Reference

The structure holds CSV export options.

`#include <common.h>`

Public Attributes

- std::string filename

  The export filename with full path.
- EXPORT_FLAGS flags

  The flags for export.

- FrameTime interval

  The interval to be averaged in milliseconds.
- std::vector< Marker > markers

  The markers for the export.
- FrameTime start

  The period's start point.
- FrameTime stop

  The period's stop point.
- int time_offset

  The offset change for each marker time in seconds.

6.9.1 Detailed Description

The structure holds CSV export options.

Author

Petri Partanen

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

- common.h

6.10 cavapa_gui::FrameCapture Struct Reference

The structure is used for storing a single source frame.

`#include <common.h>`

Public Attributes

- SourceID id

  The source ID number.
- FrameTime time

  The time of the frame.
- cv::Mat image

  The actual frame image.

6.10.1 Detailed Description

The structure is used for storing a single source frame.

Author

Petri Partanen

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

- common.h

6.11 cavapa_gui::FrameStats Struct Reference

Structure for holding frame statistics on activity.

`#include <common.h>`

Public Attributes

- FrameTime time

  The calculation time of the frame.
- float activity

  The total activity on the frame.
- float count

  The number of detected sightings on the frame.

6.11.1 Detailed Description

Structure for holding frame statistics on activity.

This is the main statistical information that is passed from the AnalysisController to the Controller when calculation updates.

Author

Petri Partanen

Remarks

All members of the structure must be defined as fixed size in order to maintain compatibility between 32bit and 64bit systems as the structure is written to result files as it is.

6.11.2 Member Data Documentation

6.11.2.1 float cavapa_gui::FrameStats::activity

The total activity on the frame.

Remarks

Smaller precision is preferred for this member to provide smaller footprint on the file data.

6.11.2.2 FrameTime cavapa_gui::FrameStats::time

The calculation time of the frame.

This will always be relative to the start of calculation.

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

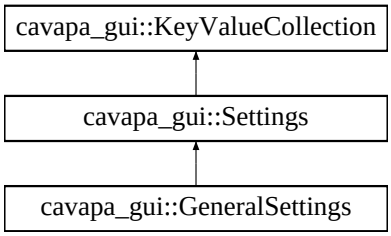
- common.h

6.12 cavapa_gui::GeneralSettings Class Reference

Represents the general settings of the application.

`#include <generalsettings.h>`

Inheritance diagram for cavapa_gui::GeneralSettings:



Public Member Functions

- GeneralSettings ()

  Constructs a new GeneralSettings object with default values.
- GraphSettings getGraphSettings () const

  Gets the GraphSettings object based on the current key values.
- void setGraphSettings (const GraphSettings &settings)

  Sets the keys related to the graph settings.

Additional Inherited Members

6.12.1 Detailed Description

Represents the general settings of the application.

Author

Mika Lehtinen

6.12.2 Member Function Documentation

6.12.2.1 GraphSettings cavapa_gui::GeneralSettings::getGraphSettings ( ) const

Gets the GraphSettings object based on the current key values.

Returns

The GraphSettings object.

6.12.2.2 void cavapa_gui::GeneralSettings::setGraphSettings (const GraphSettings & *settings*)

Sets the keys related to the graph settings.

Parameters

settings	The graph settings as a GraphSettings object.

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

- generalsettings.h
- generalsettings.cpp

6.13 cavapa_gui::GraphRegion Struct Reference

Struct for one region of the graph.

`#include <activitygraph.h>`

Public Attributes

- FrameTime start

  The start point of the region.
- FrameTime end

  The end point of the region.

6.13.1 Detailed Description

Struct for one region of the graph.

Author

Joel Kivelä

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

- gui/activitygraph.h

6.14 cavapa_gui::GraphSettings Struct Reference

Graph settings type for font/colors/line widths.

`#include <activitygraph.h>`

Public Attributes

- QFont font

  The graph text font as QFont, default = "Arial".
- QBrush background

  The graph background color as QBrush, default = Qt::black.
- QBrush selection

  The graph selection color as QBrush, default = Qt::darkRed.
- QPen activity_curve

  The activity curve style as QPen, default = QPen(Qt::magenta, 1, Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin); Change only the first two parameters (color and width).
- QPen count_curve

The count curve style as QPen, default = QPen(Qt::cyan, 1, Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin); Change only the first two parameters (color and width).

- QPen def_pen

  The default pen style for lines, etc.

- QPen text_selection

  The default pen for text selection, etc.

- QPen half_line

  The default pen for horizontal half lines, etc.

- int linewidth

  The line width for curves.

6.14.1 Detailed Description

Graph settings type for font/colors/line widths.

Author

Joel Kivelä

6.14.2 Member Data Documentation

6.14.2.1 QPen cavapa_gui::GraphSettings::def_pen

The default pen style for lines, etc.

as QPen, default = QPen(Qt::white, 1, Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin); Change only the first two parameters (color and width).

6.14.2.2 QPen cavapa_gui::GraphSettings::half_line

The default pen for horizontal half lines, etc.

as QPen, default = QPen(Qt::white, 1, Qt::DashLine, Qt::RoundCap, Qt::RoundJoin); Change only the first two parameters (color and width).

6.14.2.3 QPen cavapa_gui::GraphSettings::text_selection

The default pen for text selection, etc.

as QPen, default = QPen(Qt::cyan, 1, Qt::SolidLine, Qt::RoundCap, Qt::RoundJoin); Change only the first two parameters (color and width).

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

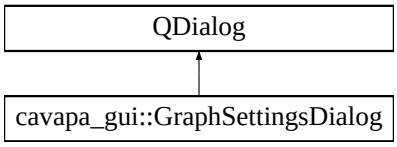
- gui/activitygraph.h

6.15 cavapa_gui::GraphSettingsDialog Class Reference

Class for settings dialog of graphwidget.

`#include <graphsettingsdialog.h>`

Inheritance diagram for cavapa_gui::GraphSettingsDialog:



Signals

- void sendSettings (GraphSettings settings)

  Sends newly edited graph settings to the parent.
- void sendfinalSettings (GraphSettings settings)

  Sends and saves the graph settings to the parent.
- void requestDefaultSettings ()

  Requests default settings from the parent.
- void requestPreviousSettings ()

  Requests previous settings from the parent.

Public Member Functions

- GraphSettingsDialog (QWidget *parent=0)

  Constructor for the graph settings dialog.
- void getSettings (GraphSettings settings)

  Gets graph settings from the parent.
- GraphSettings sendCurrent ()

  Sends current settings to the parent.

6.15.1 Detailed Description

Class for settings dialog of graphwidget.

Offers options for font, colors and linewidths.

Author

Joel Kivelä

6.15.2 Constructor & Destructor Documentation

6.15.2.1 cavapa_gui::GraphSettingsDialog::GraphSettingsDialog (QWidget * *parent* = `0`) [explicit]

Constructor for the graph settings dialog.

Parameters

parent	Default QWidget parent.

6.15.3 Member Function Documentation

6.15.3.1 void cavapa_gui::GraphSettingsDialog::getSettings (GraphSettings *settings*)

Gets graph settings from the parent.

Parameters

settings	Settings as GraphSettings.

6.15.3.2 GraphSettings cavapa_gui::GraphSettingsDialog::sendCurrent ( )

Sends current settings to the parent.

Returns

Current settings as GraphSettings.

6.15.3.3 void cavapa_gui::GraphSettingsDialog::sendfinalSettings (GraphSettings *settings*) [signal]

Sends and saves the graph settings to the parent.

Parameters

settings	The graph settings object.

6.15.3.4 void cavapa_gui::GraphSettingsDialog::sendSettings (GraphSettings *settings*) [signal]

Sends newly edited graph settings to the parent.

Parameters

settings	Settings as GraphSettings.

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

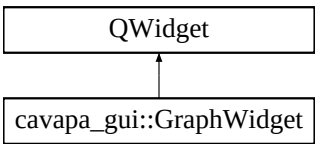
- gui/graphsettingsdialog.h
- gui/graphsettingsdialog.cpp

6.16 cavapa_gui::GraphWidget Class Reference

The widget holds the ActivityGraph and all its controls.

`#include <graphwidget.h>`

Inheritance diagram for cavapa_gui::GraphWidget:



Public Slots

- void requestStatistics (FrameTime start, FrameTime stop, int points)

  Requests statistical data.
- void sendDefaultSettings ()

  Sets default settings into graph settings dialog.
- void sendPreviousSettings ()

Gets the previous graph settings (when settings dialog canceled).

- void printGraphToFile ()

  Saves the graph in to scalable svg image file.
- void removeFromHistory (int i)

  Removes the selected marker text from marker history.
- void reset ()

  Resets the graph and nulls the clocks.
- void setZoomInActive (bool active)

  Sets zoom in button active or inactive.
- void setZoomOutActive (bool active)

  Sets zoom out button active or inactive.
- void setScrollLeftActive (bool active)

  Sets slide left button active or inactive.
- void setScrollRightActive (bool active)

  Sets slide right button active or inactive.
- void setScrollBarSize (int size)

  Sets the scroll bar size.
- void changeFixed ()

  Set fixed time window -button disabled.
- void changeAbsoluteEnabled (bool ae)

  Sets whether absolute time button is active or inactive.
- void openExportSelectionDialog ()

  Opens export dialog and send values from ui objects into it.

Signals

- void statisticsRequested (FrameTime start, FrameTime stop, int points)

  Signal to be emitted when the graph wants data to be retrieved.
- void sendGraphSettings (GraphSettings currentsettings)

  Sends current settings of the graph to parent.
- void settingsChanged ()

  Sends notify to parent that graph's settings have been changed.
- bool exportCommand (const ExportOptions &options)

  Passes export command with generated options from ExportDialog to parent.
- void pointedFrame (FrameTime point)

  Sends pointed frametime from graph to parent.
- void markerChanged ()

  Informs the parent of marker adding.

Public Member Functions

- GraphWidget (QWidget *parent=0)

  Constructs the GraphWidget.
- void updateData (const FrameStats &new_data)

  Informs the graph about a new frame statistics.
- void updateStatistics (const std::vector< FrameStats > &stats)

  Updates the graph with new statistics.
- void getSettings (GraphSettings settings)

  Gets settings from the parent and sets them as the current ones.
- GraphSettings giveSettings ()

Gives current settings to parent.

- void setMarkers (std::vector< GraphMarker > markers)

  Sets the markers from parent when loading saved measurement.

- void setMarkerHistory (std::vector< std::string > history)

  Sets the marker history from parent when loading saved measurement.

- std::vector< GraphMarker > getMarkers ()

  Gets markers from the graph and passes them to parent.

- std::vector< std::string > getMarkerHistory ()

  Passes marker history to parent.

6.16.1 Detailed Description

The widget holds the ActivityGraph and all its controls.

The main idea of the class is to separate the main window from all these controls and act as a wrapper for everything related to the graph. By simply hiding the widget we can easily hide all the graph-related elements.

Author

Joel Kivelä

6.16.2 Constructor & Destructor Documentation

6.16.2.1 cavapa_gui::GraphWidget::GraphWidget (QWidget * *parent* = 0) [explicit]

Constructs the GraphWidget.

Parameters

parent	Parent of The Widget.

6.16.3 Member Function Documentation

6.16.3.1 void cavapa_gui::GraphWidget::changeAbsoluteEnabled (bool *ae*) [slot]

Sets whether absolute time button is active or inactive.

Parameters

ae	True/false for if active or not.

6.16.3.2 bool cavapa_gui::GraphWidget::exportCommand (const ExportOptions & *options*) [signal]

Passes export command with generated options from ExportDialog to parent.

Parameters

options	Export options as ExportOptions.

Returns

True/false flag of export success.

6.16.3.3 std::vector< std::string > cavapa_gui::GraphWidget::getMarkerHistory ( )

Passes marker history to parent.

Returns

Marker history as string vector array.

6.16.3.4 std::vector< GraphMarker > cavapa_gui::GraphWidget::getMarkers ( )

Gets markers from the graph and passes them to parent.

Returns

Markers as vector array.

6.16.3.5 void cavapa_gui::GraphWidget::getSettings (GraphSettings *settings*)

Gets settings from the parent and sets them as the current ones.

Parameters

settings	New settings from parent in GraphSettings.

6.16.3.6 GraphSettings cavapa_gui::GraphWidget::giveSettings ( )

Gives current settings to parent.

Returns

Current settings of the graph.

6.16.3.7 void cavapa_gui::GraphWidget::pointedFrame (FrameTime *point*) [signal]

Sends pointed frametime from graph to parent.

Parameters

point	Point in FrameTime.

6.16.3.8 void cavapa_gui::GraphWidget::removeFromHistory (int *i*) [slot]

Removes the selected marker text from marker history.

Parameters

i	Index in markers vector array.

6.16.3.9 void cavapa_gui::GraphWidget::requestStatistics (FrameTime *start*, FrameTime *stop*, int *points*) [inline], [slot]

Requests statistical data.

Parameters

start	Period start time. If 0, the period will start from the first possible frame.
stop	Period end time. If 0, the period will end at the last possible frame.
points	Number of points desired to be returned.

6.16.3.10 void cavapa_gui::GraphWidget::sendGraphSettings (GraphSettings *currentsettings*) [signal]

Sends current settings of the graph to parent.

Parameters

currentsettings	Current settings in GraphSettings.

6.16.3.11 void cavapa_gui::GraphWidget::setMarkerHistory (std::vector< std::string > *history*)

Sets the marker history from parent when loading saved measurement.

Parameters

history	History as vector array of strings.

6.16.3.12 void cavapa_gui::GraphWidget::setMarkers (std::vector< GraphMarker > *markers*)

Sets the markers from parent when loading saved measurement.

Parameters

markers	Markers as vector array of GraphMarkers.

6.16.3.13 void cavapa_gui::GraphWidget::setScrollBarSize (int *size*) [slot]

Sets the scroll bar size.

Parameters

size	Scroll bar size as integer.

6.16.3.14 void cavapa_gui::GraphWidget::setScrollLeftActive (bool *active*) [slot]

Sets slide left button active or inactive.

Parameters

active	True/false for if active or not.

6.16.3.15 void cavapa_gui::GraphWidget::setScrollRightActive (bool *active*) [slot]

Sets slide right button active or inactive.

Parameters

active	True/false for if active or not.

6.16.3.16 void cavapa_gui::GraphWidget::setZoomInActive (bool *active*) [slot]

Sets zoom in button active or inactive.

Parameters

active	True/false for if active or not.

6.16.3.17 void cavapa_gui::GraphWidget::setZoomOutActive (bool *active*) [slot]

Sets zoom out button active or inactive.

Parameters

active	True/false for if active or not.

6.16.3.18 void cavapa_gui::GraphWidget::statisticsRequested (FrameTime *start,* FrameTime *stop,* int *points*) [signal]

Signal to be emitted when the graph wants data to be retrieved.

Parameters

start	Period start time. If 0, the period will start from the first possible frame.
stop	Period end time. If 0, the period will end at the last possible frame.
points	Number of points desired to be returned.

6.16.3.19 void cavapa_gui::GraphWidget::updateData (const FrameStats & *new_data*)

Informs the graph about a new frame statistics.

Unimplemented but required by the parent.

Parameters

new_data	Statistics from the current frame update.
new_data	Not used.

6.16.3.20 void cavapa_gui::GraphWidget::updateStatistics (const std::vector< FrameStats > & *stats*)

Updates the graph with new statistics.

Passes new statistics into graph.

Either the graph requested new statistics with requestData()-signal or the graph is being initialized with new statistical information.

Parameters

stats	Statistical information from a certain period.

stats	Statisticcs as vector array of FrameStats.

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

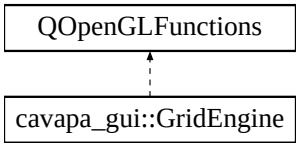
- gui/graphwidget.h
- gui/graphwidget.cpp

6.17 cavapa_gui::GridEngine Class Reference

Draws a grid with OpenGL.

`#include <gridengine.h>`

Inheritance diagram for cavapa_gui::GridEngine:



Public Member Functions

- void initialize ()
- void drawGridGeometry (QGLShaderProgram ∗program)

  Draws the grid.

6.17.1 Detailed Description

Draws a grid with OpenGL.

The grid is drawn in the XZ plane and its center is in (0,0,0).

Remarks

The original implementation used vertex buffer objects but for some reason this seemed to be incompatible with QPainter overdawing.

Author

Oskari Leppäaho

6.17.2 Member Function Documentation

6.17.2.1 void cavapa_gui::GridEngine::drawGridGeometry (QGLShaderProgram ∗ *program*)

Draws the grid.

Parameters

∗*program*	Specifies the OpenGL shader program to be used.

6.17.2.2 void cavapa_gui::GridEngine::initialize ( )

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

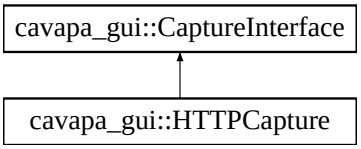
- gui/gridengine.h
- gui/gridengine.cpp

6.18 cavapa_gui::HTTPCapture Class Reference

The Capture device for HTTP images.

`#include <httpcapture.h>`

Inheritance diagram for cavapa_gui::HTTPCapture:



Public Member Functions

- DISALLOW_COPY_AND_ASSIGN (HTTPCapture)

  Copy and assign of the class is not allowed.
- double getBarrelCorrection () override

  Retrieves the barrel correction applied to the source.
- FrameCapture getBuffered () override

  Retrieves the currently buffered frame.
- std::string getDescription () const override

  Retrieves the description of the device.
- int getDeviceID () const override

  Returns the ID number the device was initialized with.
- double getFramerate () const override

  Retrieves the capture device framerate.
- int getMissedFrames () const override

  Retrieves the logged frame misses.
- FrameCapture getNext (bool skip, bool *error) override

  Retrieves the next frame.
- std::string getPath () const override

  Retrieves the path given during the initialization.
- cv::Size getResolution () const override

  Retrieves the capture device resolution.
- int getRetrievedFrames () const override

  Retrieves the total logged frames.
- INTERFACE_TYPE getType () const override

  Retrieves the type of the device.
- bool hasNew () override

  Checks whether the device has a new image to retrieve or not.
- bool isOpen () const override

  Checks whether the capturing device is open or not.

