SolarSystemSimulation

Generated by Doxygen 1.7.5

Sat May 26 2012 22:23:01

Contents

Chapter 1

Documentation

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

Chapter 2

Class Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

Class Documentation

4.1 About Class Reference

Public Member Functions

- **About** (QWidget *parent=0)

  Open the about dialog.
- ~**About** ()

  Close and delete the about dialog.

4.1.1 Constructor & Destructor Documentation

4.1.1.1 About::About (QWidget * *parent* = `0`) `[explicit]`

Open the about dialog.

Parameters

parent	

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

- forms/main/about.h
- forms/main/about.cpp

4.2 CGLMatrix Class Reference

```
#include <glmatrix.h>
```

Public Member Functions

- **CGLMatrix** ()
- **CGLMatrix** (GLfloat M[16])
- virtual ∼**CGLMatrix** ()
- virtual void **setMatrix** (const GLfloat ∗_newVal)
- virtual GLfloat ∗ **getMatrix** ()
- virtual void **loadIdentity** ()
- **CGLMatrix** (const **CGLMatrix** &toCopy)
- const **CGLMatrix operator=** (const **CGLMatrix** &toCopy)
- const **CGLMatrix operator**∗ (const **CGLMatrix** &m2) const
- void **debug** (QString caption="")
- GLfloat **m** (int row, int column)
- **CGLMatrix transpose** () const
- GLfloat **determinant** ()
- GLfloat **calcSarrus** (int deletedColumn)
- void **setM** (int row, int column, GLfloat value)
- void **loadModelview** ()
- void **multToStack** () const

4.2.1 Detailed Description

A 4 ∗4 matrix for use with OpenGl functions and C3dVectors declared as m[16]. The standard column-major order of an OpenGL matrix is:

m[0] m[4] m[8] m[12] m11 m12 m13 m14 m[1] m[5] m[9] m[13] m21 m22 m23 m24 m[2] m[6] m[10] m[14] m31 m32 m33 m34 m[3] m[7] m[11] m[15] m41 m42 m43 m44

Note that m11 is the top left coefficient m[0], m41 = m[3], m12 = m[4] and m44 = m[15].

If you read this in an html file, refer to the cglmatrix.h file for correct formatting.

Author

Walter Roth

4.2.2 Constructor & Destructor Documentation

4.2.2.1 CGLMatrix::CGLMatrix ( )

Standard constructor. Initializes a zero matrix.

4.2.2.2 CGLMatrix::CGLMatrix (GLfloat *M[16]*)

Constructor that takes a full set of matrix coefficients. For column major order see top comment in header file.

4.2.2.3 CGLMatrix::~CGLMatrix ( ) [virtual]

Destructor. Does nothing.

4.2.2.4 CGLMatrix::CGLMatrix (const CGLMatrix & *toCopy*)

Copy constructor.

4.2.3 Member Function Documentation

4.2.3.1 GLfloat CGLMatrix::calcSarrus (int *deletedColumn*)

Calculates the determinant using Sarrus' law fo 3 by 3 matrices. The 3 by 3 matrix is obtained by deleting the last row (row 4) and the specified column.

Calculates the determinant using Sarrus' law fo 3 by 3 matrices. The 3 by 3 sub-matrix is obtained by deleting the last row (row 4) and the specified column.

4.2.3.2 void CGLMatrix::debug (QString *caption* = " ")

Writes matrix coefficients to stderr.

4.2.3.3 GLfloat CGLMatrix::determinant ( )

Returns the determinant.

Returns the determinant. Because it is rather probable, that the elements of the last row are 0, 0, 0 , 1, the determinant is calculated based on the last row (row number 4). The determinantes of the 3 by 3 matrices are calculated using Sarrus' law.

4.2.3.4 GLfloat * CGLMatrix::getMatrix ( ) [virtual]

Read property of GLfloat m[16]. Returns the address of m[0].

Read property of GLfloat m_M[16].

4.2.3.5 void CGLMatrix::loadIdentity ( ) [virtual]

Loads the identity matrix.

4.2.3.6 void CGLMatrix::loadModelview ( )

Loads the current modelview matrix.

4.2.3.7 GLfloat CGLMatrix::m (int *row,* int *column*)

Returns matrix element row, column. For mathematical convenience, row and column numbers start with 1. Use getMatrix for direct access to single index zero based matrix element array, as required by OpenGL functions.

4.2.3.8 void CGLMatrix::multToStack ( ) const

Multiplies the matrix onto the current matrix stack **without saving** the current matrix on that stack.

4.2.3.9 const CGLMatrix CGLMatrix::operator∗ (const CGLMatrix & *m2*) const

Matrix multiplication m ∗ m2. !!!WARNING: ORDER OF MATRICES IS CRITICAL!!! (m∗m2 != m2∗m) Calculations run on main fpu. Use glMultMatrix for calculation on graphics processor.

Calculations run on main fpu. Use glMultMatrix for calculation on graphics processor. Matrix multiplication m_M ∗ m2. !!!WARNING: ORDER OF MATRICES IS CRITICAL!!! (m_M∗m2 != m2∗m_M) The result element result[i,j] is obtained by multiplying all elements of row i of matrix m_M with the elements of column j of matrix m2 and adding the products. Mathematical row and column numbers usually start with a 1 and not with a 0 as in C++. This has to be translated into the column major order of the OpenGL matrix, where the elements are held in m_M[16]. A 1 based row i is addressed by: m_M[i-1], m_M [i+3], m_M[i+7], m_M[i+11]. A 0 based row i is addressed by: m_M[i], m_M [i+4], m_M[i+8], m_M[i+12] = m_M[row + 4 ∗ column]. A 1 based column j is addressed by: m2[(j-1)∗4], m_M[(j-1)∗4 +1, m_M[(j-1)∗4 +2, m_M[(j-1)∗4 +3]. A 0 based column j is addressed by: m2[j∗4], m_M[j∗4 +1], m_M[j∗4 +2], m_M[j∗4 +3] = m_M[4∗column + row]. The operator ∗ runs through m_M[16] by zero based rows and columns, which is not the fastest, however a pretty obvious way of calculating the product. For top speed, use 16 separately hard coded sums.

4.2.3.10 const CGLMatrix CGLMatrix::operator= (const CGLMatrix & *toCopy*)

Operator = copies m.

Operator = copies m_M.

4.2.3.11 void CGLMatrix::setM (int *row,* int *column,* GLfloat *value*)

Sets matrix element in row and column. Row and column numbers start with 1.

4.2.3.12 void CGLMatrix::setMatrix (const GLfloat ∗ *_newVal*) [virtual]

Write property of GLfloat m[16].

Write property of GLfloat m_M[16].

4.2.3.13 CGLMatrix CGLMatrix::transpose ( ) const

Returns the transposed matrix m^T.

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

- OpenGL/glmatrix.h
- OpenGL/glmatrix.cpp

4.3 DatabaseConnectionFailedException Class Reference

`#include <databaseconnectionfailedexception.h>`

Public Member Functions

- **DatabaseConnectionFailedException** (const QString message, QString sqlError)

  Exception class for failed database connection.
- virtual const QString **getMessage** () const throw ()

  Getter for the message.
- virtual const QString **getSqlError** () const throw ()

  Getter for the sqlError.

4.3.1 Detailed Description

Class for database connection failed exceptions .

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.3.2 Constructor & Destructor Documentation

4.3.2.1 DatabaseConnectionFailedException::DatabaseConnectionFailedException (const QString *message,* QString *sqlError*)

Exception class for failed database connection.

Parameters

message	The error message.
sqlError	The SQL-error.

4.3.3 Member Function Documentation

4.3.3.1 const QString DatabaseConnectionFailedException::getMessage ( ) const throw () [virtual]

Getter for the message.

