

Reference Manual

Generated by Doxygen 1.5.6

Mon Jan 19 03:19:32 2009

Contents

1	Class Index	1
1.1	Class List	1
2	Class Documentation	3
2.1	FlowChannel Class Reference	3
2.1.1	Detailed Description	4
2.1.2	Member Function Documentation	4
2.1.2.1	copyValues	4
2.2	FlowData Class Reference	5
2.2.1	Detailed Description	6
2.3	FlowGeometry Class Reference	7
2.3.1	Detailed Description	9
2.3.2	Member Function Documentation	9
2.3.2.1	getInterpolationAt	9
2.4	GLWidget Class Reference	10
2.4.1	Detailed Description	11
2.4.2	Constructor & Destructor Documentation	11
2.4.2.1	GLWidget	11
2.4.3	Member Function Documentation	11
2.4.3.1	minimumSizeHint	11
2.4.3.2	sizeHint	11
2.4.3.3	setFlowData	12
2.4.3.4	setBackgroundColorsPtr	12
2.4.3.5	setArrowColorsPtr	12
2.4.3.6	setStreamlineColorsPtr	12
2.4.3.7	setBackgroundColors	12
2.4.3.8	setArrowColors	12
2.4.3.9	setStreamlineColors	12

2.4.3.10	setDrawBackground	12
2.4.3.11	setBackgroundChannel	13
2.4.3.12	setDrawArrows	13
2.4.3.13	setArrowQuantity	13
2.4.3.14	setArrowColorChannel	13
2.4.3.15	setDrawStreamlines	13
2.4.3.16	setStreamlineMode	13
2.4.3.17	setStreamlineDtest	14
2.4.3.18	setStreamlineDsep	14
2.4.3.19	setStreamlineDt	14
2.4.3.20	setStreamlinePeriod	14
2.4.3.21	setStreamlineLineMode	14
2.4.3.22	setStreamlineColorChannel	14
2.4.3.23	initializeGL	14
2.4.3.24	paintGL	15
2.4.3.25	resizeGL	15
2.4.3.26	mousePressEvent	15
2.4.3.27	mouseReleaseEvent	15
2.4.3.28	mouseMoveEvent	15
2.4.3.29	wheelEvent	15
2.4.3.30	draw	15
2.4.3.31	normalizeAngle	16
2.4.3.32	file2string	16
2.4.3.33	loadShader	16
2.4.3.34	printLog	16
2.4.3.35	drawArrowPlot	16
2.4.3.36	calcEuler	17
2.4.3.37	calcRunge	17
2.4.3.38	calculateBackground	17
2.4.3.39	drawStreamline	17
2.4.3.40	testRange	17
2.4.3.41	testCell	17
2.4.3.42	getTapering	18
2.4.3.43	testCellTapering	18
2.4.3.44	testSLDist	18
2.4.3.45	calculateEvenStreamlines	18

2.4.3.46	addPoint2Grid	18
2.4.3.47	calcStartCandidates	19
2.4.3.48	initStreamlines	19
2.4.3.49	removeLastPointfromCell	19
2.5	GradientEditor Class Reference	20
2.5.1	Detailed Description	20
2.5.2	Constructor & Destructor Documentation	20
2.5.2.1	GradientEditor	20
2.5.3	Member Function Documentation	21
2.5.3.1	setGradientStops	21
2.5.3.2	getColorsPtr	21
2.5.3.3	updateColorTexture	21
2.5.3.4	setGradientEditorRangeLabel	21
2.5.3.5	pointsUpdated	21
2.5.3.6	colorDialog	21
2.5.3.7	moveColorsWithAlpha	21
2.5.3.8	newColorPoints	22
2.5.3.9	deleteColorPoints	22
2.5.3.10	gradientStopsChanged	22
2.5.3.11	setEnabledSignal	22
2.5.3.12	resizeEvent	22
2.6	HoverPoints Class Reference	23
2.6.1	Detailed Description	24
2.6.2	Member Enumeration Documentation	24
2.6.2.1	PointShape	24
2.6.2.2	LockType	24
2.6.2.3	SortType	24
2.6.2.4	ConnectionType	24
2.6.3	Constructor & Destructor Documentation	24
2.6.3.1	HoverPoints	24
2.6.4	Member Function Documentation	24
2.6.4.1	eventFilter	24
2.6.4.2	paintPoints	25
2.6.4.3	boundingRect	25
2.6.4.4	setBoundingRect	25
2.6.4.5	points	25

2.6.4.6	setPoints	25
2.6.4.7	pointSize	25
2.6.4.8	setPointSize	25
2.6.4.9	sortType	26
2.6.4.10	setSortType	26
2.6.4.11	connectionType	26
2.6.4.12	setConnectionType	26
2.6.4.13	setConnectionPen	26
2.6.4.14	setShapePen	26
2.6.4.15	setShapeBrush	26
2.6.4.16	setPointLock	27
2.6.4.17	setEditable	27
2.6.4.18	editable	27
2.6.4.19	setEnabled	27
2.6.4.20	setDisabled	27
2.6.4.21	pointsChanged	27
2.6.4.22	chooseColor	28
2.6.4.23	alphaChanged	28
2.6.4.24	alphaPointCreated	28
2.6.4.25	alphaPointDeleted	28
2.6.4.26	firePointChange	28
2.6.4.27	movePoint	28
2.6.4.28	newPoint	28
2.6.4.29	deletePoint	29
2.6.4.30	fireChooseColor	29
2.7	ShadeWidget Class Reference	30
2.7.1	Detailed Description	30
2.7.2	Member Enumeration Documentation	30
2.7.2.1	ShadeType	30
2.7.3	Constructor & Destructor Documentation	30
2.7.3.1	ShadeWidget	30
2.7.4	Member Function Documentation	31
2.7.4.1	setGradientStops	31
2.7.4.2	paintEvent	31
2.7.4.3	minimumSizeHint	31
2.7.4.4	sizeHint	31

2.7.4.5	points	31
2.7.4.6	hoverPoints	31
2.7.4.7	colorAt	31
2.7.4.8	colorsChanged	32
2.7.4.9	generateShade	32
2.8	vec3 Class Reference	33
2.8.1	Detailed Description	34
2.9	Window Class Reference	35
2.9.1	Detailed Description	35
2.9.2	Constructor & Destructor Documentation	35
2.9.2.1	Window	35
2.9.3	Member Function Documentation	35
2.9.3.1	setDefaultGradientStops	35
2.9.3.2	setOpenFileName	35
2.9.3.3	saveGradientStops	36
2.9.3.4	setBackgroundColorEditorRangeLabel	36
2.9.3.5	setArrowColorEditorRangeLabel	36
2.9.3.6	setStreamlineColorEditorRangeLabel	36
2.9.3.7	backgroundColorTextureChanged	36
2.9.3.8	arrowColorTextureChanged	36
2.9.3.9	streamlineColorTextureChanged	36