- bool open (const std::string &path) override

  Opens the video file or the stream.
- void resetStats () override

  Resets the statistical counters.
- void setBarrelCorrection (double amount) override

  Sets the amount of the barrel correction to be applied.
- bool setResolution (const cv::Size &new_size) override

  Sets the capturing device resolution.

Additional Inherited Members

6.18.1 Detailed Description

The Capture device for HTTP images.

The class can be used as the interface device for capturing the HTTP images from websites (like webcam). It uses CURL library for it's image retrieval.

Remarks

The class was design to be a backup interface for network cameras. As it turned out, there was no real need for the class so it was excluded from the build. It is still here in case it is needed.

Author

Petri Partanen

6.18.2 Member Function Documentation

6.18.2.1 double cavapa_gui::HTTPCapture::getBarrelCorrection ( ) [inline], [override], [virtual]

Retrieves the barrel correction applied to the source.

Returns

The amount of the barrel correction.

Implements cavapa_gui::CaptureInterface.

6.18.2.2 FrameCapture cavapa_gui::HTTPCapture::getBuffered ( ) [override], [virtual]

Retrieves the currently buffered frame.

Returns

The buffered frame.

Remarks

This does not signal for a new image fetch from the device.

Implements cavapa_gui::CaptureInterface.

6.18.2.3 std::string cavapa_gui::HTTPCapture::getDescription ( ) const [inline], [override], [virtual]

Retrieves the description of the device.

Returns

For the devices initialized with HW index, this will be the name of the HW device. For other devices this is the path given during initialization.

Implements cavapa_gui::CaptureInterface.

6.18.2.4 int cavapa_gui::HTTPCapture::getDeviceID ( ) const [inline], [override], [virtual]

Returns the ID number the device was initialized with.

Returns

The device ID used in the initialization or UNKNOWN_DEVICE if not specified.
UNKNOWN_DEVICE. The HTTP devices are not real devices.

Implements cavapa_gui::CaptureInterface.

6.18.2.5 double cavapa_gui::HTTPCapture::getFramerate ( ) const [inline], [override], [virtual]

Retrieves the capture device framerate.

Returns

The current framerate.
0. The HTTP devices do not have framerate.

Implements cavapa_gui::CaptureInterface.

6.18.2.6 int cavapa_gui::HTTPCapture::getMissedFrames ( ) const [inline], [override], [virtual]

Retrieves the logged frame misses.

A frame miss occurs everytime when getNext() is called and the previous retrieval was still ongoing.

Returns

The count of the missed frames.

Implements cavapa_gui::CaptureInterface.

6.18.2.7 FrameCapture cavapa_gui::HTTPCapture::getNext (bool *skip*, bool * *error*) [override], [virtual]

Retrieves the next frame.

Basically it returns the currently buffered frame and signals the device to grab a new image for next call to the function. If the device has not returned from previous image grab the call is recorded as a missed frame.

Parameters

skip	Specifies if the frame can be skipped. When set to true, this will force the main thread to wait for a new image from the device. This can result in a huge lag if the device is not capable of retrieving images fast enough.
error	Will receive an error message if something went wrong.

Returns

The next frame with the reference to the image! You must make a clone if you wish to alter it!

Implements cavapa_gui::CaptureInterface.

6.18.2.8 std::string cavapa_gui::HTTPCapture::getPath ( ) const [inline], [override], [virtual]

Retrieves the path given during the initialization.

Returns

The path of the device or an empty string if not found.

Implements cavapa_gui::CaptureInterface.

6.18.2.9 cv::Size cavapa_gui::HTTPCapture::getResolution ( ) const [override], [virtual]

Retrieves the capture device resolution.

Returns

The resolution of the device.

Remarks

On HTTP sources this can only be known after the first successful image retrieval.

Implements cavapa_gui::CaptureInterface.

6.18.2.10 int cavapa_gui::HTTPCapture::getRetrievedFrames ( ) const [inline], [override], [virtual]

Retrieves the total logged frames.

Basically, this is a counter on how many times getNext() has been called.

Returns

The count of the total frames returned.

Implements cavapa_gui::CaptureInterface.

6.18.2.11 INTERFACE_TYPE cavapa_gui::HTTPCapture::getType ( ) const [inline], [override], [virtual]

Retrieves the type of the device.

Returns

The interface type.

Implements cavapa_gui::CaptureInterface.

6.18.2.12 bool cavapa_gui::HTTPCapture::hasNew ( ) [inline], [override], [virtual]

Checks whether the device has a new image to retrieve or not.

This checks if the FrameTime on the buffered image differs from what was retrieved on the last getNext().

Returns

True if new image or false otherwise.
true. HTTP devices always have new images.

Implements cavapa_gui::CaptureInterface.

6.18.2.13 bool cavapa_gui::HTTPCapture::isOpen ( ) const [inline], [override], [virtual]

Checks whether the capturing device is open or not.

Returns

True if the device is open or false otherwise.

Implements cavapa_gui::CaptureInterface.

6.18.2.14 bool cavapa_gui::HTTPCapture::open (const std::string & *path*) [override], [virtual]

Opens the video file or the stream.

Parameters

path	The path of the video file or the network stream.

Returns

True if it was opened successfully or false otherwise.

Implements cavapa_gui::CaptureInterface.

6.18.2.15 void cavapa_gui::HTTPCapture::resetStats ( ) [override], [virtual]

Resets the statistical counters.

This affects the missed and total frame counters.

Implements cavapa_gui::CaptureInterface.

6.18.2.16 void cavapa_gui::HTTPCapture::setBarrelCorrection (double *amount*) [inline], [override], [virtual]

Sets the amount of the barrel correction to be applied.

Parameters

amount	The barrel effect. 0.0 means none.

Implements cavapa_gui::CaptureInterface.

6.18.2.17 bool cavapa_gui::HTTPCapture::setResolution (const cv::Size & *new_size*) [inline], [override], [virtual]

Sets the capturing device resolution.

Parameters

new_size	The size to be set.

Returns

True if at least one of the dimension was set or false otherwise.
false. You cannot change resolution of HTTP sources.

Implements cavapa_gui::CaptureInterface.

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

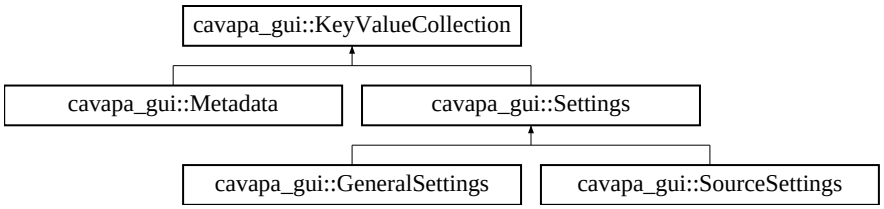
- source/httpcapture.h
- source/httpcapture.cpp

6.19 cavapa_gui::KeyValueCollection Class Reference

Represents a collection of key-value pairs.

`#include <keyvaluecollection.h>`

Inheritance diagram for cavapa_gui::KeyValueCollection:



Public Types

- using const_iterator = std::map< std::string, std::string >::const_iterator

  Provides a bidirectional iterator that can read a const element in the collection.

Public Member Functions

- const_iterator begin () const

  Returns a const iterator to the beginning of the collection.
- const_iterator end () const

  Returns a const iterator to the end of the collection.
- std::string get (const std::string &name) const

  Gets the value of the specified key as a string.
- bool getBool (const std::string &name) const

  Gets the value of the specified key as a bool.
- QColor getColor (const std::string &name) const

  Gets the value of the specified key as a QColor.
- QString getQString (const std::string &name) const

  Gets the value of the specified key as a QString.
- double getDouble (const std::string &name) const

  Gets the value of the specified key as a double.
- int getInt (const std::string &name) const

Gets the value of the specified key as an integer.

- template<typename ValueType , int N>
  std::array< ValueType, N > getValues (const std::string &name) const

  Gets an array of values from the specified key.
- void set (const std::string &name, const std::string &value)

  Sets the value of the specified key.
- void setBool (const std::string &name, bool value)

  Sets the value of the specified key as a bool.
- void setColor (const std::string &name, const QColor &value)

  Sets the value of the specified key as a QColor.
- void setInt (const std::string &name, int value)

  Sets the value of the specified key as an int.
- template<typename... T>
  void setValues (const std::string &name, T...values)

  Sets the value of the specified key as an array.
- void setQString (const std::string &name, const QString &value)

  Sets the value of the specified key as a QString.
- bool trySet (const std::string &name, const std::string &value)

  Attempts to set the value of the specified key.

Protected Attributes

- std::map< std::string,
  std::string > keyValueMap

  Contains the key-value mappings in an std::map.

6.19.1 Detailed Description

Represents a collection of key-value pairs.

Author

Mika Lehtinen

6.19.2 Member Function Documentation

6.19.2.1 KeyValueCollection::const_iterator cavapa_gui::KeyValueCollection::begin ( ) const

Returns a const iterator to the beginning of the collection.

Returns

The const iterator.

6.19.2.2 KeyValueCollection::const_iterator cavapa_gui::KeyValueCollection::end ( ) const

Returns a const iterator to the end of the collection.

Returns

The const iterator.

6.19.2.3 std::string cavapa_gui::KeyValueCollection::get (const std::string & *name*) const

Gets the value of the specified key as a string.

Parameters

name	The name of the key to be retrieved.

Returns

The value of the key as a string.

6.19.2.4 bool cavapa_gui::KeyValueCollection::getBool (const std::string & *name*) const

Gets the value of the specified key as a bool.

Parameters

name	The name of the key to be retrieved.

Returns

The value of the key as a bool.

6.19.2.5 QColor cavapa_gui::KeyValueCollection::getColor (const std::string & *name*) const

Gets the value of the specified key as a QColor.

Parameters

name	The name of the key to be retrieved.

Returns

The value of the key as a QColor.

6.19.2.6 double cavapa_gui::KeyValueCollection::getDouble (const std::string & *name*) const

Gets the value of the specified key as a double.

Parameters

name	The name of the key to be retrieved.

Returns

The value of the key.

6.19.2.7 int cavapa_gui::KeyValueCollection::getInt (const std::string & *name*) const

Gets the value of the specified key as an integer.

Parameters

name	The name of the key to be retrieved.

Returns

The value of the key.

6.19.2.8 QString cavapa_gui::KeyValueCollection::getQString (const std::string & *name*) const

Gets the value of the specified key as a QString.

Parameters

name	The name of the key to be retrieved.

Returns

The value of the key.

6.19.2.9 template<typename ValueType , int N> std::array<ValueType, N> cavapa_gui::KeyValueCollection::getValues (const std::string & *name*) const [inline]

Gets an array of values from the specified key.

Parameters

name	The name of the key to be retrieved.

Template Parameters

ValueType	The type of the values, either double or int.
N	The size of the array.

Returns

The values of the key as an array.

6.19.2.10 void cavapa_gui::KeyValueCollection::set (const std::string & *name,* const std::string & *value*)

Sets the value of the specified key.

Parameters

name	The name of the key to be set.
value	The value of the key.

6.19.2.11 void cavapa_gui::KeyValueCollection::setBool (const std::string & *name,* bool *value*)

Sets the value of the specified key as a bool.

Parameters

name	The name of the key to be set.
value	The value of the key as a bool.

6.19.2.12 void cavapa_gui::KeyValueCollection::setColor (const std::string & *name,* const QColor & *value*)

Sets the value of the specified key as a QColor.

Parameters

name	The name of the key to be set.
value	The value of the key as a QColor.

6.19.2.13 void cavapa_gui::KeyValueCollection::setInt (const std::string & *name,* int *value*)

Sets the value of the specified key as an int.

Parameters

name	The name of the key to be set.
value	The value of the key as an int.

6.19.2.14 void cavapa_gui::KeyValueCollection::setQString (const std::string & *name,* const QString & *value*)

Sets the value of the specified key as a QString.

Parameters

name	The name of the key to be set.
value	The value of the key as a QString.

6.19.2.15 template<typename... T> void cavapa_gui::KeyValueCollection::setValues (const std::string & *name,* T... *values*) [inline]

Sets the value of the specified key as an array.

Parameters

name	The name of the key to be set.
values	The values.

Template Parameters

T	The type of the values, usually deduced automatically by the compiler.

6.19.2.16 bool cavapa_gui::KeyValueCollection::trySet (const std::string & *name,* const std::string & *value*)

Attempts to set the value of the specified key.

Parameters

name	The name of the key to be set.
value	The value of the key.

Returns

True if the value was successfully set, false otherwise.

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

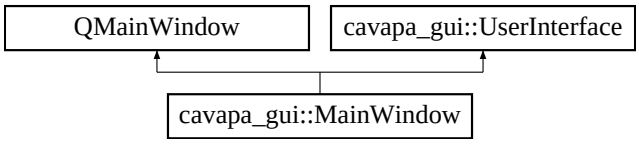
- keyvaluecollection.h
- keyvaluecollection.cpp

6.20 cavapa_gui::MainWindow Class Reference

The main window for the CAVAPA-GUI application.

`#include <mainwindow.h>`

Inheritance diagram for cavapa_gui::MainWindow:



Signals

- void calculationCancelRequested ()

  Emitted when the user requests to cancel the calculation.
- void calculationContinueRequested ()

  Emitted when the user requests to continue the calculation.
- void calculationPauseRequested ()

  Emitted when the user requests to pause the calculation.
- void calculationStartRequested ()

  Emitted when the user requests to start the calculation.
- void calculationStopRequested ()

  Emitted when the user requests to stop the calculation.
- void calibrationPointsReady (const std::vector< cavapa::CalibrationPoint > &points)

  Emitted when the user has set all the calibration points.
- void generalSettingsChanged (const GeneralSettings &settings)

  Emitted when the user changes a general setting.
- void graphPositionSelected (FrameTime time)

  Emitted when the user selects a position in the activity graph.
- void locationChanged (int x, int y)

  Emitted when the interface is moved.
- void markersReady (const std::vector< GraphMarker > &markers)

  Emitted when the user has placed all the markers.
- void metadataOpened (const std::string &path)

  Emitted when the user requests to open a metadata file.
- void metadataReady (const Metadata &metadata)

  Emitted when the user has input all the metadata.
- void newMeasurementRequested ()

  Emitted when the user wants to start a new measurement.
- void openMeasurementRequested (const std::string &path)

  Emitted when the user wants to open an existing measurement.
- void redoMeasurementRequested ()

  Emitted when the user wants to redo the measurement.
- void refreshRequested ()

  Emitted when the user requests to refresh cameras.
- void sizeChanged (int width, int height, bool fullscreen)

  Emitted when the size of the interface changes.
- void sourceAddRequested (const std::string &path)

  Emitted when the user adds a new video source.
- void sourcePauseRequested (SourceID source)

  Emitted when the user requests to pause a source.
- void sourcePlayRequested (SourceID source)

  Emitted when the user requests to play a source.
- void sourceRemoveRequested (SourceID source)

  Emitted when the user removes a source.

- void sourceSeekRequested (SourceID source, FrameTime position)

  Emitted when the user requests to seek a source to a certain position.
- void sourceSettingsChanged (SourceID source, const SourceSettings &settings)

  Emitted when the user changes a setting for a source.
- void sourceStepBackward (SourceID source)

  Emitted when the user steps a video one frame back.
- void sourceStepForward (SourceID source)

  Emitted when the user steps a video one frame forward.
- void statisticsRequested (FrameTime start, FrameTime stop, int points)

  Emitted when the user wants the frame statistics to be retrieved.
- void interfaceClosed ()

  Emitted when the user closes the interface.
- void exportRequested (const ExportOptions &options)

  Emitted when the user requests to export data from the graph.

Public Member Functions

- MainWindow (QMainWindow *parent=0)

  Constructs a new MainWindow having the specified parent.
- void addSource (SourceID source, SourceType type, FrameTime length, const std::string &description) override

  Adds a new source to the interface.
- void calculationCanceled () override

  Informs the interface that the calculation was cancelled.
- void calculationCompleted () override

  Informs the interface that the calculation has completed.
- void calculationContinued () override

  Informs the interface that the calculation was continued.
- void calculationPaused () override

  Informs the interface that the calculation was paused.
- void calculationStarted () override

  Informs the interface that the calculation has started.
- void calculationStopped () override

  Informs the interface that the calculation has stopped.
- std::vector< std::string > getMarkerHistory () override

  Returns the history of the recently used markers.
- void newMeasurement () override

  Informs the interface that the user wants to start a new measurement.
- void openMeasurement () override

  Informs the interface that a measurement has been opened.
- void removeSource (SourceID source) override

  Removes the specified source from the interface.
- void setCalibrationPoints (SourceID source, const std::vector< CalibrationPoint > &points) override

  Sets the calibration points for the specified source.
- void setCanPause (bool flag) override

  Informs the interface whether the calculation can be paused.
- void setCanSaveVideos (bool flag) override

  Sets whether videos can be saved.
- void setLocation (int x, int y) override

  Sets the location of the interface.

- void setMarkerHistory (const std::vector< std::string > &history) override

  Sets the history of previously input markers.
- void setSize (int width, int height, bool fullscreen) override

  Sets the size of the interface.
- void showMessage (const std::string &message, ErrorLevel errorLevel) override

  Displays a message in the interface.
- void sourcePaused (SourceID source) override

  Informs the interface that a source was paused.
- void sourcePlayStarted (SourceID source) override

  Informs the interface that playing was started for a source.
- void startInterface () override

  Starts the interface.
- void updateFrame (const FrameCapture &frame) override

  Updates the frame for a source.
- void updateCalculation (const FrameStats &statistics) override

  Updates the calculation data.
- void updateGeneralSettings (const GeneralSettings &settings) override

  Updates the general settings for the interface.
- void updateMarkers (const std::vector< GraphMarker > &markers) override

  Updates the graph markers.
- void updateMetadata (const Metadata &metadata) override

  Updates the metadata to the interface.
- void updateSourceSettings (SourceID source, const SourceSettings &settings) override

  Updates the settings of the speficied source.
- void updateStatistics (const std::vector< FrameStats > &stats) override

  Updates the interface with the frame statistics from a certain period.

Protected Member Functions

- virtual void changeEvent (QEvent *event)

  Reimplemented for handling a state change event.
- virtual void closeEvent (QCloseEvent *event)

  Reimplemented for handling a window close event.
- virtual void moveEvent (QMoveEvent *event)

  Reimplemented for handling a window move event.
- virtual void resizeEvent (QResizeEvent *event)

  Reimplemented for handling a window resize event.

6.20.1 Detailed Description

The main window for the CAVAPA-GUI application.

Author

Mika Lehtinen, Oskari Leppäaho

6.20.2 Constructor & Destructor Documentation

6.20.2.1 cavapa_gui::MainWindow::MainWindow (QMainWindow * *parent* = 0) [explicit]

Constructs a new MainWindow having the specified parent.

Parameters

parent	The parent window.

6.20.3 Member Function Documentation

6.20.3.1 void cavapa_gui::MainWindow::addSource (SourceID *source*, SourceType *type*, FrameTime *length*, const std::string & *path*) [override], [virtual]

Adds a new source to the interface.

Parameters

source	The ID of the source.
type	The type of the source.
length	The length of the source.
path	The path of the source.

Implements cavapa_gui::UserInterface.

6.20.3.2 void cavapa_gui::MainWindow::calculationCancelRequested ( ) [signal]

Emitted when the user requests to cancel the calculation.

6.20.3.3 void cavapa_gui::MainWindow::calculationContinueRequested ( ) [signal]

Emitted when the user requests to continue the calculation.

6.20.3.4 void cavapa_gui::MainWindow::calculationPauseRequested ( ) [signal]

Emitted when the user requests to pause the calculation.

6.20.3.5 void cavapa_gui::MainWindow::calculationStartRequested ( ) [signal]

Emitted when the user requests to start the calculation.

6.20.3.6 void cavapa_gui::MainWindow::calculationStopRequested ( ) [signal]

Emitted when the user requests to stop the calculation.

6.20.3.7 void cavapa_gui::MainWindow::calibrationPointsReady (const std::vector< cavapa::CalibrationPoint > & *points*) [signal]

Emitted when the user has set all the calibration points.

Parameters

points	The set of calibration points.

6.20.3.8 void cavapa_gui::MainWindow::changeEvent (QEvent * *event*) [protected], [virtual]

Reimplemented for handling a state change event.

Parameters

event	The event object.

6.20.3.9 void cavapa_gui::MainWindow::closeEvent (QCloseEvent * *event*) [protected], [virtual]

Reimplemented for handling a window close event.

Parameters

event	The close event object.

6.20.3.10 void cavapa_gui::MainWindow::exportRequested (const ExportOptions & *options*) [signal]

Emitted when the user requests to export data from the graph.

Parameters

options	The parameters for the export.

6.20.3.11 void cavapa_gui::MainWindow::generalSettingsChanged (const GeneralSettings & *settings*) [signal]

Emitted when the user changes a general setting.

Parameters

settings	The new GeneralSettings object.

6.20.3.12 std::vector< string > cavapa_gui::MainWindow::getMarkerHistory ( ) [override], [virtual]

Returns the history of the recently used markers.

Returns

The history of the recently used markers as a vector of strings.

Implements cavapa_gui::UserInterface.

6.20.3.13 void cavapa_gui::MainWindow::graphPositionSelected (FrameTime *time*) [signal]

Emitted when the user selects a position in the activity graph.

Parameters

time	The time of the position.

6.20.3.14 void cavapa_gui::MainWindow::interfaceClosed ( ) [signal]

Emitted when the user closes the interface.

6.20.3.15 void cavapa_gui::MainWindow::locationChanged (int *x*, int *y*) [signal]

Emitted when the interface is moved.

Parameters

x	The x-coordinate of the interface.
y	The y-coordinate of the interface.

6.20.3.16 void cavapa_gui::MainWindow::markersReady (const std::vector< GraphMarker > & *markers*) [signal]

Emitted when the user has placed all the markers.

Parameters

markers	The vector containing the markers.

Remarks

At the moment, this signal is emitted when the calculation completes.

6.20.3.17 void cavapa_gui::MainWindow::metadataOpened (const std::string & *path*) [signal]

Emitted when the user requests to open a metadata file.

Parameters

path	The path of the file that was selected.

6.20.3.18 void cavapa_gui::MainWindow::metadataReady (const Metadata & *metadata*) [signal]

Emitted when the user has input all the metadata.

Parameters

metadata	The Metadata object representing the user input.

6.20.3.19 void cavapa_gui::MainWindow::moveEvent (QMoveEvent * *event*) [protected], [virtual]

Reimplemented for handling a window move event.

Parameters

event	The move event object.

6.20.3.20 void cavapa_gui::MainWindow::newMeasurementRequested ( ) [signal]

Emitted when the user wants to start a new measurement.

6.20.3.21 void cavapa_gui::MainWindow::openMeasurementRequested (const std::string & *path*) [signal]

Emitted when the user wants to open an existing measurement.

Parameters

path	The path to the measurement's XML file.

6.20.3.22 void cavapa_gui::MainWindow::redoMeasurementRequested ( ) [signal]

Emitted when the user wants to redo the measurement.

6.20.3.23 void cavapa_gui::MainWindow::refreshRequested ( ) [signal]

Emitted when the user requests to refresh cameras.

6.20.3.24 void cavapa_gui::MainWindow::removeSource (SourceID *source*) [override], [virtual]

Removes the specified source from the interface.

Parameters

source	The id of the source to be removed.

Implements cavapa_gui::UserInterface.

6.20.3.25 void cavapa_gui::MainWindow::resizeEvent (QResizeEvent * *event*) [protected], [virtual]

Reimplemented for handling a window resize event.

Parameters

event	The resize event object.

6.20.3.26 void cavapa_gui::MainWindow::setCalibrationPoints (SourceID *source*, const std::vector< CalibrationPoint > & *points*) [override], [virtual]

Sets the calibration points for the specified source.

Parameters

source	The id of the source whose calibration points will be set.
points	The collection of calibration points.

Implements cavapa_gui::UserInterface.

6.20.3.27 void cavapa_gui::MainWindow::setCanPause (bool *flag*) [override], [virtual]

Informs the interface whether the calculation can be paused.

Parameters

flag	True if the calculation can be paused, false otherwise.

Implements cavapa_gui::UserInterface.

6.20.3.28 void cavapa_gui::MainWindow::setCanSaveVideos (bool *flag*) [override], [virtual]

Sets whether videos can be saved.

Parameters

flag	True if videos can be saved, false otherwise.

Implements cavapa_gui::UserInterface.

6.20.3.29 void cavapa_gui::MainWindow::setLocation (int *x*, int *y*) [override], [virtual]

Sets the location of the interface.

Parameters

x	The x-coordinate of the interface.
y	The y-coordinate of the interface.

Implements cavapa_gui::UserInterface.

6.20.3.30 void cavapa_gui::MainWindow::setMarkerHistory (const std::vector< std::string > & *history*) [override], [virtual]

Sets the history of previously input markers.

Parameters

history	The collection of strings representing the history.

Implements cavapa_gui::UserInterface.

6.20.3.31 void cavapa_gui::MainWindow::setSize (int *width*, int *height*, bool *fullscreen*) [override], [virtual]

Sets the size of the interface.

Parameters

width	The desired width of the interface.
height	The desired height of the interface.
fullscreen	Whether the interface is in fullscreen mode or not.

Implements cavapa_gui::UserInterface.

6.20.3.32 void cavapa_gui::MainWindow::showMessage (const std::string & *message*, ErrorLevel *level*) [override], [virtual]

Displays a message in the interface.

Parameters

message	The message to be displayed.
level	The severity level of the message.

Implements cavapa_gui::UserInterface.

6.20.3.33 void cavapa_gui::MainWindow::sizeChanged (int *width*, int *height*, bool *fullscreen*) [signal]

Emitted when the size of the interface changes.

Parameters

width	The new width of the interface.
height	The new height of the interface.
fullscreen	Whether the interface is in fullscreen mode or not.

6.20.3.34 void cavapa_gui::MainWindow::sourceAddRequested (const std::string & *path*) [signal]

Emitted when the user adds a new video source.

Parameters

path	The path of the source.

6.20.3.35 void cavapa_gui::MainWindow::sourcePaused (SourceID *source*) [override], [virtual]

Informs the interface that a source was paused.

Parameters

source	The id of the source that was paused.

Implements cavapa_gui::UserInterface.

6.20.3.36 void cavapa_gui::MainWindow::sourcePauseRequested (SourceID *source*) [signal]

Emitted when the user requests to pause a source.

Parameters

source	The id of the source to be paused.

6.20.3.37 void cavapa_gui::MainWindow::sourcePlayRequested (SourceID *source*) [signal]

Emitted when the user requests to play a source.

Parameters

source	The id of the source to play.

6.20.3.38 void cavapa_gui::MainWindow::sourcePlayStarted (SourceID *source*) [override], [virtual]

Informs the interface that playing was started for a source.

Parameters

source	The source that is being played.

Implements cavapa_gui::UserInterface.

6.20.3.39 void cavapa_gui::MainWindow::sourceRemoveRequested (SourceID *source*) [signal]

Emitted when the user removes a source.

Parameters

source	The source ID to be removed.

6.20.3.40 void cavapa_gui::MainWindow::sourceSeekRequested (SourceID *source*, FrameTime *position*) [signal]

Emitted when the user requests to seek a source to a certain position.

Parameters

source	The id of the source to be sought.
position	The requested position.

6.20.3.41 void cavapa_gui::MainWindow::sourceSettingsChanged (SourceID *source*, const SourceSettings & *settings*) [signal]

Emitted when the user changes a setting for a source.

Parameters

source	The source ID for which a setting was changed.
settings	The new settings for the source.

6.20.3.42 void cavapa_gui::MainWindow::sourceStepBackward (SourceID *source*) [signal]

Emitted when the user steps a video one frame back.

Parameters

source	The source that was stepped.

6.20.3.43 void cavapa_gui::MainWindow::sourceStepForward (SourceID *source*) [signal]

Emitted when the user steps a video one frame forward.

Parameters

source	The source that was stepped.

6.20.3.44 void cavapa_gui::MainWindow::statisticsRequested (FrameTime *start*, FrameTime *stop*, int *points*) [signal]

Emitted when the user wants the frame statistics to be retrieved.

Parameters

start	The starting point of the period.
stop	The end point of the period.
points	The number of points desired to be returned.

6.20.3.45 void cavapa_gui::MainWindow::updateCalculation (const FrameStats & *statistics*) [override], [virtual]

Updates the calculation data.

Parameters

statistics	The frame calculation statistics.

Implements cavapa_gui::UserInterface.

6.20.3.46 void cavapa_gui::MainWindow::updateFrame (const FrameCapture & *frame*) [override], [virtual]

Updates the frame for a source.

Parameters

frame	The FrameCapture structure containing the new frame and the source ID.

Implements cavapa_gui::UserInterface.

6.20.3.47 void cavapa_gui::MainWindow::updateGeneralSettings (const GeneralSettings & *settings*) [override], [virtual]

Updates the general settings for the interface.

Parameters

settings	The object describing the new settings.

Implements cavapa_gui::UserInterface.

6.20.3.48 void cavapa_gui::MainWindow::updateMarkers (const std::vector< GraphMarker > & *markers*) [override], [virtual]

Updates the graph markers.

Parameters

markers	A vector containing the markers.

Implements cavapa_gui::UserInterface.

6.20.3.49 void cavapa_gui::MainWindow::updateMetadata (const Metadata & *metadata*) [override], [virtual]

Updates the metadata to the interface.

Parameters

metadata	The Metadata object.

Implements cavapa_gui::UserInterface.

6.20.3.50 void cavapa_gui::MainWindow::updateSourceSettings (SourceID *source*, const SourceSettings & *settings*) [override], [virtual]

Updates the settings of the speficied source.

Parameters

source	The ID of the source whose settings should be updated.

settings	The new settings for the source.

Implements cavapa_gui::UserInterface.

6.20.3.51 void cavapa_gui::MainWindow::updateStatistics (const std::vector< FrameStats > & *stats*) [override], [virtual]

Updates the interface with the frame statistics from a certain period.

Parameters

stats	Statistical information.

Implements cavapa_gui::UserInterface.

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

- gui/mainwindow.h
- gui/mainwindow.cpp

6.21 cavapa_gui::Marker Struct Reference

The structure holds the properties of a single marker on the graph.

`#include <common.h>`

Public Member Functions

- bool operator< (const Marker &str) const

  Operator for sorting by time with std::sort.

Public Attributes

- bool active

  Defines if the marker is active or not.
- FrameTime pos

  The position of the marker.
- std::string text

  The text of the marker.

6.21.1 Detailed Description

The structure holds the properties of a single marker on the graph.

Author

Joel Kivelä

6.21.2 Member Function Documentation

6.21.2.1 bool cavapa_gui::Marker::operator< (const Marker & *str*) const [inline]

Operator for sorting by time with std::sort.

Parameters

str	The marker to compare against.

Returns

True if the current Marker is before the second operator or false otherwise.

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

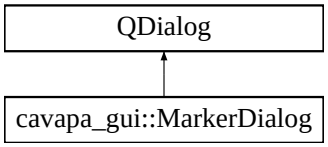
- common.h

6.22 cavapa_gui::MarkerDialog Class Reference

Class for marker creation dialog of the graphwidget.

`#include <markerdialog.h>`

Inheritance diagram for cavapa_gui::MarkerDialog:



Signals

- void sendMarkerNameAndTime (std::string text, QDateTime time)

  Sends new marker text and time position to parent.
- void sendSelectedText (QString typedtext)

  Sends selected text to parent.
- void sendMarkerPosition (QDateTime time)

  Sends marker position to parent.
- void removeFromHistory (int i)

  Notifies parent to remove selected text item from marker history.

Public Member Functions

- MarkerDialog (QWidget *parent=0)

  Constructs the marker dialog.
- void setMarkerHistoryAndPositionTime (std::vector< std::string > history, QDateTime time)

  Sets marker history and the selected time point for new marker for marker dialog.

6.22.1 Detailed Description

Class for marker creation dialog of the graphwidget.

Offers list of previously added marker texts and a time editor for marker time position fine tuning.

Author

Joel Kivelä

6.22.2 Constructor & Destructor Documentation

6.22.2.1 cavapa_gui::MarkerDialog::MarkerDialog (QWidget * *parent* = 0) [explicit]

Constructs the marker dialog.

Parameters

parent	Qt default parent.

6.22.3 Member Function Documentation

6.22.3.1 void cavapa_gui::MarkerDialog::removeFromHistory (int *i*) [signal]

Notifies parent to remove selected text item from marker history.

Parameters

i	Index of text to be deleted.

6.22.3.2 void cavapa_gui::MarkerDialog::sendMarkerNameAndTime (std::string *text*, QDateTime *time*) [signal]

Sends new marker text and time position to parent.

Parameters

text	Marker text in string.
time	Time position in QDateTime.

6.22.3.3 void cavapa_gui::MarkerDialog::sendMarkerPosition (QDateTime *time*) [signal]

Sends marker position to parent.

Parameters

time	Time position in QDateTime.

6.22.3.4 void cavapa_gui::MarkerDialog::sendSelectedText (QString *typedtext*) [signal]

Sends selected text to parent.

Parameters

typedtext	Typeded text in QString.

6.22.3.5 void cavapa_gui::MarkerDialog::setMarkerHistoryAndPositionTime (std::vector< std::string > *history*, QDateTime *time*)

Sets marker history and the selected time point for new marker for marker dialog.

Parameters

history	The marker text history as vector array of strings.
time	The selected time point from the graph as QDateTime.

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

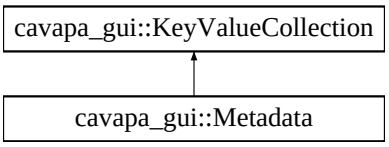
- gui/markerdialog.h
- gui/markerdialog.cpp

6.23 cavapa_gui::Metadata Class Reference

Represents the metadata associated with a measurement.

`#include <metadata.h>`

Inheritance diagram for cavapa_gui::Metadata:



Public Member Functions

- Metadata ()

  Constructs a new Metadata object with empty values.
- void addSource (SourceID id, SourceType type, const SourceSettings &settings, FrameTime startOffset=0)

  Adds a new source to the metadata.
- void addVideo (SourceID id, const std::string &path)

  Adds a new recorded video file for the specified source.
- void clearSources ()

  Removes all sources from the Metadata object.
- std::vector< CalibrationPoint > getCalibrationPoints (SourceID source)

  Gets the calibration points of the specified source.
- QDir getContainingDirectory () const

  Gets the path of the directory specifying where the metadata file is located.
- std::vector< GraphMarker > getMarkers () const

  Gets the markers associated with the metadata.
- std::vector< SourceID > getSources () const

  Gets the ids of the sources associated with the metadata.
- SourceSettings getSourceSettings (SourceID source) const

  Gets the source settings for the specified source from the metadata.
- FrameTime getStartOffset (SourceID source) const

  Gets the start offset for the specified source.
- int getVideoCount () const

  Gets the total video count of all sources.
- std::vector< std::string > getVideos (SourceID id)

  Gets the recorded videos associated with the specified source.
- bool readFromFile (const std::string &path)

  Reads the metadata information from the specified XML file.
- void setCalibrationPoints (const std::vector< CalibrationPoint > &points)

  Sets the calibration points for the metadata.
- void setCalibrationPoints (SourceID id, const std::vector< CalibrationPoint > &points)

  Sets the calibration points for the specified source.
- void setMarkers (const std::vector< GraphMarker > &markers)

  Sets the graph markers for the metadata.
- bool writeToFile () const

  Writes the metadata to the default file.
- bool writeToFile (const std::string &path) const

  Writes the metadata to the specified file.

Additional Inherited Members

6.23.1 Detailed Description

Represents the metadata associated with a measurement.

Provides methods for reading and writing the metadata from and to an XML file.

Author

Mika Lehtinen

6.23.2 Member Function Documentation

6.23.2.1 void cavapa_gui::Metadata::addSource (SourceID *id,* SourceType *type,* const SourceSettings & *settings,* FrameTime *startOffset* = 0)

Adds a new source to the metadata.

Parameters

id	The id of the source.
type	The type of the source.
settings	The settings for the source.
startOffset	The start offset of the source. This is only applicable to video files.

6.23.2.2 void cavapa_gui::Metadata::addVideo (SourceID *id,* const std::string & *path*)

Adds a new recorded video file for the specified source.

Parameters

id	The id of the source.
path	The video file to be added.

6.23.2.3 std::vector< CalibrationPoint > cavapa_gui::Metadata::getCalibrationPoints (SourceID *source*)

Gets the calibration points of the specified source.

Parameters

source	The id of the source.

Returns

The collection of the calibration points of the source.

6.23.2.4 QDir cavapa_gui::Metadata::getContainingDirectory ( ) const

Gets the path of the directory specifying where the metadata file is located.

Returns

The path of the directory.

6.23.2.5 std::vector< GraphMarker > cavapa_gui::Metadata::getMarkers ( ) const

Gets the markers associated with the metadata.

Returns

The markers.

6.23.2.6 std::vector< SourceID > cavapa_gui::Metadata::getSources ( ) const

Gets the ids of the sources associated with the metadata.

Returns

The ids of the sources.

6.23.2.7 SourceSettings cavapa_gui::Metadata::getSourceSettings (SourceID *source*) const

Gets the source settings for the specified source from the metadata.

Parameters

source	The id of the source.

Returns

The source settings.

6.23.2.8 FrameTime cavapa_gui::Metadata::getStartOffset (SourceID *source*) const

Gets the start offset for the specified source.

Parameters

source	The id of the source.

Returns

The start offset for the source.

6.23.2.9 int cavapa_gui::Metadata::getVideoCount ( ) const

Gets the total video count of all sources.

Returns

The video count of all sources.

6.23.2.10 std::vector< std::string > cavapa_gui::Metadata::getVideos (SourceID *id*)

Gets the recorded videos associated with the specified source.

Parameters

id	The id of the source whose videos will be returned.

Returns

The file names of the videos.

6.23.2.11 bool cavapa_gui::Metadata::readFromFile (const std::string & *path*)

Reads the metadata information from the specified XML file.

Parameters

path	The path of the XML file to be read.

Returns

True if the file was read successfully, false otherwise.

6.23.2.12 void cavapa_gui::Metadata::setCalibrationPoints (const std::vector< CalibrationPoint > & *points*)

Sets the calibration points for the metadata.

Parameters

points	The vector containing the points.

6.23.2.13 void cavapa_gui::Metadata::setCalibrationPoints (SourceID *id,* const std::vector< CalibrationPoint > & *points*)

Sets the calibration points for the specified source.

Parameters

id	The id of the source.
points	The calibration points for the source.

6.23.2.14 void cavapa_gui::Metadata::setMarkers (const std::vector< GraphMarker > & *markers*)

Sets the graph markers for the metadata.

Parameters

markers	The vector containing the markers.

6.23.2.15 bool cavapa_gui::Metadata::writeToFile ( ) const

Writes the metadata to the default file.

Returns

True if the data was successfully written to the file, false otherwise.

6.23.2.16 bool cavapa_gui::Metadata::writeToFile (const std::string & *path*) const

Writes the metadata to the specified file.

Parameters

path	The location of the file.

Returns

True if the data was successfully written to the file, false otherwise.

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

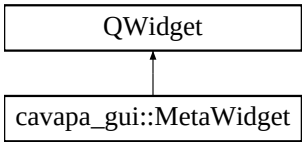
- metadata.h
- metadata.cpp

6.24 cavapa_gui::MetaWidget Class Reference

Represents the widget containing all metadata related to a measurement.

`#include <metawidget.h>`

Inheritance diagram for cavapa_gui::MetaWidget:



Public Member Functions

- MetaWidget (QWidget *parent=0)
  Constructs a new MetaWidget having the specified parent.
- Metadata getMetadata ()
  Gets the Metadata object representing the currently input data.
- void setCanSaveVideos (bool enable)
  Sets whether the user can choose to save videos.
- void setReadOnly (bool flag)
  Sets whether the input elements are read-only.
- void setDefaultDirectory (const QString &path)
  Sets the default directory for the widget.
- void setMetadata (const Metadata &metadata)
  Sets the metadata for the widget.

6.24.1 Detailed Description

Represents the widget containing all metadata related to a measurement.

Author

Mika Lehtinen

6.24.2 Constructor & Destructor Documentation

6.24.2.1 cavapa_gui::MetaWidget::MetaWidget (QWidget * *parent* = 0) [explicit]

Constructs a new MetaWidget having the specified parent.

Parameters

parent	The parent widget.

6.24.3 Member Function Documentation

6.24.3.1 Metadata cavapa_gui::MetaWidget::getMetadata ( )

Gets the Metadata object representing the currently input data.

Returns

The Metadata object.

6.24.3.2 void cavapa_gui::MetaWidget::setCanSaveVideos (bool *enable*)

Sets whether the user can choose to save videos.

Parameters

enable	True if the user can choose to save videos, false otherwise.

6.24.3.3 void cavapa_gui::MetaWidget::setDefaultDirectory (const QString & *path*)

Sets the default directory for the widget.

Parameters

path	The directory path.

6.24.3.4 void cavapa_gui::MetaWidget::setMetadata (const Metadata & *metadata*)

Sets the metadata for the widget.

Parameters

metadata	The Metadata object.

6.24.3.5 void cavapa_gui::MetaWidget::setReadOnly (bool *flag*)

Sets whether the input elements are read-only.

Parameters

flag	True if the input elements should be in read-only mode, false otherwise.

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

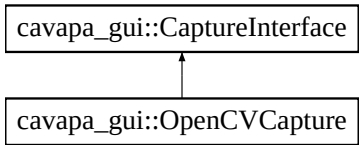
- gui/metawidget.h
- gui/metawidget.cpp

6.25 cavapa_gui::OpenCVCapture Class Reference

The OpenCV capturing class.

`#include <opencvcapture.h>`

Inheritance diagram for cavapa_gui::OpenCVCapture:



Public Member Functions

- DISALLOW_COPY_AND_ASSIGN (OpenCVCapture)

  Copy and assign of the class is not allowed.
- double getBarrelCorrection () override

  Retrieves the barrel correction applied to the source.
- FrameCapture getBuffered () override

  Retrieves the currently buffered frame.
- std::string getDescription () const override

  Retrieves the description of the device.
- int getDeviceID () const override

  Returns the ID number the device was initialized with.
- double getFramerate () const override

  Retrieves the capture device framerate.
- int getMissedFrames () const override

  Retrieves the logged frame misses.
- FrameCapture getNext (bool skip, bool ∗error) override

  Retrieves the next frame.
- std::string getPath () const override

  Retrieves the path given during the initialization.
- double getProperty (int propId)

  Gets the OpenCV specific source properties.
- cv::Size getResolution () const override

  Retrieves the capture device resolution.
- int getRetrievedFrames () const override

  Retrieves the total logged frames.
- INTERFACE_TYPE getType () const overridefinal

  Retrieves the type of the device.
- bool hasNew () override

  Checks whether the device has a new image to be retrieved or not.
- bool isOpen () const override

  Checks whether the source has been initialized or not.
- bool open (int index)

  Opens OpenCV VideoCapture for the hardware device.
- bool open (const std::string &path) override

  Opens the video file or the stream.
- void release ()

  Forces the release of the worker thread.
- void resetStats () override

  Resets the statistical counters.
- void setBarrelCorrection (double amount) override

Sets the amount of the barrel correction to be applied.

- bool setProperty (int propId, double value)

  Sets OpenCV specific source properties.

- bool setResolution (const cv::Size &new_size) override

  Sets the capturing device resolution.

Additional Inherited Members

6.25.1 Detailed Description

The OpenCV capturing class.

The class provides OpenCV-powered VideoCapture. It can be used for hardware cameras, network streams or video files. The class multithreads the actual retrievals and offers some buffering for possible frame misses and device lag time. The class does not handle HTTP image sources as current version of OpenCV (2.4.9) is unable to retrieve them multiple times.

OpenCV allows limited access to resolution and framerate changing on most device sources. Unfortunately, it seriously lacks any kind of error catching method. Special statistical values are used to catch missed frames and other problems.

As an extension of the CaptureInterface class, the class is initialized with open() that opens the connection to the actual capturing device. Once connection has been established, images can be retrieved with getNext(). Some OpenCV devices also support the change of image size and framerate.

Author

Petri Partanen

6.25.2 Member Function Documentation

6.25.2.1 double cavapa_gui::OpenCVCapture::getBarrelCorrection ( ) [override], [virtual]

Retrieves the barrel correction applied to the source.

Returns

The amount of the barrel correction.

Implements cavapa_gui::CaptureInterface.

6.25.2.2 FrameCapture cavapa_gui::OpenCVCapture::getBuffered ( ) [override], [virtual]

Retrieves the currently buffered frame.

Returns

The buffered frame.

Remarks

This does not signal for a new image fetch from the device.

Implements cavapa_gui::CaptureInterface.

6.25.2.3 string cavapa_gui::OpenCVCapture::getDescription ( ) const [override], [virtual]

Retrieves the description of the device.

Returns

For the devices initialized with HW index, this will be the name of the HW device. For other devices this is the path given during initialization.

Implements cavapa_gui::CaptureInterface.

6.25.2.4 int cavapa_gui::OpenCVCapture::getDeviceID ( ) const [inline], [override], [virtual]

Returns the ID number the device was initialized with.

Returns

The device ID used in the initialization or UNKNOWN_DEVICE if not specified.

Implements cavapa_gui::CaptureInterface.

6.25.2.5 double cavapa_gui::OpenCVCapture::getFramerate ( ) const [inline], [override], [virtual]

Retrieves the capture device framerate.

Returns

The current framerate.

Implements cavapa_gui::CaptureInterface.

6.25.2.6 int cavapa_gui::OpenCVCapture::getMissedFrames ( ) const [inline], [override], [virtual]

Retrieves the logged frame misses.

A frame miss occurs everytime when getNext() is called and the previous retrieval was still ongoing.

Returns

The count of the missed frames.

Implements cavapa_gui::CaptureInterface.

6.25.2.7 FrameCapture cavapa_gui::OpenCVCapture::getNext (bool *skip*, bool * *error*) [override], [virtual]

Retrieves the next frame.

Basically it returns the currently buffered frame and signals the device to grab a new image for next call to the function. If the device has not returned from previous image grab the call is recorded as a missed frame.

Parameters

skip	Specifies if the frame can be skipped. When set to true, this will force the main thread to wait for a new image from the device. This can result in a huge lag if the device is not capable of retrieving images fast enough.

error	Will receive an error message if something went wrong.

Returns

The next frame with the reference to the image! You must make a clone if you wish to alter it!

Implements cavapa_gui::CaptureInterface.

6.25.2.8 std::string cavapa_gui::OpenCVCapture::getPath ( ) const [inline], [override], [virtual]

Retrieves the path given during the initialization.

Returns

The path of the device or an empty string if not found.

Implements cavapa_gui::CaptureInterface.

6.25.2.9 double cavapa_gui::OpenCVCapture::getProperty (int *propId*)

Gets the OpenCV specific source properties.

See OpenCV VideoCapture class for more info.

Parameters

propId	VideoCapture::get property.

Returns

The specified property or 0.0 if the source is not open.

Remarks

Accessing any of the source properties can produce lag if the source is running as we have to do mutex locking due to multithreading.

6.25.2.10 cv::Size cavapa_gui::OpenCVCapture::getResolution ( ) const [inline], [override], [virtual]

Retrieves the capture device resolution.

Returns

The resolution of the device.

Implements cavapa_gui::CaptureInterface.

6.25.2.11 int cavapa_gui::OpenCVCapture::getRetrievedFrames ( ) const [inline], [override], [virtual]

Retrieves the total logged frames.

Basically, this is a counter on how many times getNext() has been called.

Returns

The count of the total frames returned.

Implements cavapa_gui::CaptureInterface.

6.25.2.12 INTERFACE_TYPE cavapa_gui::OpenCVCapture::getType ( ) const [inline], [final], [override], [virtual]

Retrieves the type of the device.

Returns

The interface type.

Implements cavapa_gui::CaptureInterface.

6.25.2.13 bool cavapa_gui::OpenCVCapture::hasNew ( ) [override], [virtual]

Checks whether the device has a new image to be retrieved or not.

This checks if the FrameTime on the buffered image differs from what was retrieved on the last get().

Returns

True if a new image available or false otherwise.

Remarks

This is protected by a mutex and should not be used when time is critical. Use it only when you know that any lag produced by this is not going to be a problem.

Implements cavapa_gui::CaptureInterface.

6.25.2.14 bool cavapa_gui::OpenCVCapture::isOpen ( ) const [inline], [override], [virtual]

Checks whether the source has been initialized or not.

Returns

True if it is initialized or false otherwise.

Remarks

Because OpenCV fails to provide functionality to check for unplugged devices and other lost connections, this might not give any real indication on device's actual state. It can be however used to see if the initialization was successful.

Implements cavapa_gui::CaptureInterface.

6.25.2.15 bool cavapa_gui::OpenCVCapture::open (int *index*)

Opens OpenCV VideoCapture for the hardware device.

Parameters

index	The device index number.

Returns

True if the device was opened successfully or false otherwise.

6.25.2.16 bool cavapa_gui::OpenCVCapture::open (const std::string & *path*) [override], [virtual]

Opens the video file or the stream.

Parameters

path	The path of the video file or the network stream.

Returns

True if it was opened successfully or false otherwise.

Implements cavapa_gui::CaptureInterface.

6.25.2.17 void cavapa_gui::OpenCVCapture::release ( )

Forces the release of the worker thread.

This is used when multiple video files are combined as a one source of video. When the particular video file is currently not being accessed, the worker thread can be released to free up system's resources. The worker thread is again automatically started when the getNext() function is called.

Remarks

Releasing the thread can produce lag as the main thread has to wait for it to finish.

6.25.2.18 void cavapa_gui::OpenCVCapture::resetStats ( ) [inline], [override], [virtual]

Resets the statistical counters.

This affects the missed and total frame counters.

Implements cavapa_gui::CaptureInterface.

6.25.2.19 void cavapa_gui::OpenCVCapture::setBarrelCorrection (double *amount*) [override], [virtual]

Sets the amount of the barrel correction to be applied.

Parameters

amount	The barrel effect. 0.0 means none.

Implements cavapa_gui::CaptureInterface.

6.25.2.20 bool cavapa_gui::OpenCVCapture::setProperty (int *propId*, double *value*)

Sets OpenCV specific source properties.

See OpenCV VideoCapture for more info.

Parameters

propId	VideoCapture::set property. You should NOT change resolution or framerate directly by using this.
value	The new value of the property.

Returns

Unknown, undocumented in VideoCapture::set. It will be false at least if the source is not open.

Remarks

Accessing the properties while the source is running can produce lag as we have to wait for the source to finish retrieving images first.

6.25.2.21 bool cavapa_gui::OpenCVCapture::setResolution (const cv::Size & *new_size*) [override], [virtual]

Sets the capturing device resolution.

Parameters

new_size	The size to be set.

Returns

True if at least one of the dimension was set or false otherwise.

Implements cavapa_gui::CaptureInterface.

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

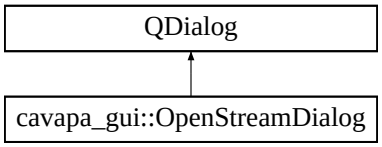
- source/opencvcapture.h
- source/opencvcapture.cpp

6.26 cavapa_gui::OpenStreamDialog Class Reference

Represents a dialog for opening a stream.

`#include <openstreamdialog.h>`

Inheritance diagram for cavapa_gui::OpenStreamDialog:



Public Member Functions

- OpenStreamDialog (QWidget *parent=0)

  Constructs a new OpenStreamDialog having the specified parent.
- std::string getSelectedUrl () const

  Gets the selected stream URL.
- void setRecentSources (const std::list< QString > &sources)

  Sets the list of the recent sources for the dialog.

6.26.1 Detailed Description

Represents a dialog for opening a stream.

Author

Mika Lehtinen

6.26.2 Constructor & Destructor Documentation

6.26.2.1 cavapa_gui::OpenStreamDialog::OpenStreamDialog (QWidget * *parent* = 0) [explicit]

Constructs a new OpenStreamDialog having the specified parent.

Parameters

parent	The parent widget.

6.26.3 Member Function Documentation

6.26.3.1 std::string cavapa_gui::OpenStreamDialog::getSelectedUrl ( ) const

Gets the selected stream URL.

Returns

The selected stream URL as a string.

6.26.3.2 void cavapa_gui::OpenStreamDialog::setRecentSources (const std::list< QString > & *sources*)

Sets the list of the recent sources for the dialog.

Parameters

sources	The list of sources.

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

- gui/openstreamdialog.h
- gui/openstreamdialog.cpp

6.27 cavapa_gui::Recorder Class Reference

The buffered recorder class creates the video file from the images.

`#include <recorder.h>`

Public Member Functions

- DISALLOW_COPY_AND_ASSIGN (Recorder)
  Copy and assign of the class is not allowed.
- unsigned int countBuffered ()
  Counts the currently buffered images.
- unsigned int getMissedFrames ()
  Returns the count of the missed frames.
- unsigned int getRecordedFrames ()
  Returns the recorded frames.
- bool isOpen ()
  Returns the initialization status of the recorder.
- bool isTerminated ()
  Retrieves the recorder status.
- bool open (const std::string &filename, int fourcc, double fps, cv::Size frameSize, bool isColor=true)
  Opens the video recorder.
- RecorderStats push (const cv::Mat &image)
  Pushes the new image to the buffer.
- bool repush ()
  Repushes the last image again to the buffer.
- void stop ()
  Softly stops the recording.

Static Public Member Functions

- static void detach (std::unique_ptr< Recorder > &&ptr)

  Detaches the recorder.

6.27.1 Detailed Description

The buffered recorder class creates the video file from the images.

This is basically just a wrapper around cv::VideoWriter so that there is a simple buffer that allows multithreading. The class should only be used for one video file recording and then deleted. Once the recorder is succesfully opened, it cannot be re-opened even after termination.

When the recorder is to be opened, it is provided with all necessary information for the video file creation. The images are added to the buffer with push(). Recorder will write the contents of the buffer to the video file with the initialized parameters, and will continue to do so until it is terminated. Recorder can be stopped with stop() in order to finish writing the buffer contents. This allows it to flush the buffer contents and close the file properly.

Method isTerminated() can be used to check when Recorder has finished flushing the buffer after stop() signal. If Recorder object is destroyed before termination is successful, it will also destroy the rest of the buffer contents, thus leading into lost video frames. The destruction of Recorder should not leave video files corrupted, as it tries to close the files properly.

Author

Petri Partanen

Remarks

All public methods are thread-safe.

6.27.2 Member Function Documentation

6.27.2.1 unsigned int cavapa_gui::Recorder::countBuffered ( )

Counts the currently buffered images.

Returns

The number of images in the buffer.

6.27.2.2 static void cavapa_gui::Recorder::detach (std::unique_ptr< Recorder > && *ptr*) [inline], [static]

Detaches the recorder.

It adds the recorder to a cleanup list that holds unfinished records, and let's them flush their buffer peacefully. The recorders on this list are destroyed on flushDetached() call, unless they have not yet terminated.

Parameters

ptr	The recorder to be detached.

Remarks

Recorder should be stopped at this point already.

6.27.2.3 unsigned int cavapa_gui::Recorder::getMissedFrames ( )

Returns the count of the missed frames.

The missed frames tell the amount of the frames that have been lost due to file writing errors. Most likely the file has no longer been open when the frames were supposed to be written.

Returns

The missed frames.

Remarks

Untested. Not sure if OpenCV actually even fails when the file is no longer open.

6.27.2.4 unsigned int cavapa_gui::Recorder::getRecordedFrames ( )

Returns the recorded frames.

Returns

The succesfully recorded frames.

Remarks

The images currently in the buffer are not counted.

6.27.2.5 bool cavapa_gui::Recorder::isOpen ( ) [inline]

Returns the initialization status of the recorder.

Returns

True if the file writer was opened, false otherwise.

Remarks

This does *NOT* change if the file is closed!

6.27.2.6 bool cavapa_gui::Recorder::isTerminated ( )

Retrieves the recorder status.

Returns

True if the recorder has finished writing frames and can be killed without destroying any unwritten frames.

6.27.2.7 bool cavapa_gui::Recorder::open (const std::string & *filename,* int *fourcc,* double *fps,* cv::Size *frameSize,* bool *isColor* = `true`)

Opens the video recorder.

The parameters are the same as in cv::VideoWriter, and are passed directly to cv::VideoWriter::open.

Parameters

filename	The name of the output video file.
fourcc	The 4-character code of codec.
fps	The framerate of the created video.
frameSize	The size of the video frames.
isColor	If it is not false, the encoder will expect and encode color frames.

Returns

True if the file was opened successfully or false otherwise.

6.27.2.8 RecorderStats cavapa_gui::Recorder::push (const cv::Mat & *image*)

Pushes the new image to the buffer.

Parameters

image	The new OpenCV image. All the images buffered must match in size, otherwise the video file gets broken.

Returns

The current statistics of the Recorder.

6.27.2.9 bool cavapa_gui::Recorder::repush ( )

Repushes the last image again to the buffer.

Returns

True if repushing was done or false otherwise.

Remarks

Does nothing if no image exists.

6.27.2.10 void cavapa_gui::Recorder::stop ( )

Softly stops the recording.

This allows the recorder to write the rest of the frames to the buffer. Recorder cannot be restarted after this. If the recorder is deleted before the buffer has finished writing, the frames will be lost forever. Use terminated() to see when the buffer has finished flushing.

Remarks

In order to leave Recorder to peacefully write rest of the frames, it should be detached().

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

- source/recorder.h
- source/recorder.cpp

6.28 cavapa_gui::RecorderStats Struct Reference

Statistics structure used for Recorder.

`#include <recorder.h>`

Public Attributes

- unsigned int missed_frames

  The number of the frames that have missed the recording.
- unsigned int written_frames

  The frames that have been written by the recorder.

6.28.1 Detailed Description

Statistics structure used for Recorder.

Author

Petri Partanen

6.28.2 Member Data Documentation

6.28.2.1 unsigned int cavapa_gui::RecorderStats::missed_frames

The number of the frames that have missed the recording.

Remarks

This indicates serious error with the recorder.

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

- source/recorder.h

6.29 cavapa_gui::Results Class Reference

The class holds the calculation results for each frame.

`#include <results.h>`

Public Member Functions

- FrameStats addFrame (FrameTime time, const std::vector< cavapa::Sightings > &sightings)

  Adds the new frame to the results.
- void clear ()

  Deletes all the stored calculation results.
- bool empty () const

  Checks if the are no results.
- bool exportToCSV (const ExportOptions &options) const

  Exports the results to the CSV file.
- unsigned int getCount () const

Returns the number of the frames saved.

- FrameTime getFirstFrameTime () const

  Retrieves the time of the first frame in Results.

- int getFrameNumber (FrameTime time) const

  Returns the frame number for the given FrameTime.

- std::vector< cavapa::Sightings > getSightings (FrameTime time) const

  Retrieves the sighting information from the frame.

- std::vector< FrameStats > getStatistics (FrameTime start, FrameTime stop, int points) const

  Returns the calculation statistics from the given period.

- bool load (const std::string &path)

  Loads the calculation results from the file.

- bool save (const std::string &path) const

  Saves the calculation results to the file.

6.29.1 Detailed Description

The class holds the calculation results for each frame.

It keeps the results in order and provides export, save, load and averaging functions for the other parts of the program.

The current results can be removed with clear(). New frames can be inserted with addFrame() which calculates the total activity and the sighting count for the frame.

The frame specific results can be retrieved with getStatistics() or they can be exported to the CSV file with exportToCSV(). The Results and sightings can also be saved to the file and loaded later.

Author

Petri Partanen

Remarks

The save file format is as follows:

```
frame_count (i) : uint32_t
[FRAME 1]
    frame_stats   : FrameStats
    cameras (n)   : uint32_t
    sightings (m) : uint32_t

    [CAMERA 1]
        sighting_count (j) : uint32_t
    [CAMERA 2]
        sighting_count     : uint32_t
    [CAMERA 3]
    ...
    [CAMERA n]

    [CAMERA 1 SIGHTINGS]
        [SIGHTING 1]
            sighting      : SightingFixed
        [SIGHTING 2]
            sighting      : SightingFixed
        [SIGHTING 3]
        ...
        [SIGHTING j]
    [CAMERA 2 SIGHTINGS]
    ...
    [SIGHTINGS m]
[FRAME 2]
...
[FRAME i]
```

6.29.2 Member Function Documentation

6.29.2.1 FrameStats cavapa_gui::Results::addFrame (FrameTime *time*, const std::vector< cavapa::Sightings > & *sightings*)

Adds the new frame to the results.

This also calculates the frame's statistics and returns them.

Parameters

time	A time of the frame.
sightings	The sightings on the frame.

Returns

The statistics of the new frame.

Remarks

Currently Results class does not sort the frames in any way, so the frame's time must have a later timestamp than the previous frame.

6.29.2.2 bool cavapa_gui::Results::empty ( ) const [inline]

Checks if the are no results.

Returns

True if no results were saved or false otherwise.

6.29.2.3 bool cavapa_gui::Results::exportToCSV (const ExportOptions & *options*) const

Exports the results to the CSV file.

It exports the basic frame information to the file in a CSV format. Any old data will be overwritten, unless APPEND flag is set.

Parameters

options	The options for the export.

Returns

True if the file was written succesfully or false otherwise.

6.29.2.4 unsigned int cavapa_gui::Results::getCount ( ) const [inline]

Returns the number of the frames saved.

Returns

The total frame count.

6.29.2.5 FrameTime cavapa_gui::Results::getFirstFrameTime ( ) const [inline]

Retrieves the time of the first frame in Results.

Returns

The frametime of the first frame. It will return 0 if none exists.

6.29.2.6 int cavapa_gui::Results::getFrameNumber (FrameTime *time*) const

Returns the frame number for the given FrameTime.

Parameters

time	FrameTime to look for.

Returns

The frame number for the given FrameTime or a negative value if failed.

6.29.2.7 std::vector< Sightings > cavapa_gui::Results::getSightings (FrameTime *time*) const

Retrieves the sighting information from the frame.

Parameters

time	The frame time.

Returns

The sightings from the frame.

TODO: Actually return something.

6.29.2.8 vector< FrameStats > cavapa_gui::Results::getStatistics (FrameTime *start*, FrameTime *stop*, int *points*) const

Returns the calculation statistics from the given period.

It can be used to return the frame statistics from a given period. The average values are calculated automatically from between the returned period points.

Parameters

start	The period start time. If 0, the period will start from the first possible frame.
stop	The period end time. If 0, the period will end at the last possible frame.
points	The number of points to be returned from the specified period.

Returns

The calculation results from a given period. There will be from 0 to 'points' number of FrameStats returned.

6.29.2.9 bool cavapa_gui::Results::load (const std::string & *path*)

Loads the calculation results from the file.

Parameters

path	The full file path.

Returns

True if loading was successful or false otherwise.

Remarks

The old results will be destroyed.

6.29.2.10 bool cavapa_gui::Results::save (const std::string & *path*) const

Saves the calculation results to the file.

Parameters

path	The full file path.

Returns

True if saving was successful or false otherwise.

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

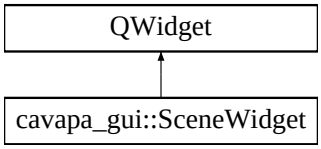
- results.h
- results.cpp

6.30 cavapa_gui::SceneWidget Class Reference

A widget that holds the output and media controls of a video source.

`#include <scenewidget.h>`

Inheritance diagram for cavapa_gui::SceneWidget:



Public Slots

- void setCameraXRot (int value)

  Sets the camera rotation around X axis.
- void setCameraYRot (int value)

  Sets the camera rotation around Y axis.
- void setCameraZRot (int value)

  Sets the camera rotation around Z axis.
- void setCameraHeight (int value)

  Sets the camera height.
- void setCameraFov (int value)

Sets the field of view of the camera.

- void setDrawGridEnabled (bool enabled)

  Sets whether the grid is drawn or not.
- void setDrawCalibrationPointsEnabled (bool enabled)

  Sets whether the calibration points are drawn or not.
- void disableForConfiguration ()

  Changes the frame shadow to raised and enables the "Set active" button.
- void activateForConfiguration ()

  Changes the frame shadow to Sunken and disables the "Set active" button.

Signals

- void closeRequested (SourceID sender)

  The signal is emitted when the user want's to close this source widget.
- void requestedActivation (SceneWidget *sender)

  The signal is emitted when user wants to activate this source widget for calibration.
- void playRequested (SourceID sender)

  The signal is emitted when the user wants to play the video.
- void pauseRequested (SourceID sender)

  The signal is emitted when the user wants to pause the video.
- void seekBarValueChanged (SourceID sender, FrameTime value)

  The signal is emitted when video seek bar value has changed.

Public Member Functions

- SceneWidget (SourceID sid, SourceType type, const QString &description, QWidget *parent=0)

  Constructs a new SceneWidget with the specified information.
- int getCameraXRot ()

  Returns the camera rotation around X axis.
- int getCameraYRot ()

  Returns the camera rotation around Y axis.
- int getCameraZRot ()

  Returns the camera rotation around Z axis.
- int getCameraHeight ()

  Returns the camera height.
- int getCameraFov ()

  Returns the field of view of the camera.
- SourceID getSourceID () const

  Returns the ID number of the source that the scene represents.
- void updateFrame (const FrameCapture &frame)

  Updates the scene image.
- void updateSettings (const SourceSettings &settings)

  Updates the scene settings.
- cavapa::camera getCameraSettings ()

  Gets the camera settings.
- std::vector
  < cavapa::calibration_point > getCalibrationPoints ()

  Gets the calibration points.
- void setCalibrationPoints (std::vector< cavapa::calibration_point > points)

  Sets the calibration points.

- void setAddCalibrationPointsEnabled (bool enabled)

  Tells the SceneWidget whether adding calibration points is possible.
- void setSeekBarValue (FrameTime value)

  Sets the seek bar to a new value.
- void showPlayButton ()

  Shows the play video button (pause button is hidden).
- void showPauseButton ()

  Shows the pause video button (play button is hidden).
- void setVideoLength (FrameTime frameTime)

  Sets video length for the seek bar.

6.30.1 Detailed Description

A widget that holds the output and media controls of a video source.

Purpose of this class is to provide easy way to add or remove the scenes and to intermediate calibration data between the Main Window and the SourceScene.

Author

Oskari Leppäaho

6.30.2 Constructor & Destructor Documentation

6.30.2.1 cavapa_gui::SceneWidget::SceneWidget (SourceID *sid*, SourceType *type*, const QString & *description*, QWidget * *parent* = `0`)

Constructs a new SceneWidget with the specified information.

Parameters

sid	The id of the source.
type	The type of the source.
description	The description of the source.
parent	The parent widget.

6.30.3 Member Function Documentation

6.30.3.1 void cavapa_gui::SceneWidget::closeRequested (SourceID *sender*) `[signal]`

The signal is emitted when the user want's to close this source widget.

Parameters

sender	Source ID-number.

6.30.3.2 std::vector< cavapa::calibration_point > cavapa_gui::SceneWidget::getCalibrationPoints ( )

Gets the calibration points.

Returns

A vector of calibration points

6.30.3.3 int cavapa_gui::SceneWidget::getCameraFov ( ) [inline]

Returns the field of view of the camera.

The value is specified in 0.1 degrees.

Returns

The field of view of the camera in 0.1 degrees.

6.30.3.4 int cavapa_gui::SceneWidget::getCameraHeight ( ) [inline]

Returns the camera height.

The value is specified in 0.01 meters.

Returns

The camera height specified in 0.01 meters.

6.30.3.5 cavapa::camera cavapa_gui::SceneWidget::getCameraSettings ( )

Gets the camera settings.

Returns

The camera settings

6.30.3.6 int cavapa_gui::SceneWidget::getCameraXRot ( ) [inline]

Returns the camera rotation around X axis.

The value is specified in 0.1 degrees.

Returns

The camera rotation around X axis specified in 0.1 degrees.

6.30.3.7 int cavapa_gui::SceneWidget::getCameraYRot ( ) [inline]

Returns the camera rotation around Y axis.

The value is specified in 0.1 degrees.

Returns

The camera rotation around Y axis specified in 0.1 degrees.

6.30.3.8 int cavapa_gui::SceneWidget::getCameraZRot ( ) [inline]

Returns the camera rotation around Z axis.

The value is specified in 0.1 degrees.

Returns

The camera rotation around Z axis specified in 0.1 degrees.

6.30.3.9 SourceID cavapa_gui::SceneWidget::getSourceID ( ) const [inline]

Returns the ID number of the source that the scene represents.

Returns

Source ID number or UNDEFINED_SOURCE if no source is set.

6.30.3.10 void cavapa_gui::SceneWidget::pauseRequested (SourceID *sender*) [signal]

The signal is emitted when the user wants to pause the video.

Parameters

sender	Source ID-number.

6.30.3.11 void cavapa_gui::SceneWidget::playRequested (SourceID *sender*) [signal]

The signal is emitted when the user wants to play the video.

Parameters

sender	Source ID-number.

6.30.3.12 void cavapa_gui::SceneWidget::requestedActivation (SceneWidget * *sender*) [signal]

The signal is emitted when user wants to activate this source widget for calibration.

Parameters

sender	Source ID-number.

6.30.3.13 void cavapa_gui::SceneWidget::seekBarValueChanged (SourceID *sender,* FrameTime *value*) [signal]

The signal is emitted when video seek bar value has changed.

Parameters

sender	Source ID-number.
value	The new seek bar value.

6.30.3.14 void cavapa_gui::SceneWidget::setAddCalibrationPointsEnabled (bool *enabled*) [inline]

Tells the SceneWidget whether adding calibration points is possible.

Parameters

enabled	Whether adding calibration points should be possible or not.

6.30.3.15 void cavapa_gui::SceneWidget::setCalibrationPoints (std::vector< cavapa::calibration_point > *points*)

Sets the calibration points.

Parameters

points	The new calibration points.

6.30.3.16 void cavapa_gui::SceneWidget::setCameraFov (int *value*) [slot]

Sets the field of view of the camera.

The value is specified in 0.1 degrees.

Parameters

value	The new fcamera field of view specified in 0.1 degrees.

6.30.3.17 void cavapa_gui::SceneWidget::setCameraHeight (int *value*) [slot]

Sets the camera height.

The value is specified in 0.01 degrees.

Parameters

value	The new camera height specified in 0.01 units.

6.30.3.18 void cavapa_gui::SceneWidget::setCameraXRot (int *value*) [slot]

Sets the camera rotation around X axis.

The value is specified in 0.1 degrees.

Parameters

value	The new X rotation specified in 0.1 degrees.

6.30.3.19 void cavapa_gui::SceneWidget::setCameraYRot (int *value*) [slot]

Sets the camera rotation around Y axis.

The value is specified in 0.1 degrees.

Parameters

value	The new Y rotation specified in 0.1 degrees.

6.30.3.20 void cavapa_gui::SceneWidget::setCameraZRot (int *value*) [slot]

Sets the camera rotation around Z axis.

The value is specified in 0.1 degrees.

Parameters

value	The new Z rotation specified in 0.1 degrees.

6.30.3.21 void cavapa_gui::SceneWidget::setDrawCalibrationPointsEnabled (bool *enabled*) [inline], [slot]

Sets whether the calibration points are drawn or not.

Parameters

enabled	Whether the calibration points are drawn or not.

6.30.3.22 void cavapa_gui::SceneWidget::setDrawGridEnabled (bool *enabled*) [inline], [slot]

Sets whether the grid is drawn or not.

Parameters

enabled	Whether the grid is drawn or not.

6.30.3.23 void cavapa_gui::SceneWidget::setSeekBarValue (FrameTime *value*)

Sets the seek bar to a new value.

Parameters

value	The new value.

6.30.3.24 void cavapa_gui::SceneWidget::setVideoLength (FrameTime *frameTime*)

Sets video length for the seek bar.

Parameters

frameTime	The video length specified in FrameTime.

6.30.3.25 void cavapa_gui::SceneWidget::updateFrame (const FrameCapture & *frame*)

Updates the scene image.

Parameters

frame	The new frame information.

6.30.3.26 void cavapa_gui::SceneWidget::updateSettings (const SourceSettings & *settings*)

Updates the scene settings.

Parameters

settings	The settings for the source.

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

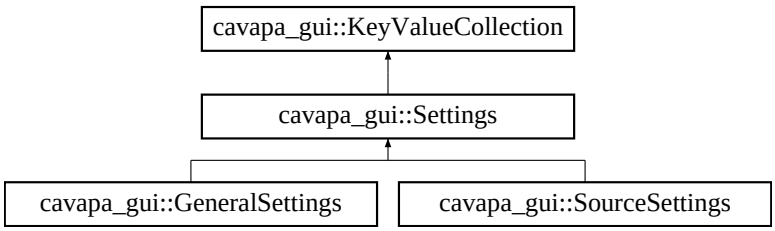
- gui/scenewidget.h
- gui/scenewidget.cpp

6.31 cavapa_gui::Settings Class Reference

Provides methods for loading and saving the application settings.

`#include <settings.h>`

Inheritance diagram for cavapa_gui::Settings:



Public Member Functions

- std::vector< std::string > getVector (const std::string &key) const
  Gets a collection of values related to the specified key.
- void setVector (const std::string &key, const std::vector< std::string > &vector)
  Sets a collection of values related to the specified key.

Protected Member Functions

- virtual std::string getGroupName ()
  Gets the name of the setting group.

Protected Attributes

- std::map< std::string, std::vector< std::string > > vectorMap
  The map associates a string key with a vector of strings.

Friends

- class **Controller**

Additional Inherited Members

6.31.1 Detailed Description

Provides methods for loading and saving the application settings.

Author

Mika Lehtinen

6.31.2 Member Function Documentation

6.31.2.1 virtual std::string cavapa_gui::Settings::getGroupName ( ) [inline], [protected], [virtual]

Gets the name of the setting group.

Returns

The name of the setting group. The default value is an empty string.

Reimplemented in cavapa_gui::SourceSettings.

6.31.2.2 std::vector< std::string > cavapa_gui::Settings::getVector (const std::string & *key*) const

Gets a collection of values related to the specified key.

Parameters

key	The key whose values are to be fetched.

Returns

The collection of values as a vector.

6.31.2.3 void cavapa_gui::Settings::setVector (const std::string & *key*, const std::vector< std::string > & *vector*)

Sets a collection of values related to the specified key.

Parameters

key	The key whose values are to be set.
vector	The collection of values as a vector.

6.31.3 Member Data Documentation

6.31.3.1 std::map<std::string, std::vector<std::string> > cavapa_gui::Settings::vectorMap [protected]

The map associates a string key with a vector of strings.

It is used to save arrays of values in the settings.

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

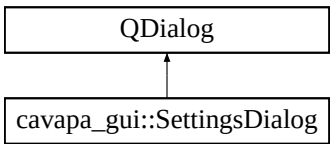
- settings.h
- settings.cpp

6.32 cavapa_gui::SettingsDialog Class Reference

Represents the dialog for the general settings of the application.

`#include <settingsdialog.h>`

Inheritance diagram for cavapa_gui::SettingsDialog:



Public Member Functions

- SettingsDialog (QWidget *parent=0)

  Constructs a new SettingsDialog having the specified parent.
- GeneralSettings getSettings ()

  Gets the settings from the widget based on the current values.
- void setSettings (const GeneralSettings &settings)

  Sets the settings for the widget.

6.32.1 Detailed Description

Represents the dialog for the general settings of the application.

Author

Mika Lehtinen

6.32.2 Constructor & Destructor Documentation

6.32.2.1 cavapa_gui::SettingsDialog::SettingsDialog (QWidget * *parent* = 0) [explicit]

Constructs a new SettingsDialog having the specified parent.

Parameters

parent	The parent widget.

6.32.3 Member Function Documentation

6.32.3.1 GeneralSettings cavapa_gui::SettingsDialog::getSettings ( )

Gets the settings from the widget based on the current values.

Returns

The GeneralSettings object representing the settings.

6.32.3.2 void cavapa_gui::SettingsDialog::setSettings (const GeneralSettings & *settings*)

Sets the settings for the widget.

Parameters

settings	The GeneralSettings object.

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

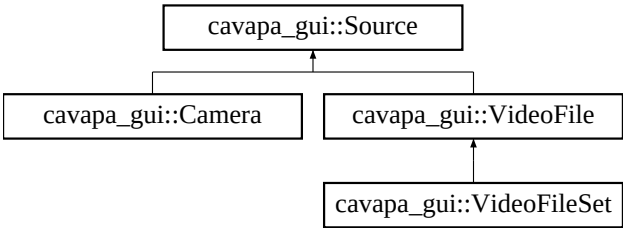
- gui/settingsdialog.h
- gui/settingsdialog.cpp

6.33 cavapa_gui::Source Class Reference

The camera and video file source base class.

`#include <source.h>`

Inheritance diagram for cavapa_gui::Source:



Public Member Functions

- DISALLOW_COPY_AND_ASSIGN (Source)

  Copy and assign of the class is not allowed.
- virtual bool canRecord () const =0

  Returns information on the source recording abilities.
- virtual FrameCapture get (FrameTime passed_time)=0

  Retrieves the next frame from the camera.
- virtual double getBarrelCorrection ()=0

  Returns the barrel correction applied to the source frames.
- virtual std::string getDescription () const =0

  Retrieves the description of the source.
- virtual double getFramerate () const =0

  Retrieves the framerate.
- virtual SourceID getID () const final

  Gets the unique source ID number.
- virtual cv::Size getResolution () const =0

  Retrieves the resolution of the source.
- virtual FrameTime getPosition () const =0

  Retrieves the current time position of the source.
- virtual SourceType getSourceType () const

  Returns the type of the source.
- virtual SourceStats getStats ()=0

  Retrieves the source statistical information.
- virtual bool hasNew ()=0

  Checks whether the device has a new image to be retrieved or not.
- virtual bool isOpen () const =0

  Checks whether the source has been initialized or not.
- virtual bool isPlaying () const =0

  Checks whether the source is playing or not.
- virtual void play ()

  Starts to play the source.
- virtual bool record (const std::string &filename, const std::string &codec="")=0

  Starts the recording of the source to the file.
- virtual void resetStats () final

  Resets the source statistics.
- virtual void setBarrelCorrection (double amount)=0

  Sets the new barrel correction value.
- virtual bool setResolution (const cv::Size &new_size)=0

  Sets the source resolution.
- virtual void stop ()=0

  Stops the source.

Protected Member Functions

- Source (SourceID desired_id=UNDEFINED_SOURCE)

  General Source creator.

Protected Attributes

- SourceStats statistics = EMPTY_SOURCE_STATS

  The statistics of the source.

6.33.1 Detailed Description

The camera and video file source base class.

The base class for any kind of frame source that provides the frame updates with buffering. The inherits from the class can provide their own functionality such as recording, resolution changes and/or framerate alterations.

Author

Petri Partanen

6.33.2 Constructor & Destructor Documentation

6.33.2.1 cavapa_gui::Source::Source (SourceID *desired_id* = UNDEFINED_SOURCE) [inline], [protected]

General Source creator.

Parameters

desired_id	The desired ID number for the source.

Remarks

This does not perform any checks if the desired ID number is already used. You must perform that check somewhere else. The function does however ensure that the sources that have no desired ID's defined will never be given the desired source ID number.

6.33.3 Member Function Documentation

6.33.3.1 virtual bool cavapa_gui::Source::canRecord ( ) const [pure virtual]

Returns information on the source recording abilities.

Returns

True if the source can record, false otherwise.

Implemented in cavapa_gui::Camera, and cavapa_gui::VideoFile.

6.33.3.2 virtual FrameCapture cavapa_gui::Source::get (FrameTime *passed_time*) [pure virtual]

Retrieves the next frame from the camera.

The time supplied to the function informs the source about how much time has passed since the last image retrieval. The source will depend on its internal clock, either retrieving a new image or returning the buffered image that it retrieved previously.

Parameters

passed_time	The time passed since the last call (in milliseconds).

Returns

The latest frame capture. This will actually return a reference to the original image, so you are NOT allowed to edit it! Otherwise the source buffer will be altered too. This is just the way OpenCV handles Mat memory.

Implemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.3 virtual double cavapa_gui::Source::getBarrelCorrection ( ) [pure virtual]

Returns the barrel correction applied to the source frames.

Returns

The barrel effect value.

Implemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.4 virtual std::string cavapa_gui::Source::getDescription ( ) const [pure virtual]

Retrieves the description of the source.

Returns

The source description.

Implemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.5 virtual double cavapa_gui::Source::getFramerate ( ) const [pure virtual]

Retrieves the framerate.

Returns

The framerate or 0.0 if not supported.

Implemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.6 virtual SourceID cavapa_gui::Source::getID ( ) const [inline], [final], [virtual]

Gets the unique source ID number.

Returns

The source ID number.

6.33.3.7 virtual FrameTime cavapa_gui::Source::getPosition ( ) const [pure virtual]

Retrieves the current time position of the source.

Returns

The source time position.

Implemented in cavapa_gui::Camera, and cavapa_gui::VideoFile.

6.33.3.8 virtual cv::Size cavapa_gui::Source::getResolution ( ) const [pure virtual]

Retrieves the resolution of the source.

Returns

The source resolution.

Implemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.9 virtual SourceType cavapa_gui::Source::getSourceType ( ) const [inline], [virtual]

Returns the type of the source.

Returns

The source type.

Reimplemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.10 virtual SourceStats cavapa_gui::Source::getStats ( ) [pure virtual]

Retrieves the source statistical information.

Returns

The source statistics.

Implemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.11 virtual bool cavapa_gui::Source::hasNew ( ) [pure virtual]

Checks whether the device has a new image to be retrieved or not.

This checks if the FrameTime on the buffered image differs from what was retrieved on the last get().

Returns

True if a new image available or false otherwise.

Implemented in cavapa_gui::Camera, cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.12 virtual bool cavapa_gui::Source::isOpen ( ) const [pure virtual]

Checks whether the source has been initialized or not.

Returns

True if it is initialized or false otherwise.

Implemented in cavapa_gui::Camera, and cavapa_gui::VideoFile.

6.33.3.13 virtual bool cavapa_gui::Source::isPlaying ( ) const [pure virtual]

Checks whether the source is playing or not.

Returns

True if the source is playing or false otherwise.

Implemented in cavapa_gui::Camera, and cavapa_gui::VideoFile.

6.33.3.14 virtual void cavapa_gui::Source::play ( ) [inline], [virtual]

Starts to play the source.

Depending on the source type, this might have no effect.

Reimplemented in cavapa_gui::VideoFile, and cavapa_gui::VideoFileSet.

6.33.3.15 virtual bool cavapa_gui::Source::record (const std::string & *filename*, const std::string & *codec* = " ") [pure virtual]

Starts the recording of the source to the file.

This can also be used to start an entirely new video file during the recording. The recording file will change immediately.

Parameters

filename	The path of the recording file.
codec	The codec to be used for recording.

Returns

True if the recording started or false otherwise.

Implemented in cavapa_gui::Camera, and cavapa_gui::VideoFile.

6.33.3.16 virtual void cavapa_gui::Source::setBarrelCorrection (double *amount*) [pure virtual]

Sets the new barrel correction value.

The value is used to correct barrel effect (lens correction) on each image that the source will return.

Parameters

amount	The barrel value.

Implemented in cavapa_gui::VideoFile, cavapa_gui::Camera, and cavapa_gui::VideoFileSet.

6.33.3.17 virtual bool cavapa_gui::Source::setResolution (const cv::Size & *new_size*) [pure virtual]

Sets the source resolution.

Parameters

new_size	The new resolution for the source.

Returns

True if the resolution was set, false otherwise.

Implemented in cavapa_gui::VideoFile, and cavapa_gui::Camera.

6.33.3.18 virtual void cavapa_gui::Source::stop ( ) [pure virtual]

Stops the source.

Depending on the source type this will have a different effect.

Implemented in cavapa_gui::VideoFile, cavapa_gui::Camera, and cavapa_gui::VideoFileSet.

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

- source/source.h

6.34 cavapa_gui::SourceInfo Struct Reference

Holds the information related to a source including id number, type, settings, a list of the recorded videos and a list of the calibration points.

`#include <metadata.h>`

Public Attributes

- SourceID id

  The id number of the source.
- SourceType type

  The type of the source.
- SourceSettings settings

  The settings for the source.
- FrameTime startOffset

  The start offset of the source used in analysis.
- std::vector< std::string > recordedVideos

  The set of the video files related to the source.
- std::vector< CalibrationPoint > calibrationPoints

  The set of the calibration points related to the source.

6.34.1 Detailed Description

Holds the information related to a source including id number, type, settings, a list of the recorded videos and a list of the calibration points.

Author

Mika Lehtinen

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

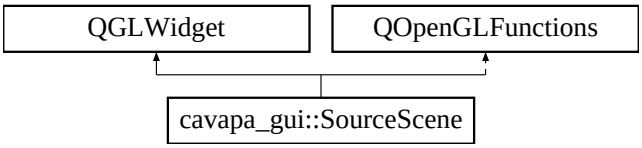
- metadata.h

6.35 cavapa_gui::SourceScene Class Reference

SourceScene draws a video stream and configures camera parameters.

`#include <sourcescene.h>`

Inheritance diagram for cavapa_gui::SourceScene:



Public Slots

- void setCameraXRot (float value)

  Sets the camera rotation around X axis.
- void setCameraYRot (float value)

  Sets the camera rotation around Y axis.
- void setCameraZRot (float value)

  Sets the camera rotation around Z axis.

- void setCameraHeight (float value)

  Sets the camera height.
- void setCameraFov (float value)

  Sets the camera field of view.
- void setDrawGridEnabled (bool enabled)

  Sets if the grid should be drawn.
- void setDrawCalibrationPointsEnabled (bool enabled)

  Sets if the calibration points should be drawn.

Public Member Functions

- SourceScene (QWidget *parent=0)

  Constructs a new SourceScene having the specified parent widget.
- float getCameraXRot ()

  Returns the camera rotation around X axis.
- float getCameraYRot ()

  Returns the camera rotation around Y axis.
- float getCameraZRot ()

  Returns the camera rotation around Z axis.
- float getCameraHeight ()

  Returns the camera height.
- float getCameraFov ()

  Sets the field of view of the camera.
- QVector3D getCameraLookDirection ()

  Gets the camera look direction.
- QVector3D getCameraPosition ()

  Gets the camera position.
- cavapa::camera getCameraSettings ()

  Gets the camera settings.
- void setCameraSettings (cavapa::camera settings)

  Sets camera settings.
- CalibrationPoints getCalibrationPoints ()

  Gets the calibration points.
- void addCalibrationIPoint (QPointF point)

  Adds a new calibration point.
- void setAddCalibrationPointsEnabled (bool enabled)

  Sets if adding calibration points should be possible.
- void updateImage (const cv::Mat &image)

  Updates the background image.

Protected Member Functions

- void mouseReleaseEvent (QMouseEvent *e)

  Reacts to a mouse release event: If right mouse button was released, display a context menu that has the actions for managing the calibration points.
- void mousePressEvent (QMouseEvent *e)

  Reacts to a mouse press event by highlighting the closest calibration point.
- void timerEvent (QTimerEvent *e)

  The timer event.
- void initializeGL ()

Initializes the OpenGL scene.

- void resizeGL (int w, int h)

  Resizes the GL viewport.
- void drawBackgroundImage ()

  Draws the background image.
- void drawGrid ()

  Draws the grid.
- void drawCalibrationPoints (QPainter &painter)

  Draws the calibration points.
- void paintGL ()

  Paints the GL scene.
- void initShaders ()

  Initializes the shaders.
- void setGlViewport ()

  Sets the GL viewport.
- void calculateViewPort ()

  Calculates the viewport maintaining the video aspect ratio.
- QPointF getRelativeVideoPosition (QPoint widgetPoint)

  Calculates the relative position of a point in the video.

6.35.1 Detailed Description

SourceScene draws a video stream and configures camera parameters.

Draws an image that can be changed to display a video. A grid that represents how the ground would look with the current camera parameters is drawn on top of the image. The camera parameters can be modified and retrieved.

It's also possible to add calibration points to the widget through the right click menu. The calibration points are ment for a situation where there are multiple cameras filming the same situation from different angles. The calibration points can be used to mark where the same spot exists in both cameras. One calibration point can be highlighted by left clicking to help differentiate points that might be close to each other.

Author

Oskari Leppäaho

6.35.2 Constructor & Destructor Documentation

6.35.2.1 cavapa_gui::SourceScene::SourceScene (QWidget * *parent* = 0) [explicit]

Constructs a new SourceScene having the specified parent widget.

Parameters

parent	The parent widget.

6.35.3 Member Function Documentation

6.35.3.1 void cavapa_gui::SourceScene::addCalibrationIPoint (QPointF *point*) [inline]

Adds a new calibration point.

Parameters

point	The new point. The point should be in relative coordinates, i.e. x and y coordinates range from 0 to 1.

6.35.3.2 CalibrationPoints cavapa_gui::SourceScene::getCalibrationPoints ( ) [inline]

Gets the calibration points.

Returns

The the calibration points.

6.35.3.3 float cavapa_gui::SourceScene::getCameraFov ( ) [inline]

Sets the field of view of the camera.

Returns

The field of view of the camera.

6.35.3.4 float cavapa_gui::SourceScene::getCameraHeight ( ) [inline]

Returns the camera height.

Returns

The camera height.

6.35.3.5 QVector3D cavapa_gui::SourceScene::getCameraLookDirection ( )

Gets the camera look direction.

Returns

A unit vector that defines the direction the camera is pointing to.

6.35.3.6 QVector3D cavapa_gui::SourceScene::getCameraPosition ( )

Gets the camera position.

Returns

The camera position as 3D vector.

6.35.3.7 cavapa::camera cavapa_gui::SourceScene::getCameraSettings ( )

Gets the camera settings.

Returns

The camera settings.

6.35.3.8 float cavapa_gui::SourceScene::getCameraXRot ( ) [inline]

Returns the camera rotation around X axis.

Returns

The camera X rotation.

6.35.3.9 float cavapa_gui::SourceScene::getCameraYRot ( ) [inline]

Returns the camera rotation around Y axis.

Returns

The camera Y rotation.

6.35.3.10 float cavapa_gui::SourceScene::getCameraZRot ( ) [inline]

Returns the camera rotation around Z axis.

Returns

The camera Z rotation.

6.35.3.11 QPointF cavapa_gui::SourceScene::getRelativeVideoPosition (QPoint *widgetPoint*) [inline], [protected]

Calculates the relative position of a point in the video.

Parameters

widgetPoint	The point which position to calculate.

Returns

A corresponding point with coordinates in range from 0 to 1.

6.35.3.12 void cavapa_gui::SourceScene::mousePressEvent (QMouseEvent * *e*) [protected]

Reacts to a mouse press event by highlighting the closest calibration point.

Parameters

e	The QMouseEvent.

6.35.3.13 void cavapa_gui::SourceScene::mouseReleaseEvent (QMouseEvent * *e*) [protected]

Reacts to a mouse release event: If right mouse button was released, display a context menu that has the actions for managing the calibration points.

Parameters

*e	The QMouseEvent.

6.35.3.14 void cavapa_gui::SourceScene::resizeGL (int *w*, int *h*) [protected]

Resizes the GL viewport.

Parameters

w	The new viewport width.
h	The new viewport height.

6.35.3.15 void cavapa_gui::SourceScene::setAddCalibrationPointsEnabled (bool *enabled*) [inline]

Sets if adding calibration points should be possible.

Parameters

enabled	Specifies if adding calibration points should be possible.

6.35.3.16 void cavapa_gui::SourceScene::setCameraFov (float *value*) [slot]

Sets the camera field of view.

Parameters

value	The new field of view.

6.35.3.17 void cavapa_gui::SourceScene::setCameraHeight (float *value*) [slot]

Sets the camera height.

Parameters

value	The new camera height.

6.35.3.18 void cavapa_gui::SourceScene::setCameraSettings (cavapa::camera *settings*)

Sets camera settings.

Parameters

settings	The new camera settings.

6.35.3.19 void cavapa_gui::SourceScene::setCameraXRot (float *value*) [slot]

Sets the camera rotation around X axis.

Parameters

value	The new camera X rotation.

6.35.3.20 void cavapa_gui::SourceScene::setCameraYRot (float *value*) [slot]

Sets the camera rotation around Y axis.

Parameters

value	The new camera Y rotation.

6.35.3.21 void cavapa_gui::SourceScene::setCameraZRot (float *value*) [slot]

Sets the camera rotation around Z axis.

Parameters

value	The new camera Z rotation.

6.35.3.22 void cavapa_gui::SourceScene::setDrawCalibrationPointsEnabled (bool *enabled*) [inline], [slot]

Sets if the calibration points should be drawn.

Parameters

enabled	Specifies whether the calibration points should be drawn or not.

6.35.3.23 void cavapa_gui::SourceScene::setDrawGridEnabled (bool *enabled*) [inline], [slot]

Sets if the grid should be drawn.

Parameters

enabled	Specifies whether the grid should be drawn or not.

6.35.3.24 void cavapa_gui::SourceScene::timerEvent (QTimerEvent * *e*) [protected]

The timer event.

Handles updating the OpenGL widget.

Parameters

*e	The QTimerEvent.

6.35.3.25 void cavapa_gui::SourceScene::updateImage (const cv::Mat & *image*)

Updates the background image.

Use this to update the video frames.

Parameters

image	The new background image.

Remarks

This assumes that the image format is RGB888.

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

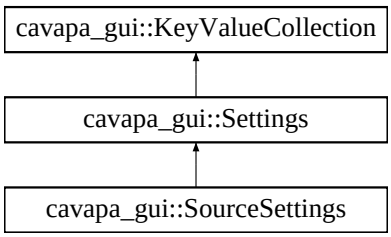
- gui/sourcescene.h
- gui/sourcescene.cpp

6.36 cavapa_gui::SourceSettings Class Reference

Represents the settings for a video source.

`#include <sourcesettings.h>`

Inheritance diagram for cavapa_gui::SourceSettings:



Public Member Functions

- SourceSettings ()

  Constructs a new SourceSettings object with default values and an empty name.
- SourceSettings (const std::string &sourceName)

  Constructs a new SourceSettings object with default values.
- std::string getSourceName () const

  Gets the name of the associated source.
- void setCameraSettings (const cavapa::CameraSettings &cameraSettings)

  Sets the camera settings for the instance of SourceSettings.
- void setSourceName (const std::string &name)

  Sets the name of the associated source.
- cavapa::CameraSettings toCameraSettings () const

  Converts the instance of SourceSettings to CameraSettings.

Protected Member Functions

- bool canLoadOrSave ()

  Gets whether the SourceSettings instance can be saved or not.
- std::string getGroupName ()

  Gets the name of the setting group.

Additional Inherited Members

6.36.1 Detailed Description

Represents the settings for a video source.

Author

Mika Lehtinen

6.36.2 Constructor & Destructor Documentation

6.36.2.1 cavapa_gui::SourceSettings::SourceSettings (const std::string & *sourceName*) [explicit]

Constructs a new SourceSettings object with default values.

Parameters

sourceName	The name of the source.

6.36.3 Member Function Documentation

6.36.3.1 bool cavapa_gui::SourceSettings::canLoadOrSave ( ) [inline], [protected], [virtual]

Gets whether the SourceSettings instance can be saved or not.

Returns

True if sourceName is not an empty string, false otherwise.

Reimplemented from cavapa_gui::Settings.

6.36.3.2 std::string cavapa_gui::SourceSettings::getGroupName ( ) [inline], [protected], [virtual]

Gets the name of the setting group.

Returns

The name of the setting group (the source name).

Reimplemented from cavapa_gui::Settings.

6.36.3.3 std::string cavapa_gui::SourceSettings::getSourceName ( ) const

Gets the name of the associated source.

Returns

The name of the source.

6.36.3.4 void cavapa_gui::SourceSettings::setCameraSettings (const cavapa::CameraSettings & *cameraSettings*)

Sets the camera settings for the instance of SourceSettings.

Parameters

cameraSettings	The camera settings.

6.36.3.5 cavapa::CameraSettings cavapa_gui::SourceSettings::toCameraSettings ( ) const

Converts the instance of SourceSettings to CameraSettings.

Returns

The CameraSettings object.

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

- sourcesettings.h
- sourcesettings.cpp

6.37 cavapa_gui::SourceStats Struct Reference

The structure to hold statistics about the source performance.

`#include <common.h>`

Public Attributes

- unsigned int missed

  The total number of the missed frames.
- unsigned int recorded

  The frames recorded to the current video file.
- unsigned int retrieved

  The total number of the frames retrieved from the source.
- unsigned int total_recorded

  The total number of the frames recorded in all video files.

6.37.1 Detailed Description

The structure to hold statistics about the source performance.

This can be used to keep an eye on possible frame drops. A large missed frames count will indicate that the source can't keep up with the current framerate. A large number of concurrent misses will definitely indicate a more serious connection error.

Author

Petri Partanen

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

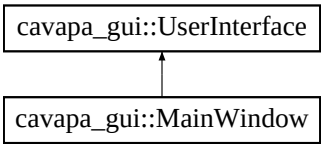
- common.h

6.38 cavapa_gui::UserInterface Class Reference

The UserInterface class defines the methods and signals that a user interface needs to have.

`#include <userinterface.h>`

Inheritance diagram for cavapa_gui::UserInterface:



Protected Member Functions

- virtual void calculationCancelRequested ()=0

  Emitted when the user requests to cancel the calculation.
- virtual void calculationContinueRequested ()=0

  Emitted when the user requests to continue the calculation.
- virtual void calculationPauseRequested ()=0

  Emitted when the user requests to pause the calculation.
- virtual void calculationStartRequested ()=0

  Emitted when the user requests to start the calculation.
- virtual void calculationStopRequested ()=0

  Emitted when the user requests to stop the calculation.
- virtual void calibrationPointsReady (const std::vector< CalibrationPoint > &points)=0

  Emitted when the user has set all the calibration points.
- virtual void exportRequested (const ExportOptions &options)=0

  Emitted when the user requests to export data from the graph.
- virtual void generalSettingsChanged (const GeneralSettings &settings)=0

  Emitted when the user changes a general setting.
- virtual void graphPositionSelected (FrameTime time)=0

  Emitted when the user selects a position in the activity graph.
- virtual void locationChanged (int x, int y)=0

  Emitted when the interface is moved.
- virtual void markersReady (const std::vector< GraphMarker > &markers)=0

  Emitted when the user has placed all the markers.
- virtual void metadataOpened (const std::string &path)=0

  Emitted when the user requests to open a metadata file.
- virtual void metadataReady (const Metadata &metadata)=0

  Emitted when the user has input all the metadata.
- virtual void newMeasurementRequested ()=0

  Emitted when the user wants to start a new measurement.
- virtual void openMeasurementRequested (const std::string &path)=0

  Emitted when the user wants to open an existing measurement.
- virtual void redoMeasurementRequested ()=0

  Emitted when the user wants to redo the measurement.
- virtual void refreshRequested ()=0

  Emitted when the user requests to refresh cameras.
- virtual void sizeChanged (int width, int height, bool fullscreen)=0

  Emitted when the size of the interface changes.

- virtual void sourceAddRequested (const std::string &path)=0

  Emitted when the user adds a new video source.
- virtual void sourcePauseRequested (SourceID source)=0

  Emitted when the user requests to pause a source.
- virtual void sourcePlayRequested (SourceID source)=0

  Emitted when the user requests to play a source.
- virtual void sourceSeekRequested (SourceID source, FrameTime position)=0

  Emitted when the user requests to seek a source to a certain position.
- virtual void sourceRemoveRequested (SourceID source)=0

  Emitted when the user removes a source.
- virtual void sourceSettingsChanged (SourceID source, const SourceSettings &settings)=0

  Emitted when the user changes a setting for a source.
- virtual void sourceStepBackward (SourceID source)=0

  Emitted when the user steps a video one frame back.
- virtual void sourceStepForward (SourceID source)=0

  Emitted when the user steps a video one frame forward.
- virtual void statisticsRequested (FrameTime start, FrameTime stop, int points)=0

  Emitted when the user wants the frame statistics to be retrieved.
- virtual void interfaceClosed ()=0

  Emitted when the user closes the interface.

Friends

- class **Controller**

6.38.1 Detailed Description

The UserInterface class defines the methods and signals that a user interface needs to have.

Author

Mika Lehtinen

6.38.2 Member Function Documentation

6.38.2.1 virtual void cavapa_gui::UserInterface::calibrationPointsReady (const std::vector< CalibrationPoint > & *points*) [protected], [pure virtual]

Emitted when the user has set all the calibration points.

Parameters

points	The set of calibration points.

6.38.2.2 virtual void cavapa_gui::UserInterface::exportRequested (const ExportOptions & *options*) [protected], [pure virtual]

Emitted when the user requests to export data from the graph.

Parameters

options	The parameters for the export.

6.38.2.3 virtual void cavapa_gui::UserInterface::generalSettingsChanged (const GeneralSettings & *settings*) [protected], [pure virtual]

Emitted when the user changes a general setting.

Parameters

settings	The new GeneralSettings object.

6.38.2.4 virtual void cavapa_gui::UserInterface::graphPositionSelected (FrameTime *time*) [protected], [pure virtual]

Emitted when the user selects a position in the activity graph.

Parameters

time	The time of the position.

6.38.2.5 virtual void cavapa_gui::UserInterface::locationChanged (int *x*, int *y*) [protected], [pure virtual]

Emitted when the interface is moved.

Parameters

x	The x-coordinate of the interface.
y	The y-coordinate of the interface.

6.38.2.6 virtual void cavapa_gui::UserInterface::markersReady (const std::vector< GraphMarker > & *markers*) [protected], [pure virtual]

Emitted when the user has placed all the markers.

Parameters

markers	The vector containing the markers.

Remarks

At the moment, this signal is emitted when the calculation completes.

6.38.2.7 virtual void cavapa_gui::UserInterface::metadataOpened (const std::string & *path*) [protected], [pure virtual]

Emitted when the user requests to open a metadata file.

Parameters

path	The path of the file that was selected.

6.38.2.8 virtual void cavapa_gui::UserInterface::metadataReady (const Metadata & *metadata*) [protected], [pure virtual]

Emitted when the user has input all the metadata.

Parameters

metadata	The Metadata object representing the user input.

6.38.2.9 virtual void cavapa_gui::UserInterface::openMeasurementRequested (const std::string & *path*) [protected], [pure virtual]

Emitted when the user wants to open an existing measurement.

Parameters

path	The path to the measurement's XML file.

6.38.2.10 virtual void cavapa_gui::UserInterface::sizeChanged (int *width*, int *height*, bool *fullscreen*) [protected], [pure virtual]

Emitted when the size of the interface changes.

Parameters

width	The new width of the interface.
height	The new height of the interface.
fullscreen	Whether the interface is in fullscreen mode or not.

6.38.2.11 virtual void cavapa_gui::UserInterface::sourceAddRequested (const std::string & *path*) [protected], [pure virtual]

Emitted when the user adds a new video source.

Parameters

path	The path of the source.

6.38.2.12 virtual void cavapa_gui::UserInterface::sourcePauseRequested (SourceID *source*) [protected], [pure virtual]

Emitted when the user requests to pause a source.

Parameters

source	The id of the source to be paused.

6.38.2.13 virtual void cavapa_gui::UserInterface::sourcePlayRequested (SourceID *source*) [protected], [pure virtual]

Emitted when the user requests to play a source.

Parameters

source	The id of the source to play.

6.38.2.14 virtual void cavapa_gui::UserInterface::sourceRemoveRequested (SourceID *source*) [protected], [pure virtual]

Emitted when the user removes a source.

Parameters

source	The source ID to be removed.

6.38.2.15 virtual void cavapa_gui::UserInterface::sourceSeekRequested (SourceID *source,* FrameTime *position*) [protected],[pure virtual]

Emitted when the user requests to seek a source to a certain position.

Parameters

source	The id of the source to be sought.
position	The requested position.

6.38.2.16 virtual void cavapa_gui::UserInterface::sourceSettingsChanged (SourceID *source,* const SourceSettings & *settings*) [protected],[pure virtual]

Emitted when the user changes a setting for a source.

Parameters

source	The source ID for which a setting was changed.
settings	The new settings for the source.

6.38.2.17 virtual void cavapa_gui::UserInterface::sourceStepBackward (SourceID *source*) [protected],[pure virtual]

Emitted when the user steps a video one frame back.

Parameters

source	The source that was stepped.

6.38.2.18 virtual void cavapa_gui::UserInterface::sourceStepForward (SourceID *source*) [protected],[pure virtual]

Emitted when the user steps a video one frame forward.

Parameters

source	The source that was stepped.

6.38.2.19 virtual void cavapa_gui::UserInterface::statisticsRequested (FrameTime *start,* FrameTime *stop,* int *points*) [protected],[pure virtual]

Emitted when the user wants the frame statistics to be retrieved.

Parameters

start	The starting point of the period.
stop	The end point of the period.

points	The number of points desired to be returned.

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

- userinterface.h

6.39 cavapa_gui::VertexData Struct Reference

Represents the information of a single vertex.

Public Attributes

- QVector3D position

  The position of the vertex.

6.39.1 Detailed Description

Represents the information of a single vertex.

Currently, this only contains the position of the vertex.

Author

Oskari Leppäaho

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

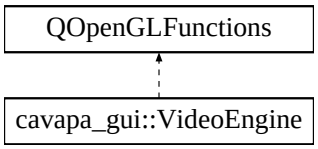
- gui/gridengine.cpp
- gui/videoengine.cpp

6.40 cavapa_gui::VideoEngine Class Reference

A class for drawing the video frame with OpenGL.

`#include <videoengine.h>`

Inheritance diagram for cavapa_gui::VideoEngine:



Public Member Functions

- void initialize ()
- void drawVideoGeometry (QGLShaderProgram *program)

  Draws the video frame.

6.40.1 Detailed Description

A class for drawing the video frame with OpenGL.

Author

Oskari Leppäaho

6.40.2 Member Function Documentation

6.40.2.1 void cavapa_gui::VideoEngine::drawVideoGeometry (QGLShaderProgram * *program*)

Draws the video frame.

Parameters

**program*	Specifies the OpenGL shader program to be used.

6.40.2.2 void cavapa_gui::VideoEngine::initialize ( )

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

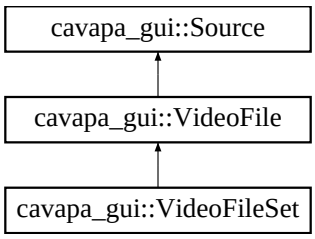
- gui/videoengine.h
- gui/videoengine.cpp

6.41 cavapa_gui::VideoFile Class Reference

The class for the video file source.

`#include <videofile.h>`

Inheritance diagram for cavapa_gui::VideoFile:



Public Member Functions

- VideoFile (SourceID desired_id=UNDEFINED_SOURCE)

  Creates a new source.
- bool canRecord () const overridefinal

  Returns information on the source recording abilities.
- virtual void fetchFirstFrame ()

  Seeks the startup point of the video and the first frame.
- virtual FrameCapture get (FrameTime passed_time) override

  Retrieves the next frame from the camera.
- virtual double getBarrelCorrection () override

  Returns the barrel correction applied to the source frames.

- int getCurrentFrame () const

  Returns the current frame of the video shown.
- virtual std::string getDescription () const override

  Retrieves the description of the source.
- virtual std::vector< std::string > getFilenames () const

  Returns the filenames of the source.
- virtual double getFramerate () const override

  Retrieves the framerate.
- virtual FrameTime getLength () const final

  Retrieves the video file length.
- virtual FrameTime getPosition () const overridefinal

  Retrieves the current time position of the source.
- virtual cv::Size getResolution () const override

  Retrieves the resolution of the source.
- virtual SourceType getSourceType () const override

  Returns the type of the source.
- virtual SourceStats getStats () override

  Retrieves the source statistical information.
- virtual bool hasNew () override

  Checks whether the device has a new image to be retrieved or not.
- virtual bool isOpen () const override

  Checks whether the source has been initialized or not.
- virtual bool isPlaying () const override

  Checks whether the source is playing or not.
- virtual bool open (const std::string &filename, bool get_first, FrameTime start_point)

  Opens a video file.
- virtual void play () override

  Starts frame updates on the file.
- bool record (const std::string &filename, const std::string &codec="") overridefinal

  Starts the recording of the source to the file.
- virtual void release ()

  Forces the release of the worker thread.
- bool seekTime (FrameTime pos)

  Seeks a specific position of the video frame.
- virtual bool skipFrame (int step)

  Skips the specific amount of frames on the video.
- virtual void setBarrelCorrection (double amount) override

  Sets the new barrel correction value.
- void setCurrentAsStartPoint ()

  Sets the current video time as the seeking start point.
- bool setResolution (const cv::Size &new_size) overridefinal

  Should change resolution, but does not have an effect on video files.
- virtual FrameTime startPoint () const final

  Returns the start point of the video file.
- virtual void stop () override

  Will stop the file from playing.
- int totalFrames () const

  Returns the total count of the frames in the video file.

Static Public Member Functions

- static void setSpeed (double multiplier)

  Sets playing speed of the video file.

Protected Member Functions

- FrameTime currentFrameTime () const

  Returns the time of the currently buffered frame.
- int frameFromFrameTime (FrameTime time) const

  Returns the expected frame number from the frame time.
- FrameTime frameTimeFromFrame (int frame) const

  Returns the expected frame time of the given frame.

Protected Attributes

- int current_frame

  The number of the currently buffered frame.
- FrameTime initial_position

  The initial position of the video file.
- bool opened = false

  The open status of the video file.
- bool playing = false

  The flag indicates if the video file is playing or not.
- int total_frames = 0

  The total frames in the video file.
- FrameTime video_time = 0

  The current time of the video file.

Static Protected Attributes

- static double speed_ratio = 1.0

  The multiplier of the video file playing speed.

6.41.1 Detailed Description

The class for the video file source.

The class uses OpenCV to retrieve images from the video files. The video files can be played, stopped and stepped frame by frame. You can also seek the specific point of the video. Unlike Camera, the class does not support recording, or any alterations to framerate or resolution of the source.

Author

Petri Partanen

6.41.2 Constructor & Destructor Documentation

6.41.2.1 cavapa_gui::VideoFile::VideoFile (SourceID *desired_id* = UNDEFINED_SOURCE) [inline]

Creates a new source.

Parameters

desired_id	The desired ID number. Use UNDEFINED_SOURCE if you do not want to define it to a specific ID. This does not check if the source with the given ID already exists. The ID number will not be given other new sources with UNDEFINED_SOURCE ID numbers.

6.41.3 Member Function Documentation

6.41.3.1 bool cavapa_gui::VideoFile::canRecord ( ) const [inline], [final], [override], [virtual]

Returns information on the source recording abilities.

Returns

True if the source can record, false otherwise.
False. The video files can't be recorded.

Implements cavapa_gui::Source.

6.41.3.2 FrameTime cavapa_gui::VideoFile::currentFrameTime ( ) const [inline], [protected]

Returns the time of the currently buffered frame.

Returns

Last updated frame time in ms.

6.41.3.3 int cavapa_gui::VideoFile::frameFromFrameTime (FrameTime *time*) const [inline], [protected]

Returns the expected frame number from the frame time.

This is used to skip video frames if the retrieval is lagging behind on what the real time actually is.

Parameters

time	The frame time where frame number is to be calculated.

Returns

The frame on which the time stamp should be.

6.41.3.4 FrameTime cavapa_gui::VideoFile::frameTimeFromFrame (int *frame*) const [inline], [protected]

Returns the expected frame time of the given frame.

Parameters

frame	The number of the frame.

Returns

FrameTime for this frame with the videos framerate.

6.41.3.5 FrameCapture cavapa_gui::VideoFile::get (FrameTime *passed_time*) [override], [virtual]

Retrieves the next frame from the camera.

The time supplied to the function informs the source about how much time has passed since the last image retrieval. The source will depend on its internal clock, either retrieving a new image or returning the buffered image that it retrieved previously.

Parameters

passed_time	The time passed since the last call (in milliseconds).

Returns

The latest frame capture. This will actually return a reference to the original image, so you are NOT allowed to edit it! Otherwise the source buffer will be altered too. This is just the way OpenCV handles Mat memory.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.6 virtual double cavapa_gui::VideoFile::getBarrelCorrection ( ) [inline], [override], [virtual]

Returns the barrel correction applied to the source frames.

Returns

The barrel effect value.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.7 int cavapa_gui::VideoFile::getCurrentFrame ( ) const [inline]

Returns the current frame of the video shown.

Returns

The frame number on 0-based linear index.

6.41.3.8 string cavapa_gui::VideoFile::getDescription ( ) const [override], [virtual]

Retrieves the description of the source.

Returns

The source description.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.9 virtual std::vector<std::string> cavapa_gui::VideoFile::getFilenames ( ) const [inline], [virtual]

Returns the filenames of the source.

Returns

The video filenames.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.10 virtual double cavapa_gui::VideoFile::getFramerate ( ) const [inline], [override], [virtual]

Retrieves the framerate.

Returns

The framerate or 0.0 if not supported.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.11 virtual FrameTime cavapa_gui::VideoFile::getLength ( ) const [inline], [final], [virtual]

Retrieves the video file length.

Returns

The video length in milliseconds.

6.41.3.12 virtual FrameTime cavapa_gui::VideoFile::getPosition ( ) const [inline], [final], [override], [virtual]

Retrieves the current time position of the source.

Returns

The source time position.

Implements cavapa_gui::Source.

6.41.3.13 virtual cv::Size cavapa_gui::VideoFile::getResolution ( ) const [inline], [override], [virtual]

Retrieves the resolution of the source.

Returns

The source resolution.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.14 virtual SourceType cavapa_gui::VideoFile::getSourceType ( ) const [inline], [override], [virtual]

Returns the type of the source.

Returns

The source type.

Reimplemented from cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.15 SourceStats cavapa_gui::VideoFile::getStats ( ) [override], [virtual]

Retrieves the source statistical information.

Returns

The source statistics.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.16 virtual bool cavapa_gui::VideoFile::hasNew ( ) [inline], [override], [virtual]

Checks whether the device has a new image to be retrieved or not.

This checks if the FrameTime on the buffered image differs from what was retrieved on the last get().

Returns

True if a new image available or false otherwise.

Remarks

This is protected by a mutex and should not be used when time is critical. Use it only when you know that any lag produced by this is not going to be a problem.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.17 virtual bool cavapa_gui::VideoFile::isOpen ( ) const [inline], [override], [virtual]

Checks whether the source has been initialized or not.

Returns

True if it is initialized or false otherwise.

Implements cavapa_gui::Source.

6.41.3.18 virtual bool cavapa_gui::VideoFile::isPlaying ( ) const [inline], [override], [virtual]

Checks whether the source is playing or not.

Returns

True if the source is playing or false otherwise.

Implements cavapa_gui::Source.

6.41.3.19 bool cavapa_gui::VideoFile::open (const std::string & *filename*, bool *get_first*, FrameTime *start_point*) [virtual]

Opens a video file.

Parameters

filename	The path of the video file.
get_first	If the first image of the video should be retrieved or not. This is useful in most cases, but if we are opening multiple video files of the same source it might be wise to set this false. This way a worker thread of the frame retrieval is not automatically started.
start_point	The video start point. This will be used as the initial position of the video. All seekTime() calls will be based on the position.

Returns

True if the file was opened successfully or false otherwise.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.20 virtual void cavapa_gui::VideoFile::play ( ) [inline], [override], [virtual]

Starts frame updates on the file.

After the video file has been set to play, it will start to retrieve new frames for each get() call.

Reimplemented from cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.21 bool cavapa_gui::VideoFile::record (const std::string & *filename*, const std::string & *codec* = " ") [inline], [final], [override], [virtual]

Starts the recording of the source to the file.

This can also be used to start an entirely new video file during the recording. The recording file will change immediately.

Parameters

filename	The path of the recording file.
codec	The codec to be used for recording.

Returns

True if the recording started or false otherwise.

Parameters

filename	Not used.
codec	Not used.

Returns

False.

Remarks

Does nothing. It is safe to call this.

Implements cavapa_gui::Source.

6.41.3.22 virtual void cavapa_gui::VideoFile::release ( ) [inline], [virtual]

Forces the release of the worker thread.

This is used when multiple video files are combined as a one source of video. When the particular video file is currently not being accessed, the worker thread can be released to free up system's resources. The worker thread is again automatically started when the get() function is called. These threads are sleeping when no frames are being fetched, but the system still has limited amount of threads at its disposal. If you plan to use hundreds of them, then you should release the ones you don't need at the time.

Remarks

Calling the function might produce a slight lag as it waits for the thread to finish first.

6.41.3.23 bool cavapa_gui::VideoFile::seekTime (FrameTime *pos*)

Seeks a specific position of the video frame.

This does not use the actual start position of the video.

Parameters

pos	The new desired video frame position.

Returns

True if seeking was successful or false otherwise.

Remarks

Unknown when this actually does fail, might be never on video files.

6.41.3.24 virtual void cavapa_gui::VideoFile::setBarrelCorrection (double *amount*) [inline], [override], [virtual]

Sets the new barrel correction value.

The value is used to correct barrel effect (lens correction) on each image that the source will return.

Parameters

amount	The barrel value.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.25 void cavapa_gui::VideoFile::setCurrentAsStartPoint ( ) [inline]

Sets the current video time as the seeking start point.

Remarks

This is usually given in the open() function but there might be a reason to do this manually too.

6.41.3.26 bool cavapa_gui::VideoFile::setResolution (const cv::Size & *new_size*) [inline], [final], [override], [virtual]

Should change resolution, but does not have an effect on video files.

It is safe to call this.

Parameters

new_size	Not used.

Returns

False.

Remarks

This method does nothing.

Implements cavapa_gui::Source.

6.41.3.27 static void cavapa_gui::VideoFile::setSpeed (double *multiplier*) [inline], [static]

Sets playing speed of the video file.

Parameters

multiplier	The speed multiplier for the video file. The default is 1.0.

6.41.3.28 bool cavapa_gui::VideoFile::skipFrame (int *step*) [virtual]

Skips the specific amount of frames on the video.

The positive numbers skip forward and negatives skip backward. The frame will be limited to the actual limits of the video, skipping 10 million frames will just seek to the last frame of the video.

Parameters

step	The desired frame change.

Returns

True if the skipping was successful or false otherwise.

Remarks

Unknown when this actually does fail. It might be never on video files.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.29 virtual FrameTime cavapa_gui::VideoFile::startPoint ( ) const [inline], [final], [virtual]

Returns the start point of the video file.

Returns

The first frame of the initial position.

6.41.3.30 virtual void cavapa_gui::VideoFile::stop ( ) [inline], [override], [virtual]

Will stop the file from playing.

Any calls to function get() will return the same buffered image and will not advance the videos internal clock. Use play() to start retrieving images again.

Implements cavapa_gui::Source.

Reimplemented in cavapa_gui::VideoFileSet.

6.41.3.31 int cavapa_gui::VideoFile::totalFrames ( ) const [inline]

Returns the total count of the frames in the video file.

Returns

The frame count. 0 usually means that the video file is broken and cannot determine the total frame count. They can still be played.

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

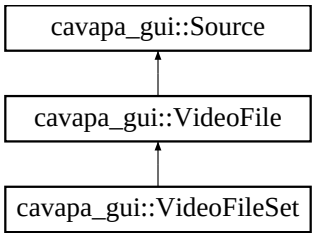
- source/videofile.h
- source/videofile.cpp

6.42 cavapa_gui::VideoFileSet Class Reference

The extension to the VideoFile class for handling multiple files.

`#include <videofileset.h>`

Inheritance diagram for cavapa_gui::VideoFileSet:



Public Types

- enum OpenError {
  FILE_OPEN_FAILED, **INVALID_FRAMECOUNT**, **OK**, **WRONG_FRAMERATE**,
  WRONG_RESOLUTION }

  Values that the VideoFileSet::open() function can return.

Public Member Functions

- VideoFileSet (SourceID desired_id=UNDEFINED_SOURCE)

  Creates a new source.
- FrameCapture get (FrameTime passed_time) override

  Retrieves the next frame from the camera.
- double getBarrelCorrection () override

  Returns the barrel correction applied to the source frames.
- std::string getDescription () const override

  Retrieves the description of the source.
- double getFramerate () const override

  Retrieves the framerate.
- std::vector< std::string > getFilenames () const override

  Returns the filenames of the source.
- cv::Size getResolution () const override

  Retrieves the resolution of the source.

- SourceType getSourceType () const override

  Returns the type of the source.
- SourceStats getStats () override

  Retrieves the source statistical information.
- bool hasNew () override

  Checks whether the device has a new image to be retrieved or not.
- bool open (const std::string &filename, bool get_first, FrameTime start_point) override

  Opens a video file.
- OpenError open (const std::vector< std::string > &paths, FrameTime start_point)

  Opens the multifile video source.
- void play () override

  Starts frame updates on the file.
- bool skipFrame (int step) override

  Skips the specific amount of frames on the video.
- void setBarrelCorrection (double amount) override

  Sets the new barrel correction value.
- void stop () override

  Will stop the file from playing.

Additional Inherited Members

6.42.1 Detailed Description

The extension to the VideoFile class for handling multiple files.

When there is a need to play multiple video files in sequence, the VideoFile class is not enough. This class provides functionality to automatically switch between the files to provide a seamless playback.

Author

Petri Partanen

Remarks

The class was created in a short time period and it is therefore a less effective way of doing things because the class could provide a better way of threading the frame retrieval from multiple sources. Currently each video file will end up having it's own std::thread.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 cavapa_gui::VideoFileSet::VideoFileSet (SourceID *desired_id* = UNDEFINED_SOURCE) [inline]

Creates a new source.

Parameters

desired_id	The Desired ID number. Use UNDEFINED_SOURCE if you do not want to define it to a specific ID. This does not check if a source with the given ID already exists. The ID-number will not be given to other new sources with UNDEFINED_SOURCE ID numbers.

6.42.3 Member Function Documentation

6.42.3.1 FrameCapture cavapa_gui::VideoFileSet::get (FrameTime *passed_time*) [override], [virtual]

Retrieves the next frame from the camera.

The time supplied to the function informs the source about how much time has passed since the last image retrieval. The source will depend on its internal clock, either retrieving a new image or returning the buffered image that it retrieved previously.

Parameters

passed_time	The time passed since the last call (in milliseconds).

Returns

The latest frame capture. This will actually return a reference to the original image, so you are NOT allowed to edit it! Otherwise the source buffer will be altered too. This is just the way OpenCV handles Mat memory.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.2 double cavapa_gui::VideoFileSet::getBarrelCorrection ( ) [override], [virtual]

Returns the barrel correction applied to the source frames.

Returns

The barrel effect value.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.3 string cavapa_gui::VideoFileSet::getDescription ( ) const [override], [virtual]

Retrieves the description of the source.

Returns

The source description.
The description of the source will contain the names of the video files separated by a character ';'.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.4 vector< string > cavapa_gui::VideoFileSet::getFilenames ( ) const [override], [virtual]

Returns the filenames of the source.

Returns

The video filenames.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.5 double cavapa_gui::VideoFileSet::getFramerate ( ) const [override], [virtual]

Retrieves the framerate.

Returns

The framerate or 0.0 if not supported.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.6 cv::Size cavapa_gui::VideoFileSet::getResolution ( ) const [override], [virtual]

Retrieves the resolution of the source.

Returns

The source resolution.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.7 SourceType cavapa_gui::VideoFileSet::getSourceType ( ) const [inline], [override], [virtual]

Returns the type of the source.

Returns

The source type.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.8 SourceStats cavapa_gui::VideoFileSet::getStats ( ) [override], [virtual]

Retrieves the source statistical information.

Returns

The source statistics.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.9 bool cavapa_gui::VideoFileSet::hasNew ( ) [inline], [override], [virtual]

Checks whether the device has a new image to be retrieved or not.

This checks if the FrameTime on the buffered image differs from what was retrieved on the last get().

Returns

True if a new image available or false otherwise.

Remarks

This is protected by a mutex and should not be used when time is critical. Use it only when you know that any lag produced by this is not going to be a problem.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.10 bool cavapa_gui::VideoFileSet::open (const std::string & *filename*, bool *get_first*, FrameTime *start_point*) [inline], [override], [virtual]

Opens a video file.

Parameters

filename	The path of the video file.
get_first	If the first image of the video should be retrieved or not. This is useful in most cases, but if we are opening multiple video files of the same source it might be wise to set this false. This way a worker thread of the frame retrieval is not automatically started.
start_point	The video start point. This will be used as the initial position of the video. All seekTime() calls will be based on the position.

Returns

True if the file was opened successfully or false otherwise.

Remarks

For a single video it is recommended to use the VideoFile class.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.11 VideoFileSet::OpenError cavapa_gui::VideoFileSet::open (const std::vector< std::string > & *paths*, FrameTime *start_point*)

Opens the multifile video source.

Parameters

paths	The paths of the video files.
start_point	The video start point. This will be used as the initial position of the video. All seekTime() calls will be based on this position.

Returns

The related VideoFileSetError key for the success or failure.

6.42.3.12 void cavapa_gui::VideoFileSet::play ( ) [override], [virtual]

Starts frame updates on the file.

After the video file has been set to play, it will start to retrieve new frames for each get() call.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.13 void cavapa_gui::VideoFileSet::setBarrelCorrection (double *amount*) [override], [virtual]

Sets the new barrel correction value.

The value is used to correct barrel effect (lens correction) on each image that the source will return.

Parameters

amount	The barrel value.

Remarks

This might produce lag as there are lot's of mutex locks.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.14 bool cavapa_gui::VideoFileSet::skipFrame (int *step*) [override], [virtual]

Skips the specific amount of frames on the video.

The positive numbers skip forward and negatives skip backward. The frame will be limited to the actual limits of the video, skipping 10 million frames will just seek to the last frame of the video.

Parameters

step	The desired frame change.

Returns

True if the skipping was successful or false otherwise.

Remarks

Unknown when this actually does fail. It might be never on video files.

Reimplemented from cavapa_gui::VideoFile.

6.42.3.15 void cavapa_gui::VideoFileSet::stop ( ) [override], [virtual]

Will stop the file from playing.

Any calls to function get() will return the same buffered image and will not advance the videos internal clock. Use play() to start retrieving images again.

Reimplemented from cavapa_gui::VideoFile.

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

- source/videofileset.h
- source/videofileset.cpp