Returns

const QString

4.3.3.2 const QString DatabaseConnectionFailedException::getSqlError ( ) const throw () [virtual]

Getter for the sqlError.

Returns

const QString

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

- database/exceptions/databaseconnectionfailedexception.h
- database/exceptions/databaseconnectionfailedexception.cpp

4.4 DeleteEntityFailedException Class Reference

`#include <deleteentityfailedexception.h>`

Public Member Functions

- **DeleteEntityFailedException** (const QString message, QString sqlError)
  Constructor for the exception with error message and SQL-error.
- **DeleteEntityFailedException** (const QString message)
  Constructor with error message only.
- virtual const QString **getMessage** () const throw ()
  Getter for message.
- virtual const QString **getSqlError** () const throw ()
  Getter for sqlError.

4.4.1 Detailed Description

Class for entity delete failed exceptions .

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.4.2 Constructor & Destructor Documentation

4.4.2.1 DeleteEntityFailedException::DeleteEntityFailedException (const QString *message*, QString *sqlError*)

Constructor for the exception with error message and SQL-error.

Parameters

message	Error message of the Object
sqlError	SQL Error of the Object

4.4.2.2 DeleteEntityFailedException::DeleteEntityFailedException (const QString *message*)

Constructor with error message only.

Parameters

message	Error message of the Object

4.4.3 Member Function Documentation

4.4.3.1 const QString DeleteEntityFailedException::getMessage ( ) const throw () [virtual]

Getter for message.

Returns

const QString

4.4.3.2 const QString DeleteEntityFailedException::getSqlError ( ) const throw () [virtual]

Getter for sqlError.

Returns

const QString

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

- data/exceptions/deleteentityfailedexception.h
- data/exceptions/deleteentityfailedexception.cpp

4.5 EntityNotUniqueException Class Reference

`#include <entitynotuniqueexception.h>`

Public Member Functions

- **EntityNotUniqueException** (const QString message)

  Constructor with error message only.
- virtual const QString **getMessage** () const throw ()

  Getter for message.

4.5.1 Detailed Description

Class for messages when the entity is not unique.

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.5.2 Constructor & Destructor Documentation

4.5.2.1 EntityNotUniqueException::EntityNotUniqueException (const QString *message*)

Constructor with error message only.

Parameters

message	Error message of the Object

4.5.3 Member Function Documentation

4.5.3.1 const QString EntityNotUniqueException::getMessage ( ) const throw () [virtual]

Getter for message.

Returns

const QString

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

- data/exceptions/entitynotuniqueexception.h
- data/exceptions/entitynotuniqueexception.cpp

4.6 Environment Class Reference

Public Member Functions

- void **drawAxes** (GLdouble axisLength)
- void **toggleCoordinateAxesVisibility** ()

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

- visualization/environment/environment.h
- visualization/environment/environment.cpp

4.7 GLColorRGBA Class Reference

```
#include <glcolorrgba.h>
```

Public Member Functions

- **GLColorRGBA** ()
- **GLColorRGBA** (GLfloat r, GLfloat g, GLfloat b, GLfloat a=1.0)
- virtual ∼**GLColorRGBA** ()
- GLfloat ∗ **fv** ()
- **GLColorRGBA operator**∗ (double f)
- double **red** ()
- **GLColorRGBA operator=** (const **GLColorRGBA** &toCopy)
- double **green** ()
- **GLColorRGBA** (const **GLColorRGBA** &toCopy)
- double **blue** ()
- double **alpha** ()

Protected Member Functions

- void **copy** (const **GLColorRGBA** &toCopy)

4.7.1 Detailed Description

A color class that supports RGBA colors. Color is saved as 4 GLfloat values 0.0 = black, 1.0 = full intensity. The "A" value is for transparency (very practical for window panes). 0.0 = transparent, 1.0 = opaque

Author

Walter Roth

4.7.2 Constructor & Destructor Documentation

4.7.2.1 GLColorRGBA::GLColorRGBA ( )

Constructs a black color.

4.7.2.2 GLColorRGBA::GLColorRGBA (GLfloat *r,* GLfloat *g,* GLfloat *b,* GLfloat *a* = `1.0`)

Constructs the specified color.

4.7.2.3 GLColorRGBA::~GLColorRGBA ( ) [virtual]

Destructor, does nothing.

4.7.2.4 GLColorRGBA::GLColorRGBA (const GLColorRGBA & *toCopy*)

Copy constructor.

4.7.3 Member Function Documentation

4.7.3.1 double GLColorRGBA::alpha ( )

Returns A (transparency) value.

4.7.3.2 double GLColorRGBA::blue ( )

Returns Blue value.

4.7.3.3 void GLColorRGBA::copy (const GLColorRGBA & *toCopy*) [protected]

Copy function. For internal use only. MUST be called by subclassed copy functions.

Copy function. For internal use only.

4.7.3.4 GLfloat * GLColorRGBA::fv ( )

Returns a pointer to fR. To be used with GLColorXXXfv functions.

4.7.3.5 double GLColorRGBA::green ( )

Returns Green value.

4.7.3.6 GLColorRGBA GLColorRGBA::operator* (double *f*)

For brightness adjustment. "a" value is not multiplied.

For brightness adjustment. "a" value is not multipiled.

4.7.3.7 GLColorRGBA GLColorRGBA::operator= (const GLColorRGBA & *toCopy*)

Copy operator.

4.7.3.8 double GLColorRGBA::red ( )

Returns Red value.

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

- OpenGL/glcolorrgba.h
- OpenGL/glcolorrgba.cpp

4.8 GLPerspective Class Reference

`#include <glperspective.h>`

Collaboration diagram for GLPerspective:

Public Member Functions

- void **apply** ()
- void **turnCameraUpDown** (double angleIncrement)
- void **turnCameraLeftRight** (double angleIncrement)
- void **stretchCameraDistance** (double factor)
- void **shiftSceneUpDown** (double distance)
- void **shiftSceneLeftRight** (double distance)
- void **shiftSceneForwardBackward** (double distance)
- void **setCamera** (const **GLVector** &newValue)
- void **setCenter** (const **GLVector** &newValue)
- void **setUp** (const **GLVector** &newValue)
- void **setAspect** (GLdouble newValue)
- void **setFar** (GLdouble newValue)
- void **setFovy** (GLdouble newValue)
- void **setNear** (GLdouble newValue)
- void **setViewport** (int width, int height)
- double **distance** () const

Protected Attributes

- **GLVector _Camera**
- **GLVector _Center**
- **GLVector _Up**
- GLdouble **_Aspect**
- GLdouble **_Far**
- GLdouble **_Fovy**
- GLdouble **_Near**
- GLint **_Width**
- GLint **_Height**

4.8.1 Detailed Description

A class for encapsulating all data for a gluPerspective and a gluLookAt call. These include: _Fovy: Opening angle of frustum in y-direction (45° is a good average to start with) _Aspect: The aspect ratio of the viewport (width / height) _Near and _Far clipping plane distances _Center: The 3d-point to appear in the center of the viewport _Camera-: The 3d-point where the camera is _Up: The 3d-vector that points upwards in the viewport

Use the **apply()** (p. 18) function to transfer the perspective settings to the OpenGL matrices.

Author

walter <walter-Roth>

4.8.2 Member Function Documentation

4.8.2.1 void GLPerspective::apply ( )

Applies the perspective settings to projection and modelview matrices. Old matrix values will be overwritten. To be called once before rendering the scene.

4.8.2.2 double GLPerspective::distance ( ) const [inline]

Getters

4.8.2.3 void GLPerspective::setCamera (const GLVector & *newValue*)

Setters

4.8.2.4 void GLPerspective::setViewport (int *width,* int *height*)

Sets viewport and adjusts _Aspect to new viewport

4.8.2.5 void GLPerspective::shiftSceneForwardBackward (double *distance*)

Shift the whole scene in x-z plane parallel to camera vector projection in xz plane

4.8.2.6 void GLPerspective::shiftSceneLeftRight (double *distance*)

Shift the whole scene in x-z plane orthogonal to camera vector

4.8.2.7 void GLPerspective::shiftSceneUpDown (double *distance*)

Shift the whole scene in y direction

4.8.2.8 void GLPerspective::stretchCameraDistance (double *factor*)

Multiply camera vector by factor

4.8.2.9 void GLPerspective::turnCameraLeftRight (double *angleIncrement*)

Moves the camera on a latitude without modifying the distance to _Center Maximum angle is +180 (east), minimum is -180 (west)

4.8.2.10 void GLPerspective::turnCameraUpDown (double *angleIncrement*)

Moves the camera on a meridian without modifying the distance to _Center Maximum angle is +90 (north), minimum is -90 (south)

4.8.3 Member Data Documentation

4.8.3.1 GLint GLPerspective::_Height [protected]

Viewport height

4.8.3.2 GLint GLPerspective::_Width [protected]

Viewport width

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

- OpenGL/glperspective.h
- OpenGL/glperspective.cpp

4.9 GLVector Class Reference

`#include <glvector.h>`

Collaboration diagram for GLVector:

Public Member Functions

- **GLVector** ()
- ∼**GLVector** ()
- **GLVector** (GLdouble x, GLdouble y, GLdouble z)
- **GLVector** (int angleMode, GLdouble radius, GLdouble longitude, GLdouble latitude)
- **GLVector** (const **GLVector** &toCopy)
- GLdouble **x** () const
- void **setX** (GLdouble _newVal)
- GLdouble **y** () const
- void **setY** (GLdouble _newVal)
- GLdouble **z** () const
- void **setZ** (GLdouble _newVal)
- const **GLVector** & **unitVector** () const
- GLdouble **operator**∗ (const **GLVector** &v2) const
- const **GLVector** & **vectorMult** (const **GLVector** &v2) const
- const **GLVector** & **normalVector** (const **GLVector** &v2) const
- GLdouble **length** () const
- const **GLVector rotateVector** (const **GLVector** &Axis, GLdouble Angle) const
- const **GLVector** & **operator-** (const **GLVector** &v2) const
- const **GLVector** & **operator+** (const **GLVector** &v2) const
- GLdouble ∗ **dv** () const
- const **GLVector** & **stretchVector** (GLdouble sx, GLdouble sy, GLdouble sz) const
- const **GLVector** & **operator**∗ (GLdouble f) const
- bool **isNull** () const
- bool **operator==** (const **GLVector** &v2) const
- bool **operator!=** (const **GLVector** &v2) const
- **GLVector** & **operator=** (const **GLVector** &toCopy)
- const **GLVector** & **operator**∗ (const **CGLMatrix** &m) const
- QString **debugString** (const QString &caption) const
- void **debugOutput** (const QString &caption) const
- GLdouble **latitude** () const
- GLdouble **theta** () const
- GLdouble **radius** () const
- GLdouble **toDegree** (GLdouble angle) const
- GLdouble **longitude** () const
- GLdouble **phi** () const
- const **GLVector** & **operator**/ (GLdouble f) const
- void **setRadiusPhiTheta** (GLdouble radius, GLdouble phi, GLdouble theta)

- void **setRadiusLongitudeLatitude** (GLdouble radius, GLdouble longitude, GLdouble latitude)
- GLdouble **toRadian** (GLdouble angle) const
- void **vertex** () const
- GLdouble **scalarMult** (const **GLVector** &v2) const
- void **limitTo** (const **GLVector** &min, const **GLVector** &max)

Static Public Member Functions

- static void **debugOutputStatistics** ()
- static void **setAngleMode** (const int &_newVal)
- static const int & **angleMode** ()
- static void **debugResetCounters** ()

Static Public Attributes

- static const int **Use360Degree** = 1
- static const int **Use400NewDegree** = 2
- static const int **UseRadian** = 3
- static const int **AvailableOpBuffers** = 3

Protected Member Functions

- void **copy** (const **GLVector** &toCopy)

Static Protected Member Functions

- static const **GLVector** & **nextBuffer** (GLdouble x, GLdouble y, GLdouble z)

Protected Attributes

- GLdouble **_X**
- GLdouble **_Y**
- GLdouble **_Z**

Static Protected Attributes

- static int **_AngleMode** = **GLVector::Use360Degree**
- static **GLVector OpBufferVectors** [**AvailableOpBuffers**]
- static int **n_Buffer** = 0
- static int **n_MaxBuffers** = 0
- static int **n_OperatorCalls** = 0
- static int **n_CopyCalls** = 0
- static int **n_ConstructorCalls** = 0
- static int **n_VertexCalls** = 0

4.9.1 Detailed Description

A vector with 3 dimensions in GLdouble numbers, especially designed for use with - OpenGL. Can be used easily with GL_xxx3dv functions. Use the **dv()** (p. 24) member function to obtain a GLdouble * to the data. Alternatively, you may use **vertex()** (p. 28) for calls to glVertex and normal() for calls to glNormal. Contains the basic vector maths as memberfunctions and operators. May be used with cartesian (x, y, z), geographic (radius, longitude angle counterclockwise around y axis starting at z axis and latitude angle from equatorial xz plane up and down 90) and polar coordinates (radius, phi angle around z axis counterclockwise from x axis, theta angle from positive z axis downwards). Radius is the length of the vector. All angle functions work with 360 degree angles, unless you set **GLVector::UseRadian** (p. 29) or **GLVector::Use400NewDegree** (p. 29) with the setAngleMode class function. However, there is a constructor for geographic coordinates with a local AngleMode, that accepts geographic coordinates and radian, 360 degree or 400 new degree angle values. Pass Use360Degree or Use400New-Degree as first parameter, if you want to use degrees. For polar coordinates, use the set-RadiusPhiTheta function. Data are kept as cartesian coordinates internally. Therefore, the geographic and polar values may have rounding errors in the order of 10E-13. All calculation functions except those operators, that include a = (=, +=, *= ...) are const functions. This means, if you want to rotate a vector foo itself, you have to call: foo = foo.rotateVector(). For convenience, there are a couple of predefined vectors at the end of this header file (v_X, v_Y..).

This class is designed for high performance and not for subclassing. Undefine DE-BUG_GLVECTOR for maximum performance (refer to top of header file). There are no virtual functions, which would create a VMT. This increases performance. Make functions virtual, if you want to subclass (NOT recommended). The operators return references to members of the protected OpBufferVectors array. This avoids permanent construction and destruction of temporary GLVectors for calculation results.**The predefined maximum number of nested calculations is 32. If you need more, increase the value of AvailableOpBuffers. See the getNextBuffer function for details. Use debugOutputStatistics() (p. 23) to obtain informations on buffer useage and call statistics. Operator buffer overflow is very likely, if results of vector calculations are passed as reference parameters to functions that perform lots of calculations. In these cases use "call by value" parameters or make an explicit copy of the reference parameter.**

Author

Walter Roth

4.9.2 Constructor & Destructor Documentation

4.9.2.1 GLVector::GLVector ( )

Standard constructor, creates a zero vector

4.9.2.2 GLVector::~GLVector ( )

Destructor, presently it does nothing.

4.9.2.3 GLVector::GLVector (GLdouble *x,* GLdouble *y,* GLdouble *z*)

Constructor for cartesian coordinates.

4.9.2.4 GLVector::GLVector (int *angleMode,* GLdouble *radius,* GLdouble *longitude,* GLdouble *latitude*)

Constructor for geographic coordinates. Pass UseRadian, Use360Degree or Use400NewDegree as value for angleMode. Static AngleMode is NOT affected. Alternatives: Construct a **GLVector** (p. 20) using the standard constructor. Then call setRadiusPhiTheta for polar or setRadiusLongitudeLatitude for geographic coordinates.

Constructor for polar coordinates. Pass UseRadian, Use360Degree or Use400NewDegree as value for angleMode.

4.9.2.5 GLVector::GLVector (const GLVector & *toCopy*)

Copy constructor. Calls **copy()** (p. 23) function.

4.9.3 Member Function Documentation

4.9.3.1 const int & GLVector::angleMode ( ) [static]

Read property of int _AngleMode.

4.9.3.2 void GLVector::copy (const GLVector & *toCopy*) [protected]

Copy function used by copy constructor and operator =.

4.9.3.3 void GLVector::debugOutput (const QString & *caption*) const

Writes list of coordinates to stderr.

4.9.3.4 void GLVector::debugOutputStatistics ( ) [static]

Produces a debug output of the number of operations, constructor calls etc.

Produces a debug output of the number of operations, constructoir calls etc.

4.9.3.5 void GLVector::debugResetCounters ( ) [static]

Sets debugging counters to 0.

4.9.3.6 QString GLVector::debugString (const QString & *caption*) const

Returns the debug output string.

4.9.3.7 GLdouble * GLVector::dv ( ) const

For use with glXXXX3dv functions. This function relies on the internal order of declaration.

For use with glXXXX3dv functions.

4.9.3.8 bool GLVector::isNull ( ) const

Returns true, if all coordinates are 0.

4.9.3.9 GLdouble GLVector::latitude ( ) const

Returns latitude angle from equatorial xz plane up and down.

Returns latitude angle form equatorial xz plane up and down.

4.9.3.10 GLdouble GLVector::length ( ) const

Caculates the length.

4.9.3.11 void GLVector::limitTo (const GLVector & *min,* const GLVector & *max*)

Limits coordinates to values between min and max.

4.9.3.12 GLdouble GLVector::longitude ( ) const

Returns the longitude angle around y axis in radian, 360 degree or 400 new degree according to angleMode. Angle starts at z-axis.

4.9.3.13 const GLVector & GLVector::nextBuffer (GLdouble *x,* GLdouble *y,* GLdouble *z*) [static, protected]

Copies x, y and z to the next (and hopefully free) buffer vector from the buffer ring - OpBufferVectors and returns its address. The number of available buffer vectors is controlled by AvailableOpBufferVectors. **There is no verification, whether a buffer**

position is no longer used. Therefore, if you are performing very deeply nested calculations, you may have to set AvailableOpBufferVectors to a higher value. Use debugOutputStatistics() (p. **23) for debugging.**

Copies x, y and z to the next (and hopefully free) buffer vector from the buffer ring OpBuffer and returns its address. The number of available buffer vectors is controlled by AvailableOpBuffers. **There is no verification, whether a buffer position is no longer used. Therefore, if you are performing very deeply nested calculations, you may have to set _BufferVectors to a higher value.**

4.9.3.14 const GLVector & GLVector::normalVector (const GLVector & *v2*) const

Normal vector with lenght 1.0 obtained by ∗this X v2.

Normal vector with length 1.0 obtained by ∗this X v2.

4.9.3.15 bool GLVector::operator!= (const GLVector & *v2*) const

Compares two vectors. Uses !operator ==.

Compares two vectors.

4.9.3.16 GLdouble GLVector::operator∗ (const GLVector & *v2*) const

Scalar product. For vector product see function vectorMult.

Scalar product.

4.9.3.17 const GLVector & GLVector::operator∗ (GLdouble *f*) const

Multiplies vector with f.

4.9.3.18 const GLVector & GLVector::operator∗ (const CGLMatrix & *m*) const

Multipies 3d Vector with 4∗4 matrix. 4th coordinate of vector is assumed to be 1.0. - Calculations run on main fpu. Use glMultMatrix for calculation on graphics processor.

4.9.3.19 const GLVector & GLVector::operator+ (const GLVector & *v2*) const

Returns the vectorsum of v1 and v2.

4.9.3.20 const GLVector & GLVector::operator- (const GLVector & *v2*) const

Returns this - v2.

4.9.3.21 const GLVector & GLVector::operator/ (GLdouble *f*) const

Multiplies vector with1/ f.

4.9.3.22 GLVector & GLVector::operator= (const GLVector & *toCopy*)

Copies toCopy and returns ∗ this.

4.9.3.23 bool GLVector::operator== (const GLVector & *v2*) const

Compares two vectors. Will only return true if ALL digits of the GLdouble coordinates are the same.

4.9.3.24 GLdouble GLVector::phi ( ) const

Returns phi angle around z axis in radian, 360 degree or 400 new degree according to angleMode.

4.9.3.25 GLdouble GLVector::radius ( ) const

Returns length of vector, for convenience.

4.9.3.26 const GLVector GLVector::rotateVector (const GLVector & *Axis,* GLdouble *Angle*) const

Returns a vector obtained by rotating ∗this around axis. Uses radian Angle.

4.9.3.27 GLdouble GLVector::scalarMult (const GLVector & *v2*) const

Returns scalar product. May be used instead of operator ∗. For vector product see function vectorMult.

Returns scalar product.

4.9.3.28 void GLVector::setAngleMode (const int & *_newVal*) [static]

Write property of int _AngleMode.

4.9.3.29 void GLVector::setRadiusLongitudeLatitude (GLdouble *radius,* GLdouble *longitude,* GLdouble *latitude*)

Sets geographic coordinates. Y is up, longitude in xz plane, latitude from xz plane up- and downwards.

4.9.3.30 void GLVector::setRadiusPhiTheta (GLdouble *radius,* GLdouble *phi,* GLdouble *theta*)

Sets polar coordinates. Z is up, phi in xy plane, theta from z downwards.

4.9.3.31 void GLVector::setX (GLdouble *_newVal*)

Write property for x coordinate.

4.9.3.32 void GLVector::setY (GLdouble *_newVal*)

Write property for y coordinate.

4.9.3.33 void GLVector::setZ (GLdouble *_newVal*)

Write property for z coordinate.

4.9.3.34 const GLVector & GLVector::stretchVector (GLdouble *sx,* GLdouble *sy,* GLdouble *sz*) const

Returns a vector x∗sx, y∗sy, z∗sz.

Stretches a vector to x∗sx, y∗sy, z∗sz.

4.9.3.35 GLdouble GLVector::theta ( ) const

Returns theta angle from z axis downwards.

4.9.3.36 GLdouble GLVector::toDegree (GLdouble *angle*) const

Returns degree value for angle. If angleMode is UseRadian, angle is returned unscaled.

Returns degree value for angle. If _AngleMode is UseRadian, angle is returned unscaled.

4.9.3.37 GLdouble GLVector::toRadian (GLdouble *angle*) const

Returns radian value for angle.

4.9.3.38 const GLVector & GLVector::unitVector ( ) const

Returns vector with length 1.0 and same direction as ∗this.

4.9.3.39 const GLVector & GLVector::vectorMult (const GLVector & *v2*) const

Vector product ∗this X v2.

4.9.3.40 void GLVector::vertex ( ) const

Calls glVertex3dv.

4.9.3.41 GLdouble GLVector::x ( ) const

Read property for x coordinate.

4.9.3.42 GLdouble GLVector::y ( ) const

Read property for y coordinate.

4.9.3.43 GLdouble GLVector::z ( ) const

Read property for z coordinate.

4.9.4 Member Data Documentation

4.9.4.1 int GLVector::_AngleMode = GLVector::Use360Degree [static, protected]

Use radian, 360 degrees or 400 new degrees. Set with **GLVector::UseRadian** (p. 29), Use360Degree or Use400NewDegree. Default: **GLVector::Use360Degree** (p. 29)

4.9.4.2 const int GLVector::AvailableOpBuffers = 3 [static]

Number of buffer vectors available for calculations using the **GLVector** (p. 20) operators. See getNextBuffer.

4.9.4.3 int GLVector::n_Buffer = 0 [static, protected]

Number of the presently used buffer for calculations.

4.9.4.4 int GLVector::n_ConstructorCalls = 0 [static, protected]

Total number of calls to constructors

4.9.4.5 int GLVector::n_CopyCalls = 0 [static, protected]

Total number of calls to function copy

4.9.4.6 int GLVector::n_MaxBuffers = 0 [static, protected]

Maximum operator buffers used

4.9.4.7 int GLVector::n_OperatorCalls = 0 [static, protected]

Total number of operator calls

4.9.4.8 int GLVector::n_VertexCalls = 0 [static, protected]

Total number of calls to dv, fv and vertex

4.9.4.9 GLVector GLVector::OpBufferVectors [static, protected]

Array with buffer vectors for return values of operators.

4.9.4.10 const int GLVector::Use360Degree = 1 [static]

360 degree mode for polar coordinates.

4.9.4.11 const int GLVector::Use400NewDegree = 2 [static]

400 new degree mode for polar coordinates.

4.9.4.12 const int GLVector::UseRadian = 3 [static]

Radian mode for polar coordinates.

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

- OpenGL/glvector.h
- OpenGL/glvector.cpp

4.10 HeavenlyBody Class Reference

Public Member Functions

- **HeavenlyBody** (QString name, int diameter, QColor color, QString type)

- **HeavenlyBody** (qint64 id, QString name, int diameter, QColor color, QString type)
- **HeavenlyBody** (qint64 id, QString name, int diameter, QString colorString, QString type)
- void **init** (QString name, int diameter, QColor color, QString type)
- qint64 **getId** ()
- QString **getName** ()
- int **getDiameter** ()
- QColor **getColor** ()
- QString **getType** ()
- void **setId** (qint64 id)
- void **setName** (QString name)
- void **setDiameter** (int diameter)
- void **setColor** (QColor color)
- void **setType** (QString type)
- bool **operator==** (const **HeavenlyBody** &heavenlyBody)

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

- model/heavenlybody/heavenlybody.h
- model/heavenlybody/heavenlybody.cpp

4.11 HeavenlyBody3d Class Reference

Inheritance diagram for HeavenlyBody3d:

Collaboration diagram for HeavenlyBody3d:

Public Member Functions

- **HeavenlyBody3d** (**HeavenlyBody** ∗heavenlyBody)
- virtual void **paintHeavenlyBody3d** ()
- virtual void **calculateHeavenlyBody3d** ()
- void **setOrbitVisisble** (bool orbitVisisble)
- double **getRadius** ()
- **GLVector getCenter** ()
- QString **getName** ()
- double **calculateDistance** (**HeavenlyBody3d** ∗heavenlyBody3d)

Protected Member Functions

- bool **isOrbitVisisble** ()

Protected Attributes

- **GLColorRGBA color**
- float **x**
- float **y**
- **GLVector heavenlyBodyCenter**

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

- visualization/heavenlybody/heavenlybody3d.h
- visualization/heavenlybody/heavenlybody3d.cpp

4.12 HeavenlyBodyComboBoxModel Class Reference

Public Member Functions

- QModelIndex **index** (int row, int column, const QModelIndex &parent=QModelIndex()) const
- QModelIndex **parent** (const QModelIndex &index) const
- int **rowCount** (const QModelIndex &index=QModelIndex()) const
- int **columnCount** (const QModelIndex &index=QModelIndex()) const
- QVariant **data** (const QModelIndex &index, int role=Qt::DisplayRole) const
- void **setData** (QList< **HeavenlyBody** * > entities)
- **HeavenlyBody** * **getHeavenlyBody** (int index)
- int **getHeavenlyBodyIndex** (**HeavenlyBody** *heavenlyBody)

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

- model/heavenlybody/heavenlybodycomboboxmodel.h
- model/heavenlybody/heavenlybodycomboboxmodel.cpp

4.13 HeavenlyBodyDetails Class Reference

Public Member Functions

- **HeavenlyBodyDetails** (QWidget *parent=0, **HeavenlyBodyModel** *heavenlyBodyModel=0, bool edit=false)

  Open dialog window to add or edit a heavenly body.
- ~**HeavenlyBodyDetails** ()

  Destructor to close dialog.

4.13.1 Constructor & Destructor Documentation

4.13.1.1 HeavenlyBodyDetails::HeavenlyBodyDetails (QWidget * *parent* = 0, HeavenlyBodyModel * *heavenlyBodyModel* = 0, bool *isEdit* = false)

Open dialog window to add or edit a heavenly body.

Parameters

parent	Parent of this dialog.
heavenly-BodyModel	Model of the heavenly body table.
isEdit	If TRUE edit, else add heavenlybody.

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

- forms/heavenlybody/heavenlybodydetails.h
- forms/heavenlybody/heavenlybodydetails.cpp

4.14 HeavenlyBodyItemDelegate Class Reference

Public Member Functions

- **HeavenlyBodyItemDelegate** (**HeavenlyBodyModel** *heavenlyBodyModel)

  Set an other color to collumn 3 (color) when row is selected.
- void **paint** (QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex &index) const

  Set the color of the selected color-collumn to the color of the heavenly body. Others to default.

4.14.1 Constructor & Destructor Documentation

4.14.1.1 HeavenlyBodyItemDelegate::HeavenlyBodyItemDelegate (HeavenlyBodyModel * *heavenlyBodyModel*)

Set an other color to collumn 3 (color) when row is selected.

Parameters

heavenly-BodyModel	

4.14.2 Member Function Documentation

4.14.2.1 void HeavenlyBodyItemDelegate::paint (QPainter * *painter,* const QStyleOptionViewItem & *option,* const QModelIndex & *index*) const

Set the color of the selected color-collumn to the color of the heavenly body. Others to default.

Parameters

painter	Paint options
option	Style options.
index	Selected row.

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

- forms/heavenlybody/heavenlybodyitemdelegate.h
- forms/heavenlybody/heavenlybodyitemdelegate.cpp

4.15 HeavenlyBodyModel Class Reference

Public Member Functions

- void **loadAllHeavenlyBodyEntities** ()
- **HeavenlyBodyTableModel** * **getHeavenlyBodyTableModel** ()
- void **setSelectionModel** (QItemSelectionModel *selectionModel)
- **HeavenlyBody** * **getSelectedEntity** ()
- bool **isEntitySelected** ()
- void **addEntity** (QString name, int diameter, QColor color, QString type)
- void **updateEntity** (QString name, int diameter, QColor color, QString type)
- void **deleteEntity** ()

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

- model/heavenlybody/heavenlybodymodel.h
- model/heavenlybody/heavenlybodymodel.cpp

4.16 HeavenlyBodyOverview Class Reference

Public Slots

- void **on_add_clicked** ()

  Add a new heavenly body to the model.
- void **on_edit_clicked** ()

  Edit the selected heavenly body from the List.
- void **on_deleteEntity_clicked** ()

Delete the selected heavenly body.

- void **doubleClicked** (QModelIndex modelIndex)

  Enable double click to select a heavenly body.

Public Member Functions

- **HeavenlyBodyOverview** (QWidget *parent=0, **HeavenlyBodyModel** *heavenlyBodyModel=0)

  Open the window of the heanenly body overview.

- ~**HeavenlyBodyOverview** ()

  Delete the heavenly body overview window.

4.16.1 Constructor & Destructor Documentation

4.16.1.1 HeavenlyBodyOverview::HeavenlyBodyOverview (QWidget * *parent* = 0, HeavenlyBodyModel * *heavenlyBodyModel* = 0) [explicit]

Open the window of the heanenly body overview.

Parameters

parent	Parent widget of this window.
heavenlyBodyModel	Model of the overview.

4.16.2 Member Function Documentation

4.16.2.1 void HeavenlyBodyOverview::doubleClicked (QModelIndex *modelIndex*) [slot]

Enable double click to select a heavenly body.

Parameters

modelIndex	Locate the data in the model

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

- forms/heavenlybody/heavenlybodyoverview.h
- forms/heavenlybody/heavenlybodyoverview.cpp

4.17 HeavenlyBodyRepository Class Reference

```
#include <heavenlybodyrepository.h>
```

Public Member Functions

- **HeavenlyBodyRepository** ()

  Constructor: Repository of the heavenly bodies with an instance of the complete database.
- QList< **HeavenlyBody** * > **fetchAllHeavenlyBodyEntities** ()

  Create a list of all heavenly bodies stored in the database.
- QList< **HeavenlyBody** * > **fetchExplizitTypedEntities** (QString type)

  Returns a list of heavely bodies of the given type.
- void **updateEntity** (**HeavenlyBody** *heavenlyBody)

  Update the data of the given heavenly body.
- void **insertEntity** (**HeavenlyBody** *heavenlyBody)

  Add a new heavenly body to the to the database.
- void **deleteEntity** (**HeavenlyBody** *heavenlyBody)

  Delete heavenly body from the database.

4.17.1 Detailed Description

Class for heavenly body repository.

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.17.2 Member Function Documentation

4.17.2.1 void HeavenlyBodyRepository::deleteEntity (HeavenlyBody * *heavenlyBody*)

Delete heavenly body from the database.

Parameters

heavenly-Body	Heavenly body to delete.

4.17.2.2 QList< HeavenlyBody * > HeavenlyBodyRepository::fetchAllHeavenlyBodyEntities ( )

Create a list of all heavenly bodies stored in the database.

Returns

QList<HeavenlyBody *> List of all stored heavenly bodies.

4.17.2.3 QList< HeavenlyBody * > HeavenlyBodyRepository::fetchExplizitTypedEntities (QString *type*)

Returns a list of heavely bodies of the given type.

Parameters

type	Type of the heavenly body

Returns

QList<HeavenlyBody *> List of the heavenly bodies of type 'type'.

4.17.2.4 void HeavenlyBodyRepository::insertEntity (HeavenlyBody * *heavenlyBody*)

Add a new heavenly body to the to the database.

Parameters

heavenly-Body	Heavenly body to add.

4.17.2.5 void HeavenlyBodyRepository::updateEntity (HeavenlyBody * *heavenlyBody*)

Update the data of the given heavenly body.

Parameters

heavenly-Body	Heavely body to be updated.

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

- data/heavenlybody/heavenlybodyrepository.h
- data/heavenlybody/heavenlybodyrepository.cpp

4.18 HeavenlyBodyTableModel Class Reference

Public Member Functions

- int **rowCount** (const QModelIndex &parent=QModelIndex()) const
- int **columnCount** (const QModelIndex &parent=QModelIndex()) const
- QVariant **data** (const QModelIndex &index, int role) const
- QVariant **headerData** (int section, Qt::Orientation orientation, int role=Qt::-DisplayRole) const

- void **setData** (QList< **HeavenlyBody** * > entities)
- **HeavenlyBody** * **getHeavenlyBody** (int row)
- void **addHeavenlyBody** (**HeavenlyBody** *heavenlyBody)
- void **removeHeavenlyBody** (**HeavenlyBody** *heavenlyBody)
- int **getEntityCount** ()

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

- model/heavenlybody/heavenlybodytablemodel.h
- model/heavenlybody/heavenlybodytablemodel.cpp

4.19 HeavenlyBodyTypeException Class Reference

`#include <heavenlybodytypeexception.h>`

Public Member Functions

- **HeavenlyBodyTypeException** (const QString message)

  Constructor with error message only.
- virtual const QString **getMessage** () const throw ()

  Getter for message.

4.19.1 Detailed Description

Class for wrong type exceptions.

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.19.2 Constructor & Destructor Documentation

4.19.2.1 HeavenlyBodyTypeException::HeavenlyBodyTypeException (const QString *message*)

Constructor with error message only.

Parameters

message	Error message of the Object

4.19.3 Member Function Documentation

4.19.3.1 const QString HeavenlyBodyTypeException::getMessage ( ) const throw () [virtual]

Getter for message.

Returns

const QString

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

- data/exceptions/heavenlybodytypeexception.h
- data/exceptions/heavenlybodytypeexception.cpp

4.20 Light Class Reference

Public Member Functions

- void **enable** ()
- void **disable** ()

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

- visualization/light/light.h
- visualization/light/light.cpp

4.21 MainWindow Class Reference

Public Member Functions

- **MainWindow** (QWidget *parent=0)

  Open the main window and establish the database connection.
- ~**MainWindow** ()

  Destructor for the main window.

4.21.1 Constructor & Destructor Documentation

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

Open the main window and establish the database connection.

Parameters

parent	

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

- forms/main/mainwindow.h
- forms/main/mainwindow.cpp

4.22 Orbit3d Class Reference

Public Member Functions

- **Orbit3d** (double angle, double orbitalPlaneAngle, **GLColorRGBA** color, float a, float b, float e)
- void **paintOrbit3d** ()
- void **drawEllipse** ()

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

- visualization/orbit/orbit3d.h
- visualization/orbit/orbit3d.cpp

4.23 Planet3d Class Reference

Inheritance diagram for Planet3d:

Collaboration diagram for Planet3d:

Public Member Functions

- **Planet3d** (**SolarSystemHeavenlyBody** ∗solarSystemHeavenlyBody, const float keplerConstant)
- void **paintHeavenlyBody3d** ()
- void **calculateHeavenlyBody3d** ()

4.23.1 Constructor & Destructor Documentation

4.23.1.1 Planet3d::Planet3d (SolarSystemHeavenlyBody ∗ *solarSystemHeavenlyBody,* const float *keplerConstant*)

Parameters

solar-System-Heavenly-Body	
kepler-Constant	Reference Value for the Planet speed

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

- visualization/heavenlybody/planet3d.h
- visualization/heavenlybody/planet3d.cpp

4.24 PostgreSQLDatabase Class Reference

`#include <postgresqldatabase.h>`

Public Member Functions

- void **transaction** ()

  Start database transaction.
- void **commit** ()

  Commit database operations.
- void **rollback** ()

  Rollback the database operations.

Static Public Member Functions

- static **PostgreSQLDatabase** * **getInstance** ()

  *Make **PostgreSQLDatabase** (p. 40) Object if none exists.*

Protected Member Functions

- **PostgreSQLDatabase** ()

  Establish database connection. On windows systems use ODBC, on other systems the native PostgreSQL driver.

4.24.1 Detailed Description

Class for connect to database.

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.24.2 Member Function Documentation

4.24.2.1 PostgreSQLDatabase * PostgreSQLDatabase::getInstance ( ) [static]

Make **PostgreSQLDatabase** (p. 40) Object if none exists.

Returns

PostgreSQLDatabase (p. 40) * Return the **PostgreSQLDatabase** (p. 40) Object.

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

- database/postgresqldatabase.h
- database/postgresqldatabase.cpp

4.25 PropertyNotValidException Class Reference

Public Member Functions

- **PropertyNotValidException** (const QString property, const QString message)
- virtual const QString **getMessage** () const throw ()
- virtual const QString **getProperty** () const throw ()

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

- model/exceptions/propertynotvalidexception.h
- model/exceptions/propertynotvalidexception.cpp

4.26 SimulationView Class Reference

Signals

- void **simulationStopped** ()
- void **collisionDetectionDeactivated** ()

Public Member Functions

- **SimulationView** (QWidget *parent=0, **SolarSystemSimulation** *solarSystem-Simulation=0)

  Class to show the scene, move the camera and initialize the simulation view.
- void **setSolarSystem** (**SolarSystem** *solarSystem)
- void **startSimulation** ()
- void **stopSimulation** ()
- void **resetPerspective** ()
- bool **isSimulationStarted** ()
- void **toggleCoordinateAxesVisibility** ()

Protected Member Functions

- void **keyPressEvent** (QKeyEvent *keyEvent)

4.26.1 Constructor & Destructor Documentation

4.26.1.1 SimulationView::SimulationView (QWidget * *parent* = 0, SolarSystemSimulation * *solarSystemSimulation* = 0) [explicit]

Class to show the scene, move the camera and initialize the simulation view.

Parameters

parent	
solar-System-Simulation	

4.26.2 Member Function Documentation

4.26.2.1 bool SimulationView::isSimulationStarted ( )

Returns

bool

4.26.2.2 void SimulationView::keyPressEvent (QKeyEvent * *keyEvent*) [protected]

Parameters

keyEvent	

4.26.2.3 void SimulationView::setSolarSystem (SolarSystem * *solarSystem*)

Parameters

solarSystem	

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

- forms/simulation/simulationview.h
- forms/simulation/simulationview.cpp

4.27 SolarSystem Class Reference

Public Member Functions

- **SolarSystem** (QString name, **HeavenlyBody** *centralStar)
- **SolarSystem** (qint64 id, QString name, **HeavenlyBody** *centralStar)
- void **init** (QString name, **HeavenlyBody** *centralStar)

- qint64 **getId** ()
- QString **getName** ()
- **HeavenlyBody** ∗ **getCentralStar** ()
- int **getPlanetCount** ()
- QList< **SolarSystemHeavenlyBody** ∗ > **getHeavenlyBodies** ()
- void **setId** (qint64 id)
- void **setName** (QString name)
- void **setCentralStar** (**HeavenlyBody** ∗centralStar)
- void **addHeavenlyBody** (**SolarSystemHeavenlyBody** ∗solarSystemHeavenlyBody)
- void **removeHeavenlyBody** (**SolarSystemHeavenlyBody** ∗solarSystemHeavenlyBody)

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

- model/solarsystem/solarsystem.h
- model/solarsystem/solarsystem.cpp

4.28 SolarSystemDetails Class Reference

Public Member Functions

- **SolarSystemDetails** (QWidget ∗parent=0, **SolarSystemModel** ∗solarSystemModel=0, bool isEdit=false)
- **HeavenlyBodyComboBoxModel** ∗ **getHeavenlyBodyComboBoxModel** ()

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

- forms/solarsystem/solarsystemdetails.h
- forms/solarsystem/solarsystemdetails.cpp

4.29 SolarSystemHeavenlyBody Class Reference

Public Member Functions

- **SolarSystemHeavenlyBody** (**HeavenlyBody** ∗heavenlyBody, double numericExcentricity, double semimajorAxis, double angle, double orbitalPlaneAngle)
- **HeavenlyBody** ∗ **getHeavenlyBody** ()
- double **getNumericExcentricity** ()
- double **getSemimajorAxis** ()
- double **getAngle** ()
- double **getOrbitalPlaneAngle** ()
- void **setHeavenlyBody** (**HeavenlyBody** ∗heavenlyBody)
- void **setNumericExcentricity** (double numericExcentricity)

- void **setSemimajorAxis** (double semimajorAxis)
- void **setAngle** (double angle)
- void **setOrbitalPlaneAngle** (double orbitalPlaneAngle)
- bool **operator==** (const **SolarSystemHeavenlyBody** &solarSystemHeavenlyBody)

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

- model/solarsystem/solarsystemheavenlybody.h
- model/solarsystem/solarsystemheavenlybody.cpp

4.30 SolarSystemHeavenlyBodyTableModel Class Reference

Public Member Functions

- int **rowCount** (const QModelIndex &parent=QModelIndex()) const
- int **columnCount** (const QModelIndex &parent=QModelIndex()) const
- QVariant **data** (const QModelIndex &index, int role) const
- QVariant **headerData** (int section, Qt::Orientation orientation, int role=Qt::DisplayRole) const
- void **setData** (QList< **SolarSystemHeavenlyBody** * > entities)
- void **addSolarSystemHeavenlyBody** (**SolarSystemHeavenlyBody** *solarSystemHeavenlyBody)
- void **deleteSolarSystemHeavenlyBody** (**SolarSystemHeavenlyBody** *solarSystemHeavenlyBody)
- **SolarSystemHeavenlyBody** * **getSolarSystemHeavenlyBody** (int row)
- int **getEntityCount** ()
- int **getSolarSystemHeavenlyBodyIndex** (**SolarSystemHeavenlyBody** *solarSystemHeavenlyBody)
- void **reset** ()
- void **resetData** ()

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

- model/solarsystem/solarsystemheavenlybodytablemodel.h
- model/solarsystem/solarsystemheavenlybodytablemodel.cpp

4.31 SolarSystemItemDelegate Class Reference

Public Member Functions

- **SolarSystemItemDelegate** (**SolarSystemModel** *solarSystemModel)

  Set the color of the planet collumn 3 (color) when row is selected.
- void **paint** (QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex &index) const

  Set the color of the selected color-collumn to the color of the heavenly body. Others to default.

4.31.1 Constructor & Destructor Documentation

4.31.1.1 SolarSystemItemDelegate::SolarSystemItemDelegate (SolarSystemModel * *solarSystemModel*)

Set the color of the planet collumn 3 (color) when row is selected.

Parameters

solar-System-Model	

4.31.2 Member Function Documentation

4.31.2.1 void SolarSystemItemDelegate::paint (QPainter * *painter,* const QStyleOptionViewItem & *option,* const QModelIndex & *index*) const

Set the color of the selected color-collumn to the color of the heavenly body. Others to default.

Parameters

painter	Paint options
option	Style options.
index	Selected row.

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

- forms/solarsystem/solarsystemitemdelegate.h
- forms/solarsystem/solarsystemitemdelegate.cpp

4.32 SolarSystemModel Class Reference

Public Member Functions

- void **loadAllSolarSystemEntities** ()
- void **loadOtherEntities** ()
- void **loadEntityData** ()
- void **setSolarSystemSelectionModel** (QItemSelectionModel *selectionModel)
- void **setSolarSystemHeavenlyBodySelectionModel** (QItemSelectionModel *selectionModel)
- **SolarSystemTableModel** * **getSolarSystemTableModel** ()
- **SolarSystemHeavenlyBodyTableModel** * **getSolarSystemHeavenlyBody-TableModel** ()
- **HeavenlyBodyComboBoxModel** * **getStarsComboBoxModel** ()
- **HeavenlyBodyComboBoxModel** * **getPlanetsComboBoxModel** ()

- **SolarSystem** ∗ **getCurrentSolarSystem** ()
- **SolarSystemHeavenlyBody** ∗ **getCurrentSolarSystemHeavenlyBody** ()
- void **createSolarSystem** (QString name, int centralStarIndex)
- void **updateSolarSystem** (QString name, int centralStarIndex)
- void **deleteSolarSystem** ()
- void **addPlanet** (int planetIndex, double excentricity, double semimajorAxis, double angle, double orbitalPlaneAngle)
- void **updatePlanet** (int planetIndex, double excentricity, double semimajorAxis, double angle, double orbitalPlaneAngle)
- void **deletePlanet** ()
- bool **isEntitySelected** ()
- int **getSelectedStarIndex** ()
- int **getSelectedHeavenlyBodyIndex** ()
- void **resetSolarSystemEntityData** ()

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

- model/solarsystem/solarsystemmodel.h
- model/solarsystem/solarsystemmodel.cpp

4.33 SolarSystemOverview Class Reference

Signals

- void **simulateSolarSystem** (**SolarSystem** ∗solarSystem)

Public Member Functions

- **SolarSystemOverview** (QWidget ∗parent=0, **SolarSystemModel** ∗solarSystemModel=0)

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

- forms/solarsystem/solarsystemoverview.h
- forms/solarsystem/solarsystemoverview.cpp

4.34 SolarSystemRepository Class Reference

```
#include <solarsystemrepository.h>
```

Public Member Functions

- **SolarSystemRepository** ()

  Constructor: Repository of the solar systems and their data.
- QList< **SolarSystem** * > **fetchAllSolarSystemEntities** ()

  Create a list of all solar systems stored in the database.
- void **insertEntity** (**SolarSystem** *solarSystem)

  Insert a new solar system.
- void **updateEntity** (**SolarSystem** *solarSystem)

  Update the main components of the database. Name and central star.
- void **deleteEntity** (**SolarSystem** *solarSystem)

  Delete complete solarsystem.
- void **insertPlanetEntity** (**SolarSystem** *solarSystem, **SolarSystemHeavenlyBody** *solarSystemHeavenlyBody)

  Add a new heavenly body to an existing solar system.
- void **updatePlanetEntity** (**SolarSystem** *solarSystem, **SolarSystemHeavenlyBody** *solarSystemHeavenlyBody, **SolarSystemHeavenlyBody** *oldSolarSystemHeavenlyBody)

  Update the data of an existing solar system.
- void **deletePlanetEntity** (**SolarSystem** *solarSystem, **SolarSystemHeavenlyBody** *solarSystemHeavenlyBody)

  Delete an existing heavenly body from a solar system.

4.34.1 Detailed Description

Class for solar system repository.

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.34.2 Member Function Documentation

4.34.2.1 void SolarSystemRepository::deleteEntity (SolarSystem * *solarSystem*)

Delete complete solarsystem.

Parameters

solarSystem	Solar system to delete.

4.34.2.2 void SolarSystemRepository::deletePlanetEntity (SolarSystem * *solarSystem*, SolarSystemHeavenlyBody * *solarSystemHeavenlyBody*)

Delete an existing heavenly body from a solar system.

Parameters

solarSystem	Solar system to change.
solarSystemHeavenlyBody	Heavenly body to be deleted.

4.34.2.3 QList< SolarSystem * > SolarSystemRepository::fetchAllSolarSystemEntities ( )

Create a list of all solar systems stored in the database.

Returns

QList<HeavenlyBody *> List of all stored solar systems including all data.

4.34.2.4 void SolarSystemRepository::insertEntity (SolarSystem * *solarSystem*)

Insert a new solar system.

Parameters

solarSystem	Object of the new solar system.

4.34.2.5 void SolarSystemRepository::insertPlanetEntity (SolarSystem * *solarSystem*, SolarSystemHeavenlyBody * *solarSystemHeavenlyBody*)

Add a new heavenly body to an existing solar system.

Parameters

solarSystem	Solar system which is to expand.
solarSystemHeavenlyBody	The new heavenly body in the solar system.

4.34.2.6 void SolarSystemRepository::updateEntity (SolarSystem * *solarSystem*)

Update the main components of the database. Name and central star.

Parameters

solarSystem	

4.34.2.7 void SolarSystemRepository::updatePlanetEntity (SolarSystem * *solarSystem*, SolarSystemHeavenlyBody * *solarSystemHeavenlyBody*, SolarSystemHeavenlyBody * *oldSolarSystemHeavenlyBody*)

Update the data of an existing solar system.

Parameters

solarSystem	Name of the solar system to be updated.
solarSystemHeavenlyBody	New parameters of the solar system.
oldSolarSystemHeavenlyBody	Old parameter of the solar system.

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

- data/solarsystem/solarsystemrepository.h
- data/solarsystem/solarsystemrepository.cpp

4.35 SolarSystemSimulation Class Reference

Signals

- void **collisionDetected** (**HeavenlyBody3d** *firstHeavenlyBody3d, **HeavenlyBody3d** *secondHeavenlyBody3d)

Public Member Functions

- void **paintSolarSystem3d** ()
- void **calculateSolarSystem3d** ()
- void **setSolarSystem** (**SolarSystem** *solarSystem)
- void **setOrbitVisible** (bool orbitVisible)
- void **setCollisionDetection** (bool collisionDetection)
- void **setKeplersLawDefault** (bool keplerDefault)
- float **getMaxSemimajorAxis** ()
- bool **isSolarSystemAvailable** ()
- QString **getSolarSystemName** ()

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

- simulation/solarsystemsimulation.h
- simulation/solarsystemsimulation.cpp

4.36 SolarSystemTableModel Class Reference

Public Member Functions

- int **rowCount** (const QModelIndex &parent=QModelIndex()) const
- int **columnCount** (const QModelIndex &parent=QModelIndex()) const
- QVariant **data** (const QModelIndex &index, int role) const
- QVariant **headerData** (int section, Qt::Orientation orientation, int role=Qt::-DisplayRole) const
- void **setData** (QList< **SolarSystem** * > entities)
- void **addSolarSystem** (**SolarSystem** *solarSystem)
- void **deleteSolarSystem** (**SolarSystem** *solarSystem)
- **SolarSystem** * **getSolarSystem** (int row)
- int **getEntityCount** ()

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

- model/solarsystem/solarsystemtablemodel.h
- model/solarsystem/solarsystemtablemodel.cpp

4.37 SqlQueryException Class Reference

`#include <sqlqueryexception.h>`

Public Member Functions

- **SqlQueryException** (const QString message, QString sqlError)

  Constructor for the exception with error message and SQL-error.
- virtual const QString **getMessage** () const throw ()

  Getter for message.
- virtual const QString **getSqlError** () const throw ()

  Getter for sqlError.

4.37.1 Detailed Description

Class for SQL query exceptions.

Author

Fabian Deitelhoff <FH@FabianDeitelhoff.de>
Christof Geisler <christof.geisler@stud.fh-swf.de>

4.37.2 Constructor & Destructor Documentation

4.37.2.1 SqlQueryException::SqlQueryException (const QString *message*, QString *sqlError*)

Constructor for the exception with error message and SQL-error.

Parameters

message	Error message of the Object
sqlError	SQL Error of the Object

4.37.3 Member Function Documentation

4.37.3.1 const QString SqlQueryException::getMessage ( ) const throw () [virtual]

Getter for message.

Returns

const QString

4.37.3.2 const QString SqlQueryException::getSqlError ( ) const throw () [virtual]

Getter for sqlError.

Returns

const QString

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

- data/exceptions/sqlqueryexception.h
- data/exceptions/sqlqueryexception.cpp

4.38 Star3d Class Reference

Inheritance diagram for Star3d:

Collaboration diagram for Star3d:

Public Member Functions

- **Star3d** (**HeavenlyBody** ∗heavenlyBody)
- void **paintHeavenlyBody3d** ()

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

- visualization/heavenlybody/star3d.h
- visualization/heavenlybody/star3d.cpp