Chapter 1

Class Index

1.1 Class List

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

FlowChannel (Handles one scalar field of floats defined for each cell)	3
FlowData (Class managing the data sets and related stuff like data loading, channels creation etc)	5
FlowGeometry (Class for handling the geometry == rectangular grids organized in vertices and cells)	7
GLWidget	10
GradientEditor	20
HoverPoints	23
ShadeWidget	30
vec3	33
Window	35

Chapter 2

Class Documentation

2.1 FlowChannel Class Reference

Handles one scalar field of floats defined for each cell.

```
#include <FlowChannel.h>
```

Public Member Functions

- void `setValue` (int vtxID, float val)
sets the value of the given vertex
- void `copyValues` (float *rawdata, int vtxSize, int offset)
takes an array containing all attributes for a vertex and copies the j-th attribute to this channel
- float `getValue` (vec3 pos)
returns the value at given position in data set coordinates (from 0 to dimX or dimY)
- float `getValue` (int vtxID)
returns the value of the given vertex
- float `getValueNormPos` (vec3 pos)
returns the value at given position in normalized coordinates for each dimension <0..1>
- float `getValueNormPos` (float x, float y)
returns the value at given position in normalized coordinates for each dimension <0..1>
- float `normalizeValue` (float val)
scales the value according to the channel minimum and maximum, so that it lies inside of <0,1>
- float `getMin` ()
returns the minimum value found in the channel
- float `getMax` ()
returns the maximum value found in the channel

- float `getRange ()`
returns the range = max - min

Private Attributes

- `FlowGeometry * geom`
reference to the geometry structure
- float * `values`
channel data storage
- float `minimum`
minimum value (of all cells in a single time step)
- float `maximum`
maximum value (of all cells in a single time step)

2.1.1 Detailed Description

Handles one scalar field of floats defined for each cell.

More dimensional vectors are split into components. E.g. a 3D velocity vector gets stored in three `FlowChannels`. A `FlowChannel` stores data only from one time step, it is not aware of any time related information.

2.1.2 Member Function Documentation

2.1.2.1 void `FlowChannel::copyValues (float * rawdata, int vtxSize, int offset)`

takes an array containing all attributes for a vertex and copies the j-th attribute to this channel

This method is used by the loading of data sets.

Parameters:

- rawdata* data gained directly from the file, without any processing. It contains all channels for all cells. Please note, there is no time information considered here.
- vtxSize* number of channels per cell (incl. velocity vector size)
- offset* offset of the parameter loaded into this channel

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

- `FlowChannel.h`
- `FlowChannel.cpp`

2.2 FlowData Class Reference

class managing the data sets and related stuff like data loading, channels creation etc.

```
#include <FlowData.h>
```

Public Member Functions

- [FlowData](#) ()
initializes the channel storage
- [~FlowData](#) ()
destroys all created channels
- bool [loadDataset](#) (string filename, bool bigEndian)
Loads a dataset, returns true if everything successful. You have to specify the byte order used in the data.
- int [getNumTimesteps](#) ()
Returns the number of timesteps.
- int [createChannel](#) ()
creates a new channel and returns it's address in the channels array (line 28)
- void [deleteChannel](#) (int i)
deletes the channel and all it's data at given adress
- [FlowChannel](#) * [getChannel](#) (int i)
returns a pointer to the instance of channel at given adress. This is the only way to access the channels storage (at line 28)
- int [createChannelGeometry](#) (int dimension)
creates a new channel containing the geometrical information of the given dimension (x = 0, y = 1). Returns address of the created channel in the channels array (line 28)
- int [createChannelVectorLength](#) (int chX, int chY, int chZ=-1)
creates a new channel containing the vector lengths for the given channels (channels given by IDs). Returns address of the created channel in the channels array (line 28)
- int [createChannelVectorLength](#) ([FlowChannel](#) *chX, [FlowChannel](#) *chY, [FlowChannel](#) *chZ=NULL)
creates a new channel containing the vector lengths for the given channels (channels given by reference). Returns address of the created channel

Private Attributes

- bool [loaded](#)
Is there any data loaded?
- int [timesteps](#)

Number of timesteps.

- [FlowGeometry geometry](#)

Stores the underlying geometry.

- bool [freeChannel](#) [max_channels]

is the channel slot free?

- [FlowChannel * channels](#) [max_channels]

stores the values of data channels for one time step. For time-dependent data, the best solution is to create a separate class handling channels in one timestep and to instantiate this class for all timesteps.

2.2.1 Detailed Description

class managing the data sets and related stuff like data loading, channels creation etc.

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

- FlowData.h
- FlowData.cpp

2.3 FlowGeometry Class Reference

class for handling the geometry == rectangular grids organized in vertices and cells

```
#include <FlowGeometry.h>
```

Public Member Functions

- [~FlowGeometry \(\)](#)
deletes the allocated geometry storage
- bool [getInterpolationAt \(vec3 pos, int *vtxID, float *coef\)](#)
Returns true if inside. Stores the vertex indices and interpolation weights for the given position in the arrays.
- bool [readFromFile \(char *header, FILE *fp, bool bigEndian\)](#)
reads the geometry grid data from a file
- int [getDimX \(\)](#)
returns the number of vertices in X dimension
- int [getDimY \(\)](#)
returns the number of vertices in Y dimension
- int [getDimZ \(\)](#)
returns the number of vertices in Z dimension, is always 1
- float [getMinX \(\)](#)
returns the minimum in the X dimension
- float [getMaxX \(\)](#)
returns the maximum in the X dimension
- float [getMinY \(\)](#)
returns the minimum in the Y dimension
- float [getMaxY \(\)](#)
returns the maximum in the Y dimension
- int [getRightNeigh \(int vtxID\)](#)
returns the vertex ID of the neighbour on its right
- int [getTopNeigh \(int vtxID\)](#)
returns the vertex ID of the neighbour on its top
- int [getLeftNeigh \(int vtxID\)](#)
returns the vertex ID of the neighbour on its left
- int [getBottomNeigh \(int vtxID\)](#)
returns the vertex ID of the neighbour on its bottom

- [vec3 normalizeCoords](#) ([vec3](#) pos)
compression of coordinates in each dimension separately, returns values scaled to <0,1>
- [vec3 unNormalizeCoords](#) ([vec3](#) pos)
inverts the compression. From values of <0,1> it restores the real geometrical coordinates

Private Member Functions

- [int getVtx](#) (int x, int y)
returns general vtxID for the vertex array indexes
- [int getVtxX](#) (int vtxID)
returns X index for the general vtxID
- [int getVtxY](#) (int vtxID)
returns Y index for the general vtxID
- [int getXylvtx](#) ([vec3](#) pos)
returns X index of the last vertex lying left to the position x and the Y index of the last vertex lying under the position y
- [vec3 getPos](#) (int vtxID)
returns the position of the vertex
- [float getPosX](#) (int vtxID)
returns the x position of the vertex
- [float getPosY](#) (int vtxID)
returns the y position of the vertex
- [int getNearestVtx](#) ([vec3](#) pos)
a very slow and dumb routine, that finds the nearest vertex to the given position

Private Attributes

- [int dim](#) [2]
resolution of the data for the dimensions X, Y
- [vec3 boundaryMin](#)
minimum boundary values for the dataset geometry sotred as {minX, minY}
- [vec3 boundaryMax](#)
maximum boundary values for the dataset geometry sotred as {maxX, maxY}
- [vec3 boundarySize](#)
boundary sizes for the dataset geometry sotred as {maxX - minX, maxY - minY}

- `vec3 * geometryData`
Storage for the geometry.
- `bool isFlipped`
indicates whether the x and y axes have to be swapped

Friends

- class `FlowData`

2.3.1 Detailed Description

class for handling the geometry == rectangular grids organized in vertices and cells

2.3.2 Member Function Documentation

2.3.2.1 `bool FlowGeometry::getInterpolationAt (vec3 pos, int * vtxID, float * coef)`

Returns true if inside. Stores the vertex indices and interpolation weights for the given position in the arrays.

Stores the indexes and weights of vertices surrounding the given position. This information can be used later on for interpolation of channel values.

Parameters:

- pos* geometrical position for the lookup
- vtxID* list of surrounding vertices (given by vertex ID)
- coef* list of surrounding vertex weights (sum == 1.0)

Returns:

- true if the given position is inside of the geometrical boundaries

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

- FlowGeometry.h
- FlowGeometry.cpp

2.4 GLWidget Class Reference

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

Public Slots

- void [setBackgroundColors](#) ()
- void [setArrowColors](#) ()
- void [setStreamlineColors](#) ()
- void [setDrawBackground](#) (bool state)
- void [setBackgroundColorChannel](#) (int channel)
- void [setDrawArrows](#) (bool state)
- void [setArrowQuantity](#) (int quantity)
- void [setArrowColorChannel](#) (int channel)
- void [setDrawStreamlines](#) (bool state)
- void [setStreamlineMode](#) (int mode)
- void [setStreamlineDtest](#) (int newDtest)
- void [setStreamlineDsep](#) (int newDsep)
- void [setStreamlineDt](#) (int newDt)
- void [setStreamlinePeriod](#) (int Period)
- void [setStreamlineLineMode](#) (int mode)
- void [setStreamlineColorChannel](#) (int channel)

Public Member Functions

- [GLWidget](#) (QWidget *parent=0)
- QSize [minimumSizeHint](#) () const
- QSize [sizeHint](#) () const
- void [setFlowData](#) (FlowData *newFlowData)
- void [setBackgroundColorsPtr](#) (QRgb *colorsPtr)
- void [setArrowColorsPtr](#) (QRgb *colorsPtr)
- void [setStreamlineColorsPtr](#) (QRgb *colorsPtr)

Protected Member Functions

- void [initializeGL](#) ()
- void [paintGL](#) ()
- void [resizeGL](#) (int width, int height)
- void [mousePressEvent](#) (QMouseEvent *event)
- void [mouseReleaseEvent](#) (QMouseEvent *event)
- void [mouseMoveEvent](#) (QMouseEvent *event)
- void [wheelEvent](#) (QWheelEvent *event)

Private Member Functions

- void `draw` ()
- void `normalizeAngle` (int *angle)
- char * `file2string` (const std::string &strFilename)
- GLuint `loadShader` (const std::string &strFilename)
- void `printLog` (GLuint obj)
- void `drawArrowPlot` (const int xScaler, const int yScaler)
- void `calcEuler` (float x, float y)
- void `calcRunge` (float x, float y)
- void `calculateBackground` ()
- void `drawStreamline` (int mode)
- bool `testRange` (`vec3` point)
- bool `testCell` (int cellx, int celly, `vec3` point)
- float `getTapering` (`vec3` point, vector< `vec3` > line, unsigned int i)
- float `testCellTapering` (int cellx, int celly, `vec3` point, vector< `vec3` > line, unsigned int i)
- bool `testSLDist` (`vec3` point, vector< `vec3` > line, unsigned int i)
- void `calculateEvenStreamlines` ()
- void `addPoint2Grid` (`vec3` point)
- void `calcStartCandidates` (`vec3` pos)
- void `initStreamlines` ()
- void `removeLastPointfromCell` (`vec3` point)

2.4.1 Detailed Description

Sets up a Qt OpenGL Canvas Widget.

2.4.2 Constructor & Destructor Documentation

2.4.2.1 GLWidget::GLWidget (QWidget *parent = 0)

Sets up a Qt OpenGL Canvas.

Parameters:

parent is the parent window of the canvas.

2.4.3 Member Function Documentation

2.4.3.1 QSize GLWidget::minimumSizeHint () const

Returns:

The minimum size for the Widget.

2.4.3.2 QSize GLWidget::sizeHint () const

Returns:

The size hint for the Widget.

2.4.3.3 void GLWidget::setFlowData (FlowData * newFlowData)

Sets up the Canvas for a new Flowdata object.

Parameters:

newFlowData is the new Flowdata.

2.4.3.4 void GLWidget::setBackgroundColorsPtr (QRgb * colorsPtr)

Initializes the pointer to the background transferfunction array.

Parameters:

colorsPtr is the pointer to the new array.

2.4.3.5 void GLWidget::setArrowColorsPtr (QRgb * colorsPtr)

Initializes the pointer to the arrow transferfunction array.

Parameters:

colorsPtr is the pointer to the new array.

2.4.3.6 void GLWidget::setStreamlineColorsPtr (QRgb * colorsPtr)

Initializes the pointer to the streamline transferfunction array.

Parameters:

colorsPtr is the pointer to the new array.

2.4.3.7 void GLWidget::setBackgroundColors () [slot]

Notifies that the background transferfunction has changed and redraws the scene.

2.4.3.8 void GLWidget::setArrowColors () [slot]

Notifies that the arrow transferfunction has changed and redraws the scene.

2.4.3.9 void GLWidget::setStreamlineColors () [slot]

Notifies that the streamline transferfunction has changed and redraws the scene.

2.4.3.10 void GLWidget::setDrawBackground (bool state) [slot]

Sets the current background Rendermode.

Parameters:

state holds if the background should be drawn or not.

2.4.3.11 void GLWidget::setBackgroundChannel (int *channel*) [slot]

Sets the current [FlowChannel](#) for the background transferfunction.

Parameters:

channel the new Channel.

2.4.3.12 void GLWidget::setDrawArrows (bool *state*) [slot]

Sets the current arrow Rendermode.

Parameters:

state holds if the background should be drawn or not.

2.4.3.13 void GLWidget::setArrowQuantity (int *quantity*) [slot]

Sets the current arrow quantity.

Parameters:

quantity holds how many arrows should be drawn.

2.4.3.14 void GLWidget::setArrowColorChannel (int *channel*) [slot]

Sets the current [FlowChannel](#) for the arrow transferfunction.

Parameters:

channel the new Channel.

2.4.3.15 void GLWidget::setDrawStreamlines (bool *state*) [slot]

Sets the current streamline Rendermode.

Parameters:

state holds if the background should be drawn or not.

2.4.3.16 void GLWidget::setStreamlineMode (int *mode*) [slot]

Sets the current streamline integrationmode.

Parameters:

mode holds the integrationmode.

2.4.3.17 void GLWidget::setStreamlineDtest (int *newDtest*) [slot]

Sets the relation of the dtest to the dsep parameter for the streamline algorithm.

Parameters:

newDtest new dtest relation parameter.

2.4.3.18 void GLWidget::setStreamlineDsep (int *newDsep*) [slot]

Sets the relation of the dsep parameter to the shorter dimension of the dataset.

Parameters:

newDsep new dsep relation parameter.

2.4.3.19 void GLWidget::setStreamlineDt (int *newDt*) [slot]

Sets the relation of the dt to the dtest parameter for the streamline algorithm.

Parameters:

newDt new dt relation parameter.

2.4.3.20 void GLWidget::setStreamlinePeriod (int *newPeriod*) [slot]

Sets length of the period for the streamline texture visualization mode.

Parameters:

newPeriod new period length.

2.4.3.21 void GLWidget::setStreamlineLineMode (int *mode*) [slot]

Sets the current streamline visualization style.

Parameters:

mode holds the streamline visualization style.

2.4.3.22 void GLWidget::setStreamlineColorChannel (int *channel*) [slot]

Sets the current [FlowChannel](#) for the streamline transferfunction.

Parameters:

channel the new Channel.

2.4.3.23 void GLWidget::initializeGL () [protected]

Initializes the OpenGL environment including antialiasing.

2.4.3.24 void GLWidget::paintGL () [protected]

Applies affine transformations and starts the drawing of the frame.

2.4.3.25 void GLWidget::resizeGL (int *width*, int *height*) [protected]

Handles resizing of the OpenGL canvas.

Parameters:

width is the new width of the canvas.

height is the new height of the canvas.

2.4.3.26 void GLWidget::mousePressEvent (QMouseEvent * *event*) [protected]

Reduces the stepsize of the raycastingshader and initialises volume rotations.

Parameters:

event is the mousePressEvent.

2.4.3.27 void GLWidget::mouseReleaseEvent (QMouseEvent * *event*) [protected]

Increases the stepsize of the raycastingshader after finished volume rotations.

Parameters:

event is the mouseReleaseEvent.

2.4.3.28 void GLWidget::mouseMoveEvent (QMouseEvent * *event*) [protected]

Handles mouse dragging and rotation of the volume.

Parameters:

event is the mouseMoveEvent.

2.4.3.29 void GLWidget::wheelEvent (QWheelEvent * *event*) [protected]

Handles mouseWheelEvents.

Parameters:

event is the mouseWheelEvent.

2.4.3.30 void GLWidget::draw () [private]

Draws texture, arrow and streamline layers (if activated).

2.4.3.31 void GLWidget::normalizeAngle (int * *angle*) [private]

Keeps angle within [0 360] interval.

Parameters:

angle is the pointer to the angle.

2.4.3.32 char * GLWidget::file2string (const std::string & *strFilename*) [private]

Loads a binary file into a Charpointer.

Parameters:

strFilename ist the filename of the file to load.

Returns:

The Charpointer with the content of the file.

2.4.3.33 GLuint GLWidget::loadShader (const std::string & *strFilename*) [private]

Handels loading, compiling and attaching of a vertex-, fragmentshader pair to a shader program. * Loading requires 2 glsl shader files with "<filepath>.vert" and "<filepath>.frag" extension to be successful.

Parameters:

strFilename is the filepath of the shaders without extension.

Returns:

The id of the new shaderprogram.

2.4.3.34 void GLWidget::printLog (GLuint *obj*) [private]

Prints the shaderobject debug log.

Parameters:

obj specifies the shaderobject.

2.4.3.35 void GLWidget::drawArrowPlot (const int *xScaler*, const int *yScaler*) [private]

Draws the Arrowlayer.

Parameters:

xScaler density of arrows in x-direction.

yScaler density of arrows in y-direction.

2.4.3.36 void GLWidget::calcEuler (float x, float y) [private]

Calculates a streamline in both directions of the given point using the Euler integration method.

Parameters:

- x* component of the startpoint.
- y* component of the startpoint.

2.4.3.37 void GLWidget::calcRunge (float x, float y) [private]

Calculates a streamline in both directions of the given point using the Runge-Kutta integration method.

Parameters:

- x* component of the startpoint.
- y* component of the startpoint.

2.4.3.38 void GLWidget::calculateBackground () [private]

Creates the background texture layer.

2.4.3.39 void GLWidget::drawStreamline (int mode) [private]

Draws the streamline layer with the given visualization mode.

Parameters:

- mode* line draw mode (0: normal | 1: tapered | 2: glyphs | 3: texture)

2.4.3.40 bool GLWidget::testRange (vec3 point) [private]

Tests if the given point is within dtest range to an already existing point.

Parameters:

- point* point to test.

2.4.3.41 bool GLWidget::testCell (int cellx, int celly, vec3 point) [private]

Tests if the given point is within dtest range to a point in the specified cell.

Parameters:

- cellx* x-coordinate of the cell.
- celly* y-coordinate of the cell.
- point* point to test.

2.4.3.42 float GLWidget::getTapering (vec3 *point*, vector< vec3 > *line*, unsigned int *i*)
[private]

Tests if the given point is within dsep range to an already existing point and returns tapering thickness.

Parameters:

point point to test.

line the line containing the testpoint.

i index of the testpoint within the line (to prevent getting hits from neighbours).

2.4.3.43 float GLWidget::testCellTapering (int *cellx*, int *celly*, vec3 *point*, vector< vec3 > *line*, unsigned int *i*) [private]

Tests if the given point is within dsep range to a point in the specified cell and returns distance to the nearest point.

Parameters:

cellx x-coordinate of the cell.

celly y-coordinate of the cell.

point point to test.

line the line containing the testpoint.

i index of the testpoint within the line (to prevent getting hits from neighbours)

2.4.3.44 bool GLWidget::testSLDist (vec3 *point*, vector< vec3 > *line*, unsigned int *i*)
[private]

Tests if the given point is member of the given line and at least ceil(dsep/dt) indices away from i.

Parameters:

point point to test.

line the line containing the testpoint.

i index of the testpoint within the line.

2.4.3.45 void GLWidget::calculateEvenStreamlines () [private]

Calculates evenly-spaced streamlines according to the method of Bruno Jobard and Wilfrid Lefer.

2.4.3.46 void GLWidget::addPoint2Grid (vec3 *point*) [private]

Adds the given point to the correct cell of the grid.

Parameters:

point point to add.

2.4.3.47 void GLWidget::calcStartCandidates (vec3 *pos*) [private]

Calculates two new startpoints from the origin, validates and adds them to the startCandidates queue.

Parameters:

pos origin point.

2.4.3.48 void GLWidget::initStreamlines () [private]

Sets up the environment to recalculate the streamlines.

2.4.3.49 void GLWidget::removeLastPointfromCell (vec3 *point*) [private]

Removes the last added point from the cell of the given point.

Parameters:

point reference to the cell.

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

- glwidget.h
- glwidget.cpp

2.5 GradientEditor Class Reference

```
#include <gradienteditor.h>
```

Public Slots

- void [pointsUpdated](#) ()
- void [colorDialog](#) (int index)
- void [moveColorsWithAlpha](#) (int index, qreal newXPos)
- void [newColorPoints](#) (int index, qreal xPos)
- void [deleteColorPoints](#) (int index)

Signals

- void [gradientStopsChanged](#) (const QGradientStops &stops, [GradientEditor](#) *editor)
- void [setEnabledSignal](#) (bool enabled)

Public Member Functions

- [GradientEditor](#) (QWidget *parent)
- void [setGradientStops](#) (const QGradientStops &stops)
- QRgb * [getColorsPtr](#) ()
- void [updateColorTexture](#) ()
- void [setGradientEditorRangeLabel](#) (float min=0, float max=1)

Protected Member Functions

- void [resizeEvent](#) (QResizeEvent *event)

2.5.1 Detailed Description

The gradienteditor class is a alpha [ShadeWidget](#) with a range label.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 GradientEditor::GradientEditor (QWidget *parent)

Constructor that creates the color editor.

Returns:

The pointer to the created color editor.

2.5.3 Member Function Documentation

2.5.3.1 void GradientEditor::setGradientStops (const QGradientStops & stops)

Sets the gradient stops position and color from the given stops.

Parameters:

stops are the gradient stops which should be set.

2.5.3.2 QRgb * GradientEditor::getColorsPtr ()

Returns:

The pointer to the texture.

2.5.3.3 void GradientEditor::updateColorTexture ()

Updates the texture with the color currently set in ShadeWidgets.

2.5.3.4 void GradientEditor::setGradientEditorRangeLabel (float min = 0, float max = 1)

Sets the left and right label of the color editor.

Parameters:

min is value which should be displayed at the left label.

max is value which should be displayed at the right label.

2.5.3.5 void GradientEditor::pointsUpdated () [slot]

Updates the gradient stops position and color from the currently set points. Sets this gradient points at the alpha [ShadeWidget](#) and emits a gradientStopsChanged signal.

2.5.3.6 void GradientEditor::colorDialog (int index) [slot]

Creates a color dialog with which a new color can be set.

Parameters:

index is the index of the point for which the color should be changed.

2.5.3.7 void GradientEditor::moveColorsWithAlpha (int index, qreal newXPos) [slot]

Moves the color with the corresponding alpha point.

Parameters:

index is the index of the color which should be moved.

newXPos is the new x position to which the corresponding color should be moved.

2.5.3.8 void GradientEditor::newColorPoints (int *index*, qreal *xPos*) [slot]

Creates the color for the corresponding alpha point.

Parameters:

- index* is the index where the color should be saved.
- xPos* is the x position where the color should be created.

2.5.3.9 void GradientEditor::deleteColorPoints (int *index*) [slot]

Deletes the color of the corresponding alpha point.

Parameters:

- index* is the index where the color should be deleted.

2.5.3.10 void GradientEditor::gradientStopsChanged (const QGradientStops & *stops*, GradientEditor * *editor*) [signal]

A signal that is send on change of the gradient stops.

Parameters:

- stops* are the gradient stops which changed.
- editor* is the color editor for whom the stops changed.

2.5.3.11 void GradientEditor::setEnabledSignal (bool *enabled*) [signal]

A signal that is send on enabling/disabling a draw group.

Parameters:

- enabled* is the state which should be set for the [ShadeWidget](#).

2.5.3.12 void GradientEditor::resizeEvent (QResizeEvent * *event*) [protected]

Updates the gradient points at each [GradientEditor](#) resizing.

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

- [gradienteditor.h](#)
- [gradienteditor.cpp](#)

2.6 HoverPoints Class Reference

```
#include <hoverpoints.h>
```

Public Types

- enum [PointShape](#)
- enum [LockType](#)
- enum [SortType](#)
- enum [ConnectionType](#)

Public Slots

- void [setEnabled](#) (bool enabled)
- void [setDisabled](#) (bool disabled)

Signals

- void [pointsChanged](#) (const QPolygonF &points)
- void [chooseColor](#) (int index)
- void [alphaChanged](#) (int index, qreal newXPos)
- void [alphaPointCreated](#) (int index, qreal xPos)
- void [alphaPointDeleted](#) (int index)

Public Member Functions

- [HoverPoints](#) (QWidget *widget, [PointShape](#) shape)
- bool [eventFilter](#) (QObject *object, QEvent *event)
- void [paintPoints](#) ()
- QRectF [boundingRect](#) () const
- void [setBoundingRect](#) (const QRectF &boundingRect)
- QPolygonF [points](#) () const
- void [setPoints](#) (const QPolygonF &points)
- QSizeF [pointSize](#) () const
- void [setPointSize](#) (const QSizeF &size)
- [SortType](#) [sortType](#) () const
- void [setSortType](#) ([SortType](#) sortType)
- [ConnectionType](#) [connectionType](#) () const
- void [setConnectionType](#) ([ConnectionType](#) connectionType)
- void [setConnectionPen](#) (const QPen &pen)
- void [setShapePen](#) (const QPen &pen)
- void [setShapeBrush](#) (const QBrush &brush)
- void [setPointLock](#) (int pos, [LockType](#) lock)
- void [setEditable](#) (bool editable)
- bool [editable](#) () const
- void [firePointChange](#) ()
- void [movePoint](#) (int i, const QPointF &newPos)

Private Slots

- void [newPoint](#) ()
- void [deletePoint](#) ()
- void [fireChooseColor](#) ()

2.6.1 Detailed Description

The `hoverpoints` class represents the points which are used in the [ShadeWidget](#).

2.6.2 Member Enumeration Documentation

2.6.2.1 enum `HoverPoints::PointShape`

Enumerates the `PointShape` options.

2.6.2.2 enum `HoverPoints::LockType`

Enumerates the `LockType` options.

2.6.2.3 enum `HoverPoints::SortType`

Enumerates the `SortType` options.

2.6.2.4 enum `HoverPoints::ConnectionType`

Enumerates the `ConnectionType` options.

2.6.3 Constructor & Destructor Documentation

2.6.3.1 `HoverPoints::HoverPoints (QWidget * widget, PointShape shape)`

Constructor that creates the [HoverPoints](#).

Returns:

The pointer to the created [HoverPoints](#).

2.6.4 Member Function Documentation

2.6.4.1 `bool HoverPoints::eventFilter (QObject * object, QEvent * event)`

Filters the events and takes appropriate actions.

Parameters:

object is the `QObject` which caused the `QEvent`.

event is the `QEvent` which happend.

Returns:

True if it was a valid object and a valid event, false otherwise.

2.6.4.2 void HoverPoints::paintPoints ()

Paints the points and connections.

2.6.4.3 QRectF HoverPoints::boundingRect () const [inline]**Returns:**

The bounding rectangle for this hover points.

2.6.4.4 void HoverPoints::setBoundingRect (const QRectF & boundingRect) [inline]

Sets the new bounding rectangle for this hover points.

Parameters:

boundingRect is the new bounding rectangle which should be set.

2.6.4.5 QPolygonF HoverPoints::points () const [inline]**Returns:**

The hover points.

2.6.4.6 void HoverPoints::setPoints (const QPolygonF & points)

Sets points position from the given points.

Parameters:

points are the points which should be set.

2.6.4.7 QSizeF HoverPoints::pointSize () const [inline]**Returns:**

The pixel size of one hover point (for drawing).

2.6.4.8 void HoverPoints::setPointSize (const QSizeF & size) [inline]

Sets the new pixel size for this hover points.

Parameters:

size is the new pixel size which should be set.

2.6.4.9 `SortType HoverPoints::sortType () const` [inline]**Returns:**

The sort type, in which the hover points should be sorted.

2.6.4.10 `void HoverPoints::setSortType (SortType sortType)` [inline]

Sets the way, in which the hover points should be sorted.

Parameters:

sortType is the new sortType which should be set.

2.6.4.11 `ConnectionType HoverPoints::connectionType () const` [inline]**Returns:**

The way in which the point connections are drawn.

2.6.4.12 `void HoverPoints::setConnectionType (ConnectionType connectionType)` [inline]

Sets the way, in which the point connections should be drawn.

Parameters:

connectionType is the new connectionType which should be set.

2.6.4.13 `void HoverPoints::setConnectionPen (const QPen & pen)` [inline]

Sets the pen, which draws the point connections.

Parameters:

pen is the new pen which should be set.

2.6.4.14 `void HoverPoints::setShapePen (const QPen & pen)` [inline]

Sets the pen, which draws the shape of the points.

Parameters:

pen is the new pen which should be set.

2.6.4.15 `void HoverPoints::setShapeBrush (const QBrush & brush)` [inline]

Sets the brush, which fills the shape of the points.

Parameters:

brush is the new brush which should be set.

2.6.4.16 void HoverPoints::setPointLock (int *pos*, LockType *lock*) [inline]

Sets the point lock for a point.

Parameters:

pos is the index of the point from which the lock should be set.

lock is the typ of lock which should be set.

2.6.4.17 void HoverPoints::setEditable (bool *editable*) [inline]

Sets this points editable state.

Parameters:

editable is true if the points are editable and flase otherwise.

2.6.4.18 bool HoverPoints::editable () const [inline]**Returns:**

true if the points are editable and flase otherwise.

2.6.4.19 void HoverPoints::setEnabled (bool *enabled*) [slot]

Enables or disables these HoverPints.

Parameters:

enabled is true if the points are enabled and flase otherwise.

2.6.4.20 void HoverPoints::setDisabled (bool *disabled*) [inline, slot]

Enables or disables these HoverPints.

Parameters:

disabled is true if the points are disabled and flase otherwise.

2.6.4.21 void HoverPoints::pointsChanged (const QPolygonF & *points*) [signal]

A signal that is send on change of the points.

Parameters:

points are the new points.

2.6.4.22 void HoverPoints::chooseColor (int *index*) [signal]

A signal that is send to initiate a color dialog.

Parameters:

index is the index of the point for which the color should be changed.

2.6.4.23 void HoverPoints::alphaChanged (int *index*, qreal *newXPos*) [signal]

A signal that is send if an alpha point got moved.

Parameters:

index is the index of the points which should be moved.

newXPos is the new x position to which the corresponding points should be moved.

2.6.4.24 void HoverPoints::alphaPointCreated (int *index*, qreal *xPos*) [signal]

A signal that is send if an alpha point got created.

Parameters:

index is the index where the points should be saved.

xPos is the x position where the points should be created.

2.6.4.25 void HoverPoints::alphaPointDeleted (int *index*) [signal]

A signal that is send if an alpha point got deleted.

Parameters:

index is the index where the points should be deleted.

2.6.4.26 void HoverPoints::firePointChange ()

Sorts the points.

2.6.4.27 void HoverPoints::movePoint (int *index*, const QPointF & *point*)

Moves the given point.

Parameters:

index is the index of the point which should be moved.

point is the point which contains the new position.

2.6.4.28 void HoverPoints::newPoint () [private, slot]

Creates a new point. Emits an alphaPointCreated.

2.6.4.29 void HoverPoints::deletePoint () [private, slot]

Deletes a point. Emits an alphaPointDeleted.

2.6.4.30 void HoverPoints::fireChooseColor () [private, slot]

Emits a chooseColor.

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

- hoverpoints.h
- hoverpoints.cpp

2.7 ShadeWidget Class Reference

```
#include <shadewidget.h>
```

Public Types

- enum [ShadeType](#)

Signals

- void [colorsChanged](#) ()

Public Member Functions

- [ShadeWidget](#) ([ShadeType](#) type, QWidget *parent)
- void [setGradientStops](#) (const QGradientStops &stops)
- void [paintEvent](#) (QPaintEvent *e)
- QSize [minimumSizeHint](#) () const
- QSize [sizeHint](#) () const
- QPolygonF [points](#) () const
- [HoverPoints](#) * [hoverPoints](#) () const
- QRgb [colorAt](#) (int x)

Private Member Functions

- void [generateShade](#) ()

2.7.1 Detailed Description

The shadewidget class is a drawable rectangle in which points and a color or alpha gradient can be drawn.

2.7.2 Member Enumeration Documentation

2.7.2.1 enum ShadeWidget::ShadeType

Enumerates the ShateType options.

2.7.3 Constructor & Destructor Documentation

2.7.3.1 ShadeWidget::ShadeWidget (ShadeType type, QWidget * parent)

Constructor that creates a color or alpha [ShadeWidget](#).

Returns:

The pointer to the created color or alpha [ShadeWidget](#).

2.7.4 Member Function Documentation

2.7.4.1 void ShadeWidget::setGradientStops (const QGradientStops & stops)

Sets the gradient stops position and color from the given stops.

Parameters:

stops are the gradient stops which should be set.

2.7.4.2 void ShadeWidget::paintEvent (QPaintEvent * e)

Paints the background, points and the connection lines.

2.7.4.3 QSize ShadeWidget::minimumSizeHint () const

Returns:

The minimum size hint for the Widget.

2.7.4.4 QSize ShadeWidget::sizeHint () const

Returns:

The size hint for the Widget.

2.7.4.5 QPolygonF ShadeWidget::points () const

Returns:

The points of this [ShadeWidget](#).

2.7.4.6 HoverPoints* ShadeWidget::hoverPoints () const [inline]

Returns:

The pointer to the hover points of this [ShadeWidget](#).

2.7.4.7 QColor ShadeWidget::colorAt (int x)

Gets the current color of the ShadeWidget at the specific position.

Parameters:

x is the position from which the color should be taken.

Returns:

The color at the position.

2.7.4.8 void ShadeWidget::colorsChanged () [signal]

A signal that is send on change of the colors.

2.7.4.9 void ShadeWidget::generateShade () [private]

Generates the backround.

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

- shadewidget.h
- shadewidget.cpp

2.8 vec3 Class Reference

```
#include <vec3.h>
```

Public Member Functions

- float & `operator[]` (unsigned)
returns an element of the vector
- `vec3 & operator=` (const `vec3` &)
assignment operator
- `vec3 & operator+=` (const `vec3` &)
adds the input vector to this one
- `vec3 & operator-=` (const `vec3` &)
subtracts the input vector from this one
- `vec3 & operator*=` (float)
scales this vector with the scalar
- `vec3 & operator/=` (float)
scales this vector with the inverse of the scalar
- const `vec3 operator+` (const `vec3` &) const
adds two vectors
- const `vec3 operator-` (const `vec3` &) const
subtracts two vectors
- const `vec3 operator*` (float) const
scales the vector with the scalar
- const `vec3 operator/` (float) const
scales the vector with the inverse of the scalar
- bool `operator==` (const `vec3` &) const
returns true if the vectors are equal
- bool `operator!=` (const `vec3` &) const
returns true if the vectors differ in at least one component
- `vec3 & operator-` ()
opposite vector
- float `norm` () const
*norm of the vector (length*length)*
- float `length` () const

length of the vector

- `vec3 & operator! ()`
normalizes the vector
- `float dist2 (const vec3 &)`
*norm of the vectors' difference (dist*dist)*
- `float dist (const vec3 &)`
length of the vectors' difference
- `float operator* (const vec3 &) const`
dot product
- `const vec3 operator^ (const vec3 &) const`
cross product
- `void print ()`
print the vector components

Public Attributes

- `float v [3]`
our vector data

2.8.1 Detailed Description

3D vector class.

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

- `vec3.h`
- `vec3.cpp`

2.9 Window Class Reference

```
#include <window.h>
```

Signals

- void [backgroundColorTextureChanged](#) ()
- void [arrowColorTextureChanged](#) ()
- void [streamlineColorTextureChanged](#) ()

Public Member Functions

- [Window](#) ()

Private Slots

- void [setDefaultGradientStops](#) ()
- void [setOpenFileName](#) ()
- void [saveGradientStops](#) (const QGradientStops &stops, [GradientEditor](#) *editor)
- void [setBackgroundEditorRangeLabel](#) (int channel=-1)
- void [setArrowColorEditorRangeLabel](#) (int channel=-1)
- void [setStreamlineColorEditorRangeLabel](#) (int channel=-1)

2.9.1 Detailed Description

The window class represents the whole application window.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 Window::Window ()

Constructor that creates the GUI.

2.9.3 Member Function Documentation

2.9.3.1 void Window::setDefaultGradientStops () [private, slot]

Loads and sets the gradient stops for the current [FlowData](#).

2.9.3.2 void Window::setOpenFileName () [private, slot]

Creates a file dialog with which [FlowData](#) can be loaded.

2.9.3.3 void Window::saveGradientStops (const QGradientStops & *stops*, GradientEditor * *editor*) [private, slot]

Saves the given gradient stops into a file, calls for a update of the transfer texture and emits a colorTextureChanged signal.

Parameters:

- stops* are the gradient stops which should be saved.
- editor* is the color editor for whom the stops should be saved.

2.9.3.4 void Window::setBackgroundEditorRangeLabel (int *channel* = -1) [private, slot]

Reads the value range from a specific [FlowData](#) channel into the backgroundColorEditor.

Parameters:

- channel* is the channel from [FlowData](#) from which the value range should be read.

2.9.3.5 void Window::setArrowColorEditorRangeLabel (int *channel* = -1) [private, slot]

Reads the value range from a specific [FlowData](#) channel into the arrowColorEditor.

Parameters:

- channel* is the channel from [FlowData](#) from which the value range should be read.

2.9.3.6 void Window::setStreamlineColorEditorRangeLabel (int *channel* = -1) [private, slot]

Reads the value range from a specific [FlowData](#) channel into the streamlineColorEditor.

Parameters:

- channel* is the channel from [FlowData](#) from which the value range should be read.

2.9.3.7 void Window::backgroundColorTextureChanged () [signal]

A signal that is send on change of the background texture.

2.9.3.8 void Window::arrowColorTextureChanged () [signal]

A signal that is send on change of the arrow texture.

2.9.3.9 void Window::streamlineColorTextureChanged () [signal]

A signal that is send on change of the streamline texture.

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

- `window.h`
- `window.cpp`

Index

- addPoint2Grid
 - GLWidget, 18
- alphaChanged
 - HoverPoints, 28
- alphaPointCreated
 - HoverPoints, 28
- alphaPointDeleted
 - HoverPoints, 28
- arrowColorTextureChanged
 - Window, 36

- backgroundColorTextureChanged
 - Window, 36
- boundingRect
 - HoverPoints, 25

- calcEuler
 - GLWidget, 16
- calcRunge
 - GLWidget, 17
- calcStartCandidates
 - GLWidget, 18
- calculateBackground
 - GLWidget, 17
- calculateEvenStreamlines
 - GLWidget, 18
- chooseColor
 - HoverPoints, 27
- colorAt
 - ShadeWidget, 31
- colorDialog
 - GradientEditor, 21
- colorsChanged
 - ShadeWidget, 31
- ConnectionType
 - HoverPoints, 24
- connectionType
 - HoverPoints, 26
- copyValues
 - FlowChannel, 4

- deleteColorPoints
 - GradientEditor, 22
- deletePoint
 - HoverPoints, 28

- draw
 - GLWidget, 15
- drawArrowPlot
 - GLWidget, 16
- drawStreamline
 - GLWidget, 17

- editable
 - HoverPoints, 27
- eventFilter
 - HoverPoints, 24

- file2string
 - GLWidget, 16
- fireChooseColor
 - HoverPoints, 29
- firePointChange
 - HoverPoints, 28
- FlowChannel, 3
 - copyValues, 4
- FlowData, 5
- FlowGeometry, 7
 - getInterpolationAt, 9

- generateShade
 - ShadeWidget, 32
- getColorsPtr
 - GradientEditor, 21
- getInterpolationAt
 - FlowGeometry, 9
- getTapering
 - GLWidget, 17
- GLWidget, 10
 - addPoint2Grid, 18
 - calcEuler, 16
 - calcRunge, 17
 - calcStartCandidates, 18
 - calculateBackground, 17
 - calculateEvenStreamlines, 18
 - draw, 15
 - drawArrowPlot, 16
 - drawStreamline, 17
 - file2string, 16
 - getTapering, 17
 - GLWidget, 11

- initializeGL, 14
- initStreamlines, 19
- loadShader, 16
- minimumSizeHint, 11
- mouseMoveEvent, 15
- mousePressEvent, 15
- mouseReleaseEvent, 15
- normalizeAngle, 15
- paintGL, 14
- printLog, 16
- removeLastPointfromCell, 19
- resizeGL, 15
- setArrowColorChannel, 13
- setArrowColors, 12
- setArrowColorsPtr, 12
- setArrowQuantity, 13
- setBackgroundChannel, 12
- setBackgroundColors, 12
- setBackgroundColorsPtr, 12
- setDrawArrows, 13
- setDrawBackground, 12
- setDrawStreamlines, 13
- setFlowData, 11
- setStreamlineColorChannel, 14
- setStreamlineColors, 12
- setStreamlineColorsPtr, 12
- setStreamlineDsep, 14
- setStreamlineDt, 14
- setStreamlineDtest, 13
- setStreamlineLineMode, 14
- setStreamlineMode, 13
- setStreamlinePeriod, 14
- sizeHint, 11
- testCell, 17
- testCellTapering, 18
- testRange, 17
- testSLDist, 18
- wheelEvent, 15
- GradientEditor, 20
 - colorDialog, 21
 - deleteColorPoints, 22
 - getColorPtr, 21
 - GradientEditor, 20
 - gradientStopsChanged, 22
 - moveColorsWithAlpha, 21
 - newColorPoints, 21
 - pointsUpdated, 21
 - resizeEvent, 22
 - setEnabledSignal, 22
 - setGradientEditorRangeLabel, 21
 - setGradientStops, 21
 - updateColorTexture, 21
- gradientStopsChanged
 - GradientEditor, 22
- HoverPoints, 23
 - alphaChanged, 28
 - alphaPointCreated, 28
 - alphaPointDeleted, 28
 - boundingRect, 25
 - chooseColor, 27
 - ConnectionType, 24
 - connectionType, 26
 - deletePoint, 28
 - editable, 27
 - eventFilter, 24
 - fireChooseColor, 29
 - firePointChange, 28
 - HoverPoints, 24
 - LockType, 24
 - movePoint, 28
 - newPoint, 28
 - paintPoints, 25
 - points, 25
 - pointsChanged, 27
 - PointShape, 24
 - pointSize, 25
 - setBoundingRect, 25
 - setConnectionPen, 26
 - setConnectionType, 26
 - setDisabled, 27
 - setEditable, 27
 - setEnabled, 27
 - setPointLock, 26
 - setPoints, 25
 - setPointSize, 25
 - setShapeBrush, 26
 - setShapePen, 26
 - setSortType, 26
 - SortType, 24
 - sortType, 25
- hoverPoints
 - ShadeWidget, 31
- initializeGL
 - GLWidget, 14
- initStreamlines
 - GLWidget, 19
- loadShader
 - GLWidget, 16
- LockType
 - HoverPoints, 24
- minimumSizeHint
 - GLWidget, 11
 - ShadeWidget, 31
- mouseMoveEvent
 - GLWidget, 15

- mousePressEvent
 - GLWidget, 15
- mouseReleaseEvent
 - GLWidget, 15
- moveColorsWithAlpha
 - GradientEditor, 21
- movePoint
 - HoverPoints, 28
- newColorPoints
 - GradientEditor, 21
- newPoint
 - HoverPoints, 28
- normalizeAngle
 - GLWidget, 15
- paintEvent
 - ShadeWidget, 31
- paintGL
 - GLWidget, 14
- paintPoints
 - HoverPoints, 25
- points
 - HoverPoints, 25
 - ShadeWidget, 31
- pointsChanged
 - HoverPoints, 27
- PointShape
 - HoverPoints, 24
- pointSize
 - HoverPoints, 25
- pointsUpdated
 - GradientEditor, 21
- printLog
 - GLWidget, 16
- removeLastPointfromCell
 - GLWidget, 19
- resizeEvent
 - GradientEditor, 22
- resizeGL
 - GLWidget, 15
- saveGradientStops
 - Window, 35
- setArrowColorChannel
 - GLWidget, 13
- setArrowColorEditorRangeLabel
 - Window, 36
- setArrowColors
 - GLWidget, 12
- setArrowColorsPtr
 - GLWidget, 12
- setArrowQuantity
 - GLWidget, 13
- setBackgroundChannel
 - GLWidget, 12
- setBackgroundEditorRangeLabel
 - Window, 36
- setBackgroundColors
 - GLWidget, 12
- setBackgroundColorsPtr
 - GLWidget, 12
- setBoundingRect
 - HoverPoints, 25
- setConnectionPen
 - HoverPoints, 26
- setConnectionType
 - HoverPoints, 26
- setDefaultGradientStops
 - Window, 35
- setDisabled
 - HoverPoints, 27
- setDrawArrows
 - GLWidget, 13
- setDrawBackground
 - GLWidget, 12
- setDrawStreamlines
 - GLWidget, 13
- setEditable
 - HoverPoints, 27
- setEnabled
 - HoverPoints, 27
- setEnabledSignal
 - GradientEditor, 22
- setFlowData
 - GLWidget, 11
- setGradientEditorRangeLabel
 - GradientEditor, 21
- setGradientStops
 - GradientEditor, 21
 - ShadeWidget, 31
- setOpenFileName
 - Window, 35
- setPointLock
 - HoverPoints, 26
- setPoints
 - HoverPoints, 25
- setPointSize
 - HoverPoints, 25
- setShapeBrush
 - HoverPoints, 26
- setShapePen
 - HoverPoints, 26
- setSortType
 - HoverPoints, 26
- setStreamlineColorChannel
 - GLWidget, 14

- setStreamlineColorEditorRangeLabel
 - Window, 36
- setStreamlineColors
 - GLWidget, 12
- setStreamlineColorsPtr
 - GLWidget, 12
- setStreamlineDsep
 - GLWidget, 14
- setStreamlineDt
 - GLWidget, 14
- setStreamlineDtest
 - GLWidget, 13
- setStreamlineLineMode
 - GLWidget, 14
- setStreamlineMode
 - GLWidget, 13
- setStreamlinePeriod
 - GLWidget, 14
- ShadeType
 - ShadeWidget, 30
- ShadeWidget, 30
 - colorAt, 31
 - colorsChanged, 31
 - generateShade, 32
 - hoverPoints, 31
 - minimumSizeHint, 31
 - paintEvent, 31
 - points, 31
 - setGradientStops, 31
 - ShadeType, 30
 - ShadeWidget, 30
 - sizeHint, 31
- sizeHint
 - GLWidget, 11
 - ShadeWidget, 31
- SortType
 - HoverPoints, 24
- sortType
 - HoverPoints, 25
- streamlineColorTextureChanged
 - Window, 36

- testCell
 - GLWidget, 17
- testCellTapering
 - GLWidget, 18
- testRange
 - GLWidget, 17
- testSLDist
 - GLWidget, 18

- updateColorTexture
 - GradientEditor, 21

- vec3, 33

- wheelEvent
 - GLWidget, 15
- Window, 35
 - arrowColorTextureChanged, 36
 - backgroundColorTextureChanged, 36
 - saveGradientStops, 35
 - setArrowColorEditorRangeLabel, 36
 - setBackgroundColorEditorRangeLabel, 36
 - setDefaultGradientStops, 35
 - setOpenFileName, 35
 - setStreamlineColorEditorRangeLabel, 36
 - streamlineColorTextureChanged, 36
 - Window, 35