Simple Analyzer

Erzeugt von Doxygen 1.8.6

Son Mai 25 2014 18:26:45

Inhaltsverzeichnis

Kapitel 1

SimpleAnalyzer

Dies ist die Dokumentation der Programmquellen des SimplaAnalyzer-Programmpakets.Für Informationen über die Verwendung der Programme konsultieren Sie bitte das Handbuch der Software.

Alle in https://github.com/vroland/SimpleAnalyzer zur Für das Simpleanalyzer-Softwarepaket gelten die Lizenzbestimmumgen der GNU Affero General Public License. Genauere Informationen sind der LICENSE-Datei zu entnehmen.

Autor

Valentin Roland

Kapitel 2

SimpleAnalyzer

Das SimpleAnalyzer-Softwarepaket enthält Programme zur Auswertung physikalischer Versuche für debianbasierte Betriebssysteme. Mithilfe der enthaltenen Software sind Sie im Stande, Temperaturmessdaten aus einer .csv-Datei oder Messwerte des ODiSI-Instruments von Luna in ein einheitliches Format umzuwandeln und zusammenzuführen.

Über eine grafische Oberfläche ist es möglich, mithilfe der so aufbereiteten Daten Auswertungen wie eine Temperaturverteilung über ein dreidimensionales Modell oder das Bestimmen des Wärmegehalts vorzunehmen und den Versuch zu visualisieren.

Zur weiteren Nutzung der Ergebnisse können diese, beispielsweise als VTK-Datei oder PNG-Grafik, exportiert werden.

Quelltext, Handbuch, Dokumentation und Beispiele sowie Binärdateien finden Sie unter `https://github.com/vroland/SimpleAnalyzer`.

Lizenz

Für das Simpleanalyzer-Softwarepaket gelten die Lizenzbestimmumgen der GNU Affero General Public License. Genauere Informationen sind der LICENSE-Datei zu entnehmen.

Handbuch

Im Handbuch zum Programm finden Sie Informationen zur Installation und Bedienung der Programme. Es liegt im pdf-Format unter simpleanalyzer-gui/Debug/simpleanalyzer-man.pdf vor und kann über das Hilfemenü in SimpleAnalyzer-GUI aufgerufen werden.

Dokumentation

Eine Dokumentation für die Weiterentwicklung der Software befindet sich im Verzeichnis doc/html.

Kapitel 3

Verzeichnis der Namensbereiche

3.1 Liste aller Namensbereiche

Liste aller Namensbereiche mit Kurzbeschreibung:

Kapitel 4

Hierarchie-Verzeichnis

4.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Kapitel 5

Klassen-Verzeichnis

5.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

Kapitel 6

Datei-Verzeichnis

6.1 Auflistung der Dateien

Hier folgt die Aufzählung aller Dateien mit einer Kurzbeschreibung:

Kapitel 7

Dokumentation der Namensbereiche

7.1 std-Namensbereichsreferenz

Klassen

- class vector

7.2 Utils-Namensbereichsreferenz

allgemeine Funktionen und Typen.

Klassen

- struct Visualization_info

  Informationen über die Farbgebung bei der Visualisierung.
- struct SortStruct

  Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.
- struct SensorPoint

  Daten eines Sensordatenpunktes.
- struct CutRender_info

  Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.
- struct SensorData

  Ein Sensordatensatz.
- struct SensorPointComparator

  Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.

Aufzählungen

- enum PIM_algorithm { ALGORITHM_TETRAHEDRONS = 0, ALGORITHM_RAY }

  Zum Punkt-in-Volumen Testen verwendeter Algorithmus.

Funktionen

- double sqr (double d)

  Quadriert eine Zahl.
- float clampHue (float h)

Begrenzt einen Wert auf den Bereich 0..1.

- string floattostr (double val)

  Hilfsfunktion zur Umwandlung einer Zahl in einen String.
- wxString floattowxstr (double val)

  Wandelt eine Fließkommazahl in einen wxWidgets-String um.
- wxString floattowxstr (double val, int digits)

  Wandelt eine Fließkommazahl in einen wxWidgets-String um.
- int rayIntersectsTriangle (Vector3D *p, Vector3D *direction, Triangle *tri, double *depth)

  Testet, ob ein Strahl ein Dreieck schneidet.
- int pointInsideMesh (Vector3D *p, tetgenio *io, PIM_algorithm algorithm)

  Testet, ob sich ein Punkt innerhalb eines Körpers befindet.
- int pointInsideTetrahedron (Vector3D *pges, Vector3D *v1, Vector3D *v2, Vector3D *v3, Vector3D *v4)

  Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.
- int pointInsideTetrahedron (double *pges, double *v1, double *v2, double *v3, double *v4)

  Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.
- int pointInsideTetrahedron (double *p, vector< SensorPoint * > *tet)

  Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.
- void nextCombination (vector< int > *indices, int depth, int dataPointCount)

  Ermöglicht das generieren aller möglichen Verteilungen von 4 Elementen auf dataPointCount Plätze.
- double getPointValue (int &status, vector< SensorPoint > *sensorpoints, double *p, Interpolator *interpolator, vector< SensorPoint * > *prev_tet=NULL, vector< SensorPoint * > *current_tet=NULL)

  Gibt den inter/extrapolierten Wert eines Punktes zurück.
- float * hsvToRgb (float h, float s, float v)

  Wandelt eine Farbe im HSV-Format ins RGB-Format um.
- void copySensorPoint (SensorPoint *from, SensorPoint *to)

  Kopiert die Eigenschaften eines Sensorpunktes in einen Anderen.

7.2.1 Ausführliche Beschreibung

allgemeine Funktionen und Typen.

7.2.2 Dokumentation der Aufzählungstypen

7.2.2.1 enum Utils::PIM_algorithm

Zum Punkt-in-Volumen Testen verwendeter Algorithmus.

Dies wird bei ALGORITHM_TETRAHEDRONS über alle Tetraeder des Objekts und deren Flächennormalen ermittelt. Bei ALGORITHM_RAY werden die Schnittpunkte aller Außenflächen mit einem Strahl gezählt (Aktuell nicht verwendet).

Aufzählungswerte

ALGORITHM_TETRAHEDRONS

ALGORITHM_RAY

Definiert in Zeile 31 der Datei utils.h.

7.2.3 Dokumentation der Funktionen

7.2.3.1 float Utils::clampHue (float *h*)

Begrenzt einen Wert auf den Bereich 0..1.

Parameter

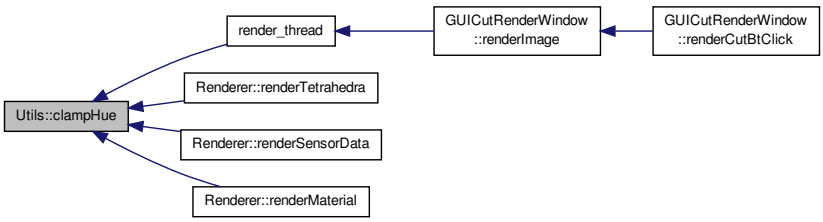
h	Die zu begrenzende Zahl.

Rückgabe

Der den Grenzen entsprechende Wert.

Definiert in Zeile 39 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.2 void Utils::copySensorPoint (SensorPoint * *from*, SensorPoint * *to*)

Kopiert die Eigenschaften eines Sensorpunktes in einen Anderen.

Parameter

from	Quelle.
to	Ziel.

Definiert in Zeile 97 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

7.2.3.3 string Utils::floattostr (double *val*) [inline]

Hilfsfunktion zur Umwandlung einer Zahl in einen String.

Parameter

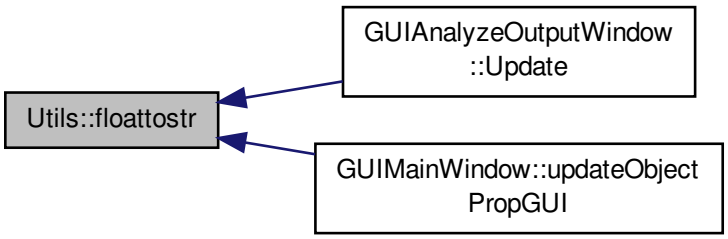
val	Die umzuwandelnde Zahl.

Rückgabe

Der resultierende String.

Definiert in Zeile 130 der Datei utils.h.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.4 wxString Utils::floattowxstr (double *val*)

Wandelt eine Fließkommazahl in einen wxWidgets-String um.

Parameter

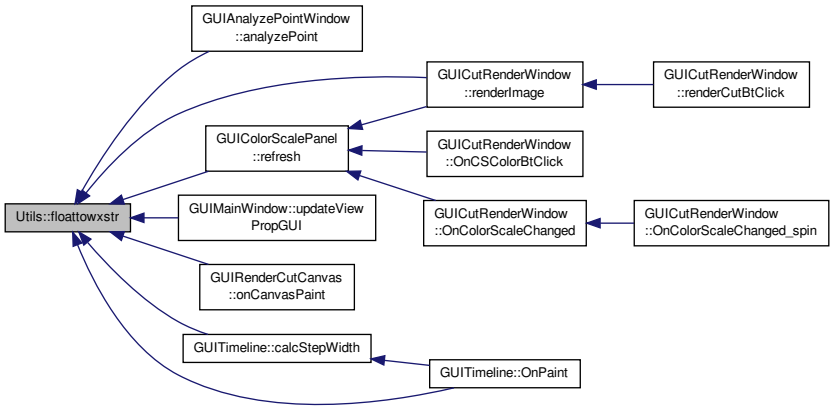
val	Die umzuwandelnde Zahl.

Rückgabe

Der entstandene String.

Definiert in Zeile 49 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.5 wxString Utils::floattowxstr (double *val,* int *digits*)

Wandelt eine Fließkommazahl in einen wxWidgets-String um.

Parameter

val	Die umzuwandelnde Zahl.
digits	Anzahl der zu übernehmenden Stellen.

Rückgabe

Der entstandene String.

Definiert in Zeile 55 der Datei utils.cpp.

7.2.3.6 double Utils::getPointValue (int & *status,* vector< SensorPoint > * *sensorpoints,* double * *p,* Interpolator * *interpolator,* vector< SensorPoint * > * *prev_tet* = NULL, vector< SensorPoint * > * *current_tet* = NULL)

Gibt den inter/extrapolierten Wert eines Punktes zurück.

Parameter

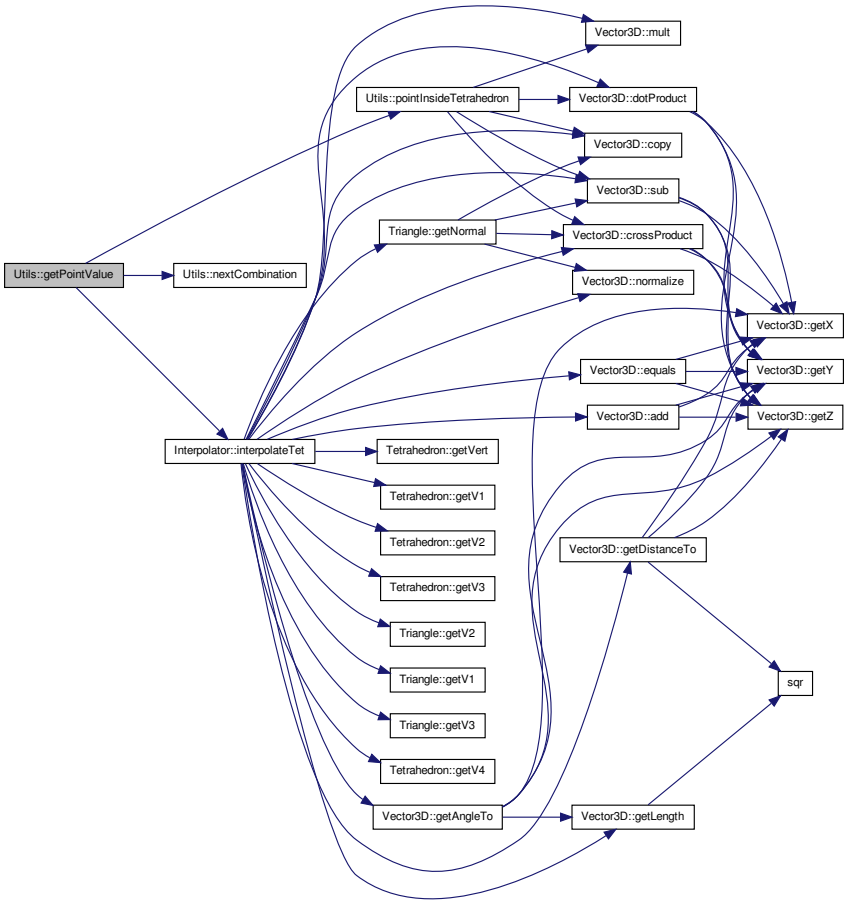
status	Rückgabevariable. 1: Punkt wurde extrapoliert 0: Punkt wurde interpoliert. -1: Alle Sensorpunkte sind komplanar.
sensorpoints	Die zu verwendenden Senosorpunkte.
p	Die Koordinaten des gesuchten Punktes.
interpolator	Das zu verwendende Interpolatorobjekt.
prev_tet	Zuerst zu Testender Tetraeder (optional, NULL zum Nichtverwenden).
current_tet	Rückgabevariable für den zuletzt verwendeten Tetraeder (optional, NULL zum Nichtverwenden).

Rückgabe

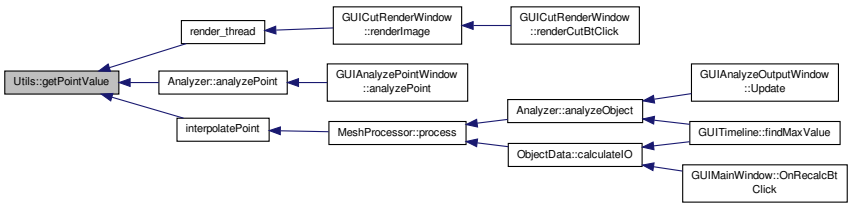
Temperatur des gesuchten Punktes.

Definiert in Zeile 354 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.7 float ∗ Utils::hsvToRgb (float *h*, float *s*, float *v*)

Wandelt eine Farbe im HSV-Format ins RGB-Format um.

Gefunden unter http://stackoverflow.com/questions/8208905/hsv-0-255-to-rgb-0-255. (5.9.13)

Parameter

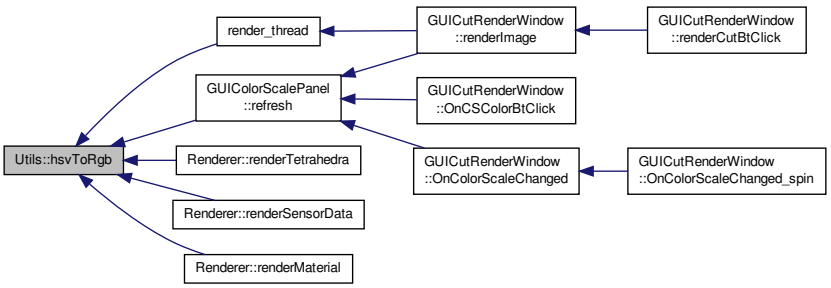
h	H-Komponente der Farbe.
s	S-Komponente der Farbe.
v	V-Komponente der Farbe.

Rückgabe

RGB-Farbe als Liste mit 3 Werten im Bereich 0..1. Muss manuell mit delete[] freigegeben werden!

Definiert in Zeile 61 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.8 void Utils::nextCombination (vector< int > * *indices*, int *depth*, int *dataPointCount*)

Ermöglicht das generieren aller möglichen Verteilungen von 4 Elementen auf dataPointCount Plätze.

Die Indices der Plätze, die die Elemente jeweils besetzten stehen in indices. Verschiedene Reihenfolgen der selben Indices werden dabei nicht generiert. Diese Funktion generiert aus der vorherigen Anordnung die Nächste, indem die Indices bis zum überlauf hochgezählt wird, woraufhin der vorhergehende erhöht wird, z.b. für dataPointCount = 8:

0 1 2 3

0 1 2 4

0 1 2 5

0 1 2 6

0 1 2 7

0 1 2 8 -> Umschlag

0 1 3 4

0 1 3 5

0 1 3 6

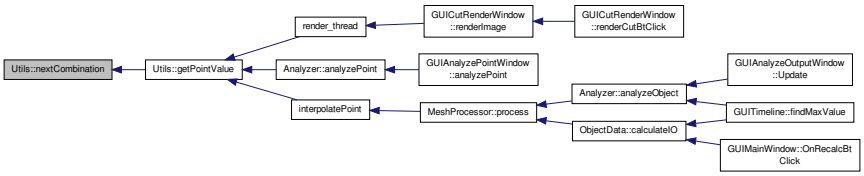
Parameter

indices	Liste der Indices der Elemente.
depth	Index des in diesem Funktionsaufruf verarbeiteten Elements. Beim ersten Aufruf also 3.

dataPointCount	Anzahl der Plätze.

Definiert in Zeile 16 der Datei utils.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.9 int Utils::pointInsideMesh (Vector3D ∗ *p*, tetgenio ∗ *io*, PIM_algorithm *algorithm*)

Testet, ob sich ein Punkt innerhalb eines Körpers befindet.

Parameter

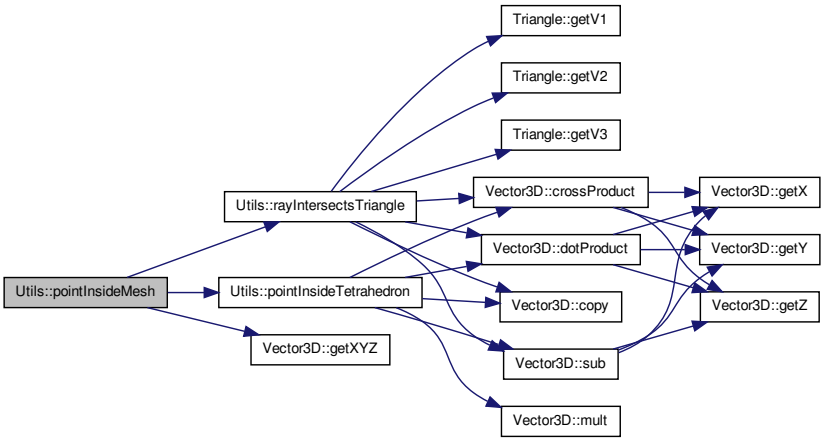
p	Der zu testende Punkt.
io	Der zu testende Körper als Tetgen-Daten (s. Tetgen Dokumentation).
algorithm	Der zu verwendende Testalgorithmus (Empfohlen und ausschließlich verwendet: ALGORITHM_TETRAHEDRONS).

Rückgabe

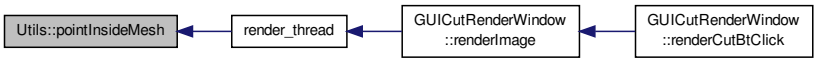
1 Wenn innerhalb, 0 wenn außerhalb. Bei einer falschen Algorithmuskonstante -1.

Definiert in Zeile 170 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.10 int Utils::pointInsideTetrahedron (Vector3D ∗ *pges,* Vector3D ∗ *v1,* Vector3D ∗ *v2,* Vector3D ∗ *v3,* Vector3D ∗ *v4*)

Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

Parameter

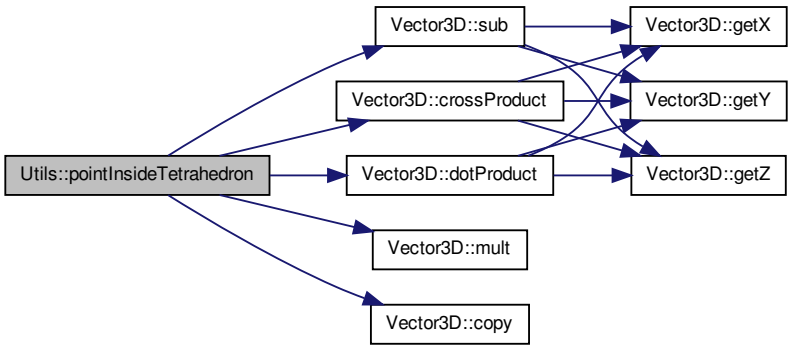
pges	Der zu testende Punkt.
v1	Der 1. Punkt des Tetraeders.
v2	Der 2. Punkt des Tetraeders.
v3	Der 3. Punkt des Tetraeders.
v4	Der 4. Punkt des Tetraeders.

Rückgabe

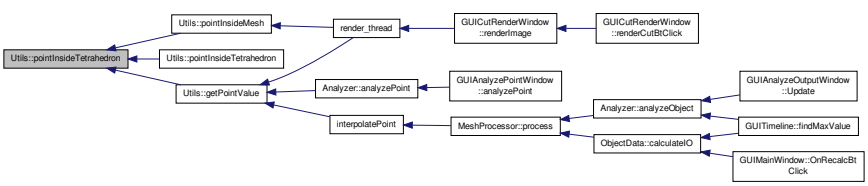
1 Wenn innerhalb, 0 wenn außerhalb. -1, wenn der Tetraeder komplanar ist.

Definiert in Zeile 249 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.11 int Utils::pointInsideTetrahedron (double * *pges*, double * *v1*, double * *v2*, double * *v3*, double * *v4*)

Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

Parameter

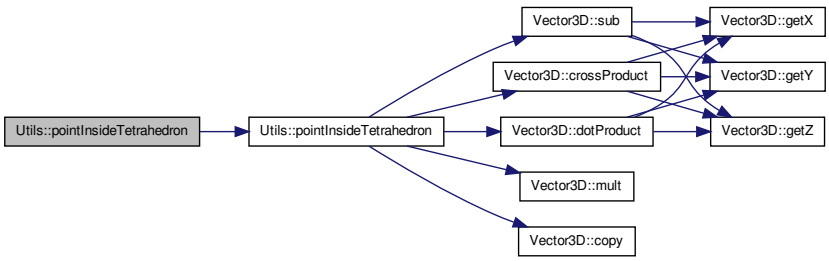
pges	Koordinaten des zu testenden Punktes.
v1	Koordinaten des 1. Punktes des Tetraeders.
v2	Koordinaten des 2. Punktes des Tetraeders.
v3	Koordinaten des 3. Punktes des Tetraeders.
v4	Koordinaten des 4. Punktes des Tetraeders.

Rückgabe

1 Wenn innerhalb, 0 wenn außerhalb. -1, wenn der Tetraeder komplanar ist.

Definiert in Zeile 343 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.2.3.12 int Utils::pointInsideTetrahedron (double * *p*, vector< SensorPoint * > * *tet*)

Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

Parameter

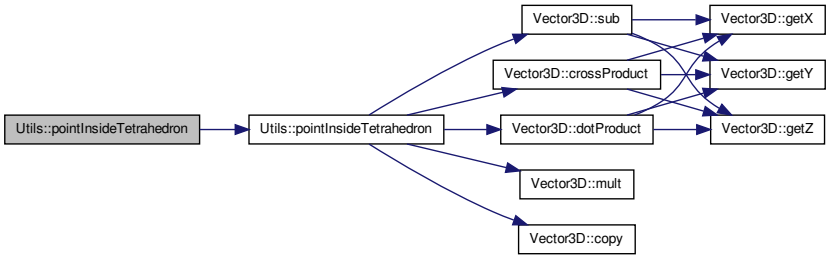
p	Koordinaten des zu testenden Punktes.
tet	Der zu untersuchende Tetraeder als Liste von Sensordaten.

Rückgabe

1 Wenn innerhalb, 0 wenn außerhalb. -1, wenn der Tetraeder komplanar ist.

Definiert in Zeile 333 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



7.2.3.13 int Utils::rayIntersectsTriangle (Vector3D * *p*, Vector3D * *direction*, Triangle * *tri*, double * *depth*)

Testet, ob ein Strahl ein Dreieck schneidet.

Gefunden unter http://en.wikipedia.org/wiki/M%C3%B6ller%E2%80%93Trumbore_intersection_algorithm am 4.9.13 und auf C++ und eigene Datentypen portiert.

Parameter

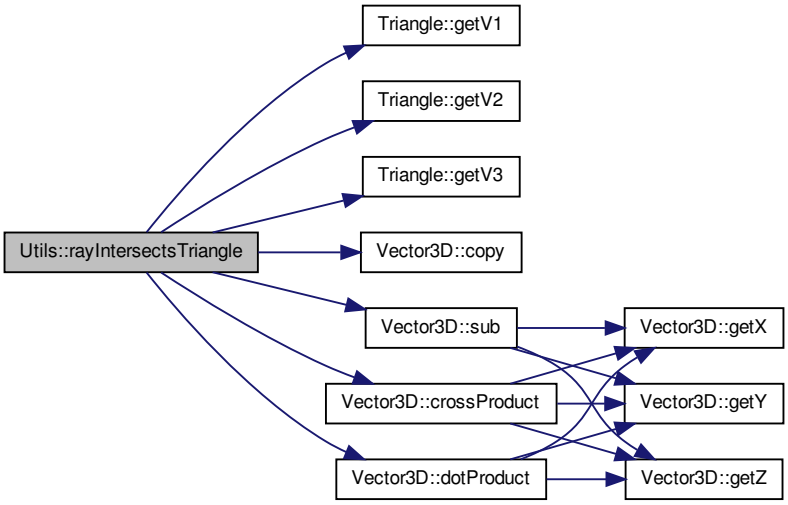
p	Ortsvektor zum Ausganspunkt des Strahls.
direction	Richtung des Strahls.
tri	Das zu testende Dreieck.
depth	Ausgabevariablie, ein Maß für den Abstand von Ausganspunkt zu Schnittpunkt.

Rückgabe

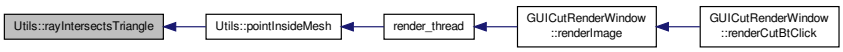
Gibt 1 zurück, wenn es einen Schnittpunkt gibt, ansonsten 0.

Definiert in Zeile 105 der Datei utils.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



7.2.3.14 double Utils::sqr (double *d*) [inline]

Quadriert eine Zahl.

Parameter

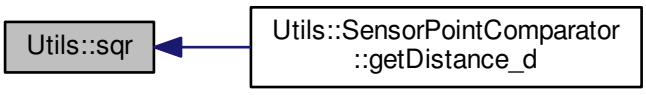
d	Die zu quadrierende Zahl.

Rückgabe

d^2.

Definiert in Zeile 40 der Datei utils.h.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Kapitel 8

Klassen-Dokumentation

8.1 Analyzer Klassenreferenz

Ermittelt Daten aus der Temperaturverteilung.

`#include <Analyzer.h>`

Klassen

- struct AnalyzerData_dataset

  Analyseergebnisse für einen Sensordatensatz.
- struct AnalyzerData_material

  Analyseergebnisse für ein Material.
- struct AnalyzerData_object

  Analyseergebnisse für ein Objekt.
- struct AnalyzerData_point

  Analyseergebnisse für einen Punkt.

Öffentliche Methoden

- Analyzer ()

  Der Konstruktor.
- void analyzeObject (ObjectData ∗obj, AnalyzerData_object ∗out, bool use_markers=true, int sdindex=-1)

  Ermittelt Daten für ein Objekt.
- void analyzePoint (ObjectData ∗obj, Vector3D ∗point, AnalyzerData_point ∗point_data, Interpolator ∗interpolator)

  Ermittelt Daten für einen Punkt am aktuell ausgewählten Zeitpunkt.
- virtual ∼Analyzer ()

  Der Destruktor.

Freundbeziehungen

- std::ostream & operator<< (std::ostream &out, const AnalyzerData_object &data)

  Operator zum Ausgeben der Analysedaten für ein Objekt in einem Stream.

8.1.1 Ausführliche Beschreibung

Ermittelt Daten aus der Temperaturverteilung.

Definiert in Zeile 21 der Datei Analyzer.h.

8.1.2 Beschreibung der Konstruktoren und Destruktoren

8.1.2.1 Analyzer::Analyzer ( )

Der Konstruktor.

Definiert in Zeile 16 der Datei Analyzer.cpp.

8.1.2.2 Analyzer::~Analyzer ( ) [virtual]

Der Destruktor.

Definiert in Zeile 191 der Datei Analyzer.cpp.

8.1.3 Dokumentation der Elementfunktionen

8.1.3.1 void Analyzer::analyzeObject (ObjectData * *obj*, AnalyzerData_object * *out*, bool *use_markers* = true, int *sdindex* = -1)

Ermittelt Daten für ein Objekt.

Objekt zum Vergleichen von Messpunkten hinsichtlich der Temperatur.

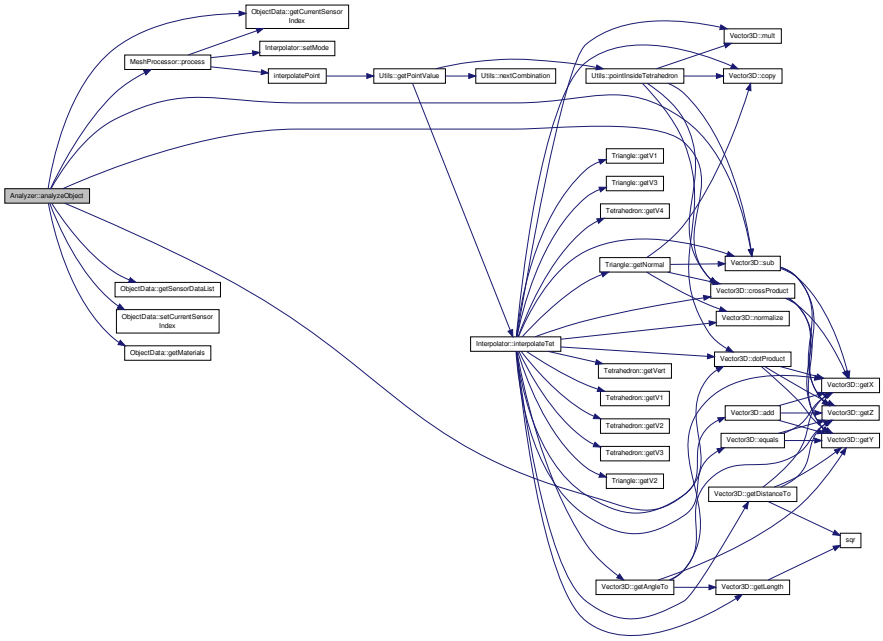
Parameter

obj	Das zu analysierende Objekt.
out	Referenz auf die AnalyzerData_object -Struktur in der die Analyseergebnisse gespeichert werden sollen.
use_markers	Die markierten Zeitpunkte eines Sensordatensatzes analysieren. Wenn false wird nur der aktuell ausgewählte Zeitpunkt analysiert.
sdindex	Nur den Sensordatensatz mit diesem Index analysieren.

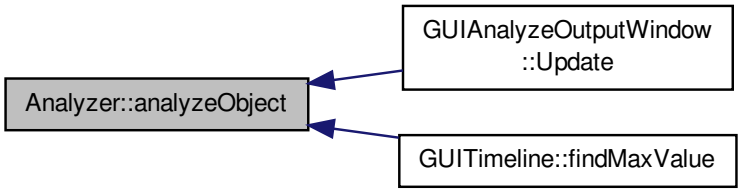
Wird von s

Definiert in Zeile 26 der Datei Analyzer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.1.3.2 void Analyzer::analyzePoint (**ObjectData** ∗ *obj*, **Vector3D** ∗ *point*, **AnalyzerData_point** ∗ *point_data*, **Interpolator** ∗ *interpolator*)

Ermittelt Daten für einen Punkt am aktuell ausgewählten Zeitpunkt.

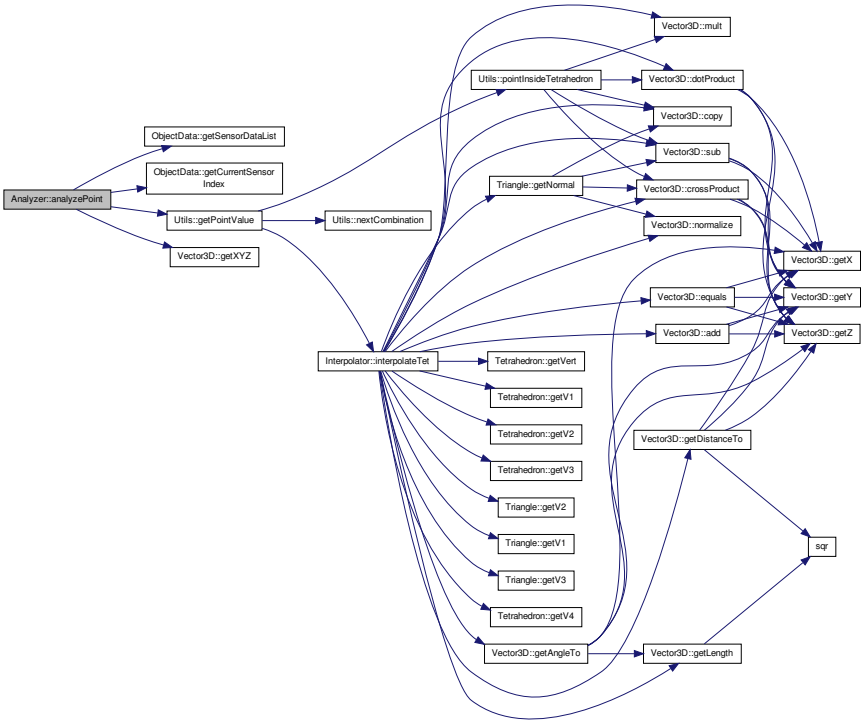
Parameter

obj	Das zu analysierende Objekt.
point	Der Ortsvektor zum zu analysierenden Punkt.
point_data	Referenz auf die AnalyzerData_point -Struktur in der die Analyseergebnisse gespeichert werden sollen.

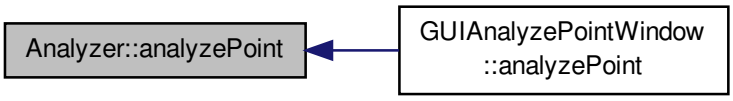
interpolator	Das zu verwendende Interpolatorobjekt.

Definiert in Zeile 147 der Datei Analyzer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.1.4 Freundbeziehungen und Funktionsdokumentation

8.1.4.1 std::ostream& operator<< (std::ostream & *out,* const AnalyzerData_object & *data*) [friend]

Operator zum Ausgeben der Analysedaten für ein Objekt in einem Stream.

Definiert in Zeile 164 der Datei Analyzer.cpp.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

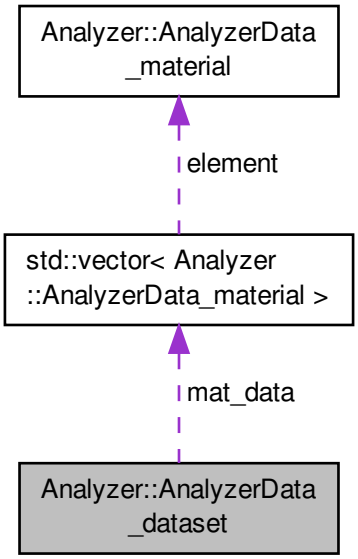
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.cpp

8.2 Analyzer::AnalyzerData_dataset Strukturreferenz

Analyseergebnisse für einen Sensordatensatz.

`#include <Analyzer.h>`

Zusammengehörigkeiten von Analyzer::AnalyzerData_dataset:



Öffentliche Attribute

- string name

  Der Name des Sensordatensatzes<.
- double heat_energy

  Die Wärmeenergie, die das Objekt für diesen Datensatz enthält.
- vector< AnalyzerData_material > mat_data

  Die Analyseergebnisse für die Einzelnen Materialien.

8.2.1 Ausführliche Beschreibung

Analyseergebnisse für einen Sensordatensatz.

Definiert in Zeile 35 der Datei Analyzer.h.

8.2.2 Dokumentation der Datenelemente

8.2.2.1 double Analyzer::AnalyzerData_dataset::heat_energy

Die Wärmeenergie, die das Objekt für diesen Datensatz enthält.

Definiert in Zeile 37 der Datei Analyzer.h.

8.2.2.2 vector<AnalyzerData_material> Analyzer::AnalyzerData_dataset::mat_data

Die Analyseergebnisse für die Einzelnen Materialien.

Definiert in Zeile 38 der Datei Analyzer.h.

8.2.2.3 string Analyzer::AnalyzerData_dataset::name

Der Name des Sensordatensatzes<.

Definiert in Zeile 36 der Datei Analyzer.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h

8.3 Analyzer::AnalyzerData_material Strukturreferenz

Analyseergebnisse für ein Material.

`#include <Analyzer.h>`

Öffentliche Attribute

- string name

  Der Name des Material.
- double volume

  Das Volumen, das dem Material zugeordnet ist.
- double heat_energy

  Die Wärmeenergie, die das dem Material zugeordnete Volumen enthält.

8.3.1 Ausführliche Beschreibung

Analyseergebnisse für ein Material.

Definiert in Zeile 26 der Datei Analyzer.h.

8.3.2 Dokumentation der Datenelemente

8.3.2.1 double Analyzer::AnalyzerData_material::heat_energy

Die Wärmeenergie, die das dem Material zugeordnete Volumen enthält.

<

Definiert in Zeile 29 der Datei Analyzer.h.

8.3.2.2 string Analyzer::AnalyzerData_material::name

Der Name des Material.

<

Definiert in Zeile 27 der Datei Analyzer.h.

8.3.2.3 double Analyzer::AnalyzerData_material::volume

Das Volumen, das dem Material zugeordnet ist.

<

Definiert in Zeile 28 der Datei Analyzer.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

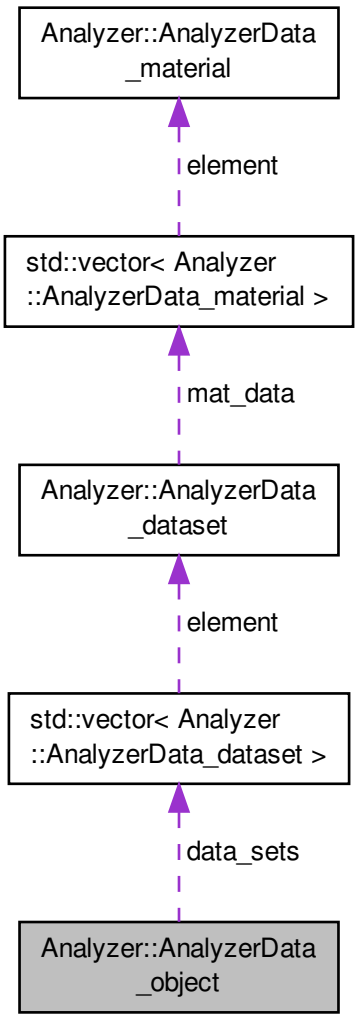
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h

8.4 Analyzer::AnalyzerData_object Strukturreferenz

Analyseergebnisse für ein Objekt.

`#include <Analyzer.h>`

Zusammengehörigkeiten von Analyzer::AnalyzerData_object:



Öffentliche Attribute

- double volume

  Das Volumen des Objekts.
- vector< AnalyzerData_dataset > data_sets

  Die Analyseergebisse für die Sensordatensätze.

8.4.1 Ausführliche Beschreibung

Analyseergebnisse für ein Objekt.

Definiert in Zeile 44 der Datei Analyzer.h.

8.4.2 Dokumentation der Datenelemente

8.4.2.1 vector<AnalyzerData_dataset> Analyzer::AnalyzerData_object::data_sets

Die Analyseergebisse für die Sensordatensätze.

<

Definiert in Zeile 46 der Datei Analyzer.h.

8.4.2.2 double Analyzer::AnalyzerData_object::volume

Das Volumen des Objekts.

<

Definiert in Zeile 45 der Datei Analyzer.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h

8.5 Analyzer::AnalyzerData_point Strukturreferenz

Analyseergebnisse für einen Punkt.

`#include <Analyzer.h>`

Öffentliche Attribute

- double value

  Die Temperatur an diesem Punkt.
- bool extrapolated

  Ist der Punkt extrapoliert?

8.5.1 Ausführliche Beschreibung

Analyseergebnisse für einen Punkt.

Definiert in Zeile 52 der Datei Analyzer.h.

8.5.2 Dokumentation der Datenelemente

8.5.2.1 bool Analyzer::AnalyzerData_point::extrapolated

Ist der Punkt extrapoliert?

Definiert in Zeile 54 der Datei Analyzer.h.

8.5.2.2 double Analyzer::AnalyzerData_point::value

Die Temperatur an diesem Punkt.

Definiert in Zeile 53 der Datei Analyzer.h.

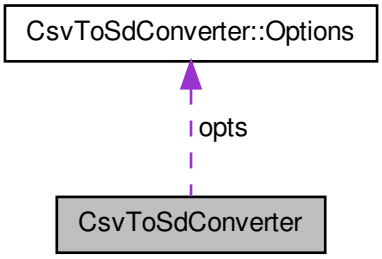
Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h

8.6 CsvToSdConverter Klassenreferenz

Konverter von .csv zu .tsd.

Zusammengehörigkeiten von CsvToSdConverter:



Klassen

- struct Options

  Strunktur für die Programmeinstellungen.

Öffentliche Methoden

- int convert (int argc, char ∗argv[])

  Wandelt die Daten der .csv-Datei ein eine .tsd-Datei um.

Geschützte Methoden

- bool contains (std::vector< string > &Vec, const string &Element)

  Testet, ob sich ein String in einer Liste von Strings befindet.

- bool contains (std::vector< int > &Vec, const int &Element)

  Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.
- string getTextBlock (string data, int n)

  Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.
- void parseLine (string line, vector< string > &out, vector< string > *timestamps, vector< string > *names, vector< int > *valid_cols)

  Sammelt Daten aus einer Textzeile (string).
- void replaceAll (string &str, const string from, const string to)

  Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.
- bool readConfiguration (string binary_path)

  Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.
- bool readSensorDefinitions (string path, vector< string > *sensor_names, vector< string > *sensor_data)

  Liest die Daten aus der Sensordefinitionsdatei.
- bool parseArguments (int argc, char *argv[], string &sdef_file, string &input_file, string &output_file)

  Wertet die Programmargumente aus.
- bool readInputFile (string path, vector< string > &sensor_names, vector< vector< string > > &values, vector< string > &timestamps, vector< string > &names)

  Liest die Daten aus der Eingabedatei.
- bool writeOutputFile (string path, vector< string > &sensor_names, vector< string > &sensor_data, vector< vector< string > > &values, vector< string > &timestamps, vector< string > &names)

  Schreibt die Ausgabedatei.

Geschützte Attribute

- string configpaths [NUMBEROFPATHS]

  Suchpfade für die Konfigurationsdatei.
- struct CsvToSdConverter::Options opts

  Hält die verwendeten Programmeinstellungen.

Statische, geschützte Attribute

- static const int NUMBEROFPATHS = 3

  Anzahl der Suchpfade für die Konfigurationsdatei.

8.6.1 Ausführliche Beschreibung

Konverter von .csv zu .tsd.

Definiert in Zeile 19 der Datei main.cpp.

8.6.2 Dokumentation der Elementfunktionen

8.6.2.1 bool CsvToSdConverter::contains (std::vector< string > & *Vec*, const string & *Element*) [inline], [protected]

Testet, ob sich ein String in einer Liste von Strings befindet.

Parameter

Vec	Liste der Strings.
Element	Der zu suchende String.

Rückgabe

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 55 der Datei main.cpp.

8.6.2.2 bool CsvToSdConverter::contains (std::vector< int > & *Vec*, const int & *Element*) [inline], [protected]

Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.

Parameter

Vec	Liste der Ganzzahlen.
Element	Die zu suchende Ganzzahl.

Rückgabe

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 71 der Datei main.cpp.

8.6.2.3 int CsvToSdConverter::convert (int *argc*, char * *argv[]*) [inline]

Wandelt die Daten der .csv-Datei ein eine .tsd-Datei um.

Wird duch die Funktion main() von außerhalb des Namespaces aufgerufen.

Parameter

argc	Anzahl der Programmargumente.
argv	Die Programmargumente.

Definiert in Zeile 621 der Datei main.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.6.2.4 string CsvToSdConverter::getTextBlock (string *data*, int *n*) [inline], [protected]

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

Parameter

data	Der Ausgansstring.
n	Index des zu findenden Blocks.

Rückgabe

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 87 der Datei main.cpp.

8.6.2.5 bool CsvToSdConverter::parseArguments (int *argc,* char * *argv[],* string & *sdef_file,* string & *input_file,* string & *output_file*) [inline], [protected]

Wertet die Programmargumente aus.

Parameter

argc	Anzahl der Programmargumente.
argv	Die Programmargumente.
sdef_file	Ausgabe für den Pfad zur Sensordefinitionsdatei.
input_file	Ausgabe für den Pfad zur Eingabedatei.
output_file	Ausgabe für den Pfad zur Ausgabedatei.

Rückgabe

Soll das Programm weiter ablaufen?

Definiert in Zeile 346 der Datei main.cpp.

8.6.2.6 void CsvToSdConverter::parseLine (string *line,* vector< string > & *out,* vector< string > * *timestamps,* vector< string > * *names,* vector< int > * *valid_cols*) [inline], [protected]

Sammelt Daten aus einer Textzeile (string).

Parameter

line	Die zu untersuchende Textzeile.
out	Ausgabevariable für die Sensordaten der Zeile. Alle Spalten nach opts.start_col werden als Sensordatenspalten betrachtet.
timestamps	Wenn nicht NULL, Ausgabevariable für den Zeitstempel der Zeile (opts.timecol). Der Zeitstempel wird an die übergebene Liste angehängt.
names	Wenn nicht NULL, Ausgabevariable für den Namen der Zeile (opts.namecol). Der Name wird an die übergebene Liste angehängt.
valid_cols	Wenn nicht NULL, werden nur die Sensordaten-Spalten mit den Indices dieser Liste ausgewertet.

Definiert in Zeile 127 der Datei main.cpp.

8.6.2.7 bool CsvToSdConverter::readConfiguration (string *binary_path*) [inline], [protected]

Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.

Parameter

binary_path	Pfad zur Binärdatei.

Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 209 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

8.6.2.8 bool CsvToSdConverter::readInputFile (string *path,* vector< string > & *sensor_names,* vector< vector< string > > & *values,* vector< string > & *timestamps,* vector< string > & *names*) [inline], [protected]

Liest die Daten aus der Eingabedatei.

Parameter

path	Der Pfad zur Eingabedatei.
sensor_names	Liste der Namen der verwendeten Sensoren.
values	Liste für die extrahierten Sensorwerte.
timestamps	Liste für die Zeitstempel der Messwerte.
names	Liste für die Namen der Datensätze.

Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 454 der Datei main.cpp.

8.6.2.9 bool CsvToSdConverter::readSensorDefinitions (string *path,* vector< string > * *sensor_names,* vector< string > * *sensor_data*) [inline], [protected]

Liest die Daten aus der Sensordefinitionsdatei.

Parameter

path	Pfad zur Binärdatei.
sensor_names	Liste für die Namen der Sensoren.
sensor_data	Liste für die Daten der Sensorden (Koordinaten).

Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 265 der Datei main.cpp.

8.6.2.10 void CsvToSdConverter::replaceAll (string & *str,* const string *from,* const string *to*) [inline], [protected]

Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.

Parameter

str	Der zu durchsuchende String.
from	Der zu ersetzende Teilstring.
to	Der Teilstring, durch den ersetzt werden soll.

Definiert in Zeile 187 der Datei main.cpp.

8.6.2.11 bool CsvToSdConverter::writeOutputFile (string *path,* vector< string > & *sensor_names,* vector< string > & *sensor_data,* vector< vector< string > > & *values,* vector< string > & *timestamps,* vector< string > & *names*) [inline], [protected]

Schreibt die Ausgabedatei.

Parameter

path	Der Pfad zur Ausgabedatei.
sensor_names	Liste der Namen der verwendeten Sensoren.
sensor_data	Liste der Koordinaten der verwendeten Sensoren.
values	Liste für die extrahierten Sensorwerte.
timestamps	Liste für die Zeitstempel der Messwerte.
names	Liste für die Namen der Datensätze.

Rückgabe

War das Schreiben erfolgreich?

Definiert in Zeile 572 der Datei main.cpp.

8.6.3 Dokumentation der Datenelemente

8.6.3.1 string CsvToSdConverter::configpaths[NUMBEROFPATHS] [protected]

Initialisierung:

```
{
            "/etc/simpleanalyzer/csvtosd.conf",
            "/usr/local/share/simpleanalyzer/csvtosd.conf",
            "/usr/share/simpleanalyzer/csvtosd.conf" }
```

Suchpfade für die Konfigurationsdatei.

Das Verzeichnis der ausführbaren Datei wird immer geprüft.

Definiert in Zeile 30 der Datei main.cpp.

8.6.3.2 const int CsvToSdConverter::NUMBEROFPATHS = 3 [static], [protected]

Anzahl der Suchpfade für die Konfigurationsdatei.

Definiert in Zeile 24 der Datei main.cpp.

8.6.3.3 struct CsvToSdConverter::Options CsvToSdConverter::opts [protected]

Hält die verwendeten Programmeinstellungen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

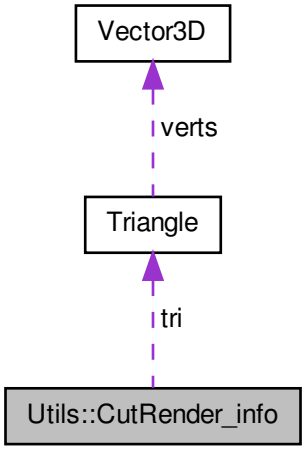
- /daten/Projekte/eclipse_workspace/csvtosd/main.cpp

8.7 Utils::CutRender_info Strukturreferenz

Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.

`#include <utils.h>`

Zusammengehörigkeiten von Utils::CutRender_info:



Öffentliche Attribute

- Triangle * tri
  Das die Ebene beschreibende Dreieck.
- float mmperpixel
  Maßstab der Darstellung der Temperaturverteilung in $\frac{mm}{Pixel}$.
- int img_width
  Breite der Darstellung der Temperaturverteilung.
- int img_height
  Höhe der Darstellung der Temperaturverteilung.
- PIM_algorithm in_volume_algorithm
  Der zu verwendende Punkt-in-Volumen-Testalgorithmus.

8.7.1 Ausführliche Beschreibung

Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.

Definiert in Zeile 78 der Datei utils.h.

8.7.2 Dokumentation der Datenelemente

8.7.2.1 int Utils::CutRender_info::img_height

Höhe der Darstellung der Temperaturverteilung.

Definiert in Zeile 82 der Datei utils.h.

8.7.2.2 int Utils::CutRender_info::img_width

Breite der Darstellung der Temperaturverteilung.

Definiert in Zeile 81 der Datei utils.h.

8.7.2.3 PIM_algorithm Utils::CutRender_info::in_volume_algorithm

Der zu verwendende Punkt-in-Volumen-Testalgorithmus.

Immer ALGORITHM_TETRAHEDRONS.

Definiert in Zeile 83 der Datei utils.h.

8.7.2.4 float Utils::CutRender_info::mmperpixel

Maßstab der Darstellung der Temperaturverteilung in $\frac{mm}{Pixel}$.

Definiert in Zeile 80 der Datei utils.h.

8.7.2.5 Triangle* Utils::CutRender_info::tri

Das die Ebene beschreibende Dreieck.

Der erste Punkt ist dabei das Zentrum der später ermittelten Temperaturverteilung.

Definiert in Zeile 79 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h

8.8 Exporter Klassenreferenz

Export der gewonnenen Daten.

`#include <Exporter.h>`

Öffentliche Methoden

- Exporter ()

  Der Konstruktor.
- ObjectData::ObjectDataStatus ExportLegacyVTK (string filename, ObjectData *data)

  Exportiert die aktuell berechnete dreidimensionale Temperaturverteilung und das Modell als VTK-Datei.
- ObjectData::ObjectDataStatus ExportCutCSV (string filename, float *values, CutRender_info *info)

  Exportiert die zweidimensionale Temperaturverteilung (Schnitt durch das Modell) als csv-Datei.
- virtual ~Exporter ()

  Der Destruktor.

Geschützte Attribute

- const char * CSV_SEPARATOR

  Das in der .csv-Datei verwendete Separatorzeichen.

8.8.1 Ausführliche Beschreibung

Export der gewonnenen Daten.

Klasse zum Export der dreidimensionalen Temperaturverteilung als VTK-Datei und der zweidimensionalen Temperaturverteilung (Schnitt durch das Modell) als .csv-Datei.

Definiert in Zeile 22 der Datei Exporter.h.

8.8.2 Beschreibung der Konstruktoren und Destruktoren

8.8.2.1 Exporter::Exporter ( )

Der Konstruktor.

Definiert in Zeile 15 der Datei Exporter.cpp.

8.8.2.2 Exporter::~Exporter ( ) [virtual]

Der Destruktor.

Definiert in Zeile 168 der Datei Exporter.cpp.

8.8.3 Dokumentation der Elementfunktionen

8.8.3.1 ObjectData::ObjectDataStatus Exporter::ExportCutCSV (string *filename*, float * *values*, CutRender_info * *info*)

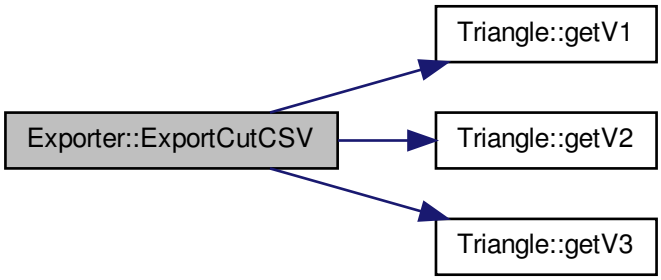
Exportiert die zweidimensionale Temperaturverteilung (Schnitt durch das Modell) als csv-Datei.

Rückgabe

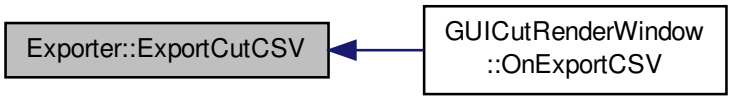
Der Fehlercode.

Definiert in Zeile 124 der Datei Exporter.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.8.3.2 ObjectData::ObjectDataStatus Exporter::ExportLegacyVTK (string *filename,* ObjectData * *data*)

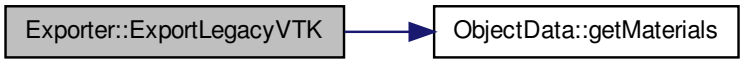
Exportiert die aktuell berechnete dreidimensionale Temperaturverteilung und das Modell als VTK-Datei.

Rückgabe

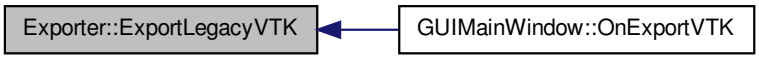
Der Fehlercode.

Definiert in Zeile 23 der Datei Exporter.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.8.4 Dokumentation der Datenelemente

8.8.4.1 const char* Exporter::CSV_SEPARATOR [protected]

Das in der .csv-Datei verwendete Separatorzeichen.

Definiert in Zeile 50 der Datei Exporter.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

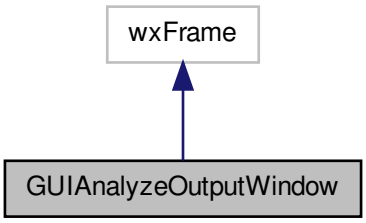
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.cpp

8.9 GUIAnalyzeOutputWindow Klassenreferenz

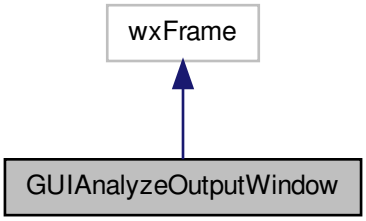
Übersichtsfenster über die Analysedaten.

`#include <GUIAnalyzeOutputWindow.h>`

Klassendiagramm für GUIAnalyzeOutputWindow:



Zusammengehörigkeiten von GUIAnalyzeOutputWindow:



Öffentliche Methoden

- GUIAnalyzeOutputWindow (wxWindow ∗parent, const wxChar ∗title, int xpos, int ypos, int width, int height)

  Der Konstruktor.
- void Update ()

  Methode zum aktualisieren des Fensters, alle Objekte werden erneut analysiert und die aktualisierten Ergebnisse angezeigt.
- virtual ∼GUIAnalyzeOutputWindow ()

  Der Destruktor.

Private Methoden

- void OnKeyPress (wxKeyEvent &event)

  Event-Tabellendeklaration für wxWidgets.
- void SelectAll ()

Selektiert alle Zellen der Tabelle.

- void ToClipboard ()

  Kopiert die Inhalte der Tabelle in die Zwischenablage.

Private Attribute

- wxGrid * table

  Die Tabellenkomponente.

8.9.1 Ausführliche Beschreibung

Übersichtsfenster über die Analysedaten.

Dieses Fenster zeigt eine Tabelle mit den zur Analyse markierten Zeitpunkten für alle Objekte und derenDatensätze und Materialen. Nicht-zeitabhängige Sensordaten werden immer angezeigt.

Definiert in Zeile 22 der Datei GUIAnalyzeOutputWindow.h.

8.9.2 Beschreibung der Konstruktoren und Destruktoren

8.9.2.1 GUIAnalyzeOutputWindow::GUIAnalyzeOutputWindow (wxWindow * *parent*, const wxChar * *title*, int *xpos*, int *ypos*, int *width*, int *height*)

Der Konstruktor.

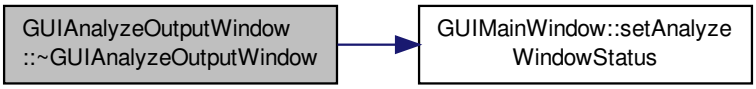
Definiert in Zeile 23 der Datei GUIAnalyzeOutputWindow.cpp.

8.9.2.2 GUIAnalyzeOutputWindow::~GUIAnalyzeOutputWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 210 der Datei GUIAnalyzeOutputWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.9.3 Dokumentation der Elementfunktionen

8.9.3.1 void GUIAnalyzeOutputWindow::OnKeyPress (wxKeyEvent & *event*) [private]

Event-Tabellendeklaration für wxWidgets.

Behandelt das Drücken von Strg+C und Strg+A.

Definiert in Zeile 197 der Datei GUIAnalyzeOutputWindow.cpp.

8.9.3.2 void GUIAnalyzeOutputWindow::SelectAll ( ) [private]

Selektiert alle Zellen der Tabelle.

Definiert in Zeile 189 der Datei GUIAnalyzeOutputWindow.cpp.

8.9.3.3 void GUIAnalyzeOutputWindow::ToClipboard ( ) [private]

Kopiert die Inhalte der Tabelle in die Zwischenablage.

Basierend auf http://forums.wxwidgets.org/viewtopic.php?f=20&t=2200#p148731.

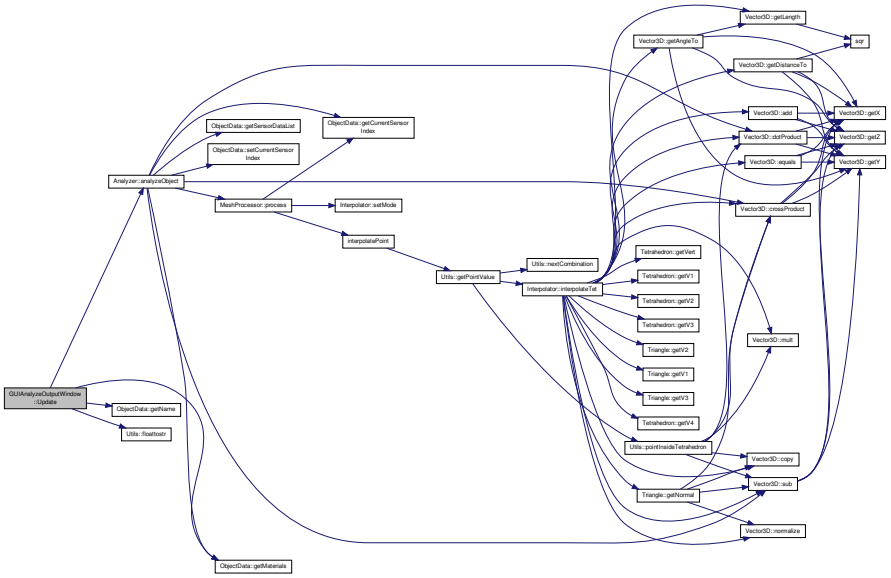
Definiert in Zeile 151 der Datei GUIAnalyzeOutputWindow.cpp.

8.9.3.4 void GUIAnalyzeOutputWindow::Update ( )

Methode zum aktualisieren des Fensters, alle Objekte werden erneut analysiert und die aktualisierten Ergebnisse angezeigt.

Definiert in Zeile 37 der Datei GUIAnalyzeOutputWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.9.4 Dokumentation der Datenelemente

8.9.4.1 wxGrid* GUIAnalyzeOutputWindow::table [private]

Die Tabellenkomponente.

Definiert in Zeile 65 der Datei GUIAnalyzeOutputWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

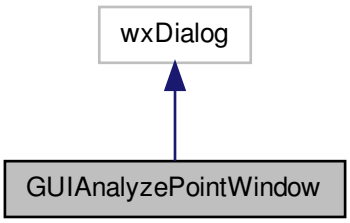
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.cpp

8.10 GUIAnalyzePointWindow Klassenreferenz

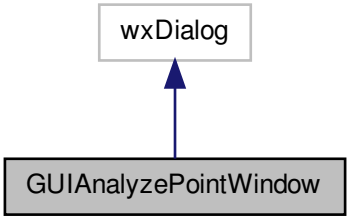
Analysefenster für einen Punkt.

`#include <GUIAnalyzePointWindow.h>`

Klassendiagramm für GUIAnalyzePointWindow:



Zusammengehörigkeiten von GUIAnalyzePointWindow:



Öffentliche Methoden

- GUIAnalyzePointWindow (wxWindow ∗parent, const wxChar ∗title, int xpos, int ypos, int width, int height)

  Der Konstruktor.
- virtual ∼GUIAnalyzePointWindow ()

  Der Destruktor.

Private Methoden

- void analyzePoint (wxCommandEvent &event)

  Event-Tabellendeklaration für wxWidgets.

Private Attribute

- wxStaticText ∗ label

Beschriftung der Fensterkomponenten.

- wxTextCtrl * xedit

  Eingabefeld für die X-Koordinate.

- wxTextCtrl * yedit

  Eingabefeld für die Y-Koordinate.

- wxTextCtrl * zedit

  Eingabefeld für die Z-Koordinate.

- wxStaticText * interpolationModeLabel

  Beschriftung für den Interpolationsmodus.

- wxComboBox * interpolationModeList

  Dropdown-Menü für den Interpolationsmodus.

- wxButton * calcbt

  Button zum Auslösen der Analyseprozedur.

8.10.1 Ausführliche Beschreibung

Analysefenster für einen Punkt.

Definiert in Zeile 16 der Datei GUIAnalyzePointWindow.h.

8.10.2 Beschreibung der Konstruktoren und Destruktoren

8.10.2.1 GUIAnalyzePointWindow::GUIAnalyzePointWindow (wxWindow * *parent*, const wxChar * *title*, int *xpos*, int *ypos*, int *width*, int *height*)

Der Konstruktor.

Definiert in Zeile 22 der Datei GUIAnalyzePointWindow.cpp.

8.10.2.2 GUIAnalyzePointWindow::~GUIAnalyzePointWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 93 der Datei GUIAnalyzePointWindow.cpp.

8.10.3 Dokumentation der Elementfunktionen

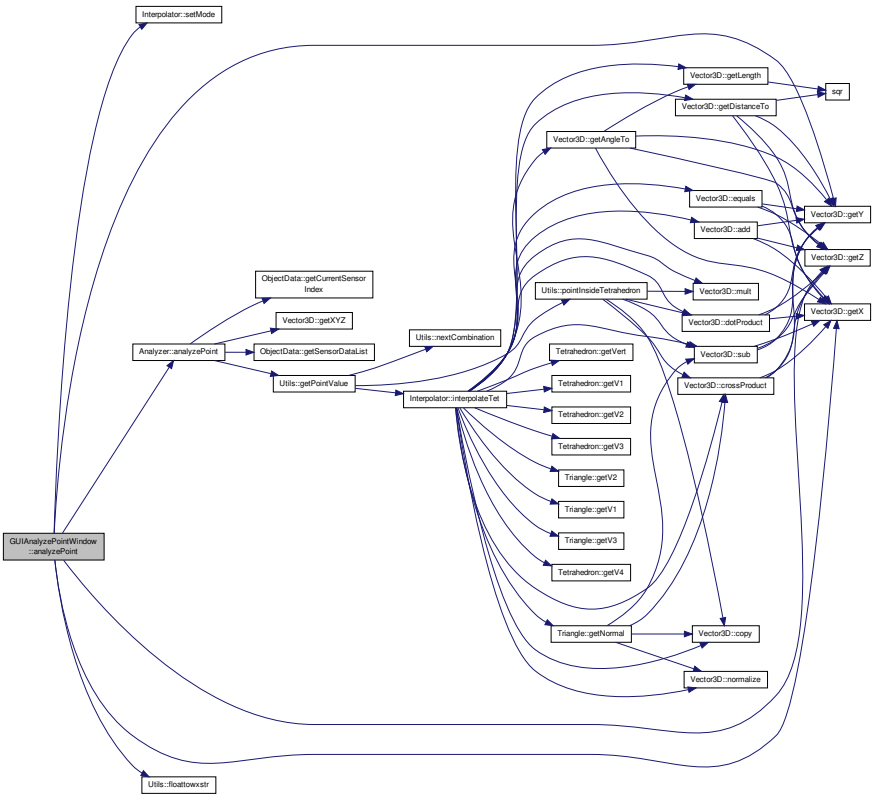
8.10.3.1 void GUIAnalyzePointWindow::analyzePoint (wxCommandEvent & *event*) [private]

Event-Tabellendeklaration für wxWidgets.

Ermittelt Temperatur und Art des Punktes (Interpoliert/Extrapoliert). Wird durch Event ausgelöst.

Definiert in Zeile 56 der Datei GUIAnalyzePointWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.10.4 Dokumentation der Datenelemente

8.10.4.1 wxButton* GUIAnalyzePointWindow::calcbt [private]

Button zum Auslösen der Analyseprozedur.

Definiert in Zeile 72 der Datei GUIAnalyzePointWindow.h.

8.10.4.2 wxStaticText* GUIAnalyzePointWindow::interpolationModeLabel [private]

Beschriftung für den Interpolationsmodus.

Definiert in Zeile 62 der Datei GUIAnalyzePointWindow.h.

8.10.4.3 wxComboBox* GUIAnalyzePointWindow::interpolationModeList [private]

Dropdown-Menü für den Interpolationsmodus.

Definiert in Zeile 67 der Datei GUIAnalyzePointWindow.h.

8.10.4.4 wxStaticText* GUIAnalyzePointWindow::label [private]

Beschriftung der Fensterkomponenten.

Definiert in Zeile 42 der Datei GUIAnalyzePointWindow.h.

8.10.4.5 wxTextCtrl* GUIAnalyzePointWindow::xedit [private]

Eingabefeld für die X-Koordinate.

Definiert in Zeile 47 der Datei GUIAnalyzePointWindow.h.

8.10.4.6 wxTextCtrl* GUIAnalyzePointWindow::yedit [private]

Eingabefeld für die Y-Koordinate.

Definiert in Zeile 52 der Datei GUIAnalyzePointWindow.h.

8.10.4.7 wxTextCtrl* GUIAnalyzePointWindow::zedit [private]

Eingabefeld für die Z-Koordinate.

Definiert in Zeile 57 der Datei GUIAnalyzePointWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePointWindow.cpp

8.11 GUIColorScalePanel Klassenreferenz

Farbige Temperaturskala für zweidimensionale Temperaturverteilung.

`#include <GUIColorScalePanel.h>`

Öffentliche Typen

- enum ScaleMode { SCM_NONE = 0, SCM_HORIZONTAL, SCM_VERTICAL }

  Modus der Skalendarstellung.

Öffentliche Methoden

- GUIColorScalePanel ()

  Der Konstruktor.
- void refresh (int img_width, int img_height)

  Zeichnet die Temperaturskala neu.
- void paintTo (wxDC &dc, float zoom, wxPoint &img_coords)

  Zeichnet die Temperaturskala mit einem bestimmten device context.
- void handleMouse (wxMouseEvent &event, wxPoint &img_coords, wxPoint &img_dim, float zoom)

  Behandelt die Mausaktionen und verändert ggf.
- void getDisplayArea (wxRect *rect, float zoom)

  Gibt die bei einem bestimmten Zoomfaktor eingenommene Fläche zurück.
- void fitBounds (wxPoint &img_dim, bool to_scale)

  Passt die Größe und Position der Skala an die Größe der Grafik an.
- bool mouseOnDisplayArea (wxPoint &img_coords, float zoom, wxPoint &mouse_pos)

  Gibt zurück, ob sich die Maus über der Fläche der Skala befindet.
- int getX ()
- int getY ()
- int getFontSize () const

- void setFontSize (int fontSize)

  Setzt die Schriftgröße der Skala.
- ScaleMode getMode () const
- void setMode (ScaleMode mode)

  Setzt den Modus der Skala.
- const wxColour & getTextColor () const
- void setTextColor (const wxColour &textColor)

  Setzt die Schriftfarbe der Skala.
- int getStepWidth () const

  Gibt die Schrittweite der Skalenbeschriftung.
- void setStepWidth (int stepWidth)

  Setzt die Schrittweite der Skalenbeschriftung.
- wxImage * getImage () const
- virtual ~GUIColorScalePanel ()

  Der Destruktor.

Private Attribute

- int step_width

  Schrittweite der Beschriftung.
- int font_size

  Die Schriftgröße.
- ScaleMode mode

  Der Darstellungsmodus.
- wxColour text_color

  Die Schriftfarbe.
- wxImage * image

  Bild, das die Skala ohne Steuerelemente enthält.
- int current_mx

  Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.
- int current_my

  Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.
- float x

  Position (X) der Skala.
- float y

  Position Y) der Skala.
- float width

  Breite der Skala.
- float height

  Höhe der Skala.
- bool scaling

  Wird gerade in der Größe verändert.
- bool transforming

  Wird gerade transformiert (Größe oder Position).
- bool prev_mouse_down

  zwischenspeicher für den Mausstatus.

8.11.1 Ausführliche Beschreibung

Farbige Temperaturskala für zweidimensionale Temperaturverteilung.

Farbige Temperaturskala für zweidimensionale Temperaturverteilung. Wird für die Darststellung einer farbigen Temperaturskala im Anzeigefenster auf der als zweidimensionale Temperaturverteilung erzeugten Grafik verwendet.

Definiert in Zeile 21 der Datei GUIColorScalePanel.h.

8.11.2 Dokumentation der Aufzählungstypen

8.11.2.1 enum GUIColorScalePanel::ScaleMode

Modus der Skalendarstellung.

Aufzählungswerte

SCM_NONE Keine Skala.

SCM_HORIZONTAL Eine horizontal ausgerichtete Skala.

SCM_VERTICAL Eine vertikal ausgerichtete Skala.

Definiert in Zeile 26 der Datei GUIColorScalePanel.h.

8.11.3 Beschreibung der Konstruktoren und Destruktoren

8.11.3.1 GUIColorScalePanel::GUIColorScalePanel ( )

Der Konstruktor.

Definiert in Zeile 21 der Datei GUIColorScalePanel.cpp.

8.11.3.2 GUIColorScalePanel::~GUIColorScalePanel ( ) [virtual]

Der Destruktor.

Definiert in Zeile 457 der Datei GUIColorScalePanel.cpp.

8.11.4 Dokumentation der Elementfunktionen

8.11.4.1 void GUIColorScalePanel::fitBounds (wxPoint & *img_dim*, bool *to_scale*)

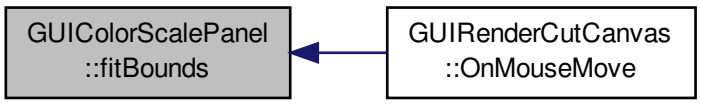
Passt die Größe und Position der Skala an die Größe der Grafik an.

Parameter

img_dim	Größe der Grafik.
to_scale	Größe statt der Position verändern.

Definiert in Zeile 296 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.11.4.2 void GUIColorScalePanel::getDisplayArea (wxRect * *rect,* float *zoom*)

Gibt die bei einem bestimmten Zoomfaktor eingenommene Fläche zurück.

Definiert in Zeile 261 der Datei GUIColorScalePanel.cpp.

8.11.4.3 int GUIColorScalePanel::getFontSize ( ) const

Rückgabe

Schriftgröße der Skala.

Definiert in Zeile 421 der Datei GUIColorScalePanel.cpp.

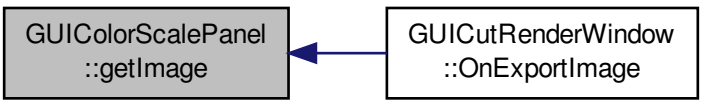
8.11.4.4 wxImage * GUIColorScalePanel::getImage ( ) const

Rückgabe

Skala als Grafik.

Definiert in Zeile 453 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.11.4.5 GUIColorScalePanel::ScaleMode GUIColorScalePanel::getMode ( ) const

Rückgabe

Modus der Skala.

Definiert in Zeile 429 der Datei GUIColorScalePanel.cpp.

8.11.4.6 int GUIColorScalePanel::getStepWidth ( ) const

Gibt die Schrittweite der Skalenbeschriftung.

Definiert in Zeile 445 der Datei GUIColorScalePanel.cpp.

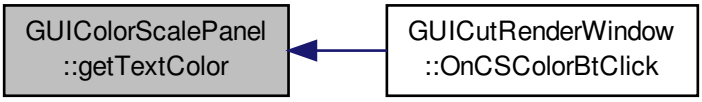
8.11.4.7 const wxColour & GUIColorScalePanel::getTextColor ( ) const

Rückgabe

Schriftfarbe der Skala.

Definiert in Zeile 437 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.11.4.8 int GUIColorScalePanel::getX ( )

Rückgabe

horizontale Position auf der Zeichenfläche.

Definiert in Zeile 288 der Datei GUIColorScalePanel.cpp.

8.11.4.9 int GUIColorScalePanel::getY ( )

Rückgabe

vertikale Position auf der Zeichenfläche.

Definiert in Zeile 292 der Datei GUIColorScalePanel.cpp.

8.11.4.10 void GUIColorScalePanel::handleMouse (wxMouseEvent & *event,* wxPoint & *img_coords,* wxPoint & *img_dim,* float *zoom*)

Behandelt die Mausaktionen und verändert ggf.

Größe oder Position des Skala.

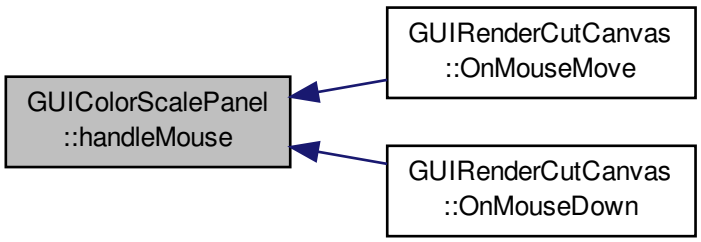
Parameter

event	Das zu behandelnde Maus-event.

img_coords	Position der Grafik auf der Zeichenfläche.
img_dim	Größe der Grafik.
zoom	aktueller Vergrößerungsfaktor des Betrachtungsfensters.

Definiert in Zeile 360 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.11.4.11 bool GUIColorScalePanel::mouseOnDisplayArea (wxPoint & *img_coords*, float *zoom*, wxPoint & *mouse_pos*)

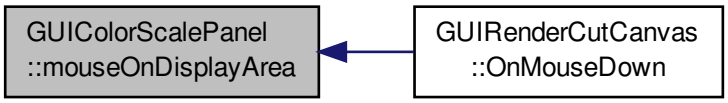
Gibt zurück, ob sich die Maus über der Fläche der Skala befindet.

Parameter

img_coords	Position der Grafik auf der Zeichenfläche.
zoom	aktueller Vergrößerungsfaktor des Betrachtungsfensters.
mouse_pos	Position der Maus auf der Zeichenfläche.

Definiert in Zeile 269 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.11.4.12 void GUIColorScalePanel::paintTo (wxDC & *dc*, float *zoom*, wxPoint & *img_coords*)

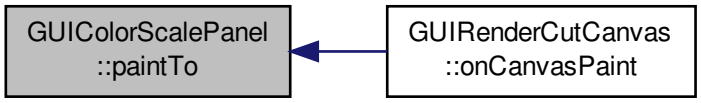
Zeichnet die Temperaturskala mit einem bestimmten device context.

Parameter

dc	Der zum Zeichnen zu verwendende device context.
zoom	Faktor zum Skalieren der Skala.
img_coords	Position der Grafik auf der Zeichenfläche.

Definiert in Zeile 42 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.11.4.13 void GUIColorScalePanel::refresh (int *img_width,* int *img_height*)

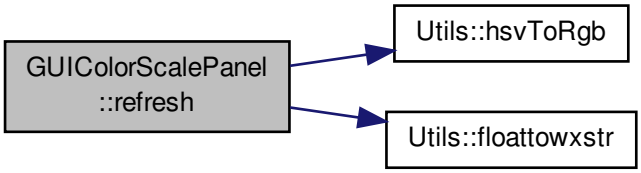
Zeichnet die Temperaturskala neu.

Parameter

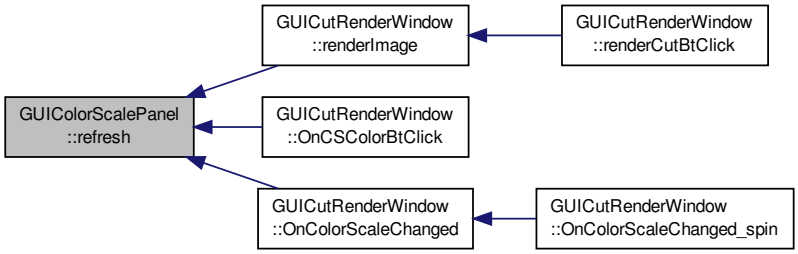
img_width	Breite des Bildes, für das die Skala gezeichnet wird.
img_height	Höhe des Bildes, für das die Skala gezeichnet wird.

Definiert in Zeile 85 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

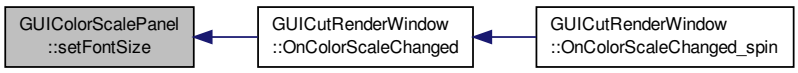


8.11.4.14 void GUIColorScalePanel::setFontSize (int *fontSize*)

Setzt die Schriftgröße der Skala.

Definiert in Zeile 425 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

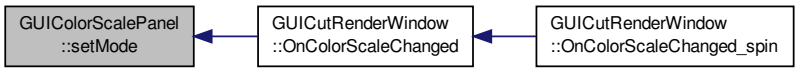


8.11.4.15 void GUIColorScalePanel::setMode (ScaleMode *mode*)

Setzt den Modus der Skala.

Definiert in Zeile 433 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

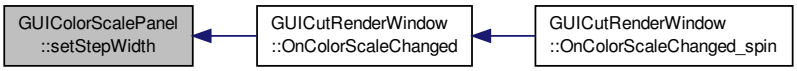


8.11.4.16 void GUIColorScalePanel::setStepWidth (int *stepWidth*)

Setzt die Schrittweite der Skalenbeschriftung.

Definiert in Zeile 449 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

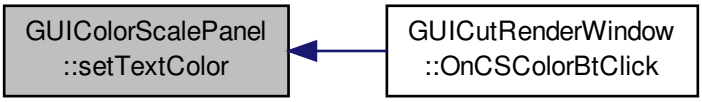


8.11.4.17 void GUIColorScalePanel::setTextColor (const wxColour & *textColor*)

Setzt die Schriftfarbe der Skala.

Definiert in Zeile 441 der Datei GUIColorScalePanel.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.11.5 Dokumentation der Datenelemente

8.11.5.1 int GUIColorScalePanel::current_mx [private]

Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.

Definiert in Zeile 170 der Datei GUIColorScalePanel.h.

8.11.5.2 int GUIColorScalePanel::current_my [private]

Zwischenspeicher für die Mausposition, zum behandeln von Mausinteraktionen.

Definiert in Zeile 175 der Datei GUIColorScalePanel.h.

8.11.5.3 int GUIColorScalePanel::font_size [private]

Die Schriftgröße.

Definiert in Zeile 150 der Datei GUIColorScalePanel.h.

8.11.5.4 float GUIColorScalePanel::height [private]

Höhe der Skala.

Definiert in Zeile 195 der Datei GUIColorScalePanel.h.

8.11.5.5 wxImage* GUIColorScalePanel::image [private]

Bild, das die Skala ohne Steuerelemente enthält.

Definiert in Zeile 165 der Datei GUIColorScalePanel.h.

8.11.5.6 ScaleMode GUIColorScalePanel::mode [private]

Der Darstellungsmodus.

Definiert in Zeile 155 der Datei GUIColorScalePanel.h.

8.11.5.7 bool GUIColorScalePanel::prev_mouse_down [private]

zwischenspeicher für den Mausstatus.

Definiert in Zeile 210 der Datei GUIColorScalePanel.h.

8.11.5.8 bool GUIColorScalePanel::scaling [private]

Wird gerade in der Größe verändert.

Definiert in Zeile 200 der Datei GUIColorScalePanel.h.

8.11.5.9 int GUIColorScalePanel::step_width [private]

Schrittweite der Beschriftung.

Definiert in Zeile 145 der Datei GUIColorScalePanel.h.

8.11.5.10 wxColour GUIColorScalePanel::text_color [private]

Die Schriftfarbe.

Definiert in Zeile 160 der Datei GUIColorScalePanel.h.

8.11.5.11 bool GUIColorScalePanel::transforming [private]

Wird gerade transformiert (Größe oder Position).

Definiert in Zeile 205 der Datei GUIColorScalePanel.h.

8.11.5.12 float GUIColorScalePanel::width [private]

Breite der Skala.

Definiert in Zeile 190 der Datei GUIColorScalePanel.h.

8.11.5.13 float GUIColorScalePanel::x [private]

Position (X) der Skala.

Definiert in Zeile 180 der Datei GUIColorScalePanel.h.

8.11.5.14 float GUIColorScalePanel::y [private]

Position Y) der Skala.

Definiert in Zeile 185 der Datei GUIColorScalePanel.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

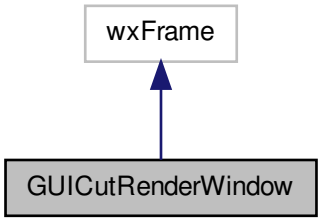
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.cpp

8.12 GUICutRenderWindow Klassenreferenz

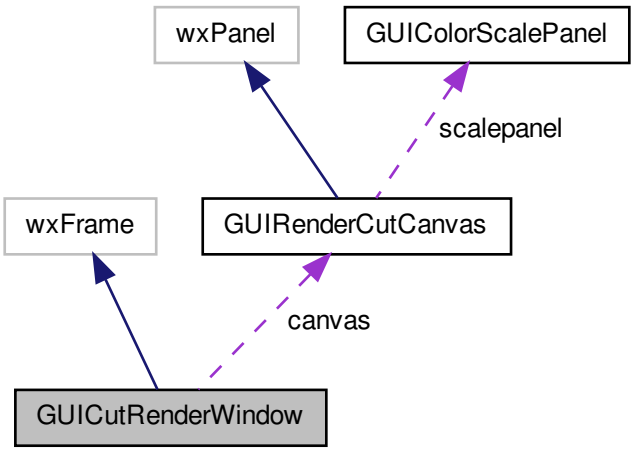
Fenster zum erstellen zweidimensionaler Temperaturverteilungen.

`#include <GUICutRenderWindow.h>`

Klassendiagramm für GUICutRenderWindow:



Zusammengehörigkeiten von GUICutRenderWindow:



Öffentliche Methoden

- GUICutRenderWindow (wxWindow ∗parent, const wxChar ∗title, int xpos, int ypos, int width, int height)

  Der Konstuktor.
- virtual ∼GUICutRenderWindow ()

  Der Destruktor.

Geschützte Methoden

- DECLARE_EVENT_TABLE ()

  Event-Tabellendeklaration für wxWidgets.

Private Methoden

- CutRender_info ∗ getCutRenderProperties ()

  Gibt die aktuell eingestellten Eigenschaften für die zweidimensionale Temperaturverteilung zurück, damit Sie später an den Renderer des 3D-Fensters zur Visualisierung übergeben werden können.
- void renderCutBtClick (wxCommandEvent &event)

  Behandelt das Drücken des Buttons zur Berechnung der zweidimensionalen Temperaturverteilung.
- void OnResize (wxSizeEvent &event)

  Behandelt Änderungen der Größe des Fensters.
- void OnCutPropsChanged (wxCommandEvent &event)

  Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.
- void refreshVisualisation ()

  Aktualisiert die Visualisierung der Schnittebene im Hauptfenster.
- void OnExportImage (wxCommandEvent &event)

  Fragt den Benutzer nach dem Pfad und Exportiert eine Grafik aus 2D-Temperaturverteilung und Temperaturskala.
- void OnExportCSV (wxCommandEvent &event)

  Fragt den Benutzer nach dem Pfad und Exportiert die 2D-Temperaturverteilung als .csv-Datei.
- void OnSCutPropsChanged_spin (wxSpinEvent &event)

  Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.
- void OnColorScaleChanged (wxCommandEvent &event)

  Behandelt das Ändern von Parametern zur darstellung der Temperaturskala.
- void OnColorScaleChanged_spin (wxSpinEvent &event)

  Behandelt das Ändern von Parametern zur darstellung der Temperaturskala.
- void OnCSColorBtClick (wxCommandEvent &event)

  Behandelt das Klicken auf den Button zur Wahl der Schriftfarbe auf der Skala.
- void renderImage (wxImage ∗image)

  Berechnet die 2D-Temperaturverteilung als Grafik.

Private Attribute

- wxScrolledWindow ∗ scroll_pane

  Scrollender Bereich, in den die anderen Komponenten außer der Zeichenfläche (canvas) eingebettet sind.
- wxTextCtrl ∗ p1xedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxTextCtrl ∗ p1yedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxTextCtrl ∗ p1zedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

- wxTextCtrl * p2xedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxTextCtrl * p2yedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxTextCtrl * p2zedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxTextCtrl * p3xedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxTextCtrl * p3yedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxTextCtrl * p3zedit

  Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.
- wxSpinCtrl * imgWidthEdit

  Textfeld zur Eingabe der Breite des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.
- wxSpinCtrl * imgHeightEdit

  Feld zur Eingabe der Höhe des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.
- wxSpinCtrl * threadcountedit

  Feld zur Eingabe der zum Berechnen zu verwendenden Prozessorkerne.
- wxTextCtrl * mmperpixeledit

  Feld zur Eingabe des Maßstabs in $\frac{mm}{px}$.
- wxStaticText * p1label

  Beschriftung für den 1.
- wxStaticText * p2label

  Beschriftung für den 2.
- wxStaticText * p3label

  Beschriftung für den 3.
- wxStaticText * mmperpixellabel

  Beschriftung für den Maßstab in $\frac{mm}{px}$.
- wxStaticText * trilabel

  Beschriftung für das die Schnittebene definierende Dreieck.
- wxStaticText * optionslbl

  Beschriftung für die die 2D-Temperaturverteilung betreffenden Parameter.
- wxStaticText * widthHeightlbl

  Beschriftung für Breite und Höhe der Grafik.
- wxStaticText * threadcountlbl

  Beschriftung für die Anzahl bei der Berechnung zu verwendender Prozessorkerne.
- wxStaticText * scalelbl

  Beschriftung für die die Skala betreffenden Optionen.
- wxStaticText * scalemodelbl

  Beschriftung für den Darstellungsmodus der Skala.
- wxComboBox * scalemodecb

  Menübox zur Auswahl des Darstellungsmodus der Skala.
- wxStaticText * scalefontpropslbl

  Beschriftung für die Schrifteigenschaften der Skala.
- wxSpinCtrl * scalefontsizeedit

  Feld zur Eingabe der Schriftgröße der Skala.
- wxButton * scalefontcolorbt

  Button zur Auswahl der Schriftfarbe.
- wxSpinCtrl * scalestepedit

  Feld zur Eingabe der Schrittweite der Skala.
- wxButton * calcbt

Button zum Starten der Berechnung der 2D-Temperaturverteilung.

- wxButton * export_img_bt

  Button zum Export der Grafik.

- wxButton * export_csv_bt

  Button zum Export der Temperaturverteilung als .csv-Datei.

- GUIRenderCutCanvas * canvas

  Die Zeichenfläche zur Darstellung der berechneten Grafik und der Skala.

- wxImage * image

  Die berechnete Temperaturverteilung als Grafik.

- float * value_img

  Die berechnete Temperaturverteilung als Temperaturwerte.

- int core_count

  Die Anzahl der zu bei der Berechnung zu verwendender Prozessorkerne.

8.12.1 Ausführliche Beschreibung

Fenster zum erstellen zweidimensionaler Temperaturverteilungen.

Das Fenster ermöglicht es, eine zweidimensionale Temperaturverteilung auf einer Schnittebene durch das dreidimensionale Modell zu berechnen. Diese Schnittebene wird im 3D-Fenster des Hauptfensters visualisiert.

Definiert in Zeile 24 der Datei GUICutRenderWindow.h.

8.12.2 Beschreibung der Konstruktoren und Destruktoren

8.12.2.1 GUICutRenderWindow::GUICutRenderWindow (wxWindow * *parent*, const wxChar * *title*, int *xpos*, int *ypos*, int *width*, int *height*)

Der Konstuktor.

Parameter

parent	Das Übergeordnete Fenster. Muss vom Typ GUIMainWindow sein.
title	Titel des Fensters.
xpos	horizontale Position des Fensters.
ypos	vertikale Position des Fensters.
width	Breite des Fensters.
height	Höhe des Fenster

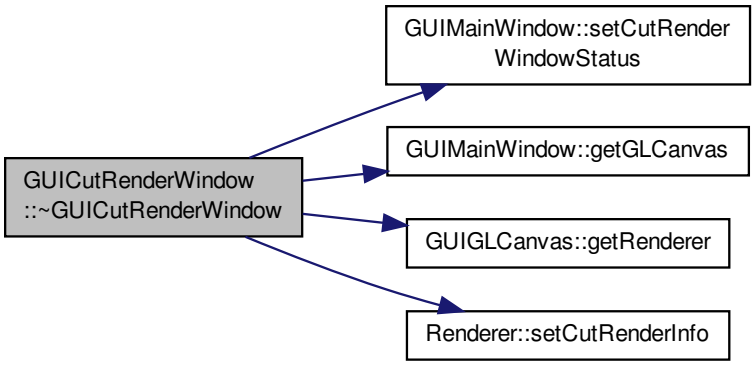
Definiert in Zeile 35 der Datei GUICutRenderWindow.cpp.

8.12.2.2 GUICutRenderWindow::~GUICutRenderWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 601 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.12.3 Dokumentation der Elementfunktionen

8.12.3.1 GUICutRenderWindow::DECLARE_EVENT_TABLE ( ) [protected]

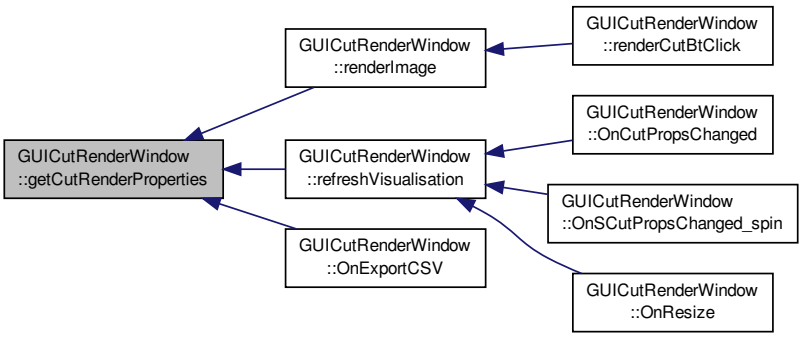
Event-Tabellendeklaration für wxWidgets.

8.12.3.2 CutRender_info * GUICutRenderWindow::getCutRenderProperties ( ) [private]

Gibt die aktuell eingestellten Eigenschaften für die zweidimensionale Temperaturverteilung zurück, damit Sie später an den Renderer des 3D-Fensters zur Visualisierung übergeben werden können.

Definiert in Zeile 546 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

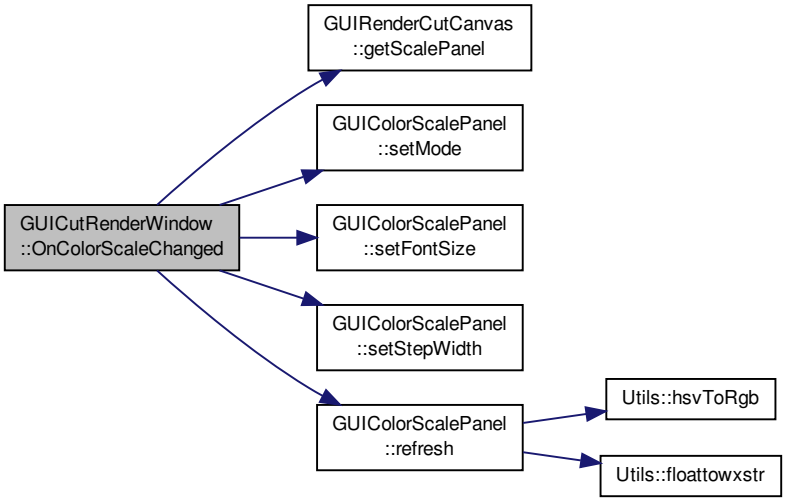


8.12.3.3 void GUICutRenderWindow::OnColorScaleChanged (wxCommandEvent & *event*) [private]

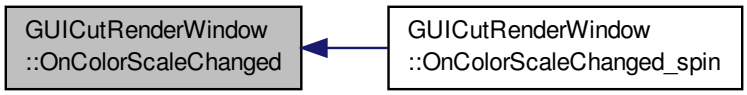
Behandelt das Ändern von Parametern zur darstellung der Temperaturskala.

Definiert in Zeile 585 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

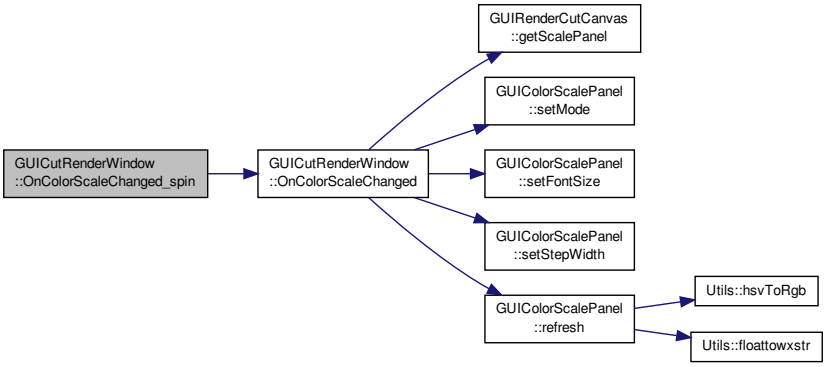


8.12.3.4 void GUICutRenderWindow::OnColorScaleChanged_spin (wxSpinEvent & *event*) [private]

Behandelt das Ändern von Parametern zur darstellung der Temperaturskala.

Definiert in Zeile 567 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

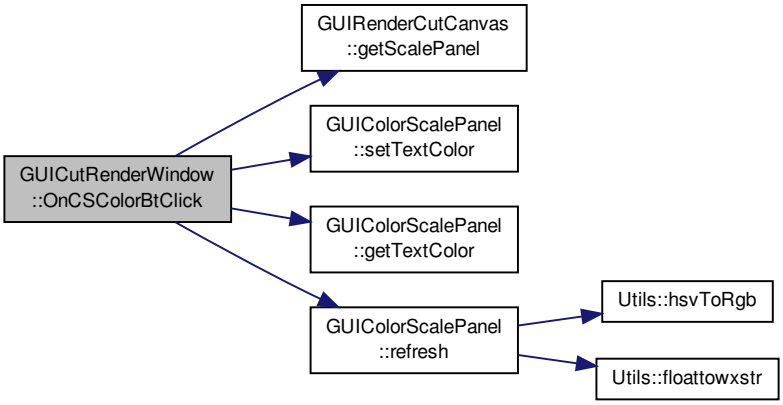


8.12.3.5 void GUICutRenderWindow::OnCSColorBtClick (wxCommandEvent & *event*) [private]

Behandelt das Klicken auf den Button zur Wahl der Schriftfarbe auf der Skala.

Definiert in Zeile 572 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

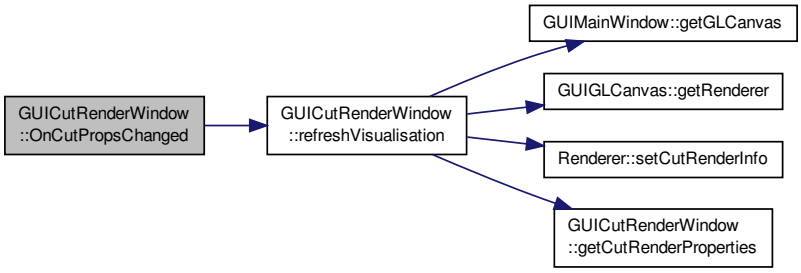


8.12.3.6 void GUICutRenderWindow::OnCutPropsChanged (wxCommandEvent & *event*) [private]

Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.

Definiert in Zeile 427 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

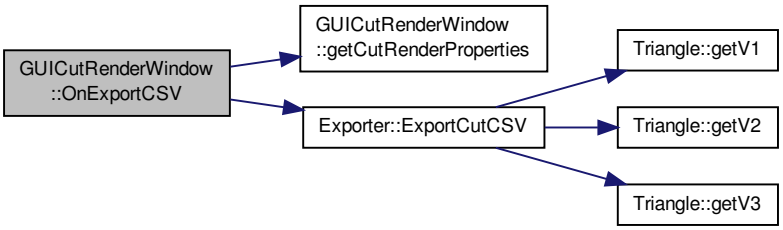


8.12.3.7 void GUICutRenderWindow::OnExportCSV (wxCommandEvent & *event*) [private]

Fragt den Benutzer nach dem Pfad und Exportiert die 2D-Temperaturverteilung als .csv-Datei.

Definiert in Zeile 486 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

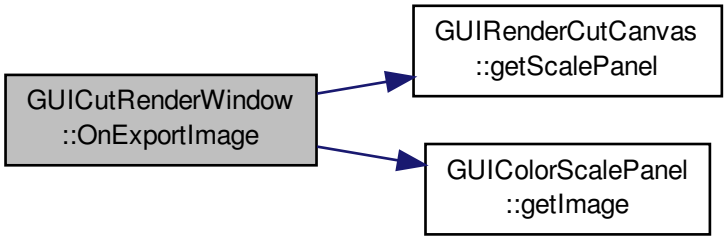


8.12.3.8 void GUICutRenderWindow::OnExportImage (wxCommandEvent & *event*) [private]

Fragt den Benutzer nach dem Pfad und Exportiert eine Grafik aus 2D-Temperaturverteilung und Temperaturskala.

Definiert in Zeile 504 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

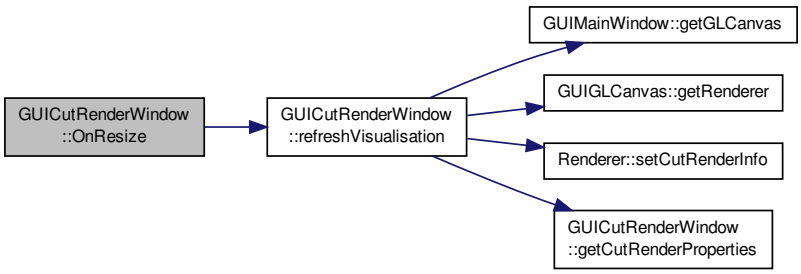


8.12.3.9 void GUICutRenderWindow::OnResize (wxSizeEvent & *event*) [private]

Behandelt Änderungen der Größe des Fensters.

Definiert in Zeile 442 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

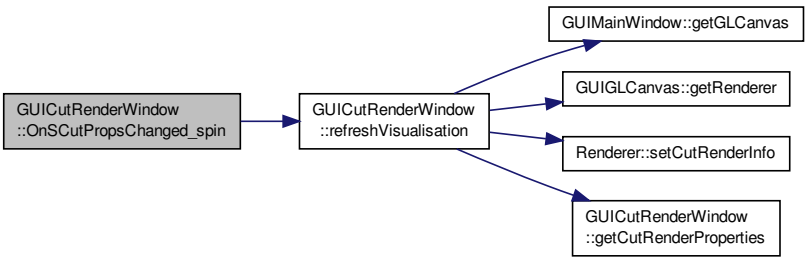


8.12.3.10 void GUICutRenderWindow::OnSCutPropsChanged_spin (wxSpinEvent & *event*) [private]

Behandelt das Ändern von Parametern zur Berechnung der 2D-Temperaturverteilung.

Definiert in Zeile 431 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

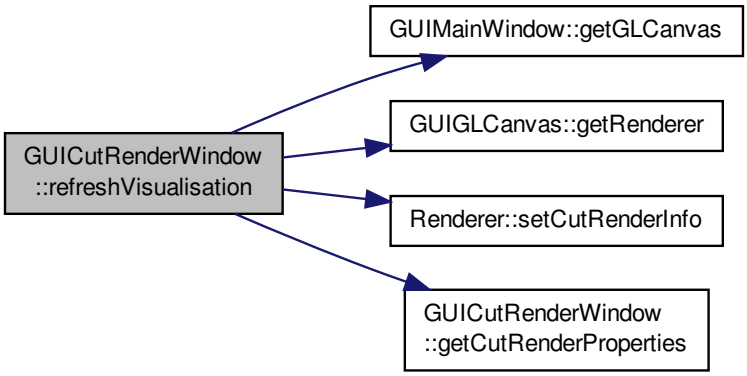


8.12.3.11 void GUICutRenderWindow::refreshVisualisation ( ) [private]

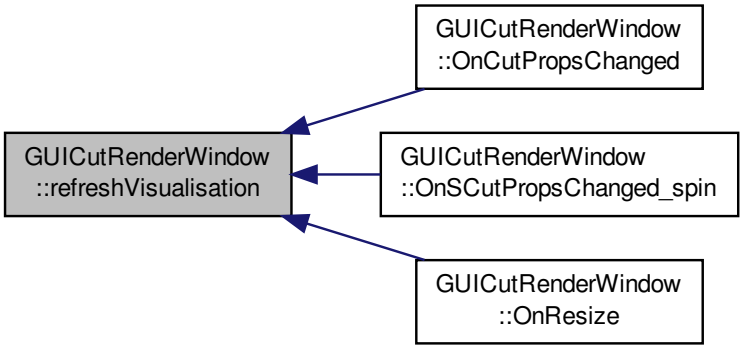
Aktualisiert die Visualisierung der Schnittebene im Hauptfenster.

Definiert in Zeile 435 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

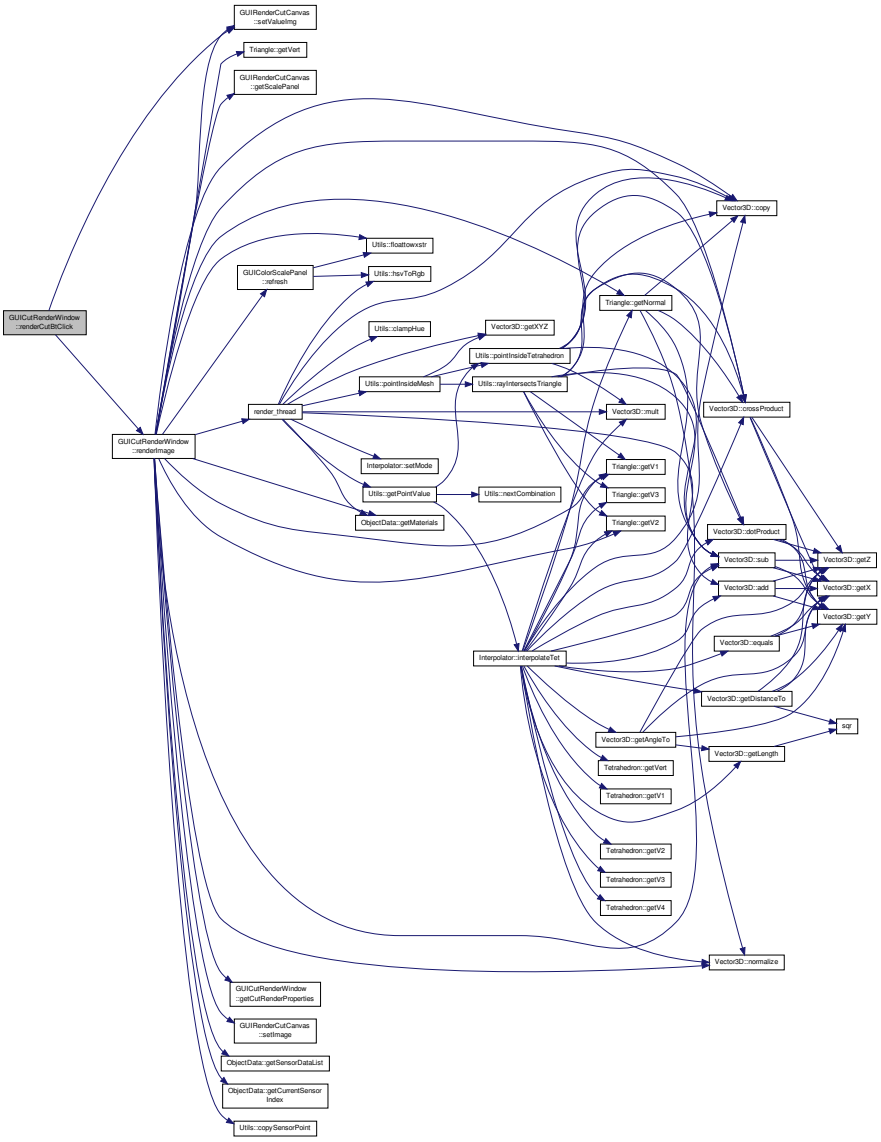


8.12.3.12 void GUICutRenderWindow::renderCutBtClick (wxCommandEvent & *event*) [private]

Behandelt das Drücken des Buttons zur Berechnung der zweidimensionalen Temperaturverteilung.

Definiert in Zeile 595 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

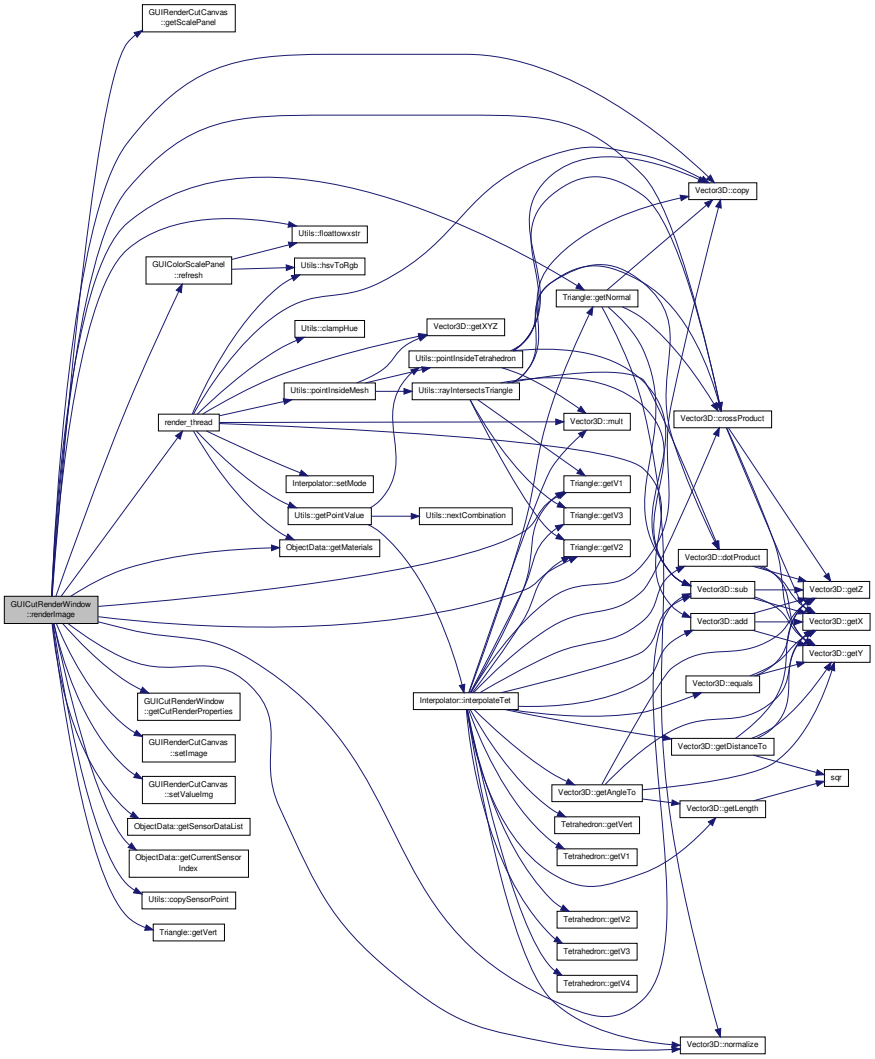


8.12.3.13 void GUICutRenderWindow::renderImage (wxImage * *image*) [private]

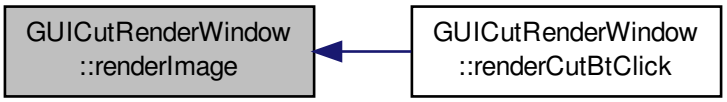
Berechnet die 2D-Temperaturverteilung als Grafik.

Definiert in Zeile 264 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.12.4 Dokumentation der Datenelemente

8.12.4.1 wxButton* GUICutRenderWindow::calcbt [private]

Button zum Starten der Berechnung der 2D-Temperaturverteilung.

Definiert in Zeile 256 der Datei GUICutRenderWindow.h.

8.12.4.2 GUIRenderCutCanvas* GUICutRenderWindow::canvas [private]

Die Zeichenfläche zur Darstellung der berechneten Grafik und der Skala.

Definiert in Zeile 271 der Datei GUICutRenderWindow.h.

8.12.4.3 int GUICutRenderWindow::core_count [private]

Die Anzahl der zu bei der Berechnung zu verwendender Prozessorkerne.

Definiert in Zeile 286 der Datei GUICutRenderWindow.h.

8.12.4.4 wxButton* GUICutRenderWindow::export_csv_bt [private]

Button zum Export der Temperaturverteilung als .csv-Datei.

Definiert in Zeile 266 der Datei GUICutRenderWindow.h.

8.12.4.5 wxButton* GUICutRenderWindow::export_img_bt [private]

Button zum Export der Grafik.

Definiert in Zeile 261 der Datei GUICutRenderWindow.h.

8.12.4.6 wxImage* GUICutRenderWindow::image [private]

Die berechnete Temperaturverteilung als Grafik.

Definiert in Zeile 276 der Datei GUICutRenderWindow.h.

8.12.4.7 wxSpinCtrl* GUICutRenderWindow::imgHeightEdit [private]

Feld zur Eingabe der Höhe des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.

Definiert in Zeile 166 der Datei GUICutRenderWindow.h.

8.12.4.8 wxSpinCtrl* GUICutRenderWindow::imgWidthEdit [private]

Textfeld zur Eingabe der Breite des Bereichs, für den die 2D-Temperaturverteilung als Grafik berechnet wird.

Definiert in Zeile 161 der Datei GUICutRenderWindow.h.

8.12.4.9 wxTextCtrl* GUICutRenderWindow::mmperpixeledit [private]

Feld zur Eingabe des Maßstabs in $\frac{mm}{px}$.

Definiert in Zeile 176 der Datei GUICutRenderWindow.h.

8.12.4.10 wxStaticText* GUICutRenderWindow::mmperpixellabel [private]

Beschriftung für den Maßstab in $\frac{mm}{px}$.

Definiert in Zeile 196 der Datei GUICutRenderWindow.h.

8.12.4.11 wxStaticText* GUICutRenderWindow::optionslbl [private]

Beschriftung für die die 2D-Temperaturverteilung betreffenden Parameter.

Definiert in Zeile 206 der Datei GUICutRenderWindow.h.

8.12.4.12 wxStaticText* GUICutRenderWindow::p1label [private]

Beschriftung für den 1.

die Schnittebene definierenden Punkt.

Definiert in Zeile 181 der Datei GUICutRenderWindow.h.

8.12.4.13 wxTextCtrl* GUICutRenderWindow::p1xedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 116 der Datei GUICutRenderWindow.h.

8.12.4.14 wxTextCtrl* GUICutRenderWindow::p1yedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 121 der Datei GUICutRenderWindow.h.

8.12.4.15 wxTextCtrl* GUICutRenderWindow::p1zedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 126 der Datei GUICutRenderWindow.h.

8.12.4.16 wxStaticText* GUICutRenderWindow::p2label [private]

Beschriftung für den 2.

die Schnittebene definierenden Punkt.

Definiert in Zeile 186 der Datei GUICutRenderWindow.h.

8.12.4.17 wxTextCtrl* GUICutRenderWindow::p2xedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 131 der Datei GUICutRenderWindow.h.

8.12.4.18 wxTextCtrl* GUICutRenderWindow::p2yedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 136 der Datei GUICutRenderWindow.h.

8.12.4.19 wxTextCtrl* GUICutRenderWindow::p2zedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 141 der Datei GUICutRenderWindow.h.

8.12.4.20 wxStaticText* GUICutRenderWindow::p3label [private]

Beschriftung für den 3.

die Schnittebene definierenden Punkt.

Definiert in Zeile 191 der Datei GUICutRenderWindow.h.

8.12.4.21 wxTextCtrl* GUICutRenderWindow::p3xedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 146 der Datei GUICutRenderWindow.h.

8.12.4.22 wxTextCtrl* GUICutRenderWindow::p3yedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 151 der Datei GUICutRenderWindow.h.

8.12.4.23 wxTextCtrl* GUICutRenderWindow::p3zedit [private]

Textfeld zur Eingabe der Position des Dreiecks, dass die Schnittebene definiert.

Definiert in Zeile 156 der Datei GUICutRenderWindow.h.

8.12.4.24 wxButton* GUICutRenderWindow::scalefontcolorbt [private]

Button zur Auswahl der Schriftfarbe.

Definiert in Zeile 246 der Datei GUICutRenderWindow.h.

8.12.4.25 wxStaticText* GUICutRenderWindow::scalefontpropslbl [private]

Beschriftung für die Schrifteigenschaften der Skala.

Definiert in Zeile 236 der Datei GUICutRenderWindow.h.

8.12.4.26 wxSpinCtrl* GUICutRenderWindow::scalefontsizeedit [private]

Feld zur Eingabe der Schriftgröße der Skala.

Definiert in Zeile 241 der Datei GUICutRenderWindow.h.

8.12.4.27 wxStaticText* GUICutRenderWindow::scalelbl [private]

Beschriftung für die die Skala betreffenden Optionen.

Definiert in Zeile 221 der Datei GUICutRenderWindow.h.

8.12.4.28 wxComboBox* GUICutRenderWindow::scalemodecb [private]

Menübox zur Auswahl des Darstellungsmodus der Skala.

Definiert in Zeile 231 der Datei GUICutRenderWindow.h.

8.12.4.29 wxStaticText* GUICutRenderWindow::scalemodelbl [private]

Beschriftung für den Darstellungsmodus der Skala.

Definiert in Zeile 226 der Datei GUICutRenderWindow.h.

8.12.4.30 wxSpinCtrl* GUICutRenderWindow::scalestepedit [private]

Feld zur Eingabe der Schrittweite der Skala.

Definiert in Zeile 251 der Datei GUICutRenderWindow.h.

8.12.4.31 wxScrolledWindow* GUICutRenderWindow::scroll_pane [private]

Scrollender Bereich, in den die anderen Komponenten außer der Zeichenfläche (canvas) eingebettet sind.

Definiert in Zeile 111 der Datei GUICutRenderWindow.h.

8.12.4.32 wxSpinCtrl* GUICutRenderWindow::threadcountedit [private]

Feld zur Eingabe der zum Berechnen zu verwendenden Prozessorkerne.

Definiert in Zeile 171 der Datei GUICutRenderWindow.h.

8.12.4.33 wxStaticText* GUICutRenderWindow::threadcountlbl [private]

Beschriftung für die Anzahl bei der Berechnung zu verwendender Prozessorkerne.

Definiert in Zeile 216 der Datei GUICutRenderWindow.h.

8.12.4.34 wxStaticText* GUICutRenderWindow::trilabel [private]

Beschriftung für das die Schnittebene definierende Dreieck.

Definiert in Zeile 201 der Datei GUICutRenderWindow.h.

8.12.4.35 float* GUICutRenderWindow::value_img [private]

Die berechnete Temperaturverteilung als Temperaturwerte.

Definiert in Zeile 281 der Datei GUICutRenderWindow.h.

8.12.4.36 wxStaticText* GUICutRenderWindow::widthHeightlbl [private]

Beschriftung für Breite und Höhe der Grafik.

Definiert in Zeile 211 der Datei GUICutRenderWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

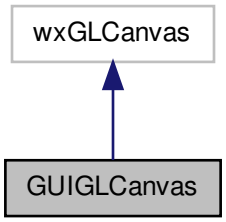
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.cpp

8.13 GUIGLCanvas Klassenreferenz

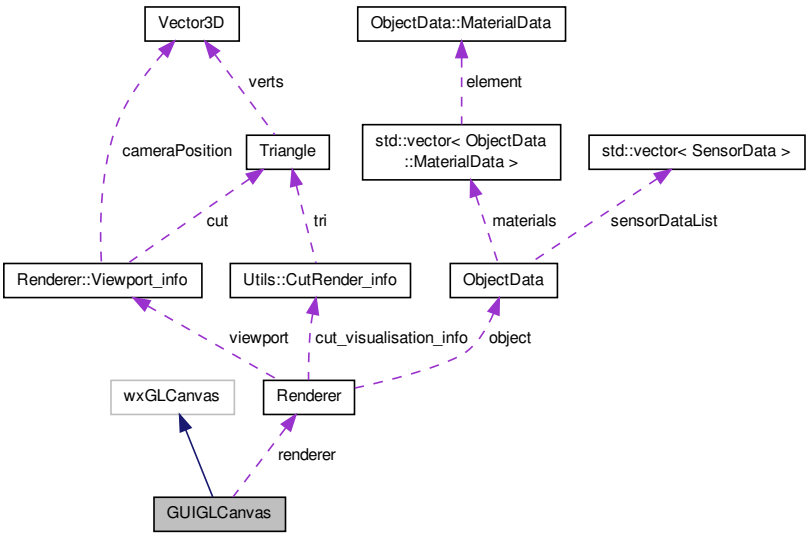
Zeichenfläche für das 3D-Fenster.

`#include <GUIGLCanvas.h>`

Klassendiagramm für GUIGLCanvas:



Zusammengehörigkeiten von GUIGLCanvas:



Öffentliche Methoden

- GUIGLCanvas (wxFrame ∗parent)

  Der Konstruktor.
- void setRenderObject (ObjectData ∗obj)

  Setzt das darzustellende Objekt.
- Renderer ∗ getRenderer ()

  Gibt den Renderer der Zeichenfläche zurück.
- void refresh ()

Zeichnet den Inhalt des 3D-Fensters neu und aktualisiert den Renderer, z.B.

- virtual ~GUIGLCanvas ()

  Der Destruktor.

Private Methoden

- void OnPaint (wxPaintEvent &event)

  Event-Tabellendeklaration für wxWidgets.
- void OnMouseWheel (wxMouseEvent &event)

  Behandelt Mausradbewegungen (zoomen).
- void OnMouseMove (wxMouseEvent &event)

  Behandelt Mausbewegungen (verschieben und drehen der Ansicht).
- void OnResize (wxSizeEvent &event)

  Behandelt Größenänderungen der Zeichenfläche.

Private Attribute

- Renderer renderer

  Der verwendete Renderer.
- bool is_initialized

  Initialisiertungsstatus des Objekts.
- bool do_refresh

  Statusvariable, gibt an ob beim Zeichnen auch der Renderer aktualisiert wird.
- int prev_mouse_x

  Zwischenspeicher für die vorherige Mausposition (X).
- int prev_mouse_y

  Zwischenspeicher für die vorherige Mausposition (Y).

8.13.1 Ausführliche Beschreibung

Zeichenfläche für das 3D-Fenster.

Klasse zum Verwalten der im 3D-Fenster angezeigten Inhalte. Auch zuständig für Drehen, Verschieben und Zoomen der Ansicht.

Definiert in Zeile 22 der Datei GUIGLCanvas.h.

8.13.2 Beschreibung der Konstruktoren und Destruktoren

8.13.2.1 GUIGLCanvas::GUIGLCanvas (wxFrame * *parent*)

Der Konstruktor.

Parameter

parent	Das Fenster, auf dem sich die Zeichenfläche befindet.

Definiert in Zeile 22 der Datei GUIGLCanvas.cpp.

8.13.2.2 GUIGLCanvas::~GUIGLCanvas ( ) [virtual]

Der Destruktor.

Definiert in Zeile 150 der Datei GUIGLCanvas.cpp.

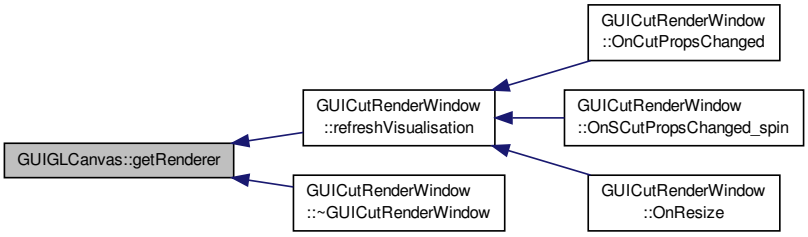
8.13.3 Dokumentation der Elementfunktionen

8.13.3.1 Renderer * GUIGLCanvas::getRenderer ( )

Gibt den Renderer der Zeichenfläche zurück.

Definiert in Zeile 101 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

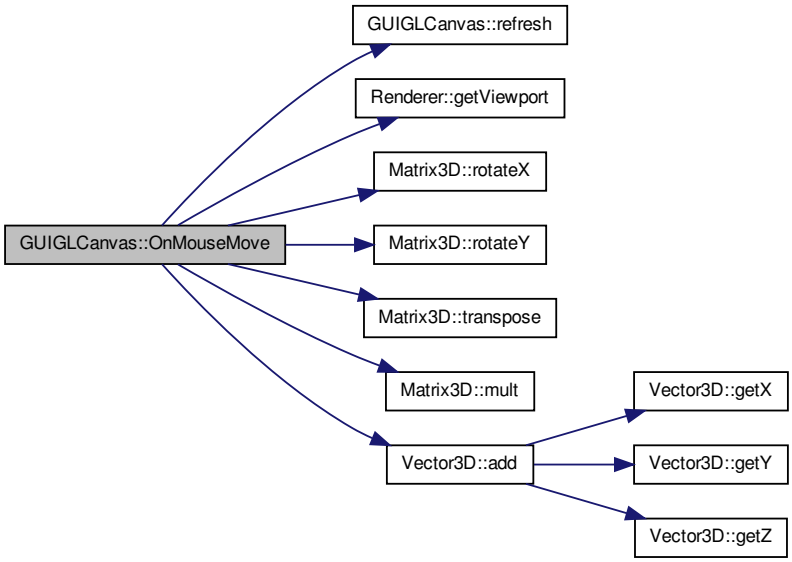


8.13.3.2 void GUIGLCanvas::OnMouseMove (wxMouseEvent & *event*) [private]

Behandelt Mausbewegungen (verschieben und drehen der Ansicht).

Definiert in Zeile 105 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

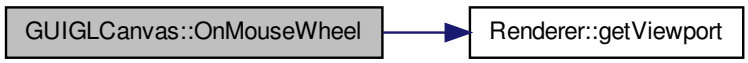


8.13.3.3 void GUIGLCanvas::OnMouseWheel (wxMouseEvent & *event*) [private]

Behandelt Mausradbewegungen (zoomen).

Definiert in Zeile 43 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



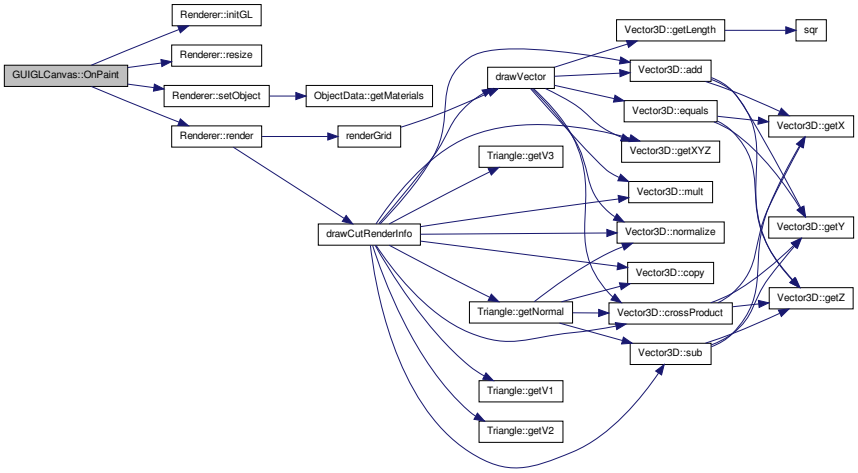
8.13.3.4 void GUIGLCanvas::OnPaint (wxPaintEvent & *event*) [private]

Event-Tabellendeklaration für wxWidgets.

Behandelt das Zeichenevent und zeichnet die Inhalte des 3D-Fensters.

Definiert in Zeile 56 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.13.3.5 void GUIGLCanvas::OnResize (wxSizeEvent & *event*) [private]

Behandelt Größenänderungen der Zeichenfläche.

Definiert in Zeile 34 der Datei GUIGLCanvas.cpp.

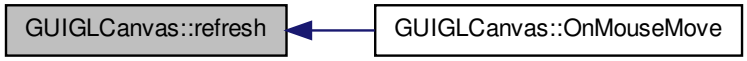
8.13.3.6 void GUIGLCanvas::refresh ( )

Zeichnet den Inhalt des 3D-Fensters neu und aktualisiert den Renderer, z.B.

bei geänderter Fenstergröße oder geänderten Eigenschaften des angezeigten Objekts.

Definiert in Zeile 95 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

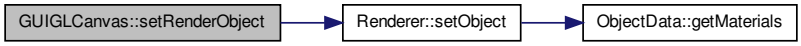


8.13.3.7 void GUIGLCanvas::setRenderObject (ObjectData * *obj*)

Setzt das darzustellende Objekt.

Definiert in Zeile 83 der Datei GUIGLCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.13.4 Dokumentation der Datenelemente

8.13.4.1 bool GUIGLCanvas::do_refresh [private]

Statusvariable, gibt an ob beim Zeichnen auch der Renderer aktualisiert wird.

Dies tritt beispielsweise bei Größenänderungen oder Änderungen am Objekt ein, da die Daten teilweise neu an den Renderer übermittelt werden müssen.

Definiert in Zeile 91 der Datei GUIGLCanvas.h.

8.13.4.2 bool GUIGLCanvas::is_initialized [private]

Initialisiertungsstatus des Objekts.

Definiert in Zeile 84 der Datei GUIGLCanvas.h.

8.13.4.3 int GUIGLCanvas::prev_mouse_x [private]

Zwischenspeicher für die vorherige Mausposition (X).

Definiert in Zeile 96 der Datei GUIGLCanvas.h.

8.13.4.4 int GUIGLCanvas::prev_mouse_y [private]

Zwischenspeicher für die vorherige Mausposition (Y).

Definiert in Zeile 101 der Datei GUIGLCanvas.h.

8.13.4.5 Renderer GUIGLCanvas::renderer [private]

Der verwendete Renderer.

Definiert in Zeile 79 der Datei GUIGLCanvas.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

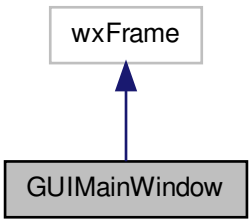
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.cpp

8.14 GUIMainWindow Klassenreferenz

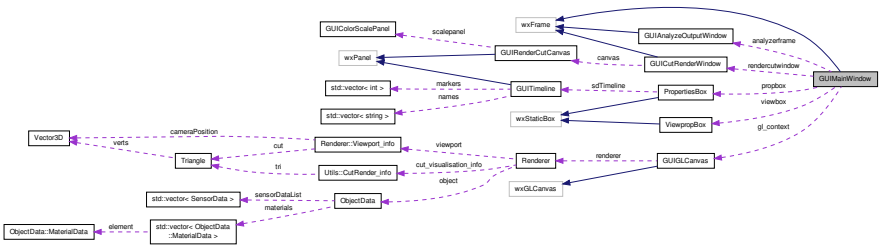
Hauptfenster mit Hauptmenü und Zugriff auf die einzelnen Programmfunktionen.

`#include <GUIMainWindow.h>`

Klassendiagramm für GUIMainWindow:



Zusammengehörigkeiten von GUIMainWindow:



Öffentliche Methoden

- GUIMainWindow (const wxChar *title, int xpos, int ypos, int width, int height)

  Der Konstruktor.
- void setAnalyzeWindowStatus (bool isValid)

  Setzt den Status des Übersichstfensters über die Analysedaten.
- void setCutRenderWindowStatus (bool isValid)

  Setzt den Status des Übersichstfensters über die Analysedaten.
- GUIGLCanvas * getGLCanvas ()

Gibt die Zeichenfläche des 3D-Fensters zurück.

- virtual ~GUIMainWindow ()

  Der Destruktor.

Geschützte Attribute

- string configpaths [NUMBEROFPATHS]

  Suchpfade für die Anwendungsdaten.

Statische, geschützte Attribute

- static const int NUMBEROFPATHS = 2

  Anzahl der Suchpfade für die Anwendungsdaten (z.B.

Private Methoden

- void OnMenuImportObj (wxCommandEvent &event)

  Event-Tabellendeklaration für wxWidgets.
- void OnMenuImportSD (wxCommandEvent &event)

  Öffnet den Dialog zum Importieren einfacher Sensordaten.
- void OnMenuImportTSD (wxCommandEvent &event)

  Öffnet den Dialog zum Importieren zeitbezogener Sensordaten.
- void OnMenuFileQuit (wxCommandEvent &event)

  Beendet das Programm.
- void OnMenuHelpAbout (wxCommandEvent &event)

  Öffnet ein Fenster mit Informationen über das Programm.
- void OnMenuOpenManual (wxCommandEvent &event)

  Öffnet das Handbuch mit dem PDF-Viewer des Systems.
- void OnRecalcBtClick (wxCommandEvent &event)

  Berechnet die 3D-Temperaturverteilung neu.
- void OnResize (wxSizeEvent &event)

  Behandelt Größenänderungen des Fensters.
- void OnMaterialSelect (wxCommandEvent &event)

  Aktualisiert die Oberfläche nach dem Auswählen eines anderen Materials im Objekteigenschaften-Fenster.
- void OnAnalyze (wxCommandEvent &event)

  Öffnet das Analysedaten-Übersichtsfenster.
- void OnImmediateUpdatePropChange (wxCommandEvent &event)

  Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften,bei denen ein sofortiges Update der Oberfläche möglich ist, durch den Nutzer.
- void OnGeneralPropChange (wxCommandEvent &event)

  Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften durch den Nutzer.
- void OnViewPropChange (wxCommandEvent &event)

  Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.
- void OnViewPropSpinChange (wxSpinEvent &event)

  Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.
- void OnSensorDataChange (wxCommandEvent &event)

  Behandelt das Auswählen eines anderen Sensordatensatzes.
- void OnSDTimelineChange (wxCommandEvent &event)

  Behandelt Änderungen an der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).
- void OnSDTLMarkerClear (wxCommandEvent &event)

Löscht alle Markierungen auf der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).

- void OnSDTLNextMarker (wxCommandEvent &event)

  Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den nächsten markierten Zeitpunkt.
- void OnSDTLPrevMarker (wxCommandEvent &event)

  Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den vorherigen markierten Zeitpunkt.
- void OnAnalyzeMarkerChange (wxCommandEvent &event)

  Behandelt das Markieren eines Zeitpunktes auf der Sensordaten-Zeitleiste.
- void OnActiveObjectChangePopup (wxCommandEvent &event)

  Setzt das aktive Objetk nach dem Auswählen im Popup-Menü.
- void OnActiveObjectChange (wxCommandEvent &event)

  Öffnet das Popup-Menü zum auswählen des aktiven Objekts.
- void OnActiveObjectDelete (wxCommandEvent &event)

  Löscht das aktive Objekt, sofern es nicht das einzige geladene Objekt ist.
- void OnAnalyzePoint (wxCommandEvent &event)

  Öffnet das Fenster zur Analyse eines Punktes (GUIAnalyzePointWindow).
- void OnRenderCut (wxCommandEvent &event)

  Öffnet das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.
- void addObject (ObjectData ∗obj)

  Registriert ein neues (Versuchs-) Objekt im Programm.
- void setActiveObject (int index)

  Setzt das aktive Objekt.
- void OnExportViewportImage (wxCommandEvent &event)

  Öffnet ein Fenster zum Exportieren der Ansicht des 3D-Fensters.
- void OnExportVTK (wxCommandEvent &event)

  Öffnet ein Fenster zum Exportieren der Temperaturverteilung und des Objekts im VTK-Format.
- void OnFindMaxTSD (wxCommandEvent &event)

  Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wäremeenergiegehalt maximal wird.
- void OnAutoUpdateChange (wxCommandEvent &event)

  Behandelt das aktivieren/deaktivieren der Option zum automatischen neuberechnen der Temperaturverteilung eines Objekts, sobald Änderungen an dessen Eigenschaften vorgenommen werden.
- void assignCurrentObjectProps ()

  Überträgt die in der GUI eingetragenen Objekteigenschaften in das aktive Objekt.
- void updateObjectPropGUI ()

  Überträgt die Eigenschaften des aktiven Objekts in die GUI.
- void assignViewProps ()

  Speichert die Visualisierungsoptionen aus der GUI.
- void updateViewPropGUI ()

  Lädt die Visualisierungsoptionen in die GUI.

Private Attribute

- GUIGLCanvas ∗ gl_context

  Die Zeichenfläche für das 3D-Fenster.
- wxToolBar ∗ toolbar

  Die Tollbarkomponente.
- wxMenuBar ∗ mwMenuBar

  Die Hauptmenükomponente.
- wxMenu ∗ mwFileMenu

  Das "Datei"-Untermenü.
- wxMenu ∗ mwHelpMenu

Das "Hilfe"-Untermenü.

- wxMenu * mwImportMenu

  Das "Import"-Untermenü.
- wxMenu * mwExportMenu

  Das "Export"-Untermenü.
- wxMenu * mwAnalyzeMenu

  Das "Analysieren"-Untermenü.
- wxMenu * mwEditMenu

  Das "Bearbeiten"-Untermenü
- PropertiesBox * propbox

  Die Unterkomponente, die die Objekteigenschaften-Oberfläche enthält.
- ViewpropBox * viewbox

  Die Unterkomponente, die die Visualisierungsoptionen-Oberfläche enthält.
- GUIAnalyzeOutputWindow * analyzerframe

  Das Analysedaten-Übersichtsfenster.
- GUICutRenderWindow * rendercutwindow

  Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.
- wxScrolledWindow * prop_scroll_win

  Scrollender Bereich, in den die Objekteigenschaften-Oberfläche eingebettet ist.
- wxScrolledWindow * view_scroll_win

  Scrollender Bereich, in den die Visualisierungsoptionen-Oberfläche eingebettet ist.
- bool updating

  Die Oberfläche wird gerade vom Programm verändert.
- bool analyze_window_valid

  Das Analysedaten-Übersichtsfenster ist gerade geöffnet.
- bool render_cut_window_valid

  Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung ist gerade geöffnet.
- string data_directory

  Der Pfad zum Verzeichnis, das die von der Anwendung verwendeten Daten (z.B.

8.14.1 Ausführliche Beschreibung

Hauptfenster mit Hauptmenü und Zugriff auf die einzelnen Programmfunktionen.

Das Hauptfenster bietet über das Hauptmenü und die Oberfläche Zugriff auf die Funktionen des Programms. Dazu kann das aktuelle Objekt gewählt werden, welches dann im eingebetteten 3D-Fenster angezeigt wird. Eigenschaften der Visualisierung und des Objekts können ebenfalls über die Oberfläche des Hauptfensters festgelegt werden.

Definiert in Zeile 28 der Datei GUIMainWindow.h.

8.14.2 Beschreibung der Konstruktoren und Destruktoren

8.14.2.1 GUIMainWindow::GUIMainWindow (const wxChar * *title*, int *xpos*, int *ypos*, int *width*, int *height*)

Der Konstruktor.

Parameter

title	Der Titel des Programmfensters.

xpos	horizontale Position des Fensters.
ypos	vertikale Position des Fensters.
width	Breite des Fensters.
height	Höhe des Fensters.

Erstellen und initialisieren der Fensterkomponenten

Definiert in Zeile 75 der Datei GUIMainWindow.cpp.

8.14.2.2 GUIMainWindow::~GUIMainWindow ( ) [virtual]

Der Destruktor.

Definiert in Zeile 984 der Datei GUIMainWindow.cpp.

8.14.3 Dokumentation der Elementfunktionen

8.14.3.1 void GUIMainWindow::addObject (ObjectData * *obj*) [private]

Registriert ein neues (Versuchs-) Objekt im Programm.

Parameter

obj	Das zu registrierende Objekt.

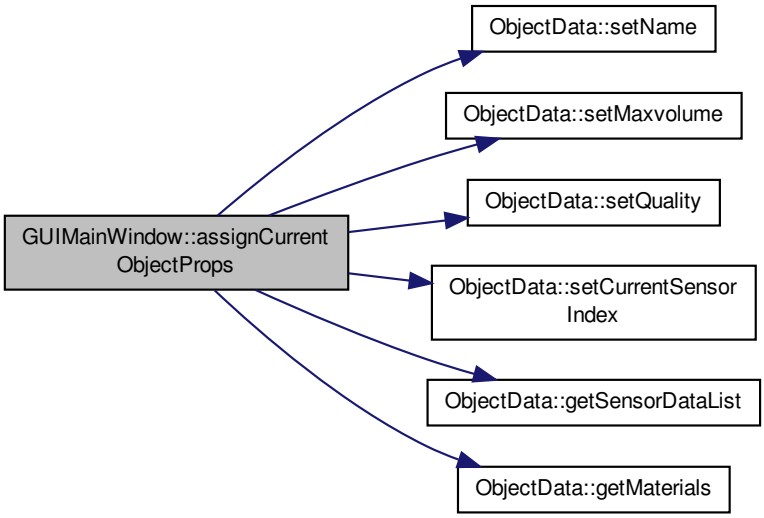
Definiert in Zeile 290 der Datei GUIMainWindow.cpp.

8.14.3.2 void GUIMainWindow::assignCurrentObjectProps ( ) [private]

Überträgt die in der GUI eingetragenen Objekteigenschaften in das aktive Objekt.

Definiert in Zeile 482 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.14.3.3 void GUIMainWindow::assignViewProps ( ) [private]

Speichert die Visualisierungsoptionen aus der GUI.

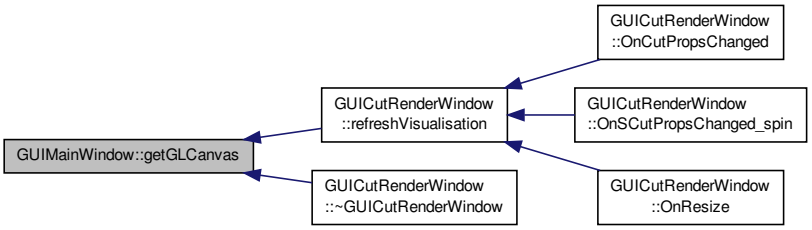
Definiert in Zeile 688 der Datei GUIMainWindow.cpp.

8.14.3.4 GUIGLCanvas * GUIMainWindow::getGLCanvas ( )

Gibt die Zeichenfläche des 3D-Fensters zurück.

Definiert in Zeile 286 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

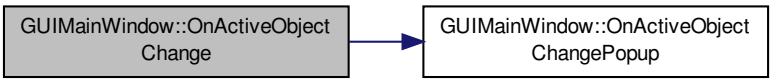


8.14.3.5 void GUIMainWindow::OnActiveObjectChange (wxCommandEvent & *event*) [private]

Öffnet das Popup-Menü zum auswählen des aktiven Objekts.

Definiert in Zeile 936 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

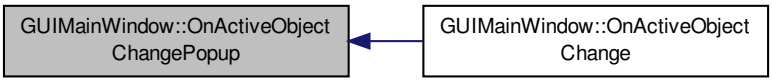


8.14.3.6 void GUIMainWindow::OnActiveObjectChangePopup (wxCommandEvent & *event*) [private]

Setzt das aktive Objetk nach dem Auswählen im Popup-Menü.

Definiert in Zeile 929 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.14.3.7 void GUIMainWindow::OnActiveObjectDelete (wxCommandEvent & *event*) [private]

Löscht das aktive Objekt, sofern es nicht das einzige geladene Objekt ist.

Definiert in Zeile 308 der Datei GUIMainWindow.cpp.

8.14.3.8 void GUIMainWindow::OnAnalyze (wxCommandEvent & *event*) [private]

Öffnet das Analysedaten-Übersichtsfenster.

Definiert in Zeile 615 der Datei GUIMainWindow.cpp.

8.14.3.9 void GUIMainWindow::OnAnalyzeMarkerChange (wxCommandEvent & *event*) [private]

Behandelt das Markieren eines Zeitpunktes auf der Sensordaten-Zeitleiste.

Definiert in Zeile 380 der Datei GUIMainWindow.cpp.

8.14.3.10 void GUIMainWindow::OnAnalyzePoint (wxCommandEvent & *event*) [private]

Öffnet das Fenster zur Analyse eines Punktes (GUIAnalyzePointWindow).

Definiert in Zeile 631 der Datei GUIMainWindow.cpp.

8.14.3.11 void GUIMainWindow::OnAutoUpdateChange (wxCommandEvent & *event*) [private]

Behandelt das aktivieren/deaktivieren der Option zum automatischen neuberechnen der Temperaturverteilung eines Objekts, sobald Änderungen an dessen Eigenschaften vorgenommen werden.

Definiert in Zeile 476 der Datei GUIMainWindow.cpp.

8.14.3.12 void GUIMainWindow::OnExportViewportImage (wxCommandEvent & *event*) [private]

Öffnet ein Fenster zum Exportieren der Ansicht des 3D-Fensters.

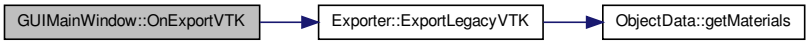
Definiert in Zeile 875 der Datei GUIMainWindow.cpp.

8.14.3.13 void GUIMainWindow::OnExportVTK (wxCommandEvent & *event*) [private]

Öffnet ein Fenster zum Exportieren der Temperaturverteilung und des Objekts im VTK-Format.

Definiert in Zeile 904 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



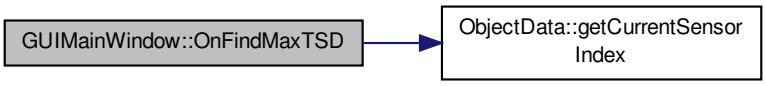
8.14.3.14 void GUIMainWindow::OnFindMaxTSD (wxCommandEvent & *event*) [private]

Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wäremeenergiegehalt maximal wird.

Dabei wird der Bereich zwischen den beiden markierten Stellen ausgewählt, zwischen denen sich der aktuell ausgewählte Zeitpunkt befindet.

Definiert in Zeile 390 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.14.3.15 void GUIMainWindow::OnGeneralPropChange (wxCommandEvent & *event*) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften durch den Nutzer.

Definiert in Zeile 674 der Datei GUIMainWindow.cpp.

8.14.3.16 void GUIMainWindow::OnImmediateUpdatePropChange (wxCommandEvent & *event*) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Objekteigenschaften,bei denen ein sofortiges Update der Oberfläche möglich ist, durch den Nutzer.

Definiert in Zeile 662 der Datei GUIMainWindow.cpp.

8.14.3.17 void GUIMainWindow::OnMaterialSelect (wxCommandEvent & *event*) [private]

Aktualisiert die Oberfläche nach dem Auswählen eines anderen Materials im Objekteigenschaften-Fenster.

Definiert in Zeile 582 der Datei GUIMainWindow.cpp.

8.14.3.18 void GUIMainWindow::OnMenuFileQuit (wxCommandEvent & *event*) [private]

Beendet das Programm.

Definiert in Zeile 960 der Datei GUIMainWindow.cpp.

8.14.3.19 void GUIMainWindow::OnMenuHelpAbout (wxCommandEvent & *event*) [private]

Öffnet ein Fenster mit Informationen über das Programm.

Definiert in Zeile 980 der Datei GUIMainWindow.cpp.

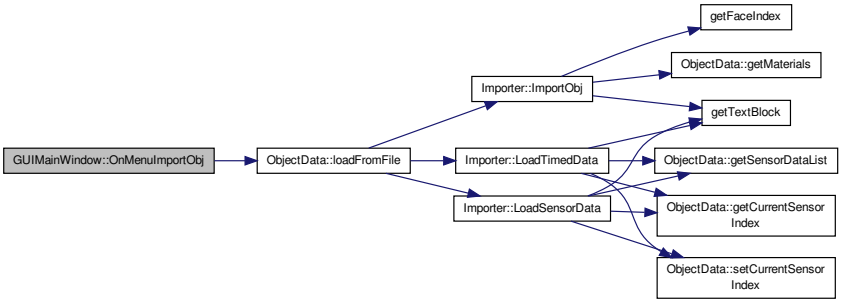
8.14.3.20 void GUIMainWindow::OnMenuImportObj (wxCommandEvent & *event*) [private]

Event-Tabellendeklaration für wxWidgets.

Öffnet den Dialog zum Importieren eines Objekts.

Definiert in Zeile 781 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

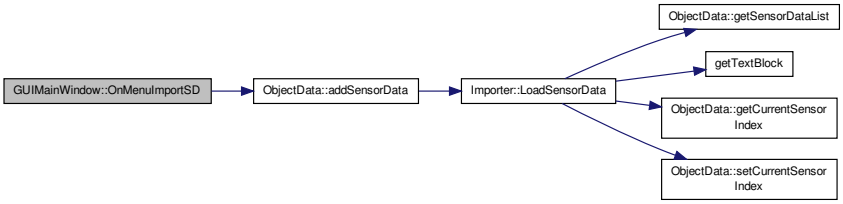


8.14.3.21 void GUIMainWindow::OnMenuImportSD (wxCommandEvent & *event*) [private]

Öffnet den Dialog zum Importieren einfacher Sensordaten.

Definiert in Zeile 833 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

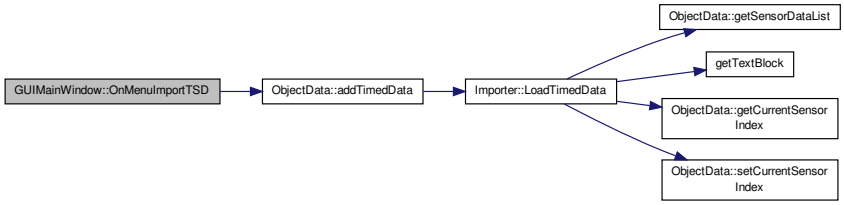


8.14.3.22 void GUIMainWindow::OnMenuImportTSD (wxCommandEvent & *event*) [private]

Öffnet den Dialog zum Importieren zeitbezogener Sensordaten.

Definiert in Zeile 854 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.14.3.23 void GUIMainWindow::OnMenuOpenManual (wxCommandEvent & *event*) [private]

Öffnet das Handbuch mit dem PDF-Viewer des Systems.

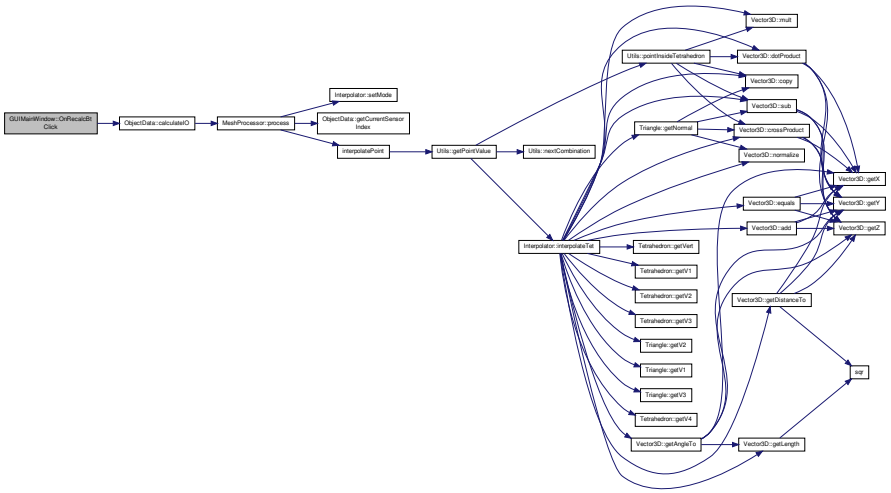
Definiert in Zeile 965 der Datei GUIMainWindow.cpp.

8.14.3.24 void GUIMainWindow::OnRecalcBtClick (wxCommandEvent & *event*) [private]

Berechnet die 3D-Temperaturverteilung neu.

Definiert in Zeile 592 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.14.3.25 void GUIMainWindow::OnRenderCut (wxCommandEvent & *event*) [private]

Öffnet das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.

Definiert in Zeile 644 der Datei GUIMainWindow.cpp.

8.14.3.26 void GUIMainWindow::OnResize (wxSizeEvent & *event*) [private]

Behandelt Größenänderungen des Fensters.

Definiert in Zeile 323 der Datei GUIMainWindow.cpp.

8.14.3.27 void GUIMainWindow::OnSDTimelineChange (wxCommandEvent & *event*) [private]

Behandelt Änderungen an der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).

Definiert in Zeile 348 der Datei GUIMainWindow.cpp.

8.14.3.28 void GUIMainWindow::OnSDTLMarkerClear (wxCommandEvent & *event*) [private]

Löscht alle Markierungen auf der Sensordaten-Zeitleiste (bei zeitbezogenen Datensätzen).

Definiert in Zeile 370 der Datei GUIMainWindow.cpp.

8.14.3.29 void GUIMainWindow::OnSDTLNextMarker (wxCommandEvent & *event*) [private]

Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den nächsten markierten Zeitpunkt.

Definiert in Zeile 420 der Datei GUIMainWindow.cpp.

8.14.3.30 void GUIMainWindow::OnSDTLPrevMarker (wxCommandEvent & *event*) [private]

Setzen des auf der Sensordaten-Zeitleiste ausgewählten Zeitpunktes auf den vorherigen markierten Zeitpunkt.

Definiert in Zeile 440 der Datei GUIMainWindow.cpp.

8.14.3.31 void GUIMainWindow::OnSensorDataChange (wxCommandEvent & *event*) [private]

Behandelt das Auswählen eines anderen Sensordatensatzes.

Definiert in Zeile 360 der Datei GUIMainWindow.cpp.

8.14.3.32 void GUIMainWindow::OnViewPropChange (wxCommandEvent & *event*) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.

Definiert in Zeile 767 der Datei GUIMainWindow.cpp.

8.14.3.33 void GUIMainWindow::OnViewPropSpinChange (wxSpinEvent & *event*) [private]

Behandelt das Aktualisieren der Oberfläche nach einer Änderung an Visualisierungsoptionen durch den Nutzer.

Definiert in Zeile 774 der Datei GUIMainWindow.cpp.

8.14.3.34 void GUIMainWindow::setActiveObject (int *index*) [private]

Setzt das aktive Objekt.

Parameter

index	Index des als aktives Objekt zu verwendeten Objekts.

Definiert in Zeile 297 der Datei GUIMainWindow.cpp.

8.14.3.35 void GUIMainWindow::setAnalyzeWindowStatus (bool *isValid*)

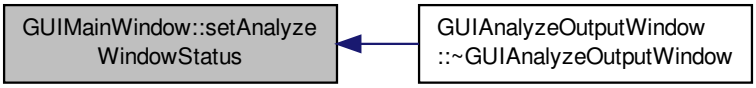
Setzt den Status des Übersichstfensters über die Analysedaten.

Parameter

isValid	Ob das Fenster ein gültiges Objekt oder ob der Speicher bereits freigegeben ist.

Definiert in Zeile 278 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.14.3.36 void GUIMainWindow::setCutRenderWindowStatus (bool *isValid*)

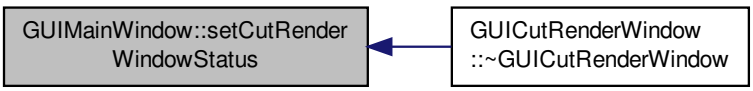
Setzt den Status des Übersichstfensters über die Analysedaten.

Parameter

isValid	Ob das Fenster ein gültiges Objekt oder ob der Speicher bereits freigegeben ist.

Definiert in Zeile 282 der Datei GUIMainWindow.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

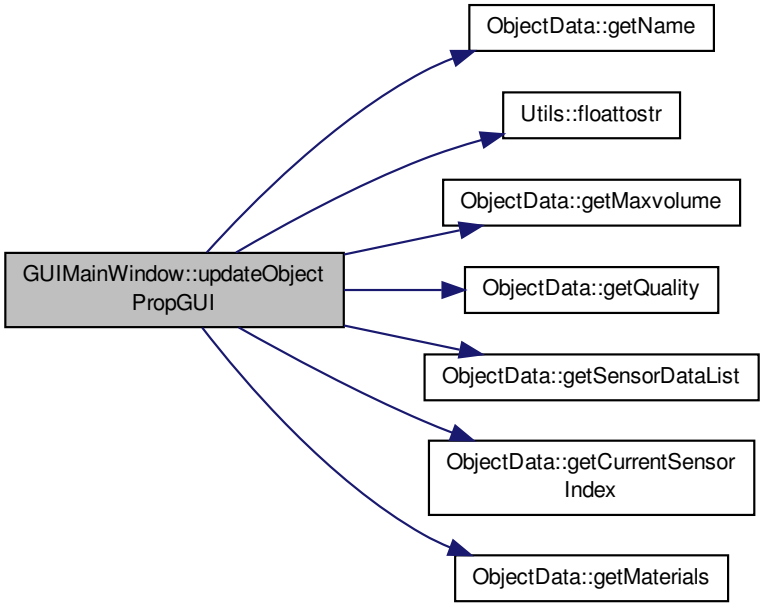


8.14.3.37 void GUIMainWindow::updateObjectPropGUI ( ) [private]

Überträgt die Eigenschaften des aktiven Objekts in die GUI.

Definiert in Zeile 514 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

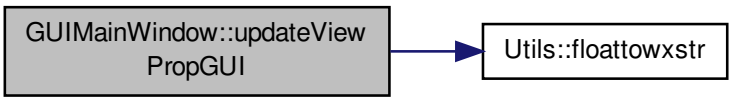


8.14.3.38 void GUIMainWindow::updateViewPropGUI ( ) [private]

Lädt die Visualisierungsoptionen in die GUI.

Definiert in Zeile 725 der Datei GUIMainWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.14.4 Dokumentation der Datenelemente

8.14.4.1 bool GUIMainWindow::analyze_window_valid [private]

Das Analysedaten-Übersichtsfenster ist gerade geöffnet.

Definiert in Zeile 348 der Datei GUIMainWindow.h.

8.14.4.2 GUIAnalyzeOutputWindow* GUIMainWindow::analyzerframe [private]

Das Analysedaten-Übersichtsfenster.

Der Zeiger ist ungültig, wenn das Analysedaten-Übersichtsfenster nicht geöffnet ist. (siehe analyze_window_valid)

Definiert in Zeile 322 der Datei GUIMainWindow.h.

8.14.4.3 string GUIMainWindow::configpaths[NUMBEROFPATHS] [protected]

Initialisierung:

```
{
            "/usr/local/share/simpleanalyzer/",
            "/usr/share/simpleanalyzer/",
    }
```

Suchpfade für die Anwendungsdaten.

Das Verzeichnis der ausführbaren Datei wird immer und zuerst geprüft.

Definiert in Zeile 72 der Datei GUIMainWindow.h.

8.14.4.4 string GUIMainWindow::data_directory [private]

Der Pfad zum Verzeichnis, das die von der Anwendung verwendeten Daten (z.B.

Icons) enthält. Wird im Konstruktor bestimmt.

Definiert in Zeile 358 der Datei GUIMainWindow.h.

8.14.4.5 GUIGLCanvas* GUIMainWindow::gl_context [private]

Die Zeichenfläche für das 3D-Fenster.

Definiert in Zeile 266 der Datei GUIMainWindow.h.

8.14.4.6 wxMenu* GUIMainWindow::mwAnalyzeMenu [private]

Das "Analysieren"-Untermenü.

Definiert in Zeile 301 der Datei GUIMainWindow.h.

8.14.4.7 wxMenu* GUIMainWindow::mwEditMenu [private]

Das "Bearbeiten"-Untermenü

Definiert in Zeile 306 der Datei GUIMainWindow.h.

8.14.4.8 wxMenu* GUIMainWindow::mwExportMenu [private]

Das "Export"-Untermenü.

Definiert in Zeile 296 der Datei GUIMainWindow.h.

8.14.4.9 wxMenu* GUIMainWindow::mwFileMenu [private]

Das "Datei"-Untermenü.

Definiert in Zeile 281 der Datei GUIMainWindow.h.

8.14.4.10 wxMenu* GUIMainWindow::mwHelpMenu [private]

Das "Hilfe"-Untermenü.

Definiert in Zeile 286 der Datei GUIMainWindow.h.

8.14.4.11 wxMenu* GUIMainWindow::mwImportMenu [private]

Das "Import"-Untermenü.

Definiert in Zeile 291 der Datei GUIMainWindow.h.

8.14.4.12 wxMenuBar* GUIMainWindow::mwMenuBar [private]

Die Hauptmenükomponente.

Definiert in Zeile 276 der Datei GUIMainWindow.h.

8.14.4.13 const int GUIMainWindow::NUMBEROFPATHS = 2 [static], [protected]

Anzahl der Suchpfade für die Anwendungsdaten (z.B.

Icons).

Definiert in Zeile 65 der Datei GUIMainWindow.h.

8.14.4.14 wxScrolledWindow* GUIMainWindow::prop_scroll_win [private]

Scrollender Bereich, in den die Objekteigenschaften-Oberfläche eingebettet ist.

Definiert in Zeile 333 der Datei GUIMainWindow.h.

8.14.4.15 PropertiesBox* GUIMainWindow::propbox [private]

Die Unterkomponente, die die Objekteigenschaften-Oberfläche enthält.

Definiert in Zeile 311 der Datei GUIMainWindow.h.

8.14.4.16 bool GUIMainWindow::render_cut_window_valid [private]

Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung ist gerade geöffnet.

Definiert in Zeile 353 der Datei GUIMainWindow.h.

8.14.4.17 GUICutRenderWindow* GUIMainWindow::rendercutwindow [private]

Das Fenster zur Berechnung einer zweidimensionalen Temperaturverteilung.

Der Zeiger ist ungültig, wenn das 2D-Fenster nicht geöffnet ist. (siehe render_cut_window_valid)

Definiert in Zeile 328 der Datei GUIMainWindow.h.

8.14.4.18 wxToolBar* GUIMainWindow::toolbar [private]

Die Tollbarkomponente.

Definiert in Zeile 271 der Datei GUIMainWindow.h.

8.14.4.19 bool GUIMainWindow::updating [private]

Die Oberfläche wird gerade vom Programm verändert.

Signalisiert, dass die Eingabe nicht durch den Nutzer erfolgt ist.

Definiert in Zeile 343 der Datei GUIMainWindow.h.

8.14.4.20 wxScrolledWindow* GUIMainWindow::view_scroll_win [private]

Scrollender Bereich, in den die Visualisierungsoptionen-Oberfläche eingebettet ist.

Definiert in Zeile 338 der Datei GUIMainWindow.h.

8.14.4.21 ViewpropBox* GUIMainWindow::viewbox [private]

Die Unterkomponente, die die Visualisierungsoptionen-Oberfläche enthält.

Definiert in Zeile 316 der Datei GUIMainWindow.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

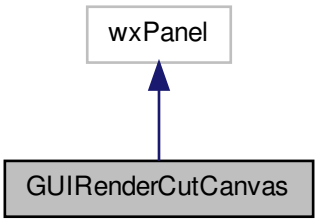
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.cpp

8.15 GUIRenderCutCanvas Klassenreferenz

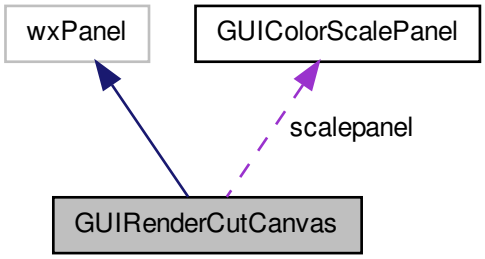
Zeichenfläche für die 2D-Temperaturverteilung.

`#include <GUIRenderCutCanvas.h>`

Klassendiagramm für GUIRenderCutCanvas:



Zusammengehörigkeiten von GUIRenderCutCanvas:



Öffentliche Methoden

- GUIRenderCutCanvas (wxWindow ∗parent)

  Der Konstruktor.
- void setImage (wxImage ∗img)

  Setzt die aktuell angezeigte Grafik.
- void setValueImg (float ∗img)

  Setzt die zum anzeigen von werten verwendete Temperaturverteilung.
- GUIColorScalePanel ∗ getScalePanel ()

  Gibt das Temperaturskala-Objekt zurück.
- virtual ∼GUIRenderCutCanvas ()

  Der Destruktor.

Private Methoden

- void onCanvasPaint (wxPaintEvent &event)

  Event-Tabellendeklaration für wxWidgets.
- void OnMouseWheel (wxMouseEvent &event)

  Behandelt das Zoomen in der Grafik.
- void OnMouseMove (wxMouseEvent &event)

  Behandelt das verschieben der Ansicht und speichert die Mauszeigerposition zur Ermittlung des Wertes an dieser Stelle in onCanvasPaint().
- void OnResize (wxSizeEvent &event)

  Behandelt Größenänderungen der Zeichenfläche.
- void OnMouseDown (wxMouseEvent &event)

  Behandelt klicken mit der Maus, deren Status zum verschieben der Ansicht benötigt wird.

Private Attribute

- float zoom

  Der aktuelle Zoomfaktor für die Zeichenfläche.
- float deltaX

  horizontale Verschiebung der Ansicht.
- float deltaY

vertikale Verschiebung der Ansicht.

- int current_mx

  Zwischenspeicher für die horizontale Mausposition.

- int current_my

  Zwischenspeicher für die vertikale Mausposition.

- bool mouse_to_scalepanel

  Müssen die Mausaktionen zur Skala weitergeleitet werden? (Wird diese gerade Transformiert?)

- wxImage * image

  Die aktuelle dargestellte Temperaturverteilung als Grafik.

- float * value_img

  Die aktuelle dargestellte Temperaturverteilung.

- GUIColorScalePanel * scalepanel

  Die Temperaturskala.

8.15.1 Ausführliche Beschreibung

Zeichenfläche für die 2D-Temperaturverteilung.

Zeichenfläche für das Fenster zur Berechnung einer 2D-Temperaturverteilung. Zeigt die berechnete Grafik, Skala und eine Statusleiste an. Verwaltet auch Mauseingaben zum Verschieben und Zoomen der Ansicht.

Definiert in Zeile 20 der Datei GUIRenderCutCanvas.h.

8.15.2 Beschreibung der Konstruktoren und Destruktoren

8.15.2.1 GUIRenderCutCanvas::GUIRenderCutCanvas (wxWindow * *parent*)

Der Konstruktor.

Parameter

parent	Das Fenster, auf dem die Zeichenfläche liegen soll.

Definiert in Zeile 29 der Datei GUIRenderCutCanvas.cpp.

8.15.2.2 GUIRenderCutCanvas::~GUIRenderCutCanvas ( ) [virtual]

Der Destruktor.

Definiert in Zeile 230 der Datei GUIRenderCutCanvas.cpp.

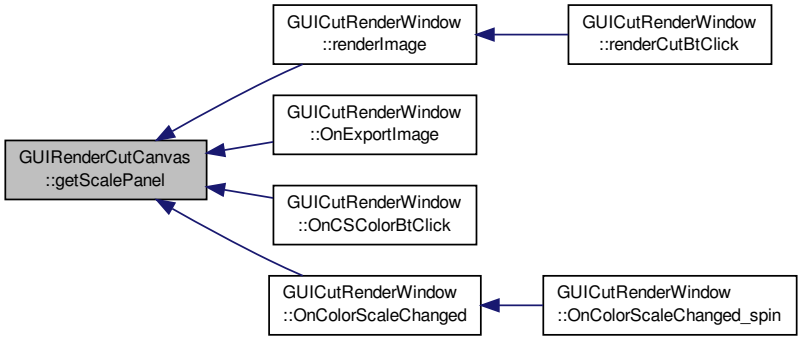
8.15.3 Dokumentation der Elementfunktionen

8.15.3.1 GUIColorScalePanel * GUIRenderCutCanvas::getScalePanel ( )

Gibt das Temperaturskala-Objekt zurück.

Definiert in Zeile 226 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



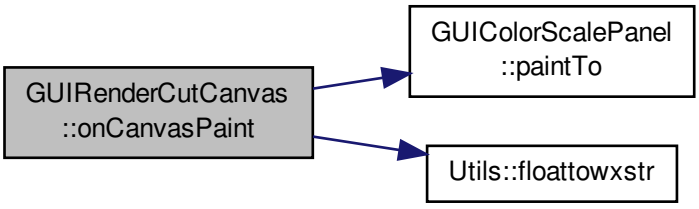
8.15.3.2 void GUIRenderCutCanvas::onCanvasPaint (wxPaintEvent & *event*) [private]

Event-Tabellendeklaration für wxWidgets.

Zeichnet die Temperaturverteilung und die Anzeigeelemente (Informationsleiste, Skala).

Definiert in Zeile 130 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

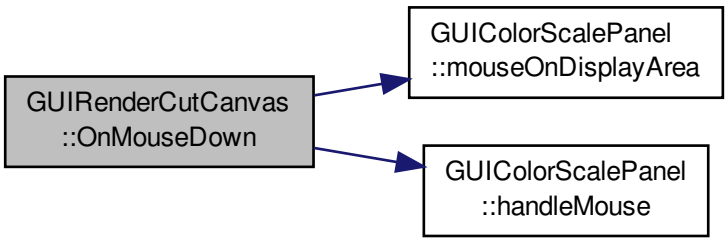


8.15.3.3 void GUIRenderCutCanvas::OnMouseDown (wxMouseEvent & *event*) [private]

Behandelt klicken mit der Maus, deren Status zum verschieben der Ansicht benötigt wird.

Definiert in Zeile 108 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

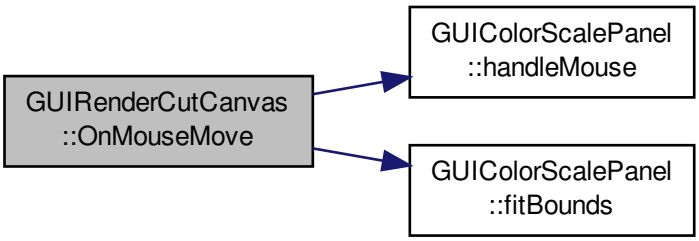


8.15.3.4 void GUIRenderCutCanvas::OnMouseMove (wxMouseEvent & *event*) [private]

Behandelt das verschieben der Ansicht und speichert die Mauszeigerposition zur Ermittlung des Wertes an dieser Stelle in onCanvasPaint().

Definiert in Zeile 65 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.15.3.5 void GUIRenderCutCanvas::OnMouseWheel (wxMouseEvent & *event*) [private]

Behandelt das Zoomen in der Grafik.

Definiert in Zeile 55 der Datei GUIRenderCutCanvas.cpp.

8.15.3.6 void GUIRenderCutCanvas::OnResize (wxSizeEvent & *event*) [private]

Behandelt Größenänderungen der Zeichenfläche.

Definiert in Zeile 104 der Datei GUIRenderCutCanvas.cpp.

8.15.3.7 void GUIRenderCutCanvas::setImage (wxImage * *img*)

Setzt die aktuell angezeigte Grafik.

Definiert in Zeile 47 der Datei GUIRenderCutCanvas.cpp.

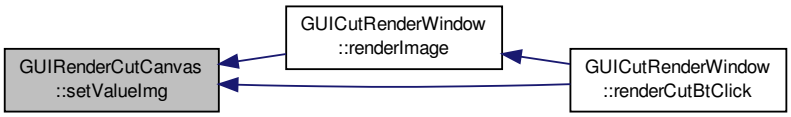
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.15.3.8 void GUIRenderCutCanvas::setValueImg (float * *img*)

Setzt die zum anzeigen von werten verwendete Temperaturverteilung.

Definiert in Zeile 51 der Datei GUIRenderCutCanvas.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.15.4 Dokumentation der Datenelemente

8.15.4.1 int GUIRenderCutCanvas::current_mx [private]

Zwischenspeicher für die horizontale Mausposition.

Definiert in Zeile 97 der Datei GUIRenderCutCanvas.h.

8.15.4.2 int GUIRenderCutCanvas::current_my [private]

Zwischenspeicher für die vertikale Mausposition.

Definiert in Zeile 102 der Datei GUIRenderCutCanvas.h.

8.15.4.3 float GUIRenderCutCanvas::deltaX [private]

horizontale Verschiebung der Ansicht.

Definiert in Zeile 87 der Datei GUIRenderCutCanvas.h.

8.15.4.4 float GUIRenderCutCanvas::deltaY [private]

vertikale Verschiebung der Ansicht.

Definiert in Zeile 92 der Datei GUIRenderCutCanvas.h.

8.15.4.5 wxImage* GUIRenderCutCanvas::image [private]

Die aktuelle dargestellte Temperaturverteilung als Grafik.

Definiert in Zeile 112 der Datei GUIRenderCutCanvas.h.

8.15.4.6 bool GUIRenderCutCanvas::mouse_to_scalepanel [private]

Müssen die Mausaktionen zur Skala weitergeleitet werden? (Wird diese gerade Transformiert?)

Definiert in Zeile 107 der Datei GUIRenderCutCanvas.h.

8.15.4.7 GUIColorScalePanel* GUIRenderCutCanvas::scalepanel [private]

Die Temperaturskala.

Definiert in Zeile 122 der Datei GUIRenderCutCanvas.h.

8.15.4.8 float* GUIRenderCutCanvas::value_img [private]

Die aktuelle dargestellte Temperaturverteilung.

Definiert in Zeile 117 der Datei GUIRenderCutCanvas.h.

8.15.4.9 float GUIRenderCutCanvas::zoom [private]

Der aktuelle Zoomfaktor für die Zeichenfläche.

Definiert in Zeile 82 der Datei GUIRenderCutCanvas.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

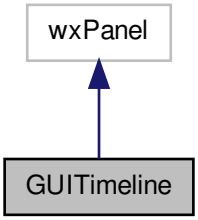
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCutCanvas.cpp

8.16 GUITimeline Klassenreferenz

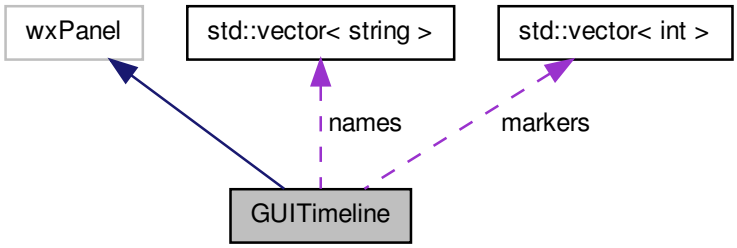
Eine Zeitleistenkomponente.

`#include <GUITimeline.h>`

Klassendiagramm für GUITimeline:



Zusammengehörigkeiten von GUITimeline:



Öffentliche Typen

- enum GUI_TIMELINE_STYLE { GTL_DEFAULT = 0 }

  Darstellungsstil der Zeitleiste.

Öffentliche Methoden

- GUITimeline (wxWindow *parent, wxWindowID id, const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxDefaultSize, long style=GTL_DEFAULT, const wxString &name=wxT("Timeline"))

  Der Konstruktor.
- void findMaxValue (ObjectData *obj, bool fast)

  Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wäremeenergie-gehalt maximal wird.
- int getValue ()

  gibt den Index des aktuell ausgewählten Zeitpunkts zurück.
- int getMaxValue ()

  Gibt den maximal auswählbaren Index zurück.
- int getMinValue ()

  Gibt den minimal auswählbaren Index zurück.

- void setValue (int val)

  Setzt den Index des aktuell ausgewählten Zeitpunkts.
- void setMaxValue (int val)

  Setzt den maximal auswählbaren Index.
- void setMinValue (int val)

  Setzt den minimal auswählbaren Index.
- void setNameList (vector< string > *namelist)

  Setzt die Liste der Namen für die jeweiligen Indices der Zeitpunkte.
- void setMarked (int pos, bool state)

  Markiert/Demarkiert einen bestimmten Zeitpunkt.
- bool isMarked (int pos)

  Gibt zurück, ob ein Zeitpunkt markiert ist.
- void clearMarkers ()

  Entfernt alle Markierungen.
- void setMarkerList (vector< int > *mlist)

  Setzt die Liste der markierten Stellen.
- vector< int > * getMarkers ()

  Gibt die Liste der markierten Stellen zurück.
- void setMarkers (vector< int > *mlist)

  Markiert eine Liste von Indices.
- virtual ~GUITimeline ()

  Der Destruktor.

Private Methoden

- void OnPaint (wxPaintEvent &)

  Event-Tabellendeklaration für wxWidgets.
- void OnMouseWheel (wxMouseEvent &event)

  Behandellt Scrolleingaben (zoomen).
- void OnMouseMove (wxMouseEvent &event)

  Behandelt Mausbewegungen (verschieben der Ansicht).
- void OnResize (wxSizeEvent &event)

  Behandelt Größenänderungen der Zeitleiste.
- void OnMouseDown (wxMouseEvent &event)

  Behandelt klicken (verschieben der Ansicht, setzten des aktuellen Zeitpunkts).
- void OnKeyDown (wxKeyEvent &event)

  Behandelt Tastendruck (setzen des aktuellen Zeitpunkts).
- void posToVal (int mouse_x)

  Setzt den aktuellen Zeitpunkt anhand der Mausposition.
- void sendTimelineEvent ()

  Löst ein wxEVT_TIMELINE_CHANGE-Event aus.
- int calcStepWidth ()

  Berechnet die für die aktuelle Darstellung günstigste Schrittweite für die Beschriftung.

Private Attribute

- int value

  Der Index des aktuell ausgewählten Zeitpunkts.
- int maxvalue

  Der größte anzuzeigende Zeitpunkt.
- int minvalue

  Der kleinste anzuzeigende Zeitpunkt.
- int maxdigits

  Maximale Anzahl an anzuzeigenden Nachkommastellen.
- float zoom

  Aktueller Zoomfaktor.
- float delta_v_view

  Verschiebung der Ansicht.
- int prev_mouse_x

  Zwischenspeicher für die vorherige horizontale Mausposition.
- vector< string > * names

  Liste der Zeitpunktnamen.
- vector< int > * markers

  Liste der markierten Zeitpunkte.

8.16.1 Ausführliche Beschreibung

Eine Zeitleistenkomponente.

Die Komponente kann Zeitpunkte als Zeitleiste darstellen, wobei die Zeitpunkte anhand von Indices ausgewählt werden können. Zusätzlich kann eine Liste von Namen für die Zeitpunkte festgelegt werden, wodurch auch der Name des gewählten Zeitpunkts angezeigt wird. Weiterhin können Zeitpunkte markiert werden.

Definiert in Zeile 32 der Datei GUITimeline.h.

8.16.2 Dokumentation der Aufzählungstypen

8.16.2.1 enum GUITimeline::GUI_TIMELINE_STYLE

Darstellungsstil der Zeitleiste.

Aufzählungswerte

GTL_DEFAULT

Definiert in Zeile 37 der Datei GUITimeline.h.

8.16.3 Beschreibung der Konstruktoren und Destruktoren

8.16.3.1 GUITimeline::GUITimeline (wxWindow * *parent*, wxWindowID *id*, const wxPoint & *pos* = `wxDefaultPosition`, const wxSize & *size* = `wxDefaultSize`, long *style* = GTL_DEFAULT, const wxString & *name* = `wxT("Timeline")`)

Der Konstruktor.

Parameter

parent	Die übergeordnete Komponente.
id	Die ID des Objekts.
pos	Die Position der Zeitleiste.
size	Die Größe der Zeitleiste.
style	Darstellungsstil der Zeitleiste.
name	Name der Zeitleiste (Komponentenname, nicht sichtbar).

Definiert in Zeile 36 der Datei GUITimeline.cpp.

8.16.3.2 GUITimeline::~GUITimeline ( ) [virtual]

Der Destruktor.

Definiert in Zeile 543 der Datei GUITimeline.cpp.

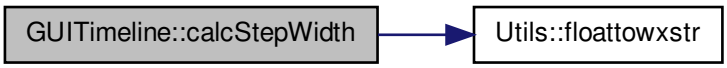
8.16.4 Dokumentation der Elementfunktionen

8.16.4.1 int GUITimeline::calcStepWidth ( ) [private]

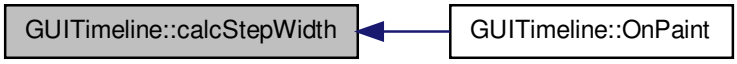
Berechnet die für die aktuelle Darstellung günstigste Schrittweite für die Beschriftung.

Definiert in Zeile 211 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.16.4.2 void GUITimeline::clearMarkers ( )

Entfernt alle Markierungen.

Definiert in Zeile 512 der Datei GUITimeline.cpp.

8.16.4.3 void GUITimeline::findMaxValue (ObjectData * *obj,* bool *fast*)

Sucht den Zeitpunkt zwischen zwei markierten Stellen auf der Sensordaten-Zeitleiste, für den der Wäremeenergiegehalt maximal wird.

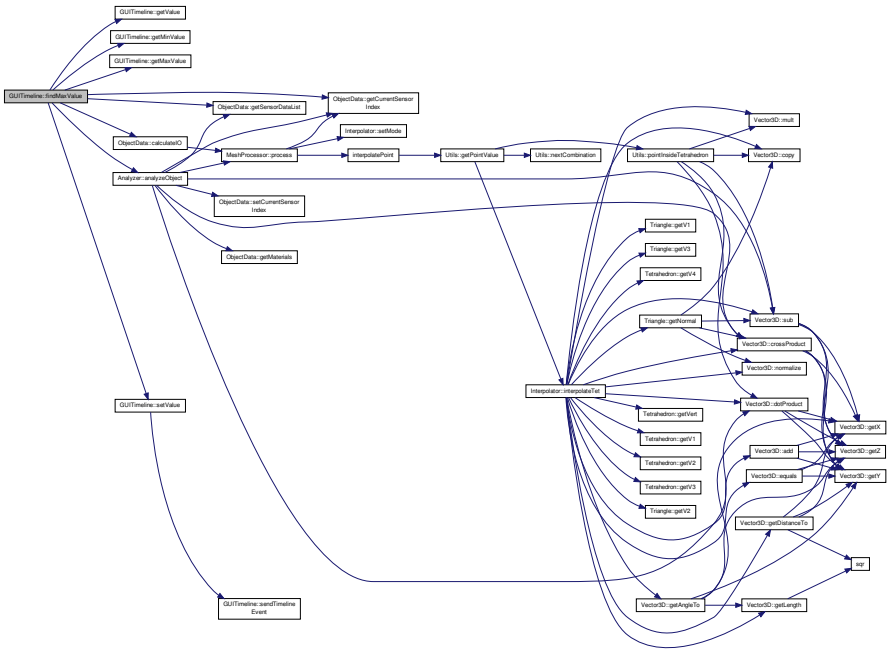
Dabei wird der Bereich zwischen den beiden markierten Stellen ausgewählt, zwischen denen sich der aktuell ausgewählte Zeitpunkt befindet.

Parameter

obj	Das zu untersuchende Objekt.
fast	Schnelle Methode verwenden. D.h., es wird statt der Temperaturverteilung nur die Durchschnittstemperatur verglichen.

Definiert in Zeile 112 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.16.4.4 vector< int > * GUITimeline::getMarkers ( )

Gibt die Liste der markierten Stellen zurück.

Definiert in Zeile 517 der Datei GUITimeline.cpp.

8.16.4.5 int GUITimeline::getMaxValue ( )

Gibt den maximal auswählbaren Index zurück.

Definiert in Zeile 431 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.16.4.6 int GUITimeline::getMinValue ( )

Gibt den minimal auswählbaren Index zurück.

Definiert in Zeile 435 der Datei GUITimeline.cpp.

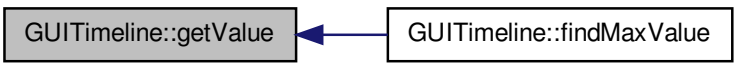
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.16.4.7 int GUITimeline::getValue ( )

gibt den Index des aktuell ausgewählten Zeitpunkts zurück.

Definiert in Zeile 427 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.16.4.8 bool GUITimeline::isMarked (int *pos*)

Gibt zurück, ob ein Zeitpunkt markiert ist.

Parameter

pos	Index des Zeitpunkts.

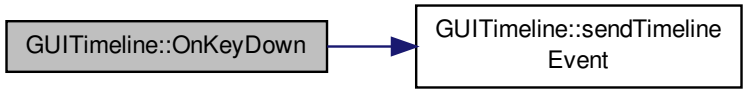
Definiert in Zeile 501 der Datei GUITimeline.cpp.

8.16.4.9 void GUITimeline::OnKeyDown (wxKeyEvent & *event*) [private]

Behandelt Tastendruck (setzen des aktuellen Zeitpunkts).

Definiert in Zeile 85 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.16.4.10 void GUITimeline::OnMouseDown (wxMouseEvent & *event*) [private]

Behandelt klicken (verschieben der Ansicht, setzten des aktuellen Zeitpunkts).

Definiert in Zeile 79 der Datei GUITimeline.cpp.

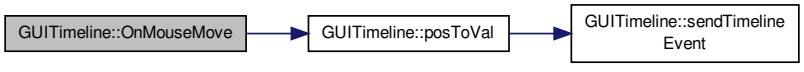
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

8.16.4.11 void GUITimeline::OnMouseMove (wxMouseEvent & *event*) [private]

Behandelt Mausbewegungen (verschieben der Ansicht).

Definiert in Zeile 278 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.16.4.12 void GUITimeline::OnMouseWheel (wxMouseEvent & *event*) [private]

Behandellt Scrolleingaben (zoomen).

Definiert in Zeile 59 der Datei GUITimeline.cpp.

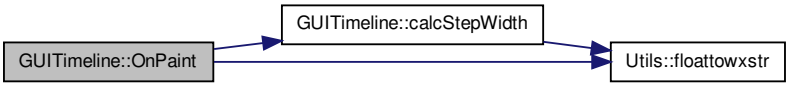
8.16.4.13 void GUITimeline::OnPaint (wxPaintEvent &) [private]

Event-Tabellendeklaration für wxWidgets.

Zeichnet die Zeitleiste.

Definiert in Zeile 307 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.16.4.14 void GUITimeline::OnResize (wxSizeEvent & *event*) [private]

Behandelt Größenänderungen der Zeitleiste.

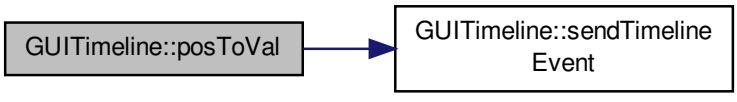
Definiert in Zeile 303 der Datei GUITimeline.cpp.

8.16.4.15 void GUITimeline::posToVal (int *mouse_x*) [private]

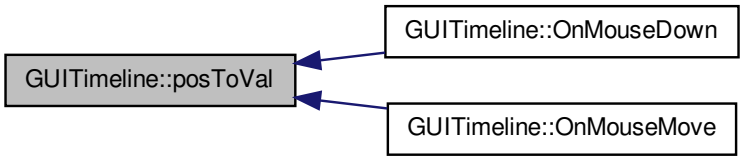
Setzt den aktuellen Zeitpunkt anhand der Mausposition.

Definiert in Zeile 254 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

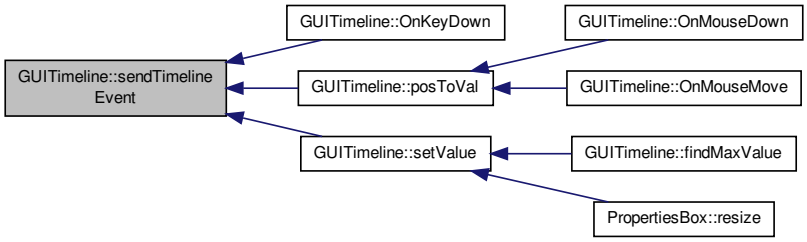


8.16.4.16 void GUITimeline::sendTimelineEvent ( ) [private]

Löst ein wxEVT_TIMELINE_CHANGE-Event aus.

Definiert in Zeile 52 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.16.4.17 void GUITimeline::setMarked (int *pos*, bool *state*)

Markiert/Demarkiert einen bestimmten Zeitpunkt.

Parameter

pos	Index des zu setzenden Zeitpunkts.
state	Status des Punktes (markiert - true, nicht markiert - false).

Definiert in Zeile 476 der Datei GUITimeline.cpp.

8.16.4.18 void GUITimeline::setMarkerList (vector< int > * *mlist*)

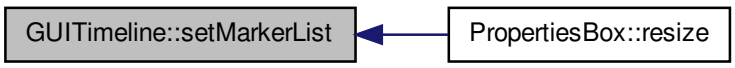
Setzt die Liste der markierten Stellen.

Parameter

mlist	Liste mit den Indices der markierten stellen.

Definiert in Zeile 471 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.16.4.19 void GUITimeline::setMarkers (vector< int > * *mlist*)

Markiert eine Liste von Indices.

Parameter

mlist	Liste aller zu markierenden Indices.

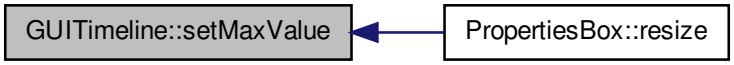
Definiert in Zeile 521 der Datei GUITimeline.cpp.

8.16.4.20 void GUITimeline::setMaxValue (int *val*)

Setzt den maximal auswählbaren Index.

Definiert in Zeile 458 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

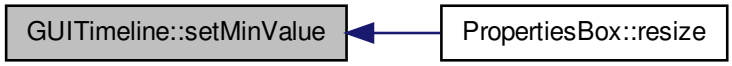


8.16.4.21 void GUITimeline::setMinValue (int *val*)

Setzt den minimal auswählbaren Index.

Definiert in Zeile 462 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.16.4.22 void GUITimeline::setNameList (vector< string > * *namelist*)

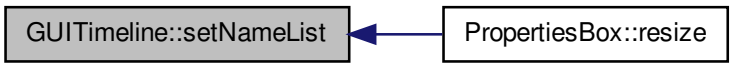
Setzt die Liste der Namen für die jeweiligen Indices der Zeitpunkte.

Parameter

namelist	Liste mit einem Namen für jeden Index.

Definiert in Zeile 466 der Datei GUITimeline.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

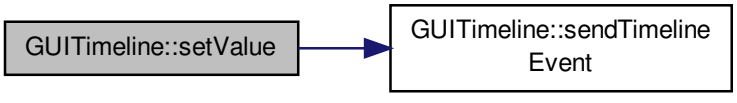


8.16.4.23 void GUITimeline::setValue (int *val*)

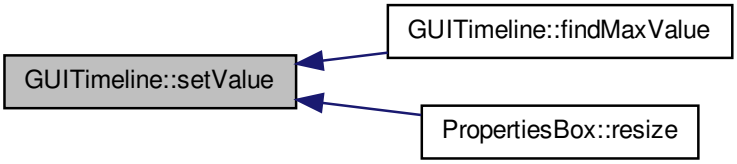
Setzt den Index des aktuell ausgewählten Zeitpunkts.

Definiert in Zeile 439 der Datei GUITimeline.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.16.5 Dokumentation der Datenelemente

8.16.5.1 float GUITimeline::delta_v_view [private]

Verschiebung der Ansicht.

Definiert in Zeile 217 der Datei GUITimeline.h.

8.16.5.2 vector<int>* GUITimeline::markers [private]

Liste der markierten Zeitpunkte.

Definiert in Zeile 232 der Datei GUITimeline.h.

8.16.5.3 int GUITimeline::maxdigits [private]

Maximale Anzahl an anzuzeigenden Nachkommastellen.

Definiert in Zeile 207 der Datei GUITimeline.h.

8.16.5.4 int GUITimeline::maxvalue [private]

Der größte anzuzeigende Zeitpunkt.

Definiert in Zeile 197 der Datei GUITimeline.h.

8.16.5.5 int GUITimeline::minvalue [private]

Der kleinste anzuzeigende Zeitpunkt.

Definiert in Zeile 202 der Datei GUITimeline.h.

8.16.5.6 vector<string>* GUITimeline::names [private]

Liste der Zeitpunktnamen.

Definiert in Zeile 227 der Datei GUITimeline.h.

8.16.5.7 int GUITimeline::prev_mouse_x [private]

Zwischenspeicher für die vorherige horizontale Mausposition.

Definiert in Zeile 222 der Datei GUITimeline.h.

8.16.5.8 int GUITimeline::value [private]

Der Index des aktuell ausgewählten Zeitpunkts.

Definiert in Zeile 192 der Datei GUITimeline.h.

8.16.5.9 float GUITimeline::zoom [private]

Aktueller Zoomfaktor.

Definiert in Zeile 212 der Datei GUITimeline.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.cpp

8.17 Importer Klassenreferenz

Importieren von 3D-Modell (.obj) und Sensordaten (.tsd oder .sd).

`#include <Importer.h>`

Öffentliche Methoden

- Importer ()

  Der Konstruktor.
- ObjectData::ObjectDataStatus LoadSensorData (string filename, ObjectData ∗data)

  Lädt von einfache Sensordaten (ohne Zeit) und Verknüpft diese mit dem Objekt.
- ObjectData::ObjectDataStatus LoadTimedData (string filename, ObjectData ∗data)

  Lädt zeitgesteuerte Sensordaten und Verknüpft diese mit dem Objekt.
- ObjectData::ObjectDataStatus ImportObj (string filename, ObjectData ∗data)

  Lädt Objektdaten aus einer .obj-Datei.
- virtual ∼Importer ()

  Der Destruktor.

8.17.1 Ausführliche Beschreibung

Importieren von 3D-Modell (.obj) und Sensordaten (.tsd oder .sd).

Definiert in Zeile 18 der Datei Importer.h.

8.17.2 Beschreibung der Konstruktoren und Destruktoren

8.17.2.1 Importer::Importer ( )

Der Konstruktor.

Definiert in Zeile 24 der Datei Importer.cpp.

8.17.2.2 Importer::~Importer ( ) [virtual]

Der Destruktor.

Definiert in Zeile 504 der Datei Importer.cpp.

8.17.3 Dokumentation der Elementfunktionen

8.17.3.1 ObjectData::ObjectDataStatus Importer::ImportObj (string *filename,* ObjectData * *data*)

Lädt Objektdaten aus einer .obj-Datei.

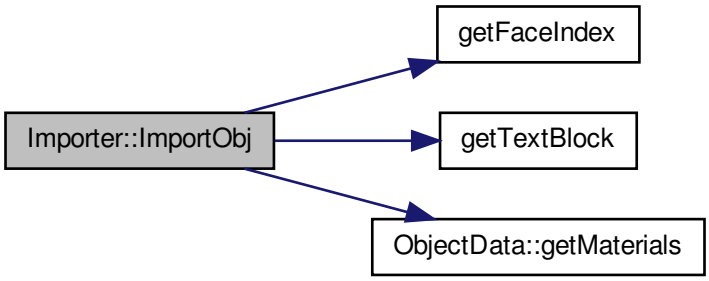
Das Objekt ist zwar schon im Speicher erstellt, wird aber erst durch diese Methode mit Daten gefüllt.

Rückgabe

Der Fehlercode.

Definiert in Zeile 83 der Datei Importer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.17.3.2 ObjectData::ObjectDataStatus Importer::LoadSensorData (string *filename,* ObjectData * *data*)

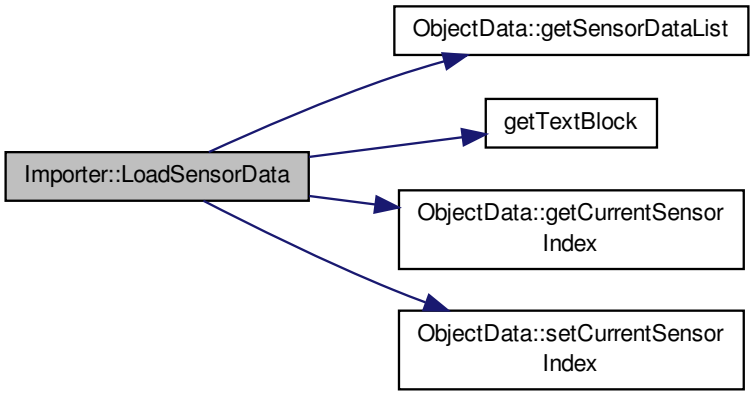
Lädt von einfache Sensordaten (ohne Zeit) und Verknüpft diese mit dem Objekt.

Rückgabe

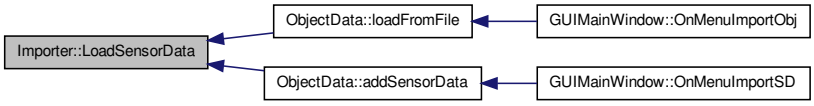
Der Fehlercode.

Definiert in Zeile 342 der Datei Importer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.17.3.3 ObjectData::ObjectDataStatus Importer::LoadTimedData (string *filename,* ObjectData * *data*)

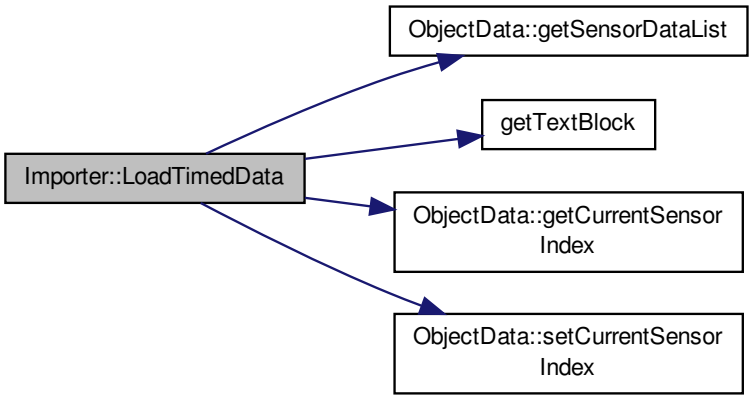
Lädt zeitgesteuerte Sensordaten und Verknüpft diese mit dem Objekt.

Rückgabe

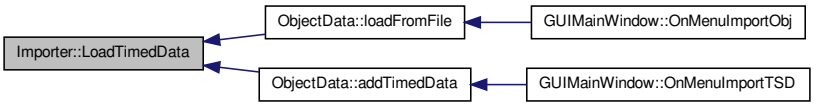
Der Fehlercode.

Definiert in Zeile 416 der Datei Importer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.cpp

8.18 Interpolator Klassenreferenz

2- und 3-dimensionale Inter-/Extrapolation

`#include <Interpolator.h>`

Öffentliche Typen

- enum InterpolationMode { LINEAR, LOGARITHMIC }

  Der Typ der verwendeten Interpolationsfunktion.

Öffentliche Methoden

- Interpolator ()

  Der Konstruktor.
- double interpolateTri (Triangle ∗tri, Vector3D ∗pos, double ∗values)

  Ermittelt den Wert für einen beliebigen Punkt in einer Ebene.
- double interpolateTet (Tetrahedron ∗tet, Vector3D ∗pos, double ∗values)

  Ermittelt den Wert für einen beliebigen Punkt im Raum.
- void setMode (InterpolationMode mode)

  Setzt den verwendeten Interpolationsmodus (die Interpolationsfunktion).
- virtual ∼Interpolator ()

  Der Destruktor.

Private Attribute

- InterpolationMode mode

  Der verwendete Interpolationsmodus bzw.

8.18.1 Ausführliche Beschreibung

2- und 3-dimensionale Inter-/Extrapolation

Klasse zur Bi- und Trilinearen Inter-/Extrapolation, wobei die Interpolationsfunktion zwischen zwei Werten entweder linear oder logarithmisch sein kann (InterpolationMode).

Definiert in Zeile 20 der Datei Interpolator.h.

8.18.2 Dokumentation der Aufzählungstypen

8.18.2.1 enum Interpolator::InterpolationMode

Der Typ der verwendeten Interpolationsfunktion.

Aufzählungswerte

LINEAR

LOGARITHMIC

Definiert in Zeile 25 der Datei Interpolator.h.

8.18.3 Beschreibung der Konstruktoren und Destruktoren

8.18.3.1 Interpolator::Interpolator ( )

Der Konstruktor.

Definiert in Zeile 12 der Datei Interpolator.cpp.

8.18.3.2 Interpolator::∼Interpolator ( ) [virtual]

Der Destruktor.

Definiert in Zeile 315 der Datei Interpolator.cpp.

8.18.4 Dokumentation der Elementfunktionen

8.18.4.1 double Interpolator::interpolateTet (Tetrahedron * *tet*, Vector3D * *pos*, double * *values*)

Ermittelt den Wert für einen beliebigen Punkt im Raum.

Dabei wird wie bei der bilinearen Interpolation (http://en.wikipedia.org/wiki/Trilinear_interpolation) vorgegangen, es kann jedoch auch eine logarithmische Interpolationsfunktion verwendet werden.

Parameter

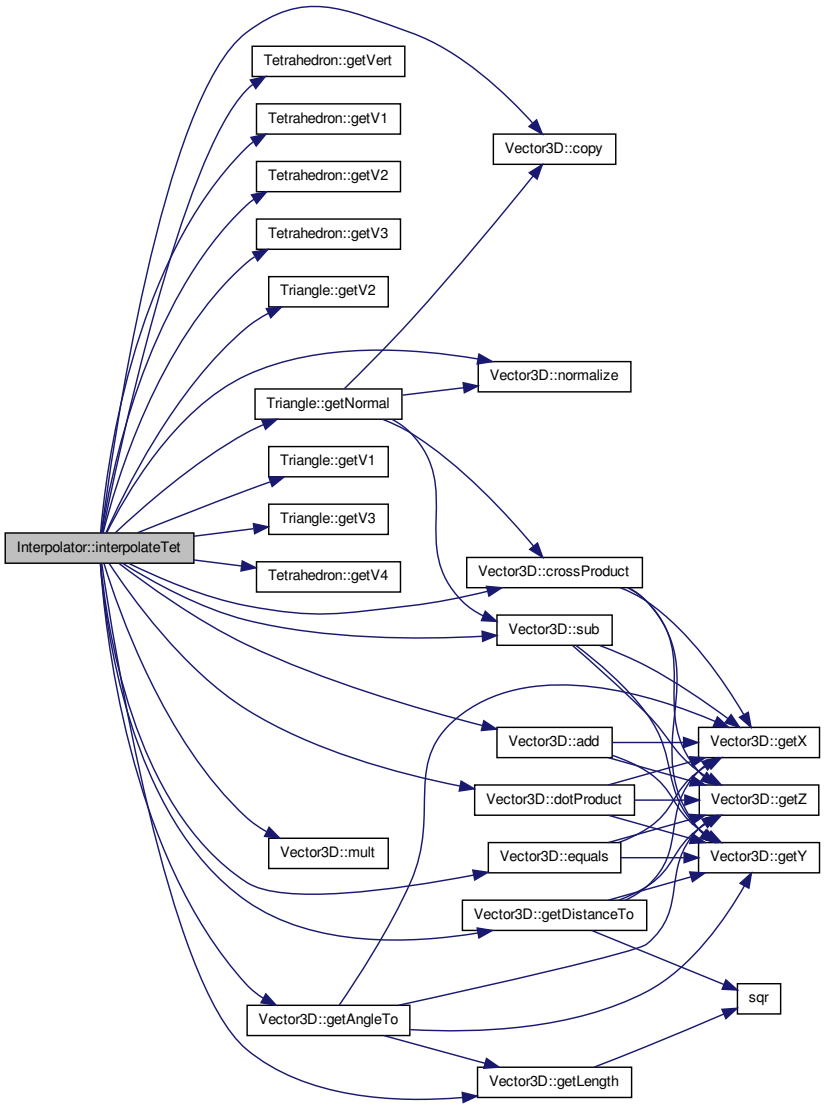
tet	Tetraeder, durch den die Punkte für die gegebenen Werte gegeben sind.
pos	Position des Punktes, für den der Wert ermittelt werden soll.
values	Die Werte, die den Punkten des Tetraeders entsprechen. Dabei ist values[0] der Wert des ersten Punktes des Tetraeders, values[0] der Zweite usw.

Rückgabe

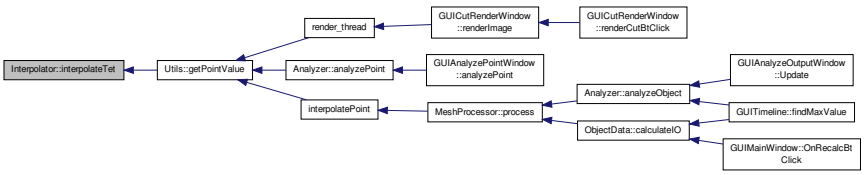
Der Wert für den angegebenen Punkt (pos).

Definiert in Zeile 149 der Datei Interpolator.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.18.4.2 double Interpolator::interpolateTri (Triangle * *tri*, Vector3D * *pos*, double * *values*)

Ermittelt den Wert für einen beliebigen Punkt in einer Ebene.

Dabei wird wie bei der bilinearen Interpolation (http://en.wikipedia.org/wiki/Bilinear_interpolation) vorgegangen, es kann jedoch auch eine logarithmische Interpolationsfunktion verwendet werden.

Parameter

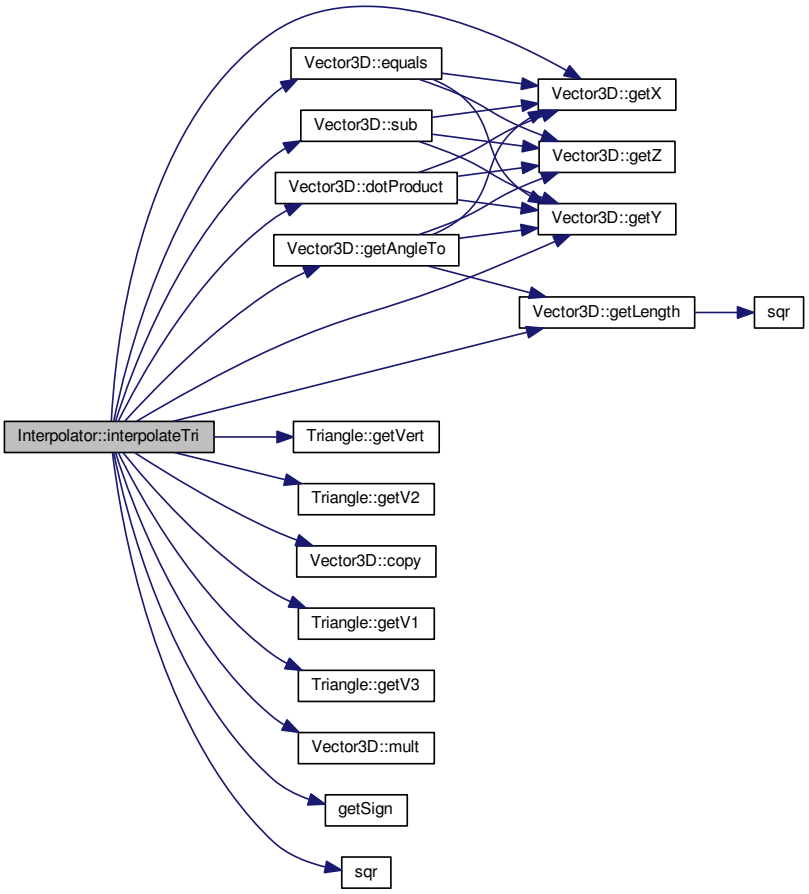
tri	Dreieck, durch das die Ebene beschrieben wird.
pos	Position des Punktes, für den der Wert ermittelt werden soll.
values	Die Werte, die den Punkten des Dreiecks entsprechen. Dabei ist values[0] der Wert des ersten Punktes des Dreiecks, values[0] der Zweite usw.

Rückgabe

Der Wert für den angegebenen Punkt (pos).

Definiert in Zeile 24 der Datei Interpolator.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

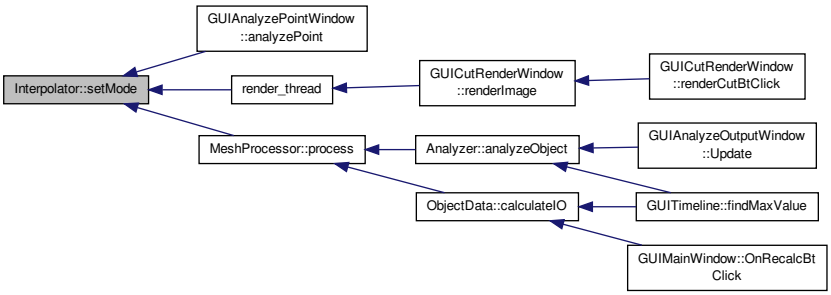


8.18.4.3 void Interpolator::setMode (InterpolationMode *mode*)

Setzt den verwendeten Interpolationsmodus (die Interpolationsfunktion).

Definiert in Zeile 16 der Datei Interpolator.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.18.5 Dokumentation der Datenelemente

8.18.5.1 InterpolationMode Interpolator::mode [private]

Der verwendete Interpolationsmodus bzw.

die Interpolationsfunktion.

Definiert in Zeile 69 der Datei Interpolator.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/Interpolator.cpp

8.19 ObjectData::MaterialData Strukturreferenz

Die Daten eines Materials.

`#include <ObjectData.h>`

Öffentliche Attribute

- string name

  Der Name des Materials.
- Interpolator::InterpolationMode interpolation_mode

  Der zu verwendende Interpolationsmodus.
- tetgenio * tetgeninput

  Originalgeometrie im Tetgen-Format (s.
- tetgenio * tetgenoutput

  Durch Zerlegung erstellte Geometrie im Tetgen-Format (s.
- bool * extrapolated

  Liste, die für jeden Punkt in der aktuellen Geometrie angibt, ob er extra- (true) oder interpoliert (false) ist.
- float color [3]

  Die Farbe des Materials im RGB-Format.
- double density

  Die Dichte in $\frac{kg}{m^3}$.
- double specificheatcapacity

Spezifische Wärmekapazität in $\frac{kJ}{kg*K}$.

- bool visible

  Soll das Material angezeigt werden?

8.19.1 Ausführliche Beschreibung

Die Daten eines Materials.

Definiert in Zeile 32 der Datei ObjectData.h.

8.19.2 Dokumentation der Datenelemente

8.19.2.1 float ObjectData::MaterialData::color[3]

Die Farbe des Materials im RGB-Format.

Definiert in Zeile 38 der Datei ObjectData.h.

8.19.2.2 double ObjectData::MaterialData::density

Die Dichte in $\frac{kg}{m^3}$.

Definiert in Zeile 39 der Datei ObjectData.h.

8.19.2.3 bool* ObjectData::MaterialData::extrapolated

Liste, die für jeden Punkt in der aktuellen Geometrie angibt, ob er extra- (true) oder interpoliert (false) ist.

Definiert in Zeile 37 der Datei ObjectData.h.

8.19.2.4 Interpolator::InterpolationMode ObjectData::MaterialData::interpolation_mode

Der zu verwendende Interpolationsmodus.

Definiert in Zeile 34 der Datei ObjectData.h.

8.19.2.5 string ObjectData::MaterialData::name

Der Name des Materials.

Definiert in Zeile 33 der Datei ObjectData.h.

8.19.2.6 double ObjectData::MaterialData::specificheatcapacity

Spezifische Wärmekapazität in $\frac{kJ}{kg*K}$.

Definiert in Zeile 40 der Datei ObjectData.h.

8.19.2.7 tetgenio* ObjectData::MaterialData::tetgeninput

Originalgeometrie im Tetgen-Format (s.

Tetgen Dokumentation)

Definiert in Zeile 35 der Datei ObjectData.h.

8.19.2.8 tetgenio* ObjectData::MaterialData::tetgenoutput

Durch Zerlegung erstellte Geometrie im Tetgen-Format (s.

Tetgen Dokumentation)

Definiert in Zeile 36 der Datei ObjectData.h.

8.19.2.9 bool ObjectData::MaterialData::visible

Soll das Material angezeigt werden?

Definiert in Zeile 41 der Datei ObjectData.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.h

8.20 Matrix3D Klassenreferenz

3x3-Matrixklasse mit Operationen.

`#include <GeometryClasses.h>`

Öffentliche Methoden

- Matrix3D ()
  Der Standardkonstruktor.
- Matrix3D (double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3)
  Erzeugt eine Matrix mit den gegebenen Elementen.
- void mult (Matrix3D *other)
  Multipliziert die Matrix mit einer anderen Matrix.
- Vector3D * mult (Vector3D *other)
  Multipliziert die Matrix mit einem Vektor.
- void rotateX (double angle)
  Rotiert die Matrix um einen bestimmten Winkel auf der X-Achse.
- void rotateY (double angle)
  Rotiert die Matrix um einen bestimmten Winkel auf der Y-Achse.
- void rotateZ (double angle)
  Rotiert die Matrix um einen bestimmten Winkel auf der Z-Achse.
- void transpose ()
  Transponiert die Matrix.
- void print ()
  Gibt die Matrix auf dem cout-Stream aus.

Private Attribute

- double elements [9]
  Die Elemente der Matrix.

8.20.1 Ausführliche Beschreibung

3x3-Matrixklasse mit Operationen.

Definiert in Zeile 151 der Datei GeometryClasses.h.

8.20.2 Beschreibung der Konstruktoren und Destruktoren

8.20.2.1 Matrix3D::Matrix3D ( )

Der Standardkonstruktor.

Erzeugt eine Standardmatrix:

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

Definiert in Zeile 141 der Datei GeometryClasses.cpp.

8.20.2.2 Matrix3D::Matrix3D (double *x1,* double *y1,* double *z1,* double *x2,* double *y2,* double *z2,* double *x3,* double *y3,* double *z3*)

Erzeugt eine Matrix mit den gegebenen Elementen.

$$\begin{matrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{matrix}$$

Definiert in Zeile 153 der Datei GeometryClasses.cpp.

8.20.3 Dokumentation der Elementfunktionen

8.20.3.1 void Matrix3D::mult (Matrix3D * *other*)

Multipliziert die Matrix mit einer anderen Matrix.

Parameter

other	Die Matrix, mit der multipliziert werden soll.

Definiert in Zeile 168 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.20.3.2 Vector3D * Matrix3D::mult (Vector3D * *other*)

Multipliziert die Matrix mit einem Vektor.

Parameter

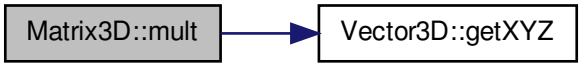
other	Der Vektor, mit dem multipliziert werden soll.

Rückgabe

Der durch die Multiplikation entstandene Vektor. Der zurückgegebene Vektor muss manuell mit delete Freigegeben werden!

Definiert in Zeile 186 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.20.3.3 void Matrix3D::print ( )

Gibt die Matrix auf dem cout-Stream aus.

Definiert in Zeile 230 der Datei GeometryClasses.cpp.

8.20.3.4 void Matrix3D::rotateX (double *angle*)

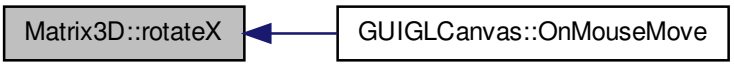
Rotiert die Matrix um einen bestimmten Winkel auf der X-Achse.

Parameter

angle	Der Winkel, um den rotiert werden soll in RAD.

Definiert in Zeile 197 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.20.3.5 void Matrix3D::rotateY (double *angle*)

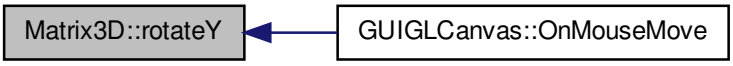
Rotiert die Matrix um einen bestimmten Winkel auf der Y-Achse.

Parameter

angle	Der Winkel, um den rotiert werden soll in RAD.

Definiert in Zeile 204 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.20.3.6 void Matrix3D::rotateZ (double *angle*)

Rotiert die Matrix um einen bestimmten Winkel auf der Z-Achse.

Parameter

angle	Der Winkel, um den rotiert werden soll in RAD.

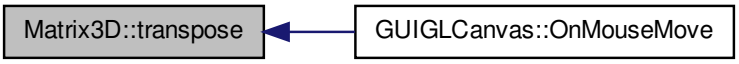
Definiert in Zeile 211 der Datei GeometryClasses.cpp.

8.20.3.7 void Matrix3D::transpose ( )

Transponiert die Matrix.

Definiert in Zeile 218 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.20.4 Dokumentation der Datenelemente

8.20.4.1 double Matrix3D::elements[9] [private]

Die Elemente der Matrix.

Definiert in Zeile 221 der Datei GeometryClasses.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp

8.21 MeshProcessor Klassenreferenz

Errechnet die Temperaturverteilung für ein Objekt.

`#include <MeshProcessor.h>`

Öffentliche Methoden

- MeshProcessor ()

  Der Konstruktor.

- void process (ObjectData *object)

  Berechnet die Temperaturverteilung für ein Objekt.

- virtual ~MeshProcessor ()

  Der Destruktor.

8.21.1 Ausführliche Beschreibung

Errechnet die Temperaturverteilung für ein Objekt.

Definiert in Zeile 16 der Datei MeshProcessor.h.

8.21.2 Beschreibung der Konstruktoren und Destruktoren

8.21.2.1 MeshProcessor::MeshProcessor ( )

Der Konstruktor.

Definiert in Zeile 17 der Datei MeshProcessor.cpp.

8.21.2.2 MeshProcessor::~MeshProcessor ( ) [virtual]

Der Destruktor.

Definiert in Zeile 85 der Datei MeshProcessor.cpp.

8.21.3 Dokumentation der Elementfunktionen

8.21.3.1 void MeshProcessor::process (ObjectData * *object*)

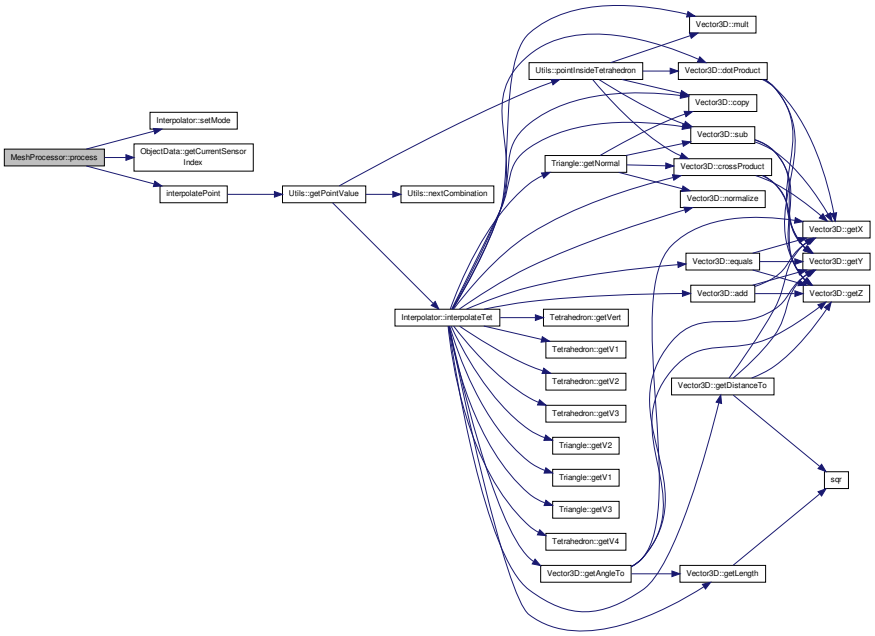
Berechnet die Temperaturverteilung für ein Objekt.

Parameter

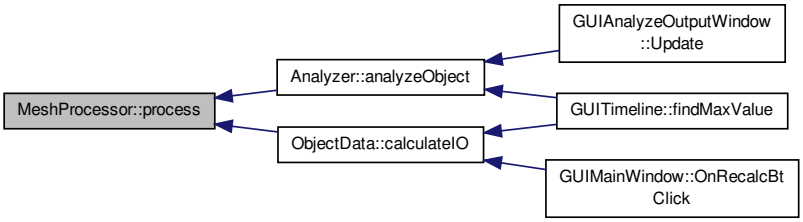
object	Das Objekt, für das die Temperaturverteilung ermittelt werden soll.

Definiert in Zeile 35 der Datei MeshProcessor.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

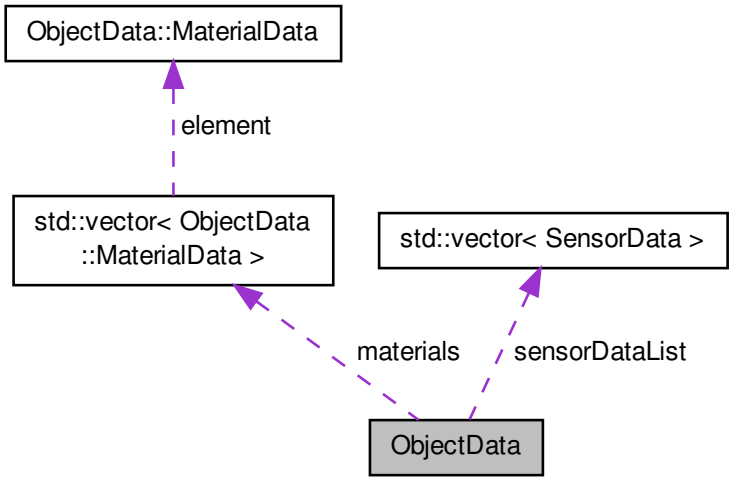
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.cpp

8.22 ObjectData Klassenreferenz

Die Daten eines Versuchsobjekts.

`#include <ObjectData.h>`

Zusammengehörigkeiten von ObjectData:



Klassen

- struct MaterialData

  Die Daten eines Materials.

Öffentliche Typen

- enum ObjectDataStatus { OD_SUCCESS = 1, OD_FAILURE, OD_LOAD_ALREADY_LOADED, OD_LOAD_INVALID_FILE, OD_LOAD_INVALID_SENSOR_FILE }

  Status einer die Objektdaten betreffenden Aktion.

Öffentliche Methoden

- ObjectData ()

  Der Konstruktor.
- int loadFromFile (wxString &path)

  Lädt ein Objekt und erste Sensordaten.
- int addSensorData (wxString &path)

  Lädt einfache Sensordaten und verknüpft sie mit dem Objekt.
- int addTimedData (wxString &path)

  Lädt zeitbezogene Sensordaten und verknüpft sie mit dem Objekt.
- int calculateIO ()

  Zerlegt das Objekt in Tetraeder (Schnittstelle zur Tetgen-Bibliothek) und Berechnet die Temperaturverteilung für die aktuell ausgewählten Sensordaten (und den aktuelle augewählten Zeitpunkt).
- vector< MaterialData > * getMaterials ()

  Gibt eine Referenz auf die Liste der Materialien (mit Materialdaten) des Objekts zurück.
- double getMaxvolume ()

Gibt das maximale Tetraedervolumen für der Zerlegung zurück.

- void setMaxvolume (double maxvolume)

  Setzt das maximale Tetraedervolumen für der Zerlegung.
- string getName ()

  Gibt den Namen des Objekts zurück.
- void setName (string name)

  Setzt den Namen des Objekts.
- double getQuality ()

  Gibt die Qualisätseinstellung für die Tetraeder bei der Zerlegung (s.
- void setQuality (double quality)

  Sezt die Qualisätseinstellung für die Tetraeder bei der Zerlegung (s.
- vector< SensorData > * getSensorDataList ()

  Gibt eine Referenz auf die Sensordaten des Objekts zurück.
- int getCurrentSensorIndex ()

  Gibt den Index des aktuell verwendeten Sensordatensatzes zurück.
- void setCurrentSensorIndex (int currentSensorIndex)

  Setzt den Index des aktuell verwendeten Sensordatensatzes.
- virtual ~ObjectData ()

  Der Destruktor.

Private Attribute

- int current_sensor_index

  Index des aktuell verwendeten Sensordatensatzes.
- string name

  Name des Objekts.
- double maxvolume

  Maximales Volumen für Tetraeder bei der Zerlegung.
- double quality

  Qualität der Tetraeder bei der Zerlegung (s.
- vector< MaterialData > materials

  Liste der Materialien des Objekts.
- vector< SensorData > sensorDataList

  Liste der Sensordaten des Objekts.

8.22.1 Ausführliche Beschreibung

Die Daten eines Versuchsobjekts.

Diese Klasse hält Objekteigenschaften, Materialien und Sensordaten eines untersuchten Objekts. Desweiteren stellt sie die Schnittstelle zur Tetgen-Bibliothek (http://wias-berlin.de/software/tetgen/) zum zerlegen des Objekts dar.

Definiert in Zeile 27 der Datei ObjectData.h.

8.22.2 Dokumentation der Aufzählungstypen

8.22.2.1 enum ObjectData::ObjectDataStatus

Status einer die Objektdaten betreffenden Aktion.

Aufzählungswerte

OD_SUCCESS

OD_FAILURE

OD_LOAD_ALREADY_LOADED

OD_LOAD_INVALID_FILE

OD_LOAD_INVALID_SENSOR_FILE

Definiert in Zeile 47 der Datei ObjectData.h.

8.22.3 Beschreibung der Konstruktoren und Destruktoren

8.22.3.1 ObjectData::ObjectData ( )

Der Konstruktor.

Definiert in Zeile 23 der Datei ObjectData.cpp.

8.22.3.2 ObjectData::~ObjectData ( ) [virtual]

Der Destruktor.

Definiert in Zeile 200 der Datei ObjectData.cpp.

8.22.4 Dokumentation der Elementfunktionen

8.22.4.1 int ObjectData::addSensorData (wxString & *path*)

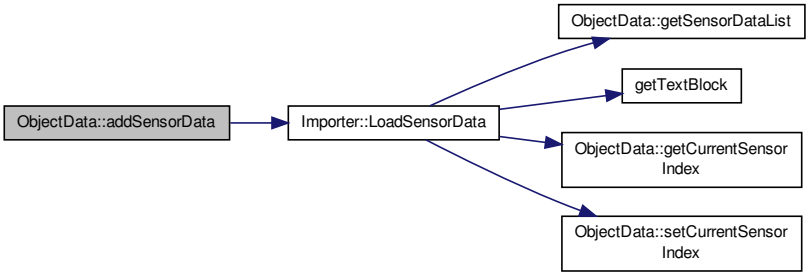
Lädt einfache Sensordaten und verknüpft sie mit dem Objekt.

Parameter

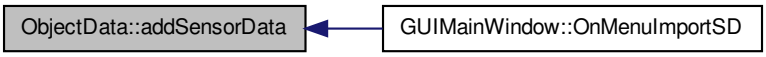
path	Pfad zur .sd-Datei.

Definiert in Zeile 108 der Datei ObjectData.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.22.4.2 int ObjectData::addTimedData (wxString & *path*)

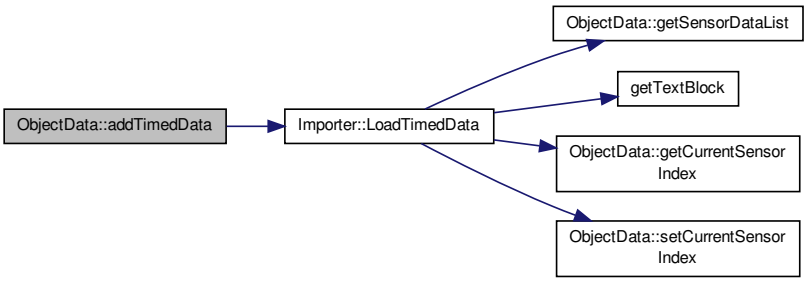
Lädt zeitbezogene Sensordaten und verknüpft sie mit dem Objekt.

Parameter

path	Pfad zur .tsd-Datei.

Definiert in Zeile 114 der Datei ObjectData.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



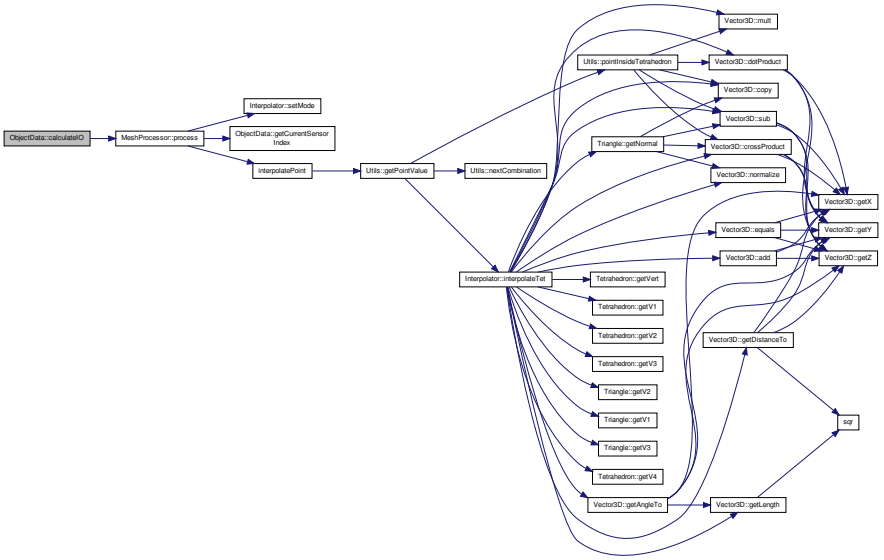
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.22.4.3 int ObjectData::calculateIO ( )

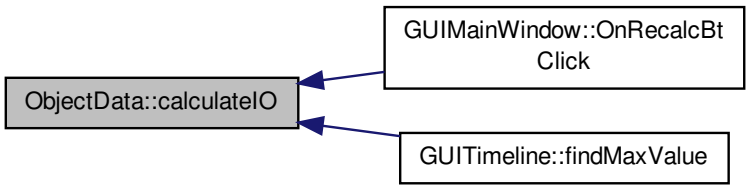
Zerlegt das Objekt in Tetraeder (Schnittstelle zur Tetgen-Bibliothek) und Berechnet die Temperaturverteilung für die aktuell ausgewählten Sensordaten (und den aktuelle augewählten Zeitpunkt).

Definiert in Zeile 120 der Datei ObjectData.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

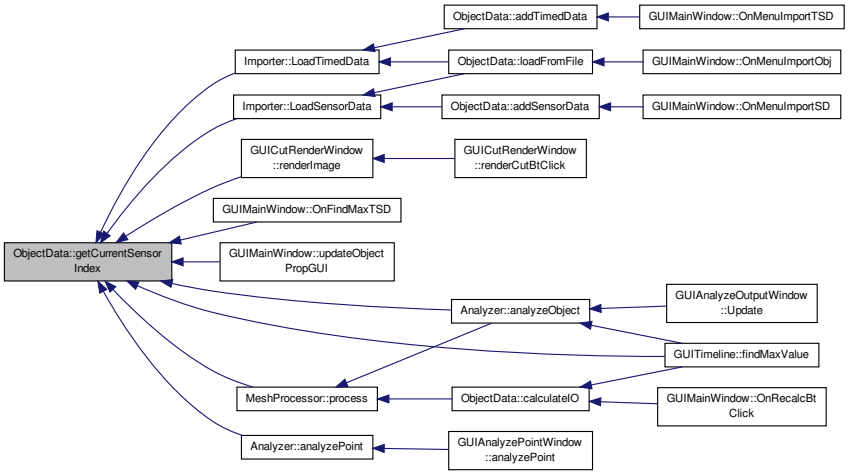


8.22.4.4 int ObjectData::getCurrentSensorIndex ( )

Gibt den Index des aktuell verwendeten Sensordatensatzes zurück.

Definiert in Zeile 192 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

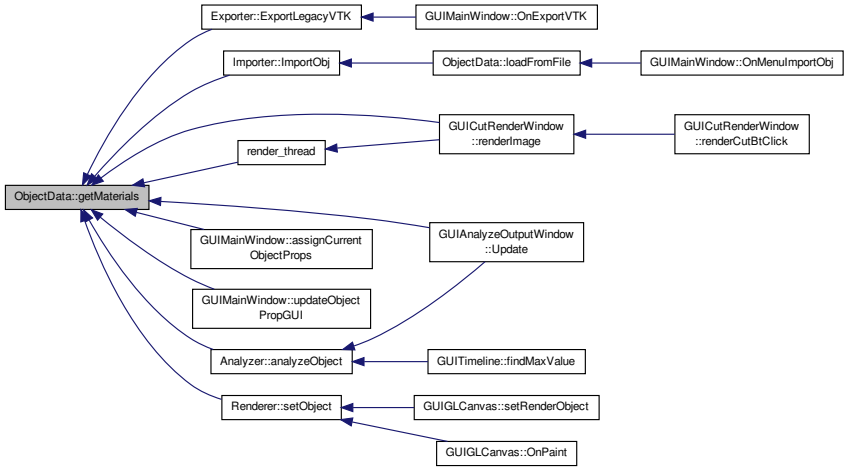


8.22.4.5 vector< ObjectData::MaterialData > * ObjectData::getMaterials ( )

Gibt eine Referenz auf die Liste der Materialien (mit Materialdaten) des Objekts zurück.

Definiert in Zeile 160 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

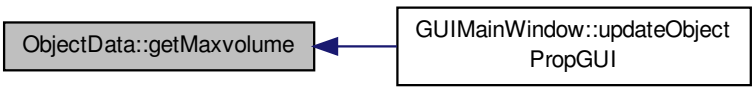


8.22.4.6 double ObjectData::getMaxvolume ( )

Gibt das maximale Tetraedervolumen für der Zerlegung zurück.

Definiert in Zeile 164 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

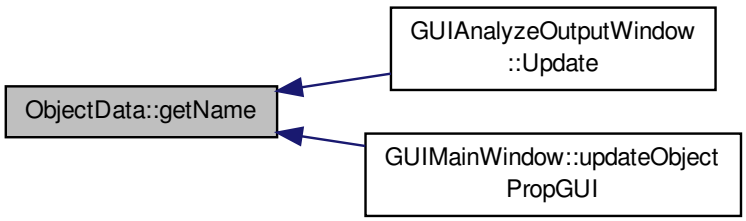


8.22.4.7 string ObjectData::getName ( )

Gibt den Namen des Objekts zurück.

Definiert in Zeile 172 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



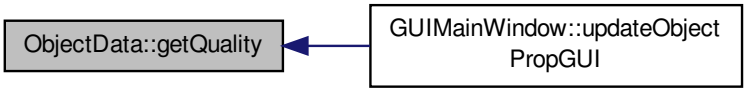
8.22.4.8 double ObjectData::getQuality ( )

Gibt die Qualisätseinstellung für die Tetraeder bei der Zerlegung (s.

Tetgen Dokumentation) zurück.

Definiert in Zeile 180 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

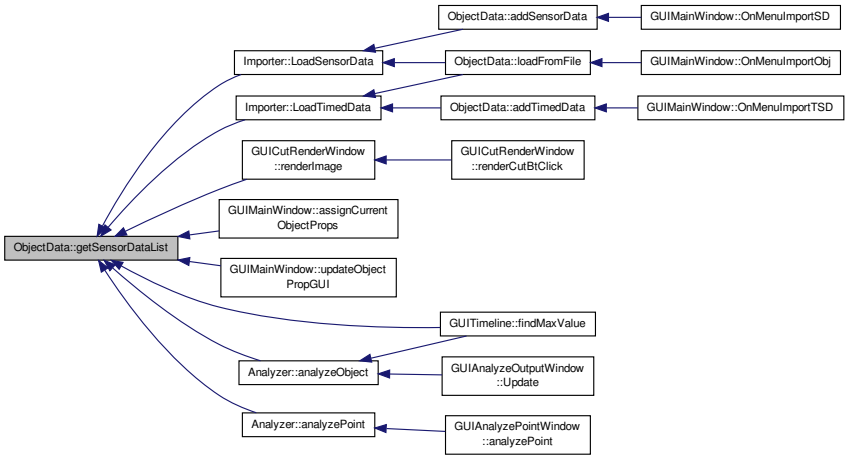


8.22.4.9 vector< SensorData > * ObjectData::getSensorDataList ( )

Gibt eine Referenz auf die Sensordaten des Objekts zurück.

Definiert in Zeile 188 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.22.4.10 int ObjectData::loadFromFile (wxString & *path*)

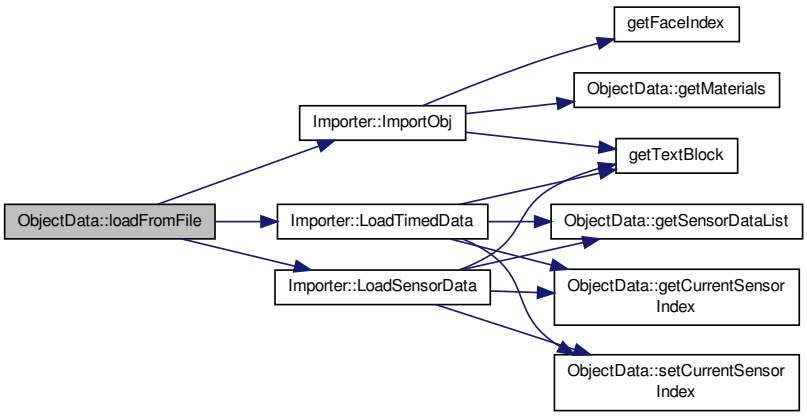
Lädt ein Objekt und erste Sensordaten.

Parameter

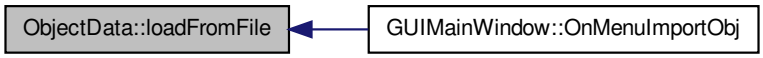
path	Pfad zur 3D-Modell(.obj)-Datei.

Definiert in Zeile 33 der Datei ObjectData.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

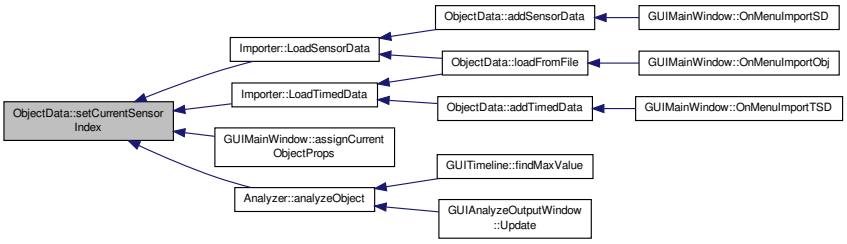


8.22.4.11 void ObjectData::setCurrentSensorIndex (int *currentSensorIndex*)

Setzt den Index des aktuell verwendeten Sensordatensatzes.

Definiert in Zeile 196 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.22.4.12 void ObjectData::setMaxvolume (double *maxvolume*)

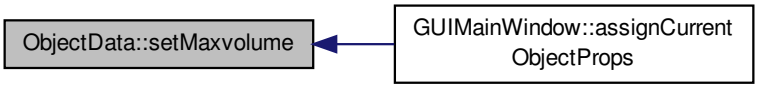
Setzt das maximale Tetraedervolumen für der Zerlegung.

Parameter

maxvolume	Maximales Tetraedervolumen.

Definiert in Zeile 168 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.22.4.13 void ObjectData::setName (string *name*)

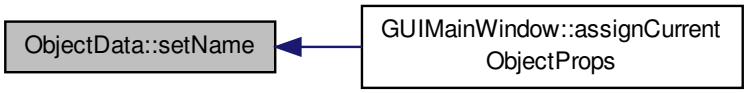
Setzt den Namen des Objekts.

Parameter

name	Der neue Name des Objekts.

Definiert in Zeile 176 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



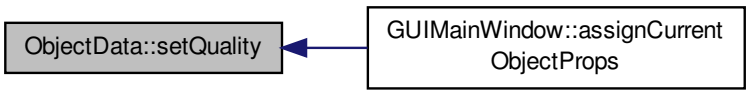
8.22.4.14 void ObjectData::setQuality (double *quality*)

Sezt die Qualisätseinstellung für die Tetraeder bei der Zerlegung (s.

Tetgen Dokumentation).

Definiert in Zeile 184 der Datei ObjectData.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.22.5 Dokumentation der Datenelemente

8.22.5.1 int ObjectData::current_sensor_index [private]

Index des aktuell verwendeten Sensordatensatzes.

Definiert in Zeile 144 der Datei ObjectData.h.

8.22.5.2 vector<MaterialData> ObjectData::materials [private]

Liste der Materialien des Objekts.

Definiert in Zeile 164 der Datei ObjectData.h.

8.22.5.3 double ObjectData::maxvolume [private]

Maximales Volumen für Tetraeder bei der Zerlegung.

Definiert in Zeile 154 der Datei ObjectData.h.

8.22.5.4 string ObjectData::name [private]

Name des Objekts.

Definiert in Zeile 149 der Datei ObjectData.h.

8.22.5.5 double ObjectData::quality [private]

Qualität der Tetraeder bei der Zerlegung (s.

Tetgen Dokumentation).

Definiert in Zeile 159 der Datei ObjectData.h.

8.22.5.6 vector<SensorData> ObjectData::sensorDataList [private]

Liste der Sensordaten des Objekts.

Definiert in Zeile 169 der Datei ObjectData.h.

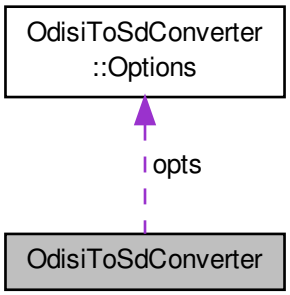
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.cpp

8.23 OdisiToSdConverter Klassenreferenz

Konverter von ODiSI zu .tsd.

Zusammengehörigkeiten von OdisiToSdConverter:



Klassen

- struct Options

  Strunktur für die Programmeinstellungen.

Öffentliche Methoden

- int convert (int argc, char ∗argv[])

Liest die Programmargumente, die Konfiguration, die Sensordefinitionsdatei und die ODiSI-Datei um eine .tsd-Datei zu generieren, bzw.

Geschützte Methoden

- bool contains (std::vector< string > &Vec, const string &Element)

  Testet, ob sich ein String in einer Liste von Strings befindet.
- bool contains (std::vector< int > &Vec, const int &Element)

  Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.
- void replaceAll (string &str, const string from, const string to)

  Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.
- string floattostr (float val)

  Wandelt eine Zeichenkette (String) um.
- string getTextBlock (string data, int n)

  Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.
- void parseLine (string line, vector< float > *out, vector< float > *times, vector< int > *debug_positions, size_t row_count)

  Sammelt Daten aus einer Textzeile (string).
- bool readConfiguration (string binary_path)

  Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.
- bool parseArguments (int argc, char *argv[], string &def_filename, string &data_filename, string &out_filename, string &err_filename)

  Wertet die Programmargumente aus.
- bool readSensorDefinitions (string path, vector< float > &inlist, vector< float > &outlist, vector< float > &in_x, vector< float > &out_x, vector< bool > &dirlist)

  Liest die Daten aus der Sensordefinitionsdatei.
- bool readInputFile (string path, vector< vector< float > > &values, vector< vector< int > > &debug_positions, vector< float > &times, vector< float > &lin_positions)

  Liest die Daten aus der Eingabedatei.
- bool writeOutputFile (string path, string logpath, vector< vector< float > > &values, vector< vector< int > > &debug_positions, vector< float > &times, vector< float > &lin_positions, vector< float > &inlist, vector< float > &outlist, vector< float > &in_x, vector< float > &out_x, vector< bool > &dirlist)

  Schreibt die Ausgabedatei.

Geschützte Attribute

- string configpaths [NUMBEROFPATHS]

  Suchpfade für die Konfigurationsdatei.
- struct OdisiToSdConverter::Options opts

  Hält die verwendeten Programmeinstellungen.

Statische, geschützte Attribute

- static const int NUMBEROFPATHS = 3

  Anzahl der Suchpfade für die Konfigurationsdatei.

8.23.1 Ausführliche Beschreibung

Konverter von ODiSI zu .tsd.

Zusätzlich können Ausreißerwerte erkannt und eliminiert werden.

Definiert in Zeile 22 der Datei main.cpp.

8.23.2 Dokumentation der Elementfunktionen

8.23.2.1 bool OdisiToSdConverter::contains (std::vector< string > & *Vec,* const string & *Element*) [inline], [protected]

Testet, ob sich ein String in einer Liste von Strings befindet.

Parameter

Vec	Liste der Strings.
Element	Der zu suchende String.

Rückgabe

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 65 der Datei main.cpp.

8.23.2.2 bool OdisiToSdConverter::contains (std::vector< int > & *Vec,* const int & *Element*) [inline], [protected]

Testet, ob sich eine Ganzzahl in einer Liste von Ganzzahlen befindet.

Parameter

Vec	Liste der Ganzzahlen.
Element	Die zu suchende Ganzzahl.

Rückgabe

true, wenn das Element gefunden wurde, sonst false.

Definiert in Zeile 81 der Datei main.cpp.

8.23.2.3 int OdisiToSdConverter::convert (int *argc,* char * *argv[]*) [inline]

Liest die Programmargumente, die Konfiguration, die Sensordefinitionsdatei und die ODiSI-Datei um eine .tsd-Datei zu generieren, bzw.

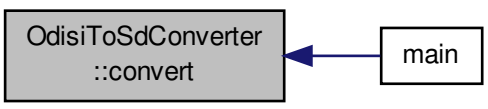
die Daten der ODiSI-Datei ein eine .tsd-Datei umzuwandeln. Wird duch die Funktion main() von außerhalb des Namespaces aufgerufen.

Parameter

argc	Anzahl der Programmargumente.
argv	Die Programmargumente.

Definiert in Zeile 908 der Datei main.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.23.2.4 string OdisiToSdConverter::floattostr (float *val*) [inline], [protected]

Wandelt eine Zeichenkette (String) um.

Parameter

val	Die umzuwandelnde Zahl.

Rückgabe

Die Entsprechung der Zahl als String.

Definiert in Zeile 119 der Datei main.cpp.

8.23.2.5 string OdisiToSdConverter::getTextBlock (string *data,* int *n*) [inline], [protected]

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

Parameter

data	Der Ausgansstring.
n	Index des zu findenden Blocks.

Rückgabe

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 132 der Datei main.cpp.

8.23.2.6 bool OdisiToSdConverter::parseArguments (int *argc,* char * *argv[],* string & *def_filename,* string & *data_filename,* string & *out_filename,* string & *err_filename*) [inline], [protected]

Wertet die Programmargumente aus.

Parameter

argc	Anzahl der Programmargumente.
argv	Die Programmargumente.
def_filename	Ausgabe für den Pfad zur Sensordefinitionsdatei.
data_filename	Ausgabe für den Pfad zur Eingabedatei.
out_filename	Ausgabe für den Pfad zur Ausgabedatei.
err_filename	Ausgabe für den Pfad zur Logdatei.

Rückgabe

Soll das Programm weiter ablaufen?

Definiert in Zeile 339 der Datei main.cpp.

8.23.2.7 void OdisiToSdConverter::parseLine (string *line,* vector< float > * *out,* vector< float > * *times,* vector< int > * *debug_positions,* size_t *row_count*) [inline], [protected]

Sammelt Daten aus einer Textzeile (string).

Parameter

line	Die zu untersuchende Textzeile.
out	Ausgabevariable für die Sensordaten der Zeile. Alle Spalten nach opts.start_col werden als Sensordatenspalten betrachtet. Wenn row_count == opts.startrow ist, werden statt der Sensordaten die Faserpositionen eingelesen!
times	Wenn nicht NULL, Ausgabevariable für den Zeitstempel der Zeile (opts.timecol). Der Zeitstempel wird an die übergebene Liste angehängt.
debug_positions	Wenn nicht NULL, Ausgabevariable für die Position der einzelnen Messwerte in der Datei. Diese Positionen werden in der Logdatei zur Fehlerkorrektur angegeben, um ein Wiederfinden der Werte für den Nutzer einfacher zu machen.
row_count	Nummer der aktuellen Zeile (Index+1).

Komma mit Punkt als Dezimaltrennzeichen ersetzen?

Definiert in Zeile 174 der Datei main.cpp.

8.23.2.8 bool OdisiToSdConverter::readConfiguration (string *binary_path*) [inline], [protected]

Liest und setzt die Programmkonfiguration aus der Konfigurationsdatei.

Parameter

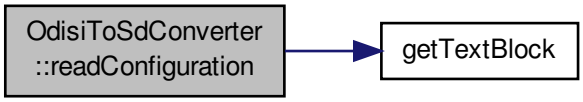
binary_path	Pfad zur Binärdatei.

Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 266 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.23.2.9 bool OdisiToSdConverter::readInputFile (string *path,* vector< vector< float > > & *values,* vector< vector< int > > & *debug_positions,* vector< float > & *times,* vector< float > & *lin_positions*) [inline], [protected]

Liest die Daten aus der Eingabedatei.

Parameter

path	Der Pfad zur Eingabedatei.
values	Liste für die extrahierten Sensorwerte.

times	Liste für die Zeitstempel der Messwerte.
debug_positions	Liste für die Positionen der Messwerte in der Datei.
lin_positions	Liste für die Positionen auf der Faser.

Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 587 der Datei main.cpp.

8.23.2.10 bool OdisiToSdConverter::readSensorDefinitions (string *path,* vector< float > & *inlist,* vector< float > & *outlist,* vector< float > & *in_x,* vector< float > & *out_x,* vector< bool > & *dirlist*) [inline], [protected]

Liest die Daten aus der Sensordefinitionsdatei.

Parameter

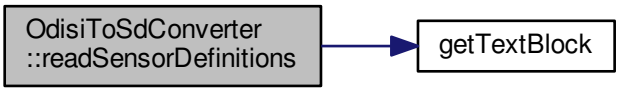
path	Pfad zur Binärdatei.
inlist	Liste für die Positionen der Fasereingänge auf der Faser.
outlist	Liste für die Positionen der Faserausgänge auf der Faser.
in_x	Liste für die X-Positionen der Fasereingänge.
out_x	Liste für die X-Positionen der Faserausgänge.
dirlist	Liste für die Richtungen der Faser zwischen Ein-und Ausgang bezüglich der Z-Richtung.

Rückgabe

War das Einlesen erfolgreich?

Definiert in Zeile 481 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.23.2.11 void OdisiToSdConverter::replaceAll (string & *str,* const string *from,* const string *to*) [inline], [protected]

Ersetzt in einem String alle Vorkommen eines Teilstrings durch einen Anderen.

Parameter

str	Der zu durchsuchende String.
from	Der zu ersetzende Teilstring.

to	Der Teilstring, durch den ersetzt werden soll.

Definiert in Zeile 97 der Datei main.cpp.

8.23.2.12 bool OdisiToSdConverter::writeOutputFile (string *path,* string *logpath,* vector< vector< float > > & *values,* vector< vector< int > > & *debug_positions,* vector< float > & *times,* vector< float > & *lin_positions,* vector< float > & *inlist,* vector< float > & *outlist,* vector< float > & *in_x,* vector< float > & *out_x,* vector< bool > & *dirlist*) [inline],[protected]

Schreibt die Ausgabedatei.

Parameter

path	Der Pfad zur Ausgabedatei.
logpath	Der Pfad zur Logdatei. Bei "" wird keine Logdatei angelegt.
debug_positions	Liste der Positionen der Sensorwerte in der Eingabedatei.
values	Die extrahierten Sensorwerte.
times	Zeitstempel der Datensätze.
lin_positions	Die Position der Messstellen auf der Faser.
inlist	Positionen der Fasereingänge auf der Faser.
outlist	Positionen der Faserausgänge auf der Faser.
in_x	X-Positionen der Fasereingänge.
out_x	X-Positionen der Faserausgänge.
dirlist	Liste für die Richtungen der Faser zwischen Ein-und Ausgang bezüglich der Z-Richtung.

Rückgabe

War das Schreiben erfolgreich?

Definiert in Zeile 687 der Datei main.cpp.

8.23.3 Dokumentation der Datenelemente

8.23.3.1 string OdisiToSdConverter::configpaths[NUMBEROFPATHS] [protected]

Initialisierung:

```
{
            "/etc/simpleanalyzer/odisitosd.conf",
            "/usr/local/share/simpleanalyzer/odisitosd.conf",
            "/usr/share/simpleanalyzer/odisitosd.conf" }
```

Suchpfade für die Konfigurationsdatei.

Das Verzeichnis der ausführbaren Datei wird immer geprüft.

Definiert in Zeile 33 der Datei main.cpp.

8.23.3.2 const int OdisiToSdConverter::NUMBEROFPATHS = 3 [static],[protected]

Anzahl der Suchpfade für die Konfigurationsdatei.

Definiert in Zeile 27 der Datei main.cpp.

8.23.3.3 struct OdisiToSdConverter::Options OdisiToSdConverter::opts [protected]

Hält die verwendeten Programmeinstellungen.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/odisitosd/main.cpp

8.24 CsvToSdConverter::Options Strukturreferenz

Strunktur für die Programmeinstellungen.

Öffentliche Attribute

- size_t start_col

  Index der Spalte, in der die ersten Sensordaten stehen.
- char separator

  Das verwendete Separatorzeichen.
- bool replace_comma_with_point

  Sollen Kommata durch Punkte ersetzt werrden?
- size_t timecol

  Index der Spalte, die die Zeitstempel enthält.
- size_t namecol

  Index der Spalte, die die Namen für die Datensätze enthält.
- int time_step_delta

  Schrittweite beim Auslesen der Sensordaten (nur jeder time_step_delta Zeitpunkt wird verwendet).
- long max_time

  Nur bis maximal zu diesem Zeitstempel auslesen.
- long min_time

  Ab diesem Zeitstempel auslesen.

8.24.1 Ausführliche Beschreibung

Strunktur für die Programmeinstellungen.

Definiert in Zeile 38 der Datei main.cpp.

8.24.2 Dokumentation der Datenelemente

8.24.2.1 long CsvToSdConverter::Options::max_time

Nur bis maximal zu diesem Zeitstempel auslesen.

Definiert in Zeile 45 der Datei main.cpp.

8.24.2.2 long CsvToSdConverter::Options::min_time

Ab diesem Zeitstempel auslesen.

Definiert in Zeile 46 der Datei main.cpp.

8.24.2.3 size_t CsvToSdConverter::Options::namecol

Index der Spalte, die die Namen für die Datensätze enthält.

Definiert in Zeile 43 der Datei main.cpp.

8.24.2.4 bool CsvToSdConverter::Options::replace_comma_with_point

Sollen Kommata durch Punkte ersetzt werrden?

Definiert in Zeile 41 der Datei main.cpp.

8.24.2.5 char CsvToSdConverter::Options::separator

Das verwendete Separatorzeichen.

Definiert in Zeile 40 der Datei main.cpp.

8.24.2.6 size_t CsvToSdConverter::Options::start_col

Index der Spalte, in der die ersten Sensordaten stehen.

Definiert in Zeile 39 der Datei main.cpp.

8.24.2.7 int CsvToSdConverter::Options::time_step_delta

Schrittweite beim Auslesen der Sensordaten (nur jeder time_step_delta Zeitpunkt wird verwendet).

Definiert in Zeile 44 der Datei main.cpp.

8.24.2.8 size_t CsvToSdConverter::Options::timecol

Index der Spalte, die die Zeitstempel enthält.

Definiert in Zeile 42 der Datei main.cpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/csvtosd/main.cpp

8.25 TsdMerger::Options Strukturreferenz

Strunktur für die Programmeinstellungen.

Öffentliche Attribute

- int offset

  Ein zusätzlicher Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.
- unsigned int max_dt

  Maximale Zeitdifferenz zwischen den Zeitstempeln um die Datensätze zusammenführen zu können.
- long int delta

  Ein Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.
- bool auto_delta

  Delta automatisch aus der Differenz der jeweils ersten Zeitstempel der Eingabedateien ermitteln.

8.25.1 Ausführliche Beschreibung

Strunktur für die Programmeinstellungen.

Definiert in Zeile 27 der Datei mergetsd.cpp.

8.25.2 Dokumentation der Datenelemente

8.25.2.1 bool TsdMerger::Options::auto_delta

Delta automatisch aus der Differenz der jeweils ersten Zeitstempel der Eingabedateien ermitteln.

Definiert in Zeile 31 der Datei mergetsd.cpp.

8.25.2.2 long int TsdMerger::Options::delta

Ein Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.

Definiert in Zeile 30 der Datei mergetsd.cpp.

8.25.2.3 unsigned int TsdMerger::Options::max_dt

Maximale Zeitdifferenz zwischen den Zeitstempeln um die Datensätze zusammenführen zu können.

Definiert in Zeile 29 der Datei mergetsd.cpp.

8.25.2.4 int TsdMerger::Options::offset

Ein zusätzlicher Versatz, der zu den Zeitstempeln der zweiten Datei addiert wird.

Definiert in Zeile 28 der Datei mergetsd.cpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/mergetsd/src/mergetsd.cpp

8.26 OdisiToSdConverter::Options Strukturreferenz

Strunktur für die Programmeinstellungen.

Öffentliche Attribute

- size_t startrow

  Index der ersten Zeile in der Odisi-Datei, die Sensordaten enthält.
- char separator

  Das verwendete Separatorzeichen.
- bool replace_comma_with_point

  Sollen Kommata durch Punkte ersetzt werrden?
- size_t timecol

  Index der Spalte, die die Zeitstempel enthält.
- float error_threshold

  Maximal zulässige Differenz zum Vorgängerwert für einen gültigen Messwert bei der Fehlerkorrektur.
- int maxfwcount

  Maximale Schrittanzahl zum finden eines gültigen Messwertes bei der Fehlerkorrektur.
- int tab_space_count

  Anzahl der Leerzeichen für TAB (Für die Positionsangabe in der Log-Datei).
- float height

  Höhe der Faserebene in m.
- float basetemp

Temperatur zu Beginn des Versuches (Die Odisi-Daten sind Differenzen zu dieser Anfangstemperatur).

- float objwidth

  Position der Messwerte auf der X-Achse um diesen Wert verschieben.

- bool flipobj

  Position auf der X-Achse spiegeln?

- int fiber_step_delta

  Schrittweite beim Auslesen der Sensordaten (nur jeder fiber_step_delta Messpunkt auf der Faser wird verwendet).

- int time_step_delta

  Schrittweite beim Auslesen der Sensordaten (nur jeder time_step_delta Zeitpunkt wird verwendet).

- double max_time

  Nur bis maximal zu diesem Zeitstempel auslesen.

- double min_time

  Ab diesem Zeitstempel auslesen.

8.26.1 Ausführliche Beschreibung

Strunktur für die Programmeinstellungen.

Definiert in Zeile 41 der Datei main.cpp.

8.26.2 Dokumentation der Datenelemente

8.26.2.1 float OdisiToSdConverter::Options::basetemp

Temperatur zu Beginn des Versuches (Die Odisi-Daten sind Differenzen zu dieser Anfangstemperatur).

Definiert in Zeile 50 der Datei main.cpp.

8.26.2.2 float OdisiToSdConverter::Options::error_threshold

Maximal zulässige Differenz zum Vorgängerwert für einen gültigen Messwert bei der Fehlerkorrektur.

Definiert in Zeile 46 der Datei main.cpp.

8.26.2.3 int OdisiToSdConverter::Options::fiber_step_delta

Schrittweite beim Auslesen der Sensordaten (nur jeder fiber_step_delta Messpunkt auf der Faser wird verwendet).

Definiert in Zeile 53 der Datei main.cpp.

8.26.2.4 bool OdisiToSdConverter::Options::flipobj

Position auf der X-Achse spiegeln?

Definiert in Zeile 52 der Datei main.cpp.

8.26.2.5 float OdisiToSdConverter::Options::height

Höhe der Faserebene in m.

Definiert in Zeile 49 der Datei main.cpp.

8.26.2.6 double OdisiToSdConverter::Options::max_time

Nur bis maximal zu diesem Zeitstempel auslesen.

Definiert in Zeile 55 der Datei main.cpp.

8.26.2.7 int OdisiToSdConverter::Options::maxfwcount

Maximale Schrittanzahl zum finden eines gültigen Messwertes bei der Fehlerkorrektur.

Definiert in Zeile 47 der Datei main.cpp.

8.26.2.8 double OdisiToSdConverter::Options::min_time

Ab diesem Zeitstempel auslesen.

Definiert in Zeile 56 der Datei main.cpp.

8.26.2.9 float OdisiToSdConverter::Options::objwidth

Position der Messwerte auf der X-Achse um diesen Wert verschieben.

Definiert in Zeile 51 der Datei main.cpp.

8.26.2.10 bool OdisiToSdConverter::Options::replace_comma_with_point

Sollen Kommata durch Punkte ersetzt werrden?

Definiert in Zeile 44 der Datei main.cpp.

8.26.2.11 char OdisiToSdConverter::Options::separator

Das verwendete Separatorzeichen.

(Hier Leerzeichen)

Definiert in Zeile 43 der Datei main.cpp.

8.26.2.12 size_t OdisiToSdConverter::Options::startrow

Index der ersten Zeile in der Odisi-Datei, die Sensordaten enthält.

Definiert in Zeile 42 der Datei main.cpp.

8.26.2.13 int OdisiToSdConverter::Options::tab_space_count

Anzahl der Leerzeichen für TAB (Für die Positionsangabe in der Log-Datei).

Definiert in Zeile 48 der Datei main.cpp.

8.26.2.14 int OdisiToSdConverter::Options::time_step_delta

Schrittweite beim Auslesen der Sensordaten (nur jeder time_step_delta Zeitpunkt wird verwendet).

Definiert in Zeile 54 der Datei main.cpp.

8.26.2.15 size_t OdisiToSdConverter::Options::timecol

Index der Spalte, die die Zeitstempel enthält.

Definiert in Zeile 45 der Datei main.cpp.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

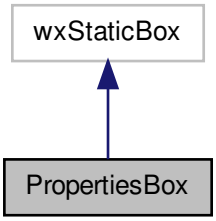
- /daten/Projekte/eclipse_workspace/odisitosd/main.cpp

8.27 PropertiesBox Klassenreferenz

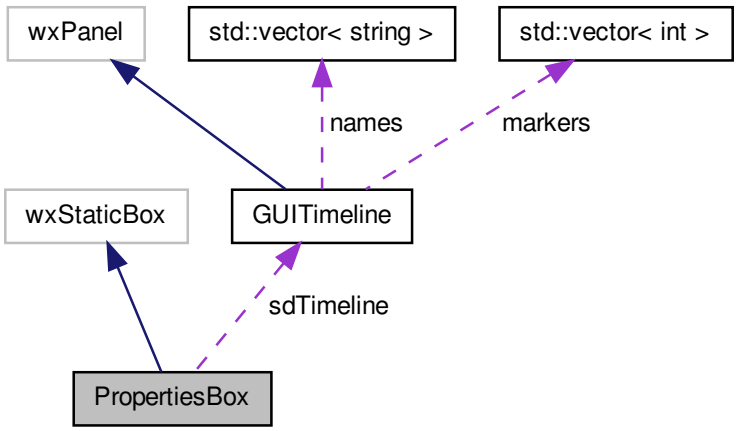
Oberfläche zum Verändern/Anzeigen der Eigenschaften eines Objekts.

`#include <PropertiesBox.h>`

Klassendiagramm für PropertiesBox:



Zusammengehörigkeiten von PropertiesBox:



Öffentliche Methoden

- PropertiesBox (wxWindow *parent)

  Der Konstruktor.
- void resize ()

  Behandelt Größenänderungen und passt die Positionen der Komponenten an.
- wxCheckBox * getAnalyzeMarkerCheckBox ()

  Gibt die Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts zurück.
- wxCheckBox * getAutoUpdateCeckBox ()

  Gibt die Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften zurück.
- wxButton * getClearAnalyzeMarkerBt ()

  Gibt den Button zum Löschen aller Markierungen (s.
- wxTextCtrl * getSpecificHeatCapEdit ()

  Gibt das Eingabefeld für die spezifische Wärmekapazität zurück.
- int getCurrentMaterial ()

  Gibt den Index des aktuell ausgewählten Materials zurück.
- void setCurrentMaterial (int index)

  Setzt den Index des aktuell ausgewählten Materials.
- wxTextCtrl * getDensityEdit ()

  Gibt das Eingabefeld für die Dichte des Materials zurück.
- wxButton * getFindMaxBt ()

  Gibt den Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s.
- wxComboBox * getInterpolationModeList ()

  Gibt das Auswahlfeld für den zu verwendenden Interpolationsmodus zurück.
- wxListBox * getMatListBox ()

  Gibt die Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen, zurück.
- wxTextCtrl * getMatNameEdit ()

  Gibt das Eingabefeld für den Materialnamen zurück.
- wxStaticBox * getMatPropBox ()

  Gibt den Bereich, der die Materialeigenschaften enthält zurück.
- wxTextCtrl * getMaxVolumeEdit ()

  Gibt das Eingabefeld für das maximale Tetraedervolumen zurück.
- wxButton * getNextMarkerBt ()

  Gibt den Button zum Auswählen der nächsten Markierung (s.
- wxTextCtrl * getObjNameEdit ()

  Gibt das Eingeabefeld für den Objektnamen.
- wxButton * getPrevMarkerBt ()

  Gibt den Button zum Auswählen der vorherigen Markierung (s.
- wxTextCtrl * getQualityEdit ()

  Gibt das Eingabefeld für die Zerlegungsqualität des Modells (s.
- wxButton * getRecalcButton ()

  Gibt den Button zum Neuberechnen der Temperaturverteilung zurück.
- GUITimeline * getSdTimeline ()

  Gibt die Die Zeitleiste für zeitbezogene Sensordaten zurück.
- wxComboBox * getSensorDataList ()

  Gibt das Auswahlfeld für den zu verwendenden Sensordatensatz zurück.
- wxStaticText * getUpToDateLbl ()

  Gibt die Beschiftungskomponente für die Warnung bei geänderten Objekteigenschaften zurück.
- virtual ~PropertiesBox ()

  Der Destruktor.

Private Attribute

- wxButton * recalcButton

  Button zum Neuberechnen der Temperaturverteilung.
- wxStaticText * objNameLbl

  Beschriftung für das Objektnamen-Eingabefeld.
- wxTextCtrl * objNameEdit

  Eingeabefeld für den Objektnamen.
- wxStaticText * matNameLbl

  Beschriftung für das Materialnamen-Eingabefeld.
- wxTextCtrl * matNameEdit

  Eingabefeld für den Materialnamen.
- wxStaticText * upToDateLbl

  Beschiftung für die Warnung bei geänderten Objekteigenschaften.
- wxStaticText * maxVolumeLbl

  Beschriftung für das max.
- wxTextCtrl * maxVolumeEdit

  Eingabefeld für das maximale Tetraedervolumen.
- wxStaticText * qualityLbl

  Beschriftung für das Zerlegungsqualität-Eingabefeld.
- wxTextCtrl * qualityEdit

  Eingabefeld für die Zerlegungsqualität des Modells (s.
- wxStaticText * sensorDataLbl

  Beschriftung für das Sensordatensatz-Auswahlfeld.
- wxComboBox * sensorDataList

  Auswahlfeld für den zu verwendenden Sensordatensatz.
- wxListBox * matListBox

  Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen.
- wxStaticText * matListBoxLbl

  Beschriftung für die Materialauswahl-Box.
- wxStaticBox * matPropBox

  Bereich, der die Materialeigenschaften enthält.
- wxComboBox * interpolationModeList

  Auswahlfeld für den zu verwendenden Interpolationsmodus.
- wxStaticText * interpolationModeLbl

  Beschriftung für das Interpolationsmodus-Auswahlfeld.
- wxTextCtrl * densityEdit

  Eingabefeld für die Dichte des Materials.
- wxStaticText * densityLbl

  Beschriftung für das Dichte-Eingabefeld.
- wxTextCtrl * specificHeatCapEdit

  Eingabefeld für die spezifische Wärmekapazität.
- wxStaticText * specificHeatCapLbl

  Beschriftung für das Wärmekapazitäts-Eingabefeld.
- GUITimeline * sdTimeline

  Die Zeitleiste für zeitbezogene Sensordaten.
- wxCheckBox * analyzeMarkerCheckBox

  Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts.
- wxButton * findMaxBt

  Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s.
- wxButton * clearAnalyzeMarkerBt

Button zum Löschen aller Markierungen (s.

- wxButton ∗ nextMarkerBt

  Button zum Auswählen der nächsten Markierung (s.

- wxButton ∗ prevMarkerBt

  Button zum Auswählen der vorherigen Markierung (s.

- wxCheckBox ∗ autoUpdateCeckBox

  Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften.

- int current_material

  Index des aktuell ausgewählten Materials.

8.27.1 Ausführliche Beschreibung

Oberfläche zum Verändern/Anzeigen der Eigenschaften eines Objekts.

Diese Klasse verwaltet nur das Layout des Objekteigenschaften-Bereichs. Die Funktionalität wird in GUIMainWindow behandelt.

Definiert in Zeile 19 der Datei PropertiesBox.h.

8.27.2 Beschreibung der Konstruktoren und Destruktoren

8.27.2.1 PropertiesBox::PropertiesBox (wxWindow ∗ *parent*)

Der Konstruktor.

Parameter

parent	Die übergeordnete Komponente.

Definiert in Zeile 19 der Datei PropertiesBox.cpp.

8.27.2.2 PropertiesBox::∼PropertiesBox ( ) [virtual]

Der Destruktor.

Definiert in Zeile 224 der Datei PropertiesBox.cpp.

8.27.3 Dokumentation der Elementfunktionen

8.27.3.1 wxCheckBox ∗ PropertiesBox::getAnalyzeMarkerCheckBox ( )

Gibt die Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts zurück.

Definiert in Zeile 141 der Datei PropertiesBox.cpp.

8.27.3.2 wxCheckBox ∗ PropertiesBox::getAutoUpdateCeckBox ( )

Gibt die Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften zurück.

Definiert in Zeile 145 der Datei PropertiesBox.cpp.

8.27.3.3 wxButton * PropertiesBox::getClearAnalyzeMarkerBt ( )

Gibt den Button zum Löschen aller Markierungen (s.

GUITimeline) zurück.

Definiert in Zeile 149 der Datei PropertiesBox.cpp.

8.27.3.4 int PropertiesBox::getCurrentMaterial ( )

Gibt den Index des aktuell ausgewählten Materials zurück.

Definiert in Zeile 157 der Datei PropertiesBox.cpp.

8.27.3.5 wxTextCtrl * PropertiesBox::getDensityEdit ( )

Gibt das Eingabefeld für die Dichte des Materials zurück.

Definiert in Zeile 164 der Datei PropertiesBox.cpp.

8.27.3.6 wxButton * PropertiesBox::getFindMaxBt ( )

Gibt den Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s.

GUITimeline) zurück.

Definiert in Zeile 168 der Datei PropertiesBox.cpp.

8.27.3.7 wxComboBox * PropertiesBox::getInterpolationModeList ( )

Gibt das Auswahlfeld für den zu verwendenden Interpolationsmodus zurück.

Definiert in Zeile 172 der Datei PropertiesBox.cpp.

8.27.3.8 wxListBox * PropertiesBox::getMatListBox ( )

Gibt die Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen, zurück.

Definiert in Zeile 176 der Datei PropertiesBox.cpp.

8.27.3.9 wxTextCtrl * PropertiesBox::getMatNameEdit ( )

Gibt das Eingabefeld für den Materialnamen zurück.

Definiert in Zeile 180 der Datei PropertiesBox.cpp.

8.27.3.10 wxStaticBox * PropertiesBox::getMatPropBox ( )

Gibt den Bereich, der die Materialeigenschaften enthält zurück.

Definiert in Zeile 184 der Datei PropertiesBox.cpp.

8.27.3.11 wxTextCtrl * PropertiesBox::getMaxVolumeEdit ( )

Gibt das Eingabefeld für das maximale Tetraedervolumen zurück.

Definiert in Zeile 188 der Datei PropertiesBox.cpp.

8.27.3.12 wxButton * PropertiesBox::getNextMarkerBt ( )

Gibt den Button zum Auswählen der nächsten Markierung (s.
GUITimeline) zurück.
Definiert in Zeile 192 der Datei PropertiesBox.cpp.

8.27.3.13 wxTextCtrl * PropertiesBox::getObjNameEdit ( )

Gibt das Eingeabefeld für den Objektnamen.
zurück.
Definiert in Zeile 196 der Datei PropertiesBox.cpp.

8.27.3.14 wxButton * PropertiesBox::getPrevMarkerBt ( )

Gibt den Button zum Auswählen der vorherigen Markierung (s.
GUITimeline) zurück.
Definiert in Zeile 200 der Datei PropertiesBox.cpp.

8.27.3.15 wxTextCtrl * PropertiesBox::getQualityEdit ( )

Gibt das Eingabefeld für die Zerlegungsqualität des Modells (s.
Tetgen-dokumentation für weitere Informationen) zurück.
Definiert in Zeile 204 der Datei PropertiesBox.cpp.

8.27.3.16 wxButton * PropertiesBox::getRecalcButton ( )

Gibt den Button zum Neuberechnen der Temperaturverteilung zurück.
Definiert in Zeile 208 der Datei PropertiesBox.cpp.

8.27.3.17 GUITimeline * PropertiesBox::getSdTimeline ( )

Gibt die Die Zeitleiste für zeitbezogene Sensordaten zurück.
Definiert in Zeile 212 der Datei PropertiesBox.cpp.

8.27.3.18 wxComboBox * PropertiesBox::getSensorDataList ( )

Gibt das Auswahlfeld für den zu verwendenden Sensordatensatz zurück.
Definiert in Zeile 216 der Datei PropertiesBox.cpp.

8.27.3.19 wxTextCtrl * PropertiesBox::getSpecificHeatCapEdit ( )

Gibt das Eingabefeld für die spezifische Wärmekapazität zurück.
Definiert in Zeile 153 der Datei PropertiesBox.cpp.

8.27.3.20 wxStaticText * PropertiesBox::getUpToDateLbl ( )

Gibt die Beschiftungskomponente für die Warnung bei geänderten Objekteigenschaften zurück.

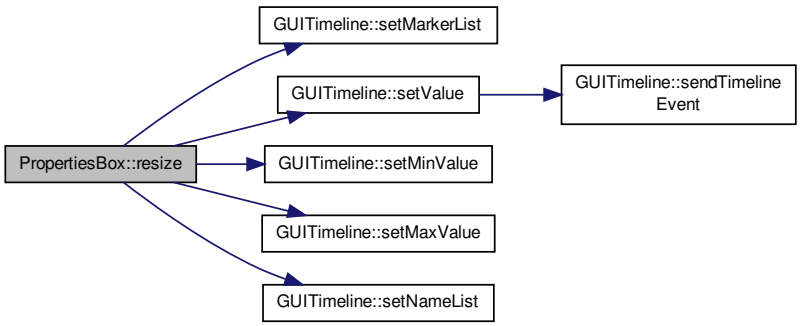
Definiert in Zeile 220 der Datei PropertiesBox.cpp.

8.27.3.21 void PropertiesBox::resize ( )

Behandelt Größenänderungen und passt die Positionen der Komponenten an.

Definiert in Zeile 72 der Datei PropertiesBox.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.27.3.22 void PropertiesBox::setCurrentMaterial (int *index*)

Setzt den Index des aktuell ausgewählten Materials.

Parameter

index	Index des auszuwählenden Materials.

Definiert in Zeile 160 der Datei PropertiesBox.cpp.

8.27.4 Dokumentation der Datenelemente

8.27.4.1 wxCheckBox* PropertiesBox::analyzeMarkerCheckBox [private]

Checkbox zum markieren des auf der Zeitleiste ausgewählten Zeitpunkts.

Dieser Zeitpunkt wird dann im Analysedaten-Übersichtsfenster (GUIAnalyzeOutputWindow) angezeigt.

Definiert in Zeile 258 der Datei PropertiesBox.h.

8.27.4.2 wxCheckBox* PropertiesBox::autoUpdateCeckBox [private]

Checkbox zum automatischen Neuberechnen der Temperaturverteilung nach einer Änderung an den Objekteigenschaften.

Definiert in Zeile 283 der Datei PropertiesBox.h.

8.27.4.3 wxButton* PropertiesBox::clearAnalyzeMarkerBt [private]

Button zum Löschen aller Markierungen (s.

GUITimeline).

Definiert in Zeile 268 der Datei PropertiesBox.h.

8.27.4.4 int PropertiesBox::current_material [private]

Index des aktuell ausgewählten Materials.

Definiert in Zeile 288 der Datei PropertiesBox.h.

8.27.4.5 wxTextCtrl* PropertiesBox::densityEdit [private]

Eingabefeld für die Dichte des Materials.

Definiert in Zeile 232 der Datei PropertiesBox.h.

8.27.4.6 wxStaticText* PropertiesBox::densityLbl [private]

Beschriftung für das Dichte-Eingabefeld.

Definiert in Zeile 237 der Datei PropertiesBox.h.

8.27.4.7 wxButton* PropertiesBox::findMaxBt [private]

Button zum Suchen des maximums zwischen zwei markierten Zeitpunkten (s.

GUITimeline).

Definiert in Zeile 263 der Datei PropertiesBox.h.

8.27.4.8 wxStaticText* PropertiesBox::interpolationModeLbl [private]

Beschriftung für das Interpolationsmodus-Auswahlfeld.

Definiert in Zeile 227 der Datei PropertiesBox.h.

8.27.4.9 wxComboBox* PropertiesBox::interpolationModeList [private]

Auswahlfeld für den zu verwendenden Interpolationsmodus.

Definiert in Zeile 222 der Datei PropertiesBox.h.

8.27.4.10 wxListBox* PropertiesBox::matListBox [private]

Auswahlbox für das Material, dessen Eigenschaften angezeigt werden sollen.

Definiert in Zeile 207 der Datei PropertiesBox.h.

8.27.4.11 wxStaticText* PropertiesBox::matListBoxLbl [private]

Beschriftung für die Materialauswahl-Box.

Definiert in Zeile 212 der Datei PropertiesBox.h.

8.27.4.12 wxTextCtrl* PropertiesBox::matNameEdit [private]

Eingabefeld für den Materialnamen.

Definiert in Zeile 167 der Datei PropertiesBox.h.

8.27.4.13 wxStaticText* PropertiesBox::matNameLbl [private]

Beschriftung für das Materialnamen-Eingabefeld.

Definiert in Zeile 162 der Datei PropertiesBox.h.

8.27.4.14 wxStaticBox* PropertiesBox::matPropBox [private]

Bereich, der die Materialeigenschaften enthält.

Definiert in Zeile 217 der Datei PropertiesBox.h.

8.27.4.15 wxTextCtrl* PropertiesBox::maxVolumeEdit [private]

Eingabefeld für das maximale Tetraedervolumen.

Definiert in Zeile 182 der Datei PropertiesBox.h.

8.27.4.16 wxStaticText* PropertiesBox::maxVolumeLbl [private]

Beschriftung für das max.

Tetraedervolumen-Eingabefeld.

Definiert in Zeile 177 der Datei PropertiesBox.h.

8.27.4.17 wxButton* PropertiesBox::nextMarkerBt [private]

Button zum Auswählen der nächsten Markierung (s.

GUITimeline).

Definiert in Zeile 273 der Datei PropertiesBox.h.

8.27.4.18 wxTextCtrl* PropertiesBox::objNameEdit [private]

Eingeabefeld für den Objektnamen.

Definiert in Zeile 157 der Datei PropertiesBox.h.

8.27.4.19 wxStaticText* PropertiesBox::objNameLbl [private]

Beschriftung für das Objektnamen-Eingabefeld.

Definiert in Zeile 152 der Datei PropertiesBox.h.

8.27.4.20 wxButton* PropertiesBox::prevMarkerBt [private]

Button zum Auswählen der vorherigen Markierung (s.

GUITimeline).

Definiert in Zeile 278 der Datei PropertiesBox.h.

8.27.4.21 wxTextCtrl* PropertiesBox::qualityEdit [private]

Eingabefeld für die Zerlegungsqualität des Modells (s.

Tetgen-dokumentation für weitere Informationen).

Definiert in Zeile 192 der Datei PropertiesBox.h.

8.27.4.22 wxStaticText* PropertiesBox::qualityLbl [private]

Beschriftung für das Zerlegungsqualität-Eingabefeld.

Definiert in Zeile 187 der Datei PropertiesBox.h.

8.27.4.23 wxButton* PropertiesBox::recalcButton [private]

Button zum Neuberechnen der Temperaturverteilung.

Definiert in Zeile 147 der Datei PropertiesBox.h.

8.27.4.24 GUITimeline* PropertiesBox::sdTimeline [private]

Die Zeitleiste für zeitbezogene Sensordaten.

Definiert in Zeile 252 der Datei PropertiesBox.h.

8.27.4.25 wxStaticText* PropertiesBox::sensorDataLbl [private]

Beschriftung für das Sensordatensatz-Auswahlfeld.

Definiert in Zeile 197 der Datei PropertiesBox.h.

8.27.4.26 wxComboBox* PropertiesBox::sensorDataList [private]

Auswahlfeld für den zu verwendenden Sensordatensatz.

Definiert in Zeile 202 der Datei PropertiesBox.h.

8.27.4.27 wxTextCtrl* PropertiesBox::specificHeatCapEdit [private]

Eingabefeld für die spezifische Wärmekapazität.

Definiert in Zeile 242 der Datei PropertiesBox.h.

8.27.4.28 wxStaticText* PropertiesBox::specificHeatCapLbl [private]

Beschriftung für das Wärmekapazitäts-Eingabefeld.

Definiert in Zeile 247 der Datei PropertiesBox.h.

8.27.4.29 wxStaticText* PropertiesBox::upToDateLbl [private]

Beschiftung für die Warnung bei geänderten Objekteigenschaften.

Definiert in Zeile 172 der Datei PropertiesBox.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

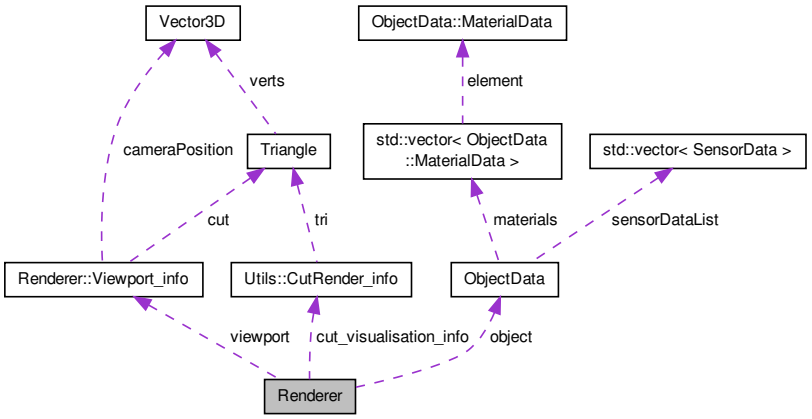
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.cpp

8.28 Renderer Klassenreferenz

Zeichnet den Inhalt der 3D-Fensters.

`#include <Renderer.h>`

Zusammengehörigkeiten von Renderer:



Klassen

- struct Viewport_info

  Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.

Öffentliche Typen

- enum RenderMode { RM_NONE = 0, RM_MATERIALCOLOR, RM_VALUECOLOR }

  Darstellungsmodus für Elemente /Punkte, Kanten, Flächen) des 3D-Objekts.

Öffentliche Methoden

- Renderer ()

  Der Konstruktor.
- void initGL (int width, int height)

  Initialisiert die OpenGL-Bibliothek.
- void resize (int width, int height)

  Verändert die Größe des Anzeigebereichs.
- void render ()

  Zeichnet das Objekt (Attribut object).
- void setObject (ObjectData ∗obj)

  Setzt das zu zeichnende Objekt.
- void setCutRenderInfo (CutRender_info ∗info)

Setzt die Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D--Temperaturverteilung benötigt werden.

- wxImage * getViewportImage ()

  Gibt den Inhalt der Zeichenfläche als Bild zurück.
- Viewport_info * getViewport ()

  Gibt eine Referenz auf die verwendeten Anzeigeeigenschaften zurück.
- virtual ~Renderer ()

  Der Destruktor.

Private Methoden

- void renderMaterial (ObjectData::MaterialData *mat)

  Zeichnet die Elemente eines Materials des Objekts.
- void renderTetrahedra (ObjectData::MaterialData *mat, RenderMode rendermode)

  Zeichnet die Tetraeder eines Materials des Objekts.
- void renderSensorData (vector< SensorPoint > *data)

  Zeichnet Sensordaten als Punkte.

Private Attribute

- Viewport_info viewport

  Informationen über die Darstellung des zu zeichnenden Inhalts.
- ObjectData * object

  Das darzustellende Objekt.
- CutRender_info * cut_visualisation_info

  Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D--Temperaturverteilung benötigt werden.
- int displayList

  Adresse der OpenGL-Displaylist, die die Geometriedaten auf der Grafikkarte vorhält.

8.28.1 Ausführliche Beschreibung

Zeichnet den Inhalt der 3D-Fensters.

Zeichnet das 3D-Objekt, Sensordaten und Koordinatensystem je nach Visualisierungsoptionen mithilfe der OpenGL-Bibliothek.

Definiert in Zeile 24 der Datei Renderer.h.

8.28.2 Dokumentation der Aufzählungstypen

8.28.2.1 enum Renderer::RenderMode

Darstellungsmodus für Elemente /Punkte, Kanten, Flächen) des 3D-Objekts.

Aufzählungswerte

RM_NONE

RM_MATERIALCOLOR

RM_VALUECOLOR

Definiert in Zeile 29 der Datei Renderer.h.

8.28.3 Beschreibung der Konstruktoren und Destruktoren

8.28.3.1 Renderer::Renderer ( )

Der Konstruktor.

Definiert in Zeile 19 der Datei Renderer.cpp.

8.28.3.2 Renderer::~Renderer ( ) [virtual]

Der Destruktor.

Definiert in Zeile 700 der Datei Renderer.cpp.

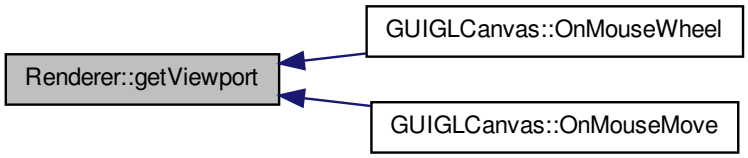
8.28.4 Dokumentation der Elementfunktionen

8.28.4.1 Renderer::Viewport_info * Renderer::getViewport ( )

Gibt eine Referenz auf die verwendeten Anzeigeeigenschaften zurück.

Definiert in Zeile 42 der Datei Renderer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.28.4.2 wxImage * Renderer::getViewportImage ( )

Gibt den Inhalt der Zeichenfläche als Bild zurück.

Definiert in Zeile 532 der Datei Renderer.cpp.

8.28.4.3 void Renderer::initGL (int *width*, int *height*)

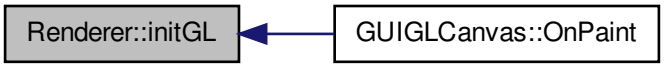
Initialisiert die OpenGL-Bibliothek.

Parameter

width	Breite des Anzeigebereichs.
height	Höhe des Anzeigebereichs.

Definiert in Zeile 346 der Datei Renderer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

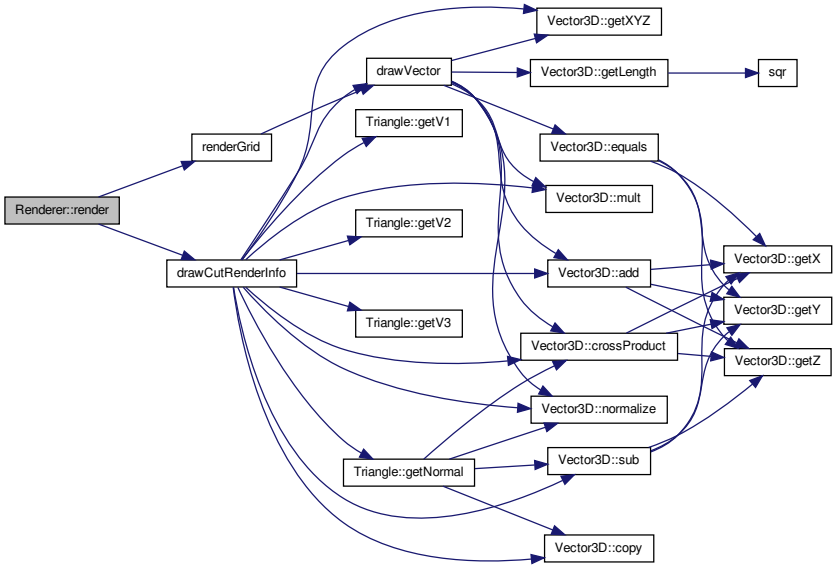


8.28.4.4 void Renderer::render ( )

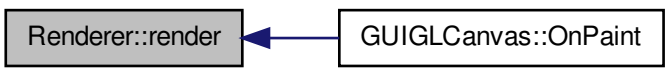
Zeichnet das Objekt (Attribut object).

Definiert in Zeile 657 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.28.4.5 void Renderer::renderMaterial (ObjectData::MaterialData * *mat*) [private]

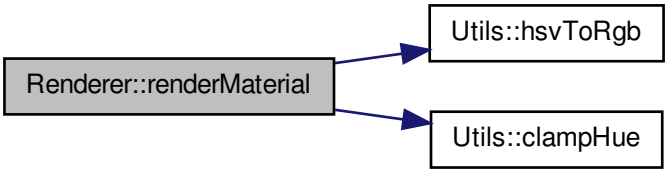
Zeichnet die Elemente eines Materials des Objekts.

Parameter

mat	Das zu zeichnende Material.

Definiert in Zeile 274 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.28.4.6 void Renderer::renderSensorData (vector< SensorPoint > * *data*) [private]

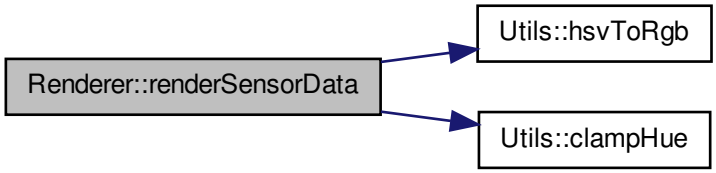
Zeichnet Sensordaten als Punkte.

Parameter

data	Sensordaten als Liste von Punkten.

Definiert in Zeile 236 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.28.4.7 void Renderer::renderTetrahedra (ObjectData::MaterialData * *mat*, RenderMode *rendermode*) [private]

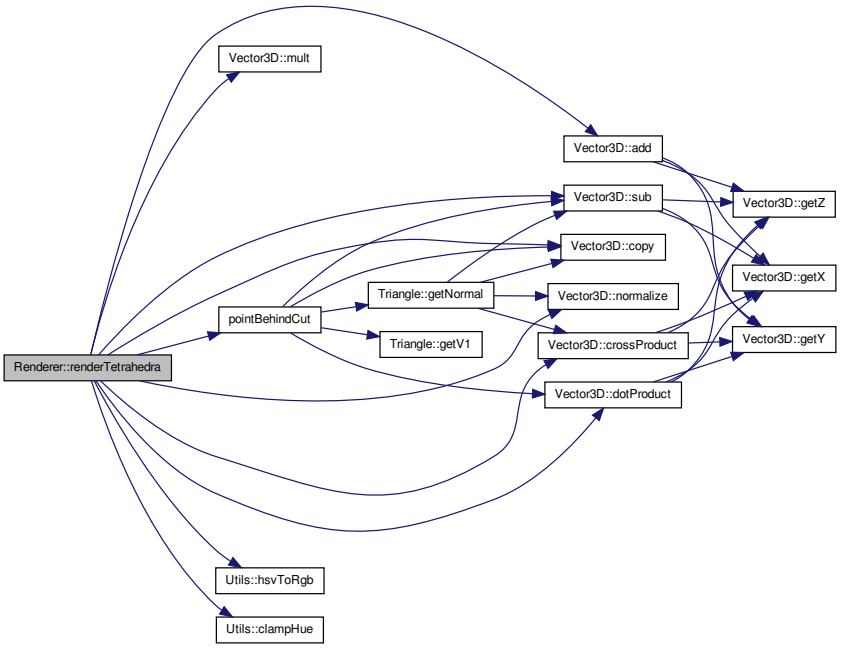
Zeichnet die Tetraeder eines Materials des Objekts.

Parameter

mat	Das zu zeichnende Material.
rendermode	Der zu verwendende Zeichenmodus.

Definiert in Zeile 94 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.28.4.8 void Renderer::resize (int *width*, int *height*)

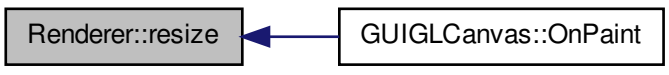
Verändert die Größe des Anzeigebereichs.

Parameter

width	Neue Breite des Anzeigebereichs.
height	Neue Höhe des Anzeigebereichs.

Definiert in Zeile 353 der Datei Renderer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.28.4.9 void Renderer::setCutRenderInfo (CutRender_info * *info*)

Setzt die Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D--Temperaturverteilung benötigt werden.

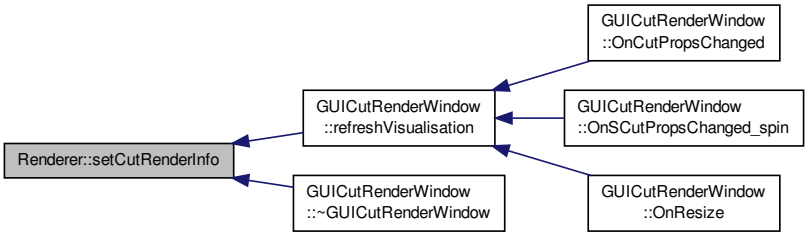
NULL bedeutet keine Visualisierung.

Parameter

info	Die Eigenschaften der 2D-Temperaturverteilung.

Definiert in Zeile 489 der Datei Renderer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.28.4.10 void Renderer::setObject (ObjectData * *obj*)

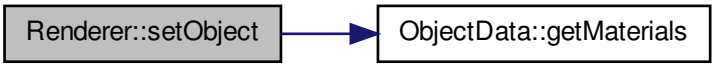
Setzt das zu zeichnende Objekt.

Parameter

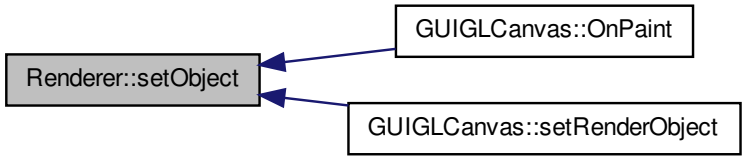
obj	Das zu zeichnende Objekt.

Definiert in Zeile 503 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.28.5 Dokumentation der Datenelemente

8.28.5.1 CutRender_info* Renderer::cut_visualisation_info [private]

Eigenschaften einer 2D-Temperaturverteilung, welche teilweise zur Visualisierung der Ebene der 2D--Temperaturverteilung benötigt werden.

NULL bedeutet keine Visualisierung.

Definiert in Zeile 142 der Datei Renderer.h.

8.28.5.2 int Renderer::displayList [private]

Adresse der OpenGL-Displaylist, die die Geometriedaten auf der Grafikkarte vorhält.

Definiert in Zeile 147 der Datei Renderer.h.

8.28.5.3 ObjectData* Renderer::object [private]

Das darzustellende Objekt.

Definiert in Zeile 135 der Datei Renderer.h.

8.28.5.4 Viewport_info Renderer::viewport [private]

Informationen über die Darstellung des zu zeichnenden Inhalts.

Definiert in Zeile 130 der Datei Renderer.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

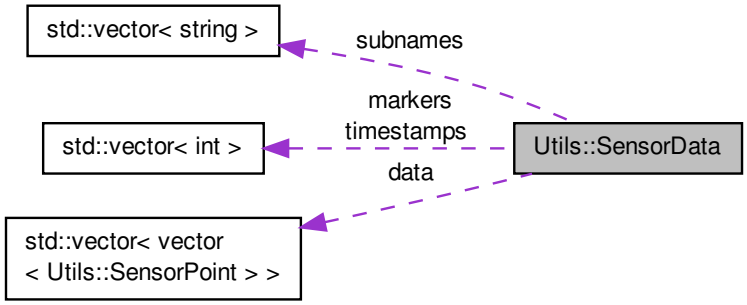
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.cpp

8.29 Utils::SensorData Strukturreferenz

Ein Sensordatensatz.

`#include <utils.h>`

Zusammengehörigkeiten von Utils::SensorData:



Öffentliche Attribute

- vector< vector< SensorPoint > > data

  Daten des Datensatzes (Sensorpunkte zu versch.
- vector< string > subnames

  Namen der einzelnen Zeitpunkte.
- vector< int > timestamps

  Zeitstempel der einzelnen Zeitpunkte.
- vector< int > markers

  Markierte Zeitpunkte.
- bool timed

  Sind die Sensordaten zeitbezogen? Wenn nein, ist die Länge von data 1.
- int current_time_index

  Index des aktuell ausgewählten Zeitpunkts.
- string name

  Name des Sensordatensatzes.

8.29.1 Ausführliche Beschreibung

Ein Sensordatensatz.

Definiert in Zeile 89 der Datei utils.h.

8.29.2 Dokumentation der Datenelemente

8.29.2.1 int Utils::SensorData::current_time_index

Index des aktuell ausgewählten Zeitpunkts.

Definiert in Zeile 95 der Datei utils.h.

8.29.2.2 vector<vector<SensorPoint> > Utils::SensorData::data

Daten des Datensatzes (Sensorpunkte zu versch.

Zeitpunkten).

Definiert in Zeile 90 der Datei utils.h.

8.29.2.3 vector<int> Utils::SensorData::markers

Markierte Zeitpunkte.

Definiert in Zeile 93 der Datei utils.h.

8.29.2.4 string Utils::SensorData::name

Name des Sensordatensatzes.

Definiert in Zeile 96 der Datei utils.h.

8.29.2.5 vector<string> Utils::SensorData::subnames

Namen der einzelnen Zeitpunkte.

Definiert in Zeile 91 der Datei utils.h.

8.29.2.6 bool Utils::SensorData::timed

Sind die Sensordaten zeitbezogen? Wenn nein, ist die Länge von data 1.

Definiert in Zeile 94 der Datei utils.h.

8.29.2.7 vector<int> Utils::SensorData::timestamps

Zeitstempel der einzelnen Zeitpunkte.

Definiert in Zeile 92 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h

8.30 Utils::SensorPoint Strukturreferenz

Daten eines Sensordatenpunktes.

`#include <utils.h>`

Öffentliche Attribute

- double coords [3]

  Koordinaten des Punktes.
- double temperature

  Temperatur des Punktes.

8.30.1 Ausführliche Beschreibung

Daten eines Sensordatenpunktes.

Definiert in Zeile 70 der Datei utils.h.

8.30.2 Dokumentation der Datenelemente

8.30.2.1 double Utils::SensorPoint::coords[3]

Koordinaten des Punktes.

Definiert in Zeile 71 der Datei utils.h.

8.30.2.2 double Utils::SensorPoint::temperature

Temperatur des Punktes.

Definiert in Zeile 72 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h

8.31 Utils::SensorPointComparator Strukturreferenz

Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.

`#include <utils.h>`

Öffentliche Methoden

- double getDistance_d (double *p1, double *p2)

  Berechnet den Abstand zwischen zwei Punkten mithilfe des Satzes des Pythagoras.
- bool operator() (SensorPoint p1, SensorPoint p2)

  Vergleichsoperator für den Abstand zum Punkt meshpoint.

Öffentliche Attribute

- double meshpoint [3]

  Punkt, zu dem der Abstand ermittelt werden soll.

8.31.1 Ausführliche Beschreibung

Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.

Der Punkt, zu dem der Abstand berechnet werden soll ist in der Struktur gespeichert. Zum Vergleich zweier Messpunkte wird dann für beide der Abstand berechnet.

Wird für Sortieralgorithmen der Standardbibliothek benötigt.

Definiert in Zeile 107 der Datei utils.h.

8.31.2 Dokumentation der Elementfunktionen

8.31.2.1 double Utils::SensorPointComparator::getDistance_d (double * *p1*, double * *p2*) [inline]

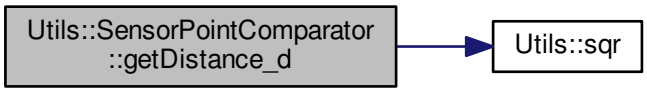
Berechnet den Abstand zwischen zwei Punkten mithilfe des Satzes des Pythagoras.

Parameter

p1	Koordinaten des ersten Punkten als Liste dreier Koordinaten.
p2	Koordinaten des zweiten Punkten als Liste dreier Koordinaten.

Definiert in Zeile 113 der Datei utils.h.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.31.2.2 bool Utils::SensorPointComparator::operator() (SensorPoint *p1*, SensorPoint *p2*) [inline]

Vergleichsoperator für den Abstand zum Punkt meshpoint.

Definiert in Zeile 120 der Datei utils.h.

8.31.3 Dokumentation der Datenelemente

8.31.3.1 double Utils::SensorPointComparator::meshpoint[3]

Punkt, zu dem der Abstand ermittelt werden soll.

Definiert in Zeile 116 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

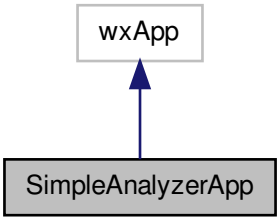
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h

8.32 SimpleAnalyzerApp Klassenreferenz

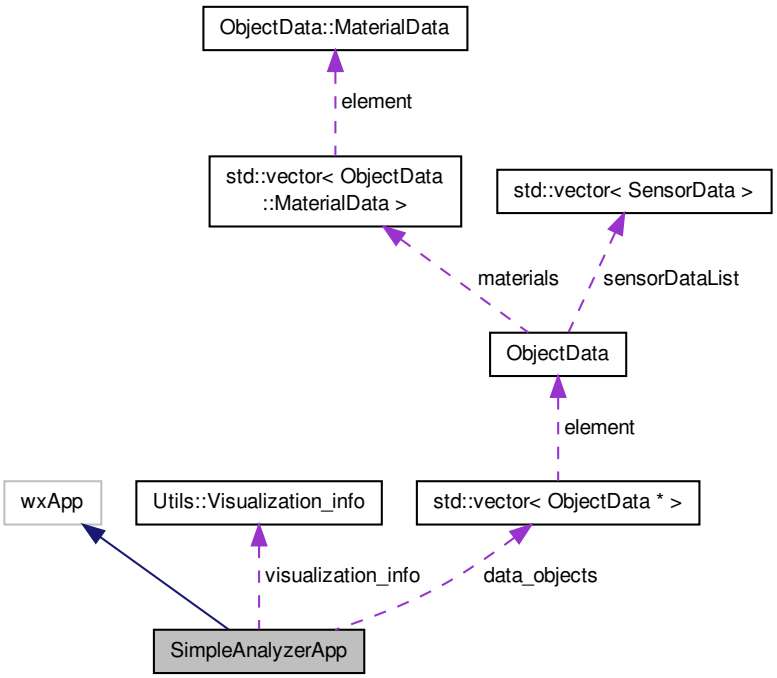
Regelt den allgemeinen Ablauf des Programms.

`#include <SimpleAnalyzerApp.h>`

Klassendiagramm für SimpleAnalyzerApp:



Zusammengehörigkeiten von SimpleAnalyzerApp:



Öffentliche Methoden

- int getCurrentDataObjectIndex ()

  Gibt den Index des aktiven Objekts zurück.
- void setCurrentDataObjectIndex (int currentDataObjectIndex)

  Setzt den Index des aktiven Objekts.
- vector< ObjectData * > * getDataObjects ()

  Gibt einen Verweis auf die Liste der geladenen Objekte zurück.
- Utils::Visualization_info * getVisualizationInfo ()

Gibt einen Verweis auf verwendeten Visualisierungsoptionen zurück.

- ObjectData * getActiveObject ()

  Gibt einen Verweis auf das aktuell aktive Objekt zurück.

- void addObject (ObjectData *obj)

  Fügt ein Objekt zur Objektliste hinzu.

- void removeCurrentObject ()

  Löscht das aktuelle Objekt aus der Objektliste.

- virtual ~SimpleAnalyzerApp ()

  Der Destruktor.

Private Methoden

- virtual bool OnInit ()

  Wird beim Start der Anwendung ausgeführt und öffnet das Hauptfenster.

Private Attribute

- vector< ObjectData * > data_objects

  Liste aller geladenen Objekte.

- int current_data_object_index = -1

  Index des aktuellen Objekts.

- Utils::Visualization_info visualization_info

  Die allgemein verwendeten Visualisierungsoptionen.

8.32.1 Ausführliche Beschreibung

Regelt den allgemeinen Ablauf des Programms.

Eine eigene Anwendungsklasse wird von wxWidgets gefordert. Das zugrunde liegende System organisiert über diese Klasse den Programmablauf (MainLoop) und Events.

Definiert in Zeile 23 der Datei SimpleAnalyzerApp.h.

8.32.2 Beschreibung der Konstruktoren und Destruktoren

8.32.2.1 SimpleAnalyzerApp::~SimpleAnalyzerApp ( ) [virtual]

Der Destruktor.

Definiert in Zeile 88 der Datei SimpleAnalyzerApp.cpp.

8.32.3 Dokumentation der Elementfunktionen

8.32.3.1 void SimpleAnalyzerApp::addObject (ObjectData * *obj*)

Fügt ein Objekt zur Objektliste hinzu.

Parameter

obj	Das hinzuzufügende Objekt.

Definiert in Zeile 53 der Datei SimpleAnalyzerApp.cpp.

8.32.3.2 ObjectData * SimpleAnalyzerApp::getActiveObject ( )

Gibt einen Verweis auf das aktuell aktive Objekt zurück.

Rückgabe

Pointer auf das aktuell aktive Objekt.

Definiert in Zeile 49 der Datei SimpleAnalyzerApp.cpp.

8.32.3.3 int SimpleAnalyzerApp::getCurrentDataObjectIndex ( )

Gibt den Index des aktiven Objekts zurück.

Rückgabe

Der Index des aktiven Objetks.

Definiert in Zeile 33 der Datei SimpleAnalyzerApp.cpp.

8.32.3.4 vector< ObjectData * > * SimpleAnalyzerApp::getDataObjects ( )

Gibt einen Verweis auf die Liste der geladenen Objekte zurück.

Rückgabe

Pointer zur Liste der geladenen Objekte.

Definiert in Zeile 41 der Datei SimpleAnalyzerApp.cpp.

8.32.3.5 Utils::Visualization_info * SimpleAnalyzerApp::getVisualizationInfo ( )

Gibt einen Verweis auf verwendeten Visualisierungsoptionen zurück.

Rückgabe

Pointer zu den verwendeten Visualisierungsoptionen.

Definiert in Zeile 45 der Datei SimpleAnalyzerApp.cpp.

8.32.3.6 bool SimpleAnalyzerApp::OnInit ( ) [private], [virtual]

Wird beim Start der Anwendung ausgeführt und öffnet das Hauptfenster.

Definiert in Zeile 72 der Datei SimpleAnalyzerApp.cpp.

8.32.3.7 void SimpleAnalyzerApp::removeCurrentObject ( )

Löscht das aktuelle Objekt aus der Objektliste.

Definiert in Zeile 57 der Datei SimpleAnalyzerApp.cpp.

8.32.3.8 void SimpleAnalyzerApp::setCurrentDataObjectIndex (int *currentDataObjectIndex*)

Setzt den Index des aktiven Objekts.

Parameter

currentData-ObjectIndex	Index des auszuwählenden Objekts.

Definiert in Zeile 37 der Datei SimpleAnalyzerApp.cpp.

8.32.4 Dokumentation der Datenelemente

8.32.4.1 int SimpleAnalyzerApp::current_data_object_index = -1 [private]

Index des aktuellen Objekts.

Definiert in Zeile 79 der Datei SimpleAnalyzerApp.h.

8.32.4.2 vector<ObjectData*> SimpleAnalyzerApp::data_objects [private]

Liste aller geladenen Objekte.

Definiert in Zeile 74 der Datei SimpleAnalyzerApp.h.

8.32.4.3 Utils::Visualization_info SimpleAnalyzerApp::visualization_info [private]

Die allgemein verwendeten Visualisierungsoptionen.

Definiert in Zeile 84 der Datei SimpleAnalyzerApp.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.cpp

8.33 Utils::SortStruct Strukturreferenz

Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.

`#include <utils.h>`

Öffentliche Attribute

- double distance

  Abstand des Punktes.
- int pointIndex

  Index des entsprechenden Sensordatenpuntkes.

8.33.1 Ausführliche Beschreibung

Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.

Definiert in Zeile 55 der Datei utils.h.

8.33.2 Dokumentation der Datenelemente

8.33.2.1 double Utils::SortStruct::distance

Abstand des Punktes.

Definiert in Zeile 56 der Datei utils.h.

8.33.2.2 int Utils::SortStruct::pointIndex

Index des entsprechenden Sensordatenpuntkes.

Definiert in Zeile 57 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

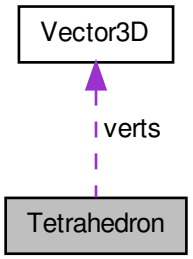
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h

8.34 Tetrahedron Klassenreferenz

Ein durch 4 Ortsvektoren beschriebener Tetraeder.

`#include <GeometryClasses.h>`

Zusammengehörigkeiten von Tetrahedron:



Öffentliche Methoden

- Tetrahedron (Vector3D *v1, Vector3D *v2, Vector3D *v3, Vector3D *v4)
  Der Konstruktor.
- Vector3D * getV1 ()
  Gibt eine Referenz auf den Ortsvektor zum 1.
- Vector3D * getV2 ()
  Gibt eine Referenz auf den Ortsvektor zum 2.
- Vector3D * getV3 ()
  Gibt eine Referenz auf den Ortsvektor zum 3.
- Vector3D * getV4 ()
  Gibt eine Referenz auf den Ortsvektor zum 3.
- Vector3D * getVert (int index)
  Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Tetraeders zurück.

Private Attribute

- Vector3D ∗ verts [4]

  Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Tetraeders.

8.34.1 Ausführliche Beschreibung

Ein durch 4 Ortsvektoren beschriebener Tetraeder.

Definiert in Zeile 285 der Datei GeometryClasses.h.

8.34.2 Beschreibung der Konstruktoren und Destruktoren

8.34.2.1 Tetrahedron::Tetrahedron (Vector3D ∗ *v1*, Vector3D ∗ *v2*, Vector3D ∗ *v3*, Vector3D ∗ *v4*)

Der Konstruktor.

Die übergebenen Vektorobjekte werden als Element der Klasse gespeichert (nicht kopiert).

Parameter

v1	Ortsvektor zum 1. Punkt des Tetraeders.
v2	Ortsvektor zum 2. Punkt des Tetraeders.
v3	Ortsvektor zum 3. Punkt des Tetraeders.
v4	Ortsvektor zum 4. Punkt des Tetraeders.

Definiert in Zeile 287 der Datei GeometryClasses.cpp.

8.34.3 Dokumentation der Elementfunktionen

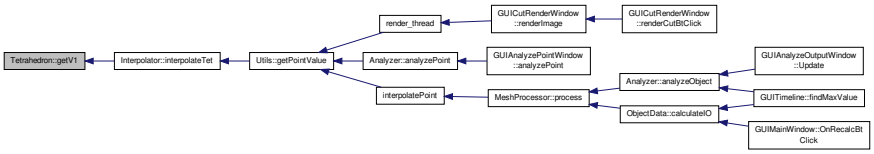
8.34.3.1 Vector3D ∗ Tetrahedron::getV1 ( )

Gibt eine Referenz auf den Ortsvektor zum 1.

Punkt des Tetraeders zurück.

Definiert in Zeile 295 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



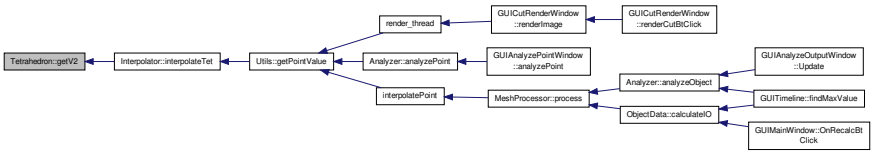
8.34.3.2 Vector3D ∗ Tetrahedron::getV2 ( )

Gibt eine Referenz auf den Ortsvektor zum 2.

Punkt des Tetraeders zurück.

Definiert in Zeile 299 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



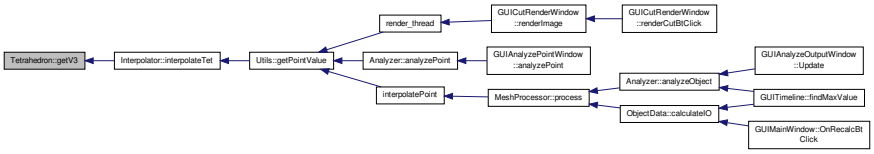
8.34.3.3 Vector3D * Tetrahedron::getV3 ( )

Gibt eine Referenz auf den Ortsvektor zum 3.

Punkt des Tetraeders zurück.

Definiert in Zeile 303 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



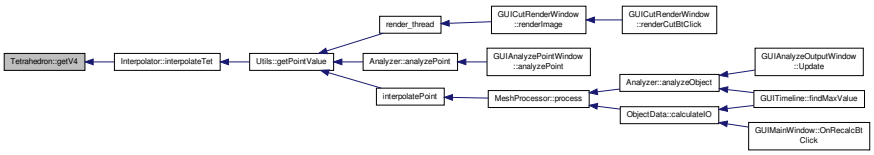
8.34.3.4 Vector3D * Tetrahedron::getV4 ( )

Gibt eine Referenz auf den Ortsvektor zum 3.

Punkt des Tetraeders zurück.

Definiert in Zeile 307 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.34.3.5 Vector3D * Tetrahedron::getVert (int *index*)

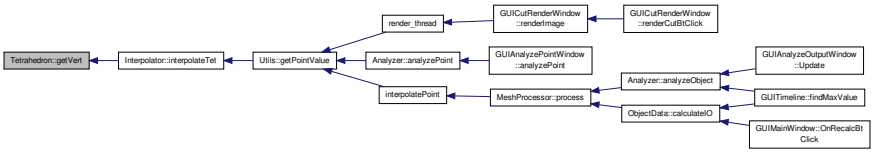
Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Tetraeders zurück.

Parameter

index	Der Index des gesuchten Punktes (0..3).

Definiert in Zeile 311 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.34.4 Dokumentation der Datenelemente

8.34.4.1 Vector3D∗ Tetrahedron::verts[4] [private]

Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Tetraeders.

Definiert in Zeile 326 der Datei GeometryClasses.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

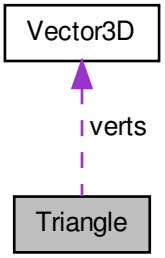
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp

8.35 Triangle Klassenreferenz

Ein durch 3 Ortsvektoren beschriebenes Dreieck.

`#include <GeometryClasses.h>`

Zusammengehörigkeiten von Triangle:



Öffentliche Methoden

- Triangle (Vector3D ∗v1, Vector3D ∗v2, Vector3D ∗v3)

  Der Konstruktor.

- Vector3D * getV1 ()

  Gibt eine Referenz auf den Ortsvektor zum ersten Punkt des Dreiecks zurück.
- Vector3D * getV2 ()

  Gibt eine Referenz auf den Ortsvektor zum zweiten Punkt des Dreiecks zurück.
- Vector3D * getV3 ()

  Gibt eine Referenz auf den Ortsvektor zum dritten Punkt des Dreiecks zurück.
- Vector3D * getVert (int index)

  Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Dreiecks zurück.
- void print ()

  Gibt die Punkte des Dreiecks auf dem cout-Stream aus.
- Vector3D * getNormal ()

  Gibt die Normale des Dreiecks zurück.
- ~Triangle ()

  Der Destruktor.

Private Attribute

- Vector3D * verts [3]

  Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Dreiecks.

8.35.1 Ausführliche Beschreibung

Ein durch 3 Ortsvektoren beschriebenes Dreieck.

Definiert in Zeile 228 der Datei GeometryClasses.h.

8.35.2 Beschreibung der Konstruktoren und Destruktoren

8.35.2.1 Triangle::Triangle (Vector3D * *v1*, Vector3D * *v2*, Vector3D * *v3*)

Der Konstruktor.

Die übergebenen Vektorobjekte werden als Element der Klasse gespeichert (nicht kopiert).

Parameter

v1	Ortsvektor zum 1. Punkt des Dreiecks.
v2	Ortsvektor zum 2. Punkt des Dreiecks.
v3	Ortsvektor zum 3. Punkt des Dreiecks.

Definiert in Zeile 240 der Datei GeometryClasses.cpp.

8.35.2.2 Triangle::~Triangle ( )

Der Destruktor.

Definiert in Zeile 283 der Datei GeometryClasses.cpp.

8.35.3 Dokumentation der Elementfunktionen

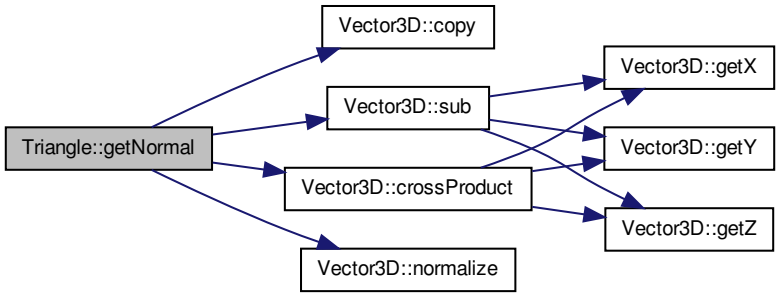
8.35.3.1 Vector3D * Triangle::getNormal ( )

Gibt die Normale des Dreiecks zurück.

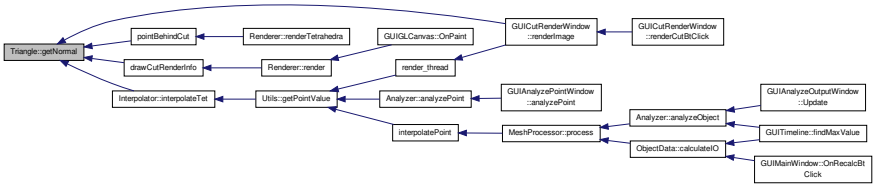
Der zurückgegebene Vektor muss manuell mit delete Freigegeben werden!

Definiert in Zeile 262 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

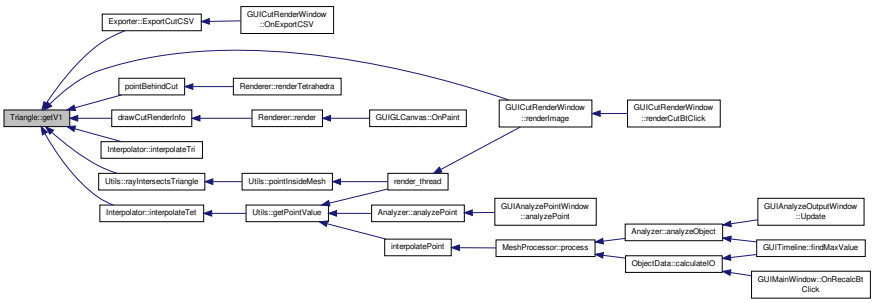


8.35.3.2 Vector3D ∗ Triangle::getV1 ( )

Gibt eine Referenz auf den Ortsvektor zum ersten Punkt des Dreiecks zurück.

Definiert in Zeile 246 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

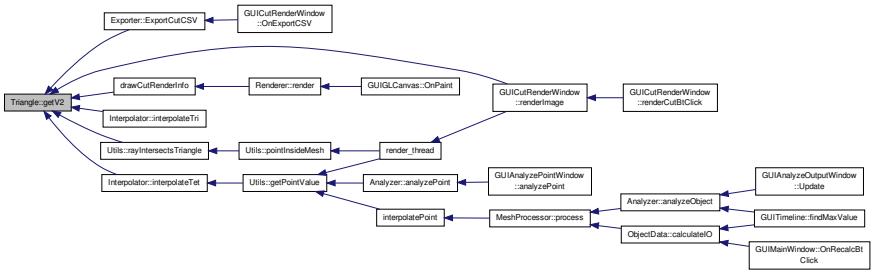


8.35.3.3 Vector3D ∗ Triangle::getV2 ( )

Gibt eine Referenz auf den Ortsvektor zum zweiten Punkt des Dreiecks zurück.

Definiert in Zeile 250 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

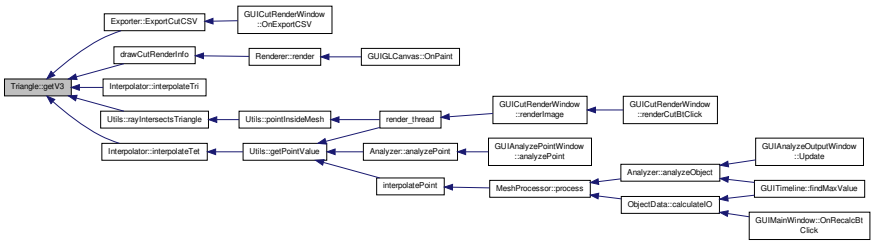


8.35.3.4 Vector3D * Triangle::getV3 ( )

Gibt eine Referenz auf den Ortsvektor zum dritten Punkt des Dreiecks zurück.

Definiert in Zeile 254 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.35.3.5 Vector3D * Triangle::getVert (int *index*)

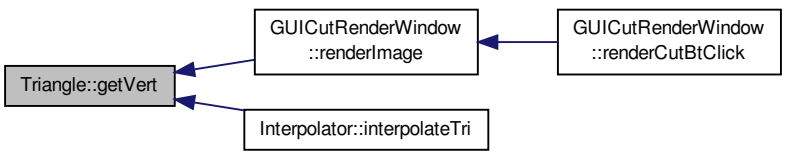
Gibt eine Referenz auf den Ortsvektor zum index+1 Punkt des Dreiecks zurück.

Parameter

index	Der Index des gesuchten Punktes (0..2).

Definiert in Zeile 258 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.35.3.6 void Triangle::print ( )

Gibt die Punkte des Dreiecks auf dem cout-Stream aus.

Definiert in Zeile 274 der Datei GeometryClasses.cpp.

8.35.4 Dokumentation der Datenelemente

8.35.4.1 Vector3D* Triangle::verts[3] [private]

Die Referenzen auf die Ortsvektoren zu den Eckpunkten des Dreiecks.

Definiert in Zeile 278 der Datei GeometryClasses.h.

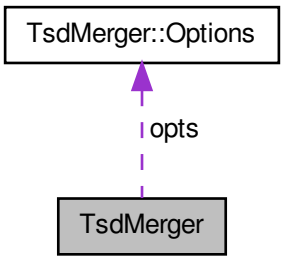
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp

8.36 TsdMerger Klassenreferenz

Zusammenführen zweier .tsd-Dateien.

Zusammengehörigkeiten von TsdMerger:



Klassen

- struct Options

  Strunktur für die Programmeinstellungen.

Öffentliche Methoden

- int merge (int argc, char *argv[])

  Liest die Programmargumente um die Eingabedateien anhand der Zeitstempel in eine .tsd-Datei zusammen zu führen.

Geschützte Methoden

- string getTextBlock (string data, int n)

  Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.
- int parseFile (string filename, vector< long > &timestamps, vector< string > &names, vector< string > &data)

  Sammelt Daten aus einer .tsd-Datei.
- bool parseArguments (int argc, char *argv[], string &input1, string &input2, string &output_file)

  Wertet die Programmargumente aus.
- bool writeOutputFile (string path, vector< long > &timestamps1, vector< string > &names1, vector< string > &data1, vector< long > &timestamps2, vector< string > &names2, vector< string > &data2)

  Schreibt die Ausgabedatei.

Geschützte Attribute

- struct TsdMerger::Options opts

  Hält die verwendeten Programmeinstellungen.

8.36.1 Ausführliche Beschreibung

Zusammenführen zweier .tsd-Dateien.

Definiert in Zeile 21 der Datei mergetsd.cpp.

8.36.2 Dokumentation der Elementfunktionen

8.36.2.1 string TsdMerger::getTextBlock (string *data*, int *n*) [inline], [protected]

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

Parameter

data	Der Ausgansstring.
n	Index des zu findenden Blocks.

Rückgabe

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 40 der Datei mergetsd.cpp.

8.36.2.2 int TsdMerger::merge (int *argc*, char * *argv[]*) [inline]

Liest die Programmargumente um die Eingabedateien anhand der Zeitstempel in eine .tsd-Datei zusammen zu führen.

Wird duch die Funktion main() von außerhalb des Namespaces aufgerufen.

Parameter

argc	Anzahl der Programmargumente.

argv	Die Programmargumente.

Definiert in Zeile 313 der Datei mergetsd.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

8.36.2.3 bool TsdMerger::parseArguments (int *argc,* char * *argv[],* string & *input1,* string & *input2,* string & *output_file*) [inline], [protected]

Wertet die Programmargumente aus.

Parameter

argc	Anzahl der Programmargumente.
argv	Die Programmargumente.
input1	Ausgabe für den Pfad zur Eingabedatei 1.
input2	Ausgabe für den Pfad zur Eingabedatei 2.
output_file	Ausgabe für den Pfad zur Ausgabedatei.

Rückgabe

Soll das Programm weiter ablaufen?

Definiert in Zeile 138 der Datei mergetsd.cpp.

8.36.2.4 int TsdMerger::parseFile (string *filename,* vector< long > & *timestamps,* vector< string > & *names,* vector< string > & *data*) [inline], [protected]

Sammelt Daten aus einer .tsd-Datei.

Parameter

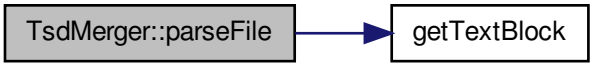
filename	Der Pfad zur .tsd-Datei.
timestamps	Ausgabevariable für die Zeitstempel der Datensätze.
names	Ausgabevariable für den Namen der Datensätze.
data	Ausgabevariable für die Sensordaten der Datensätze.

Rückgabe

Gibt 0 bei Erfolg zurück, 1, wenn die Datei nicht gefunden werden konnte.

Definiert in Zeile 80 der Datei mergetsd.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.36.2.5 bool TsdMerger::writeOutputFile (string *path,* vector< long > & *timestamps1,* vector< string > & *names1,* vector< string > & *data1,* vector< long > & *timestamps2,* vector< string > & *names2,* vector< string > & *data2*) [inline], [protected]

Schreibt die Ausgabedatei.

Parameter

path	Der Pfad zur Ausgabedatei.
timestamps1	Zeitstempel der Datensätze der ersten Datei.
timestamps2	Zeitstempel der Datensätze der zweiten Datei.
names1	Namen der Datensätze der ersten Datei.
names2	Namen der Datensätze der zweiten Datei.
data1	Daten der Datensätze der ersten Datei.
data2	Daten der Datensätze der zweiten Datei.

Rückgabe

War das Schreiben erfolgreich?

Definiert in Zeile 256 der Datei mergetsd.cpp.

8.36.3 Dokumentation der Datenelemente

8.36.3.1 struct TsdMerger::Options TsdMerger::opts [protected]

Hält die verwendeten Programmeinstellungen.

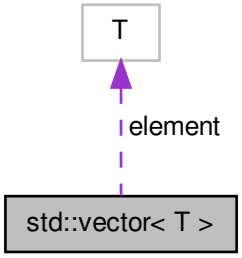
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/mergetsd/src/mergetsd.cpp

8.37 std::vector< T > Template-Klassenreferenz

```
#include <doxygen_dep_dummy.h>
```

Zusammengehörigkeiten von std::vector< T >:



Öffentliche Attribute

- T element

8.37.1 Ausführliche Beschreibung

template<class T>class std::vector< T >

STL vector class

Definiert in Zeile 3 der Datei doxygen_dep_dummy.h.

8.37.2 Dokumentation der Datenelemente

8.37.2.1 template<class T> T std::vector< T >::element

Definiert in Zeile 3 der Datei doxygen_dep_dummy.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Datei:

- doxygen_dep_dummy.h

8.38 Vector3D Klassenreferenz

3D-Vektorklasse mit nützlichen Operationen.

`#include <GeometryClasses.h>`

Öffentliche Methoden

- Vector3D (double x, double y, double z)

  Konstruktor für Konstruktion aus einzelnen Koordinaten.
- Vector3D (const double *values)

  Konstruktor für Konstruktion aus einer Koordinatenliste.
- Vector3D (Vector3D *other)

Konstruktor für die Konstruktion aus einem anderen Vektor.

- Vector3D * copy ()

  Gibt eine Kopie des Vektors zurück.
- double getX ()

  Gibt das X-Element des Vektors zurück.
- double getY ()

  Gibt das Y-Element des Vektors zurück.
- double getZ ()

  Gibt das Z-Element des Vektors zurück.
- double getLength ()

  Gibt die Länge des Vektors zurück.
- double getAngleTo (Vector3D *other)

  Gibt den Winkel zu einem anderen Vektor in RAD zurück.
- double dotProduct (Vector3D *other)

  Gibt das Skalarprodukt mit einem anderen Vektor.
- Vector3D * crossProduct (Vector3D *other)

  Gibt das Kreuzprodukt mit einem anderen Vektor zurück.
- void add (Vector3D *other)

  Addiert einen Vektor zu diesem Vektor.
- void sub (Vector3D *other)

  Subtrahiert einen Vektor von diesem Vektor.
- void mult (double scalar)

  Multipliziert den Vektor mit einem Skalar.
- void normalize ()

  Normalisiert den Vektor.
- bool equals (Vector3D *other)

  Testet, ob zwei Vektoren identisch sind.
- double getDistanceTo (Vector3D *other)

  Gibt den Abstand zu einem anderen Vektor zurück.
- double * getXYZ ()

  Gibt eine Referenz auf die Vektorelemente zurück (Vor allem zur Übergabe an OpenGL verwendet).
- void print ()

  Gibt den Vektor auf dem cout-Stream aus.
- void printTo (std::ostream &stream) const

  Gibt den Vektor auf dem gegebenen Stream aus.
- virtual ~Vector3D ()

  Der Destruktor.

Private Attribute

- double coords [3]

  Die Elemente des Vektors.

Freundbeziehungen

- std::ostream & operator<< (std::ostream &out, const Vector3D &vec)

  Definition des <<-Operators für die Ausgabe eines Vektors.

8.38.1 Ausführliche Beschreibung

3D-Vektorklasse mit nützlichen Operationen.

Definiert in Zeile 13 der Datei GeometryClasses.h.

8.38.2 Beschreibung der Konstruktoren und Destruktoren

8.38.2.1 Vector3D::Vector3D (double *x*, double *y*, double *z*)

Konstruktor für Konstruktion aus einzelnen Koordinaten.

Parameter

x	X-Element des Vektors.
y	Y-Element des Vektors.
z	Z-Element des Vektors.

Definiert in Zeile 19 der Datei GeometryClasses.cpp.

8.38.2.2 Vector3D::Vector3D (const double * *values*)

Konstruktor für Konstruktion aus einer Koordinatenliste.

Parameter

values	Liste der Koordinaten (x,y und z-Wert).

Definiert in Zeile 25 der Datei GeometryClasses.cpp.

8.38.2.3 Vector3D::Vector3D (Vector3D * *other*)

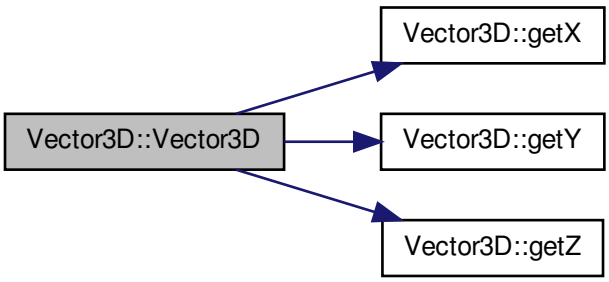
Konstruktor für die Konstruktion aus einem anderen Vektor.

Parameter

other	Der Vektor, dessen Eigenschaften übernommen werden sollen.

Definiert in Zeile 31 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



8.38.2.4 Vector3D::~Vector3D ( ) [virtual]

Der Destruktor.

Definiert in Zeile 137 der Datei GeometryClasses.cpp.

8.38.3 Dokumentation der Elementfunktionen

8.38.3.1 void Vector3D::add (Vector3D * *other*)

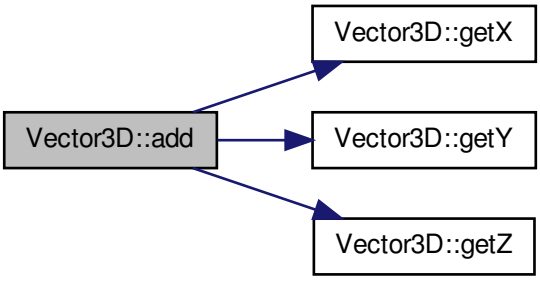
Addiert einen Vektor zu diesem Vektor.

Parameter

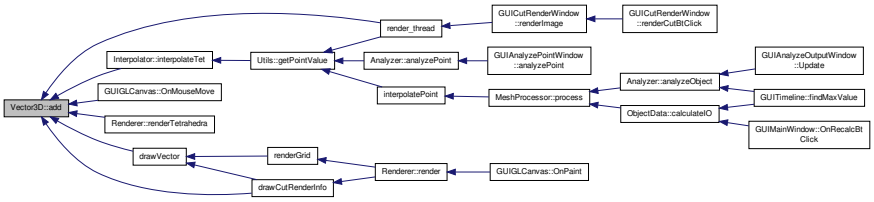
other	Der zu addierende Vektor.

Definiert in Zeile 88 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



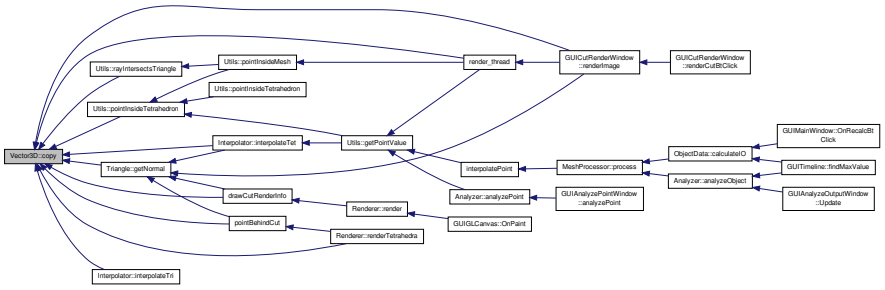
8.38.3.2 Vector3D * Vector3D::copy ( )

Gibt eine Kopie des Vektors zurück.

Der zurückgegebene Vektor muss manuell mit delete Freigegeben werden!

Definiert in Zeile 37 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.3 Vector3D * Vector3D::crossProduct (Vector3D * *other*)

Gibt das Kreuzprodukt mit einem anderen Vektor zurück.

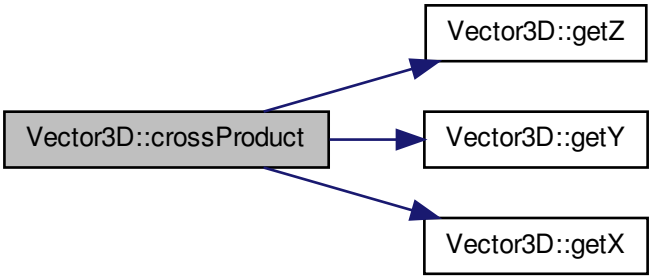
Der zurückgegebene Vektor muss manuell mit delete Freigegeben werden!

Parameter

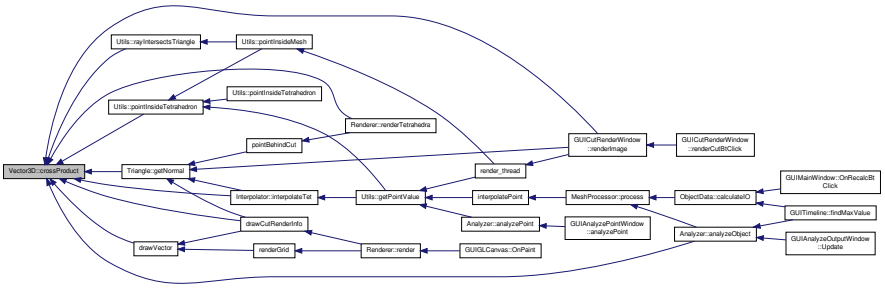
other	Der Vektor, mit dem das Kreuzprodukt gebildet werden soll.

Definiert in Zeile 81 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.4 double Vector3D::dotProduct (Vector3D ∗ *other*)

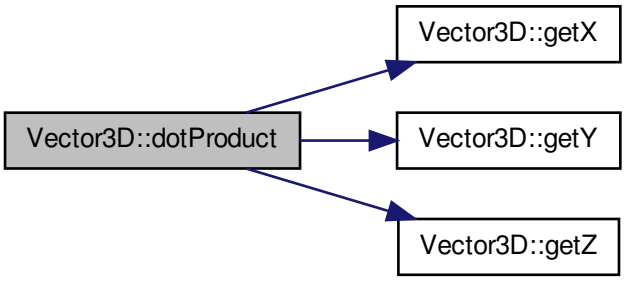
Gibt das Skalarprodukt mit einem anderen Vektor.

Parameter

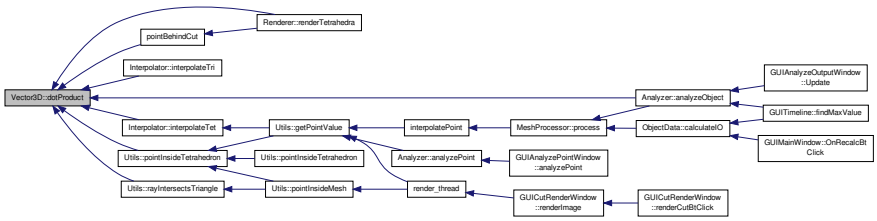
other	Der Vektor, mit dem das Skalarprodukt gebildet werden soll.

Definiert in Zeile 76 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.5 bool Vector3D::equals (Vector3D ∗ *other*)

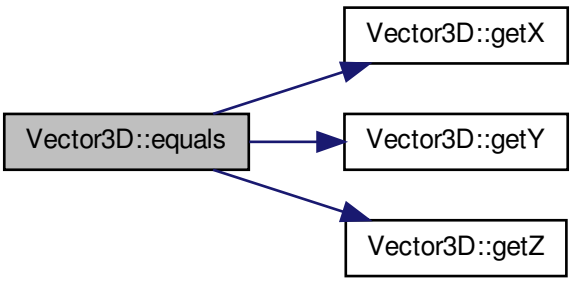
Testet, ob zwei Vektoren identisch sind.

Parameter

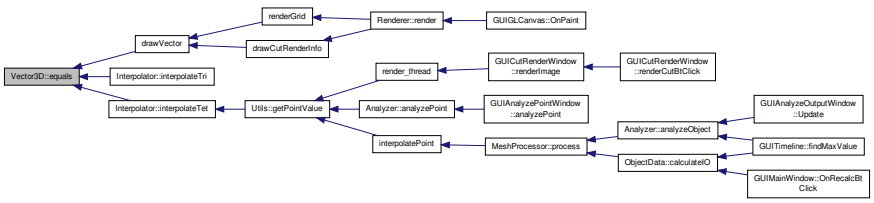
other	Der Vektor, mit dem verglichen werden soll.

Definiert in Zeile 42 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.6 double Vector3D::getAngleTo (Vector3D * *other*)

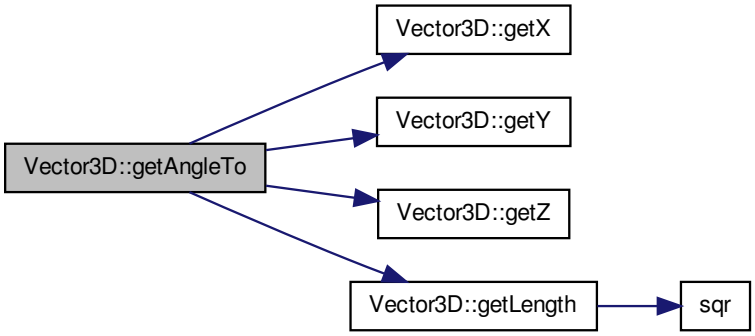
Gibt den Winkel zu einem anderen Vektor in RAD zurück.

Parameter

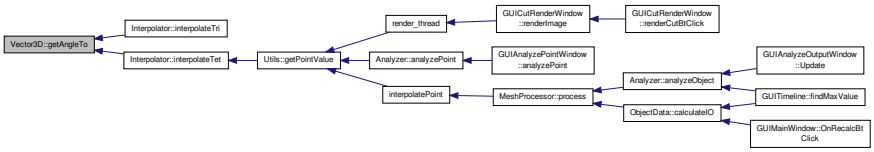
other	Der Vektor, zu dem der Winkel ermittelt werden soll.

Definiert in Zeile 64 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.7 double Vector3D::getDistanceTo (Vector3D * *other*)

Gibt den Abstand zu einem anderen Vektor zurück.

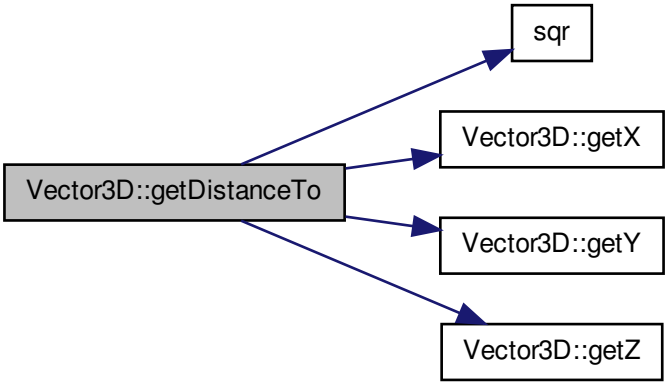
Dabei werden beide Vektoren als Ortsvektoren betrachtet.

Parameter

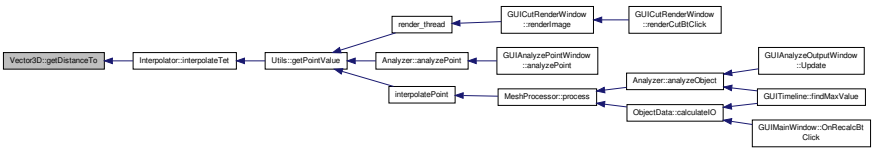
other	Der Vektor, zu dem der Abstand ermittelt werden soll.

Definiert in Zeile 117 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

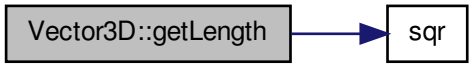


8.38.3.8 double Vector3D::getLength ( )

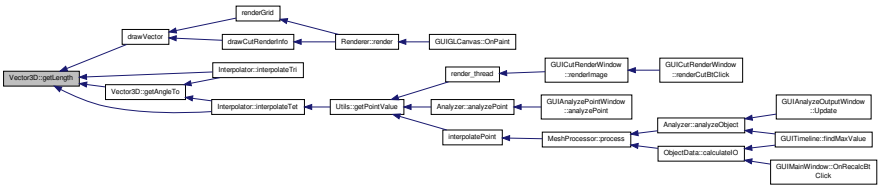
Gibt die Länge des Vektors zurück.

Definiert in Zeile 60 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

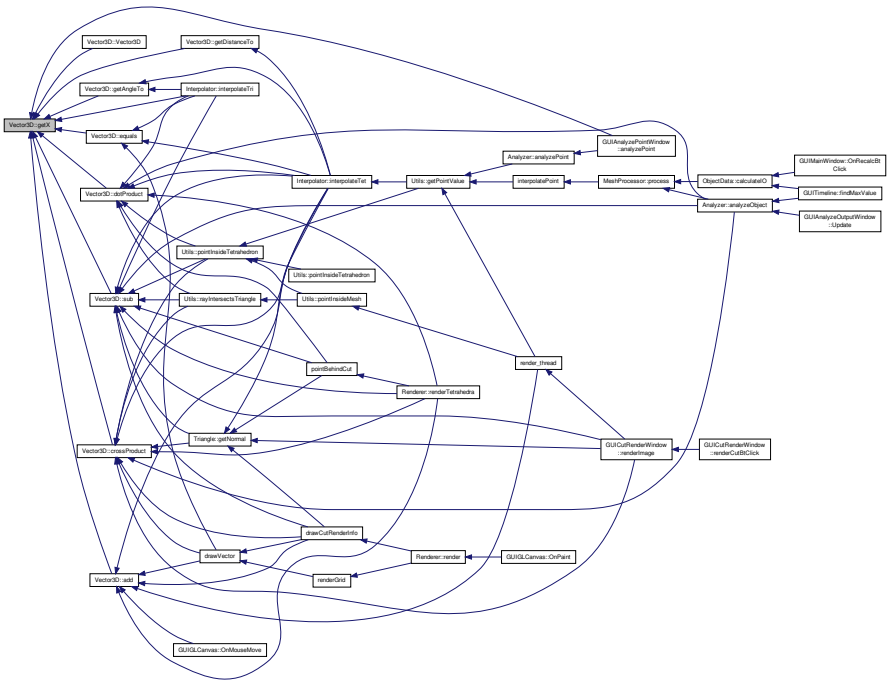


8.38.3.9 double Vector3D::getX ( )

Gibt das X-Element des Vektors zurück.

Definiert in Zeile 48 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

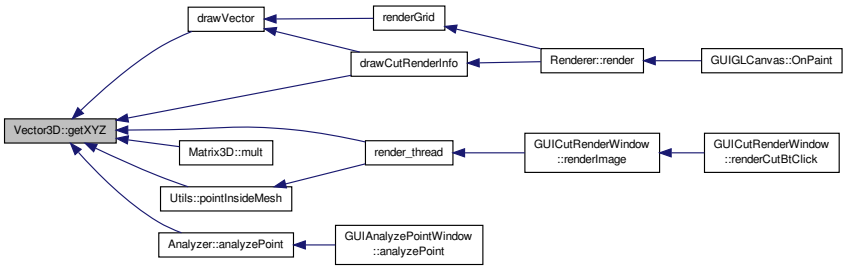


8.38.3.10 double * Vector3D::getXYZ ( )

Gibt eine Referenz auf die Vektorelemente zurück (Vor allem zur Übergabe an OpenGL verwendet).

Definiert in Zeile 113 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

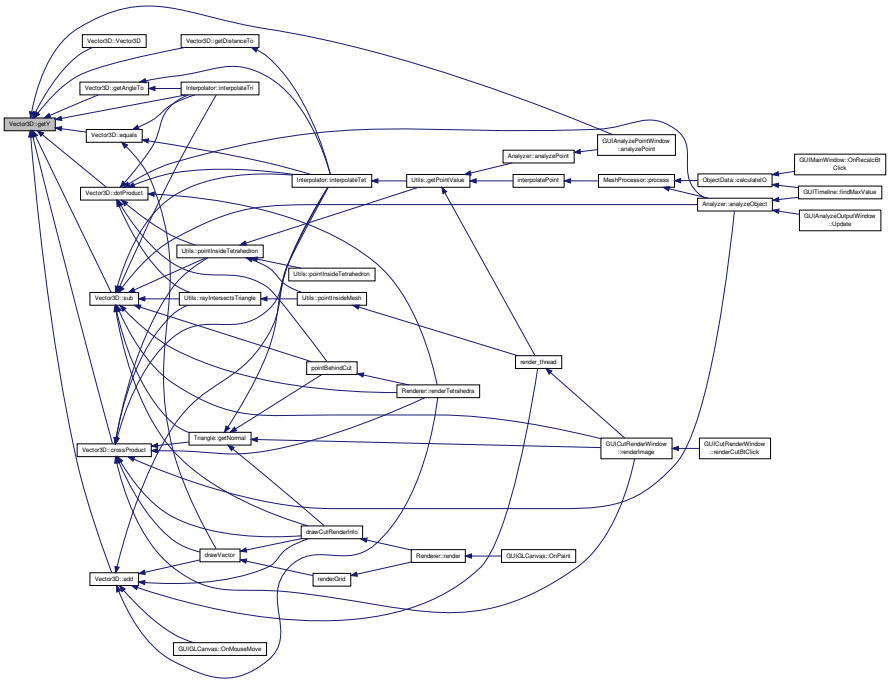


8.38.3.11 double Vector3D::getY ( )

Gibt das Y-Element des Vektors zurück.

Definiert in Zeile 52 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

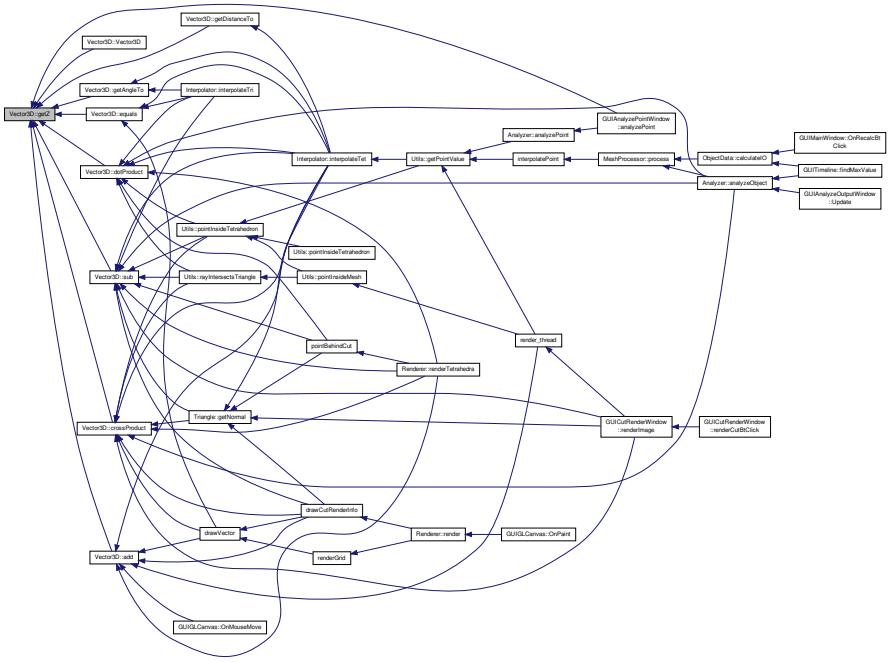


8.38.3.12 double Vector3D::getZ ( )

Gibt das Z-Element des Vektors zurück.

Definiert in Zeile 56 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.13 void Vector3D::mult (double *scalar*)

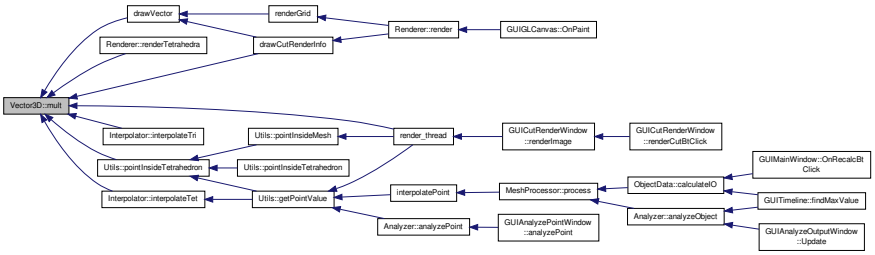
Multipliziert den Vektor mit einem Skalar.

Parameter

scalar	Der Skalar.

Definiert in Zeile 100 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

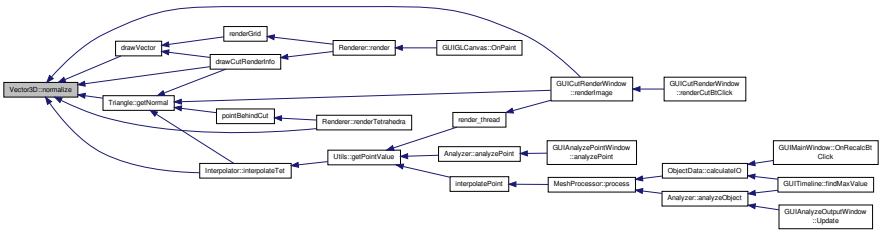


8.38.3.14 void Vector3D::normalize ( )

Normalisiert den Vektor.

Definiert in Zeile 106 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.15 void Vector3D::print ( )

Gibt den Vektor auf dem cout-Stream aus.

Definiert in Zeile 123 der Datei GeometryClasses.cpp.

8.38.3.16 void Vector3D::printTo (std::ostream & *stream*) const

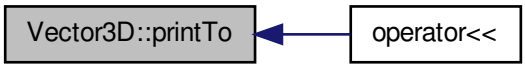
Gibt den Vektor auf dem gegebenen Stream aus.

Parameter

stream	Der zu verwendende Stream.

Definiert in Zeile 128 der Datei GeometryClasses.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.3.17 void Vector3D::sub (Vector3D * *other*)

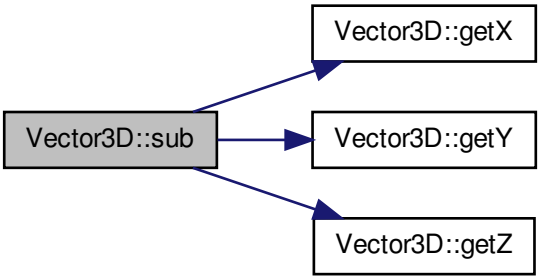
Subtrahiert einen Vektor von diesem Vektor.

Parameter

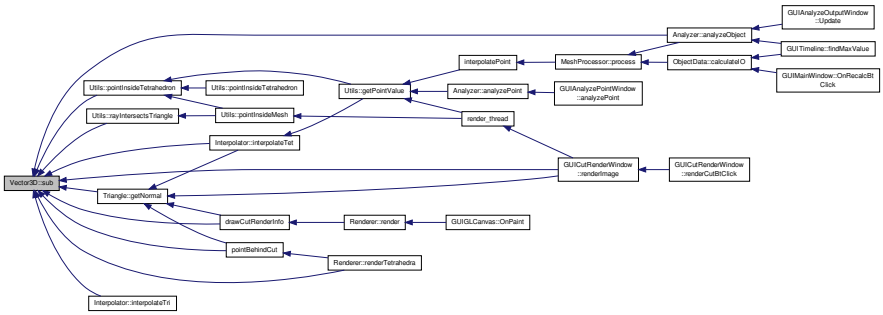
other	Der zu subtrahierende Vektor.

Definiert in Zeile 94 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



8.38.4 Freundbeziehungen und Funktionsdokumentation

8.38.4.1 std::ostream& operator<< (std::ostream & *out,* const Vector3D & *vec*) [friend]

Definition des <<-Operators für die Ausgabe eines Vektors.

Definiert in Zeile 132 der Datei GeometryClasses.cpp.

8.38.5 Dokumentation der Datenelemente

8.38.5.1 double Vector3D::coords[3] [private]

Die Elemente des Vektors.

Definiert in Zeile 139 der Datei GeometryClasses.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

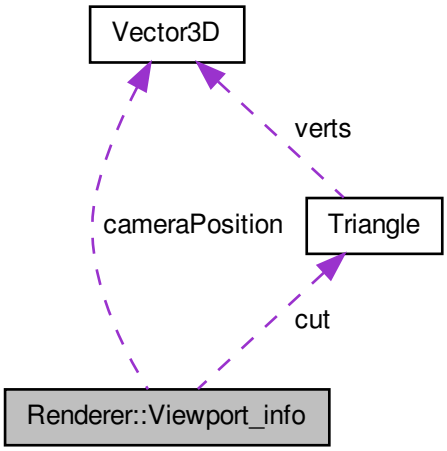
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/GeometryClasses.cpp

8.39 Renderer::Viewport_info Strukturreferenz

Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.

`#include <Renderer.h>`

Zusammengehörigkeiten von Renderer::Viewport_info:



Öffentliche Attribute

- float zoom

  Aktueller Zoomfaktor.
- float rotationY

  Rotation der Ansicht um die Y-Achse.
- float rotationX

  Rotation der Ansicht um die (Kameralokale) X-Achse.
- Vector3D * cameraPosition

  position der Virtuellen Kamera
- Triangle * cut

  Dreieck der Schnittebene, wenn nicht NULL, werden nur Elemente oberhalb der Dreiecksebene dargestellt (Momentan nicht verwendet).
- bool invertcut

  Nur Elemente unterhalb der durch cut definierte Ebene darstellen (Momentan nicht verwendet).
- RenderMode showPoints

  Modus der Darstellung von Punkten des 3D-Objekts.
- RenderMode showEdges

  Modus der Darstellung von Kanten des 3D-Objekts.
- RenderMode showFaces

  Modus der Darstellung von Flächen des 3D-Objekts.
- bool show_extrapolated

  Extrapolierte Elemente anzeigen.
- bool show_sensordata

  Sensordaten als Punkte anzeigen.

- int width

  Breite des dargestellten Bereichs.
- int height

  Höhe des dargestellten Bereichs.
- float scale

  Skalierungsfaktor für das 3D-Objekt.

8.39.1 Ausführliche Beschreibung

Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.

Definiert in Zeile 38 der Datei Renderer.h.

8.39.2 Dokumentation der Datenelemente

8.39.2.1 Vector3D* Renderer::Viewport_info::cameraPosition

position der Virtuellen Kamera

Definiert in Zeile 42 der Datei Renderer.h.

8.39.2.2 Triangle* Renderer::Viewport_info::cut

Dreieck der Schnittebene, wenn nicht NULL, werden nur Elemente oberhalb der Dreiecksebene dargestellt (Momentan nicht verwendet).

Definiert in Zeile 43 der Datei Renderer.h.

8.39.2.3 int Renderer::Viewport_info::height

Höhe des dargestellten Bereichs.

Definiert in Zeile 51 der Datei Renderer.h.

8.39.2.4 bool Renderer::Viewport_info::invertcut

Nur Elemente unterhalb der durch cut definierte Ebene darstellen (Momentan nicht verwendet).

Definiert in Zeile 44 der Datei Renderer.h.

8.39.2.5 float Renderer::Viewport_info::rotationX

Rotation der Ansicht um die (Kameralokale) X-Achse.

Definiert in Zeile 41 der Datei Renderer.h.

8.39.2.6 float Renderer::Viewport_info::rotationY

Rotation der Ansicht um die Y-Achse.

Definiert in Zeile 40 der Datei Renderer.h.

8.39.2.7 float Renderer::Viewport_info::scale

Skalierungsfaktor für das 3D-Objekt.

Definiert in Zeile 52 der Datei Renderer.h.

8.39.2.8 bool Renderer::Viewport_info::show_extrapolated

Extrapolierte Elemente anzeigen.

Definiert in Zeile 48 der Datei Renderer.h.

8.39.2.9 bool Renderer::Viewport_info::show_sensordata

Sensordaten als Punkte anzeigen.

Definiert in Zeile 49 der Datei Renderer.h.

8.39.2.10 RenderMode Renderer::Viewport_info::showEdges

Modus der Darstellung von Kanten des 3D-Objekts.

Definiert in Zeile 46 der Datei Renderer.h.

8.39.2.11 RenderMode Renderer::Viewport_info::showFaces

Modus der Darstellung von Flächen des 3D-Objekts.

Definiert in Zeile 47 der Datei Renderer.h.

8.39.2.12 RenderMode Renderer::Viewport_info::showPoints

Modus der Darstellung von Punkten des 3D-Objekts.

Definiert in Zeile 45 der Datei Renderer.h.

8.39.2.13 int Renderer::Viewport_info::width

Breite des dargestellten Bereichs.

Definiert in Zeile 50 der Datei Renderer.h.

8.39.2.14 float Renderer::Viewport_info::zoom

Aktueller Zoomfaktor.

Definiert in Zeile 39 der Datei Renderer.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

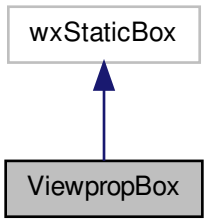
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.h

8.40 ViewpropBox Klassenreferenz

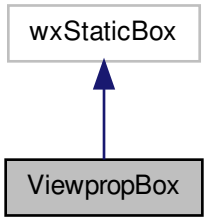
Oberfläche zum Verändern/Anzeigen der Visualisierungsoptionen.

`#include <ViewpropBox.h>`

Klassendiagramm für ViewpropBox:



Zusammengehörigkeiten von ViewpropBox:



Öffentliche Methoden

- ViewpropBox (wxWindow *parent)

  Der Konstruktor.
- void resize ()

  Behandelt Größenänderungen und passt die Positionen der Komponenten an.
- wxSpinCtrl * getColorRangeMaxEdit ()

  Gibt das Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot) zurück.
- wxSpinCtrl * getColorRangeMinEdit ()

  Gibt das Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau) zurück.
- wxRadioBox * getEdgesCheckBox ()

  Gibt das Auswahlfeld für den Darstellungsmodus von Kanten zurück.
- wxRadioBox * getFacesCheckBox ()

  Gibt das Auswahlfeld für den Darstellungsmodus von Flächen zurück.
- wxCheckListBox * getMatVisibilityListBox ()

  Gibt das Auswahlfeld für die Sichtbarkeit von Materialien zurück.
- wxRadioBox * getPointsCheckBox ()

  Gibt das Auswahlfeld für den Darstellungsmodus von Punkten zurück.

- wxCheckBox * getShowExtrapolatedCheckBox ()

  Gibt die Checkbox zum Anzeigen Extrapolierter Elemente zurück.
- wxCheckBox * getShowShowSensorData ()

  Gibt die Checkbox zum Anzeigen der Sensordaten als Punkte zurück.
- wxTextCtrl * getViewScaleEdit ()

  Gibt das Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt zurück.
- virtual ~ViewpropBox ()

  Der Destruktor.

Private Attribute

- wxRadioBox * pointsCheckBox

  Auswahlfeld für den Darstellungsmodus von Punkten.
- wxRadioBox * edgesCheckBox

  Auswahlfeld für den Darstellungsmodus von Kanten.
- wxRadioBox * facesCheckBox

  Auswahlfeld für den Darstellungsmodus von Flächen.
- wxStaticText * matVisualizationLbl

  Beschriftung für das Auswahlfeld für die Sichtbarkeit von Materialien.
- wxCheckListBox * matVisibilityListBox

  Auswahlfeld für die Sichtbarkeit von Materialien.
- wxCheckBox * showExtrapolatedCheckBox

  Checkbox zum Anzeigen Extrapolierter Elemente.
- wxCheckBox * showShowSensorData

  Checkbox zum Anzeigen der Sensordaten als Punkte.
- wxStaticText * colorRangeLbl

  Beschriftung für die Eingabefelder des zur Visualisierung verwendeten Temperaturbereichs.
- wxSpinCtrl * colorRangeMinEdit

  Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau).
- wxSpinCtrl * colorRangeMaxEdit

  Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot).
- wxStaticText * viewScaleLbl

  Beschriftung für das Eingabefeld eines Skalierungsfaktors für das 3D-Objekt.
- wxTextCtrl * viewScaleEdit

  Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt.

8.40.1 Ausführliche Beschreibung

Oberfläche zum Verändern/Anzeigen der Visualisierungsoptionen.

Diese Klasse verwaltet nur das Layout des Visualisierungsoptionen-Bereichs. Die Funktionalität wird in GUIMainWindow behandelt.

Definiert in Zeile 19 der Datei ViewpropBox.h.

8.40.2 Beschreibung der Konstruktoren und Destruktoren

8.40.2.1 ViewpropBox::ViewpropBox (wxWindow * *parent*)

Der Konstruktor.

Parameter

parent	Die übergeordnete Komponente.

Definiert in Zeile 17 der Datei ViewpropBox.cpp.

8.40.2.2 ViewpropBox::~ViewpropBox ( ) [virtual]

Der Destruktor.

Definiert in Zeile 105 der Datei ViewpropBox.cpp.

8.40.3 Dokumentation der Elementfunktionen

8.40.3.1 wxSpinCtrl * ViewpropBox::getColorRangeMaxEdit ( )

Gibt das Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot) zurück.

Definiert in Zeile 69 der Datei ViewpropBox.cpp.

8.40.3.2 wxSpinCtrl * ViewpropBox::getColorRangeMinEdit ( )

Gibt das Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau) zurück.

Definiert in Zeile 73 der Datei ViewpropBox.cpp.

8.40.3.3 wxRadioBox * ViewpropBox::getEdgesCheckBox ( )

Gibt das Auswahlfeld für den Darstellungsmodus von Kanten zurück.

Definiert in Zeile 77 der Datei ViewpropBox.cpp.

8.40.3.4 wxRadioBox * ViewpropBox::getFacesCheckBox ( )

Gibt das Auswahlfeld für den Darstellungsmodus von Flächen zurück.

Definiert in Zeile 81 der Datei ViewpropBox.cpp.

8.40.3.5 wxCheckListBox * ViewpropBox::getMatVisibilityListBox ( )

Gibt das Auswahlfeld für die Sichtbarkeit von Materialien zurück.

Definiert in Zeile 85 der Datei ViewpropBox.cpp.

8.40.3.6 wxRadioBox * ViewpropBox::getPointsCheckBox ( )

Gibt das Auswahlfeld für den Darstellungsmodus von Punkten zurück.

Definiert in Zeile 89 der Datei ViewpropBox.cpp.

8.40.3.7 wxCheckBox * ViewpropBox::getShowExtrapolatedCheckBox ( )

Gibt die Checkbox zum Anzeigen Extrapolierter Elemente zurück.

Definiert in Zeile 93 der Datei ViewpropBox.cpp.

8.40.3.8 wxCheckBox ∗ ViewpropBox::getShowShowSensorData ( )

Gibt die Checkbox zum Anzeigen der Sensordaten als Punkte zurück.

Definiert in Zeile 97 der Datei ViewpropBox.cpp.

8.40.3.9 wxTextCtrl ∗ ViewpropBox::getViewScaleEdit ( )

Gibt das Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt zurück.

Diese Skalierung ist rein optisch.

Definiert in Zeile 101 der Datei ViewpropBox.cpp.

8.40.3.10 void ViewpropBox::resize ( )

Behandelt Größenänderungen und passt die Positionen der Komponenten an.

Definiert in Zeile 48 der Datei ViewpropBox.cpp.

8.40.4 Dokumentation der Datenelemente

8.40.4.1 wxStaticText∗ ViewpropBox::colorRangeLbl [private]

Beschriftung für die Eingabefelder des zur Visualisierung verwendeten Temperaturbereichs.

Definiert in Zeile 120 der Datei ViewpropBox.h.

8.40.4.2 wxSpinCtrl∗ ViewpropBox::colorRangeMaxEdit [private]

Eingabefeld für die maximal Visualisierte Temperatur (entspricht der Farbe Rot).

Definiert in Zeile 130 der Datei ViewpropBox.h.

8.40.4.3 wxSpinCtrl∗ ViewpropBox::colorRangeMinEdit [private]

Eingabefeld für die minimal Visualisierte Temperatur (entspricht der Farbe Blau).

Definiert in Zeile 125 der Datei ViewpropBox.h.

8.40.4.4 wxRadioBox∗ ViewpropBox::edgesCheckBox [private]

Auswahlfeld für den Darstellungsmodus von Kanten.

Definiert in Zeile 90 der Datei ViewpropBox.h.

8.40.4.5 wxRadioBox∗ ViewpropBox::facesCheckBox [private]

Auswahlfeld für den Darstellungsmodus von Flächen.

Definiert in Zeile 95 der Datei ViewpropBox.h.

8.40.4.6 wxCheckListBox∗ ViewpropBox::matVisibilityListBox [private]

Auswahlfeld für die Sichtbarkeit von Materialien.

Definiert in Zeile 105 der Datei ViewpropBox.h.

8.40.4.7 wxStaticText* ViewpropBox::matVisualizationLbl [private]

Beschriftung für das Auswahlfeld für die Sichtbarkeit von Materialien.

Definiert in Zeile 100 der Datei ViewpropBox.h.

8.40.4.8 wxRadioBox* ViewpropBox::pointsCheckBox [private]

Auswahlfeld für den Darstellungsmodus von Punkten.

Definiert in Zeile 85 der Datei ViewpropBox.h.

8.40.4.9 wxCheckBox* ViewpropBox::showExtrapolatedCheckBox [private]

Checkbox zum Anzeigen Extrapolierter Elemente.

Definiert in Zeile 110 der Datei ViewpropBox.h.

8.40.4.10 wxCheckBox* ViewpropBox::showShowSensorData [private]

Checkbox zum Anzeigen der Sensordaten als Punkte.

Definiert in Zeile 115 der Datei ViewpropBox.h.

8.40.4.11 wxTextCtrl* ViewpropBox::viewScaleEdit [private]

Eingabefeld für einen Skalierungsfaktor für das 3D-Objekt.

Diese Skalierung ist rein optisch.

Definiert in Zeile 140 der Datei ViewpropBox.h.

8.40.4.12 wxStaticText* ViewpropBox::viewScaleLbl [private]

Beschriftung für das Eingabefeld eines Skalierungsfaktors für das 3D-Objekt.

Definiert in Zeile 135 der Datei ViewpropBox.h.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.h
- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.cpp

8.41 Utils::Visualization_info Strukturreferenz

Informationen über die Farbgebung bei der Visualisierung.

`#include <utils.h>`

Öffentliche Attribute

- int max_visualisation_temp = 100

  maximal Visualisierte Temperatur (entspricht der Farbe Rot).
- int min_visualisation_temp = 0

  minimal Visualisierte Temperatur (entspricht der Farbe Blau).

8.41.1 Ausführliche Beschreibung

Informationen über die Farbgebung bei der Visualisierung.

Definiert in Zeile 47 der Datei utils.h.

8.41.2 Dokumentation der Datenelemente

8.41.2.1 int Utils::Visualization_info::max_visualisation_temp = 100

maximal Visualisierte Temperatur (entspricht der Farbe Rot).

Definiert in Zeile 48 der Datei utils.h.

8.41.2.2 int Utils::Visualization_info::min_visualisation_temp = 0

minimal Visualisierte Temperatur (entspricht der Farbe Blau).

Definiert in Zeile 49 der Datei utils.h.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h

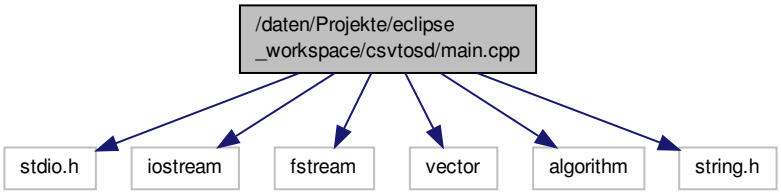
Kapitel 9

Datei-Dokumentation

9.1 /daten/Projekte/eclipse_workspace/csvtosd/main.cpp-Dateireferenz

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
#include <string.h>
```

Include-Abhängigkeitsdiagramm für main.cpp:



Klassen

- class CsvToSdConverter

  Konverter von .csv zu .tsd.
- struct CsvToSdConverter::Options

  Strunktur für die Programmeinstellungen.

Funktionen

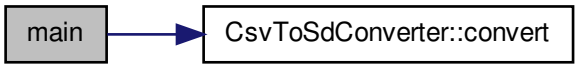
- int main (int argc, char ∗argv[])

9.1.1 Dokumentation der Funktionen

9.1.1.1 int main (int *argc*, char ∗ *argv[]*)

Definiert in Zeile 694 der Datei main.cpp.

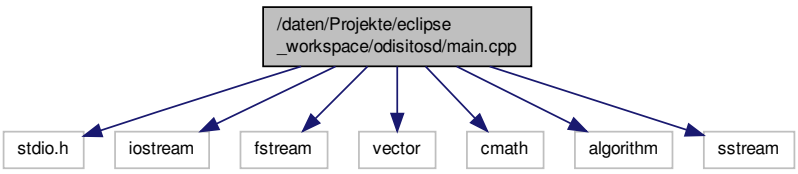
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



9.2 /daten/Projekte/eclipse_workspace/odisitosd/main.cpp-Dateireferenz

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <sstream>
```

Include-Abhängigkeitsdiagramm für main.cpp:



Klassen

- class OdisiToSdConverter

  Konverter von ODiSI zu .tsd.
- struct OdisiToSdConverter::Options

  Strunktur für die Programmeinstellungen.

Funktionen

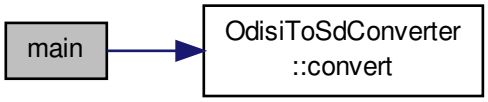
- int main (int argc, char ∗argv[])

9.2.1 Dokumentation der Funktionen

9.2.1.1 int main (int *argc,* char ∗ *argv[]*)

Definiert in Zeile 996 der Datei main.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



9.3 doxygen_dep_dummy.h-Dateireferenz

Klassen

- class std::vector< T >

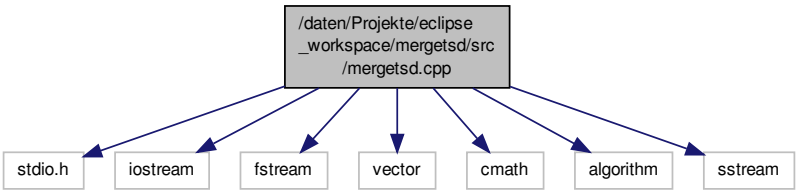
Namensbereiche

- std

9.4 /daten/Projekte/eclipse_workspace/mergetsd/src/mergetsd.cpp-Dateireferenz

```
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <sstream>
```

Include-Abhängigkeitsdiagramm für mergetsd.cpp:



Klassen

- class TsdMerger

  Zusammenführen zweier .tsd-Dateien.
- struct TsdMerger::Options

  Strunktur für die Programmeinstellungen.

Funktionen

- int main (int argc, char ∗argv[])

9.4.1 Dokumentation der Funktionen

9.4.1.1 int main (int *argc,* char ∗ *argv[]*)

Definiert in Zeile 366 der Datei mergetsd.cpp.

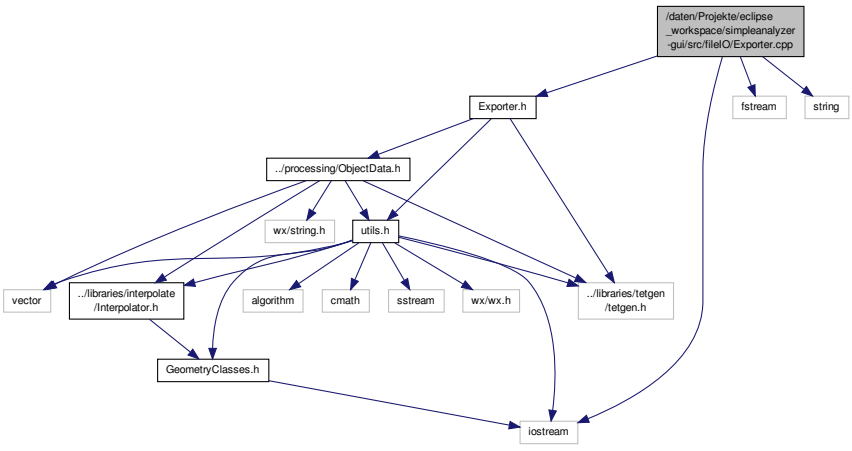
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

9.5 /daten/Projekte/eclipse_workspace/README.md-Dateireferenz

9.6 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.cpp-Dateireferenz

```
#include "Exporter.h"
#include <iostream>
#include <fstream>
#include <string>
```

Include-Abhängigkeitsdiagramm für Exporter.cpp:



Variablen

- const int tetface_indices [4][3]

9.6.1 Variablen-Dokumentation

9.6.1.1 const int tetface_indices[4][3]

Initialisierung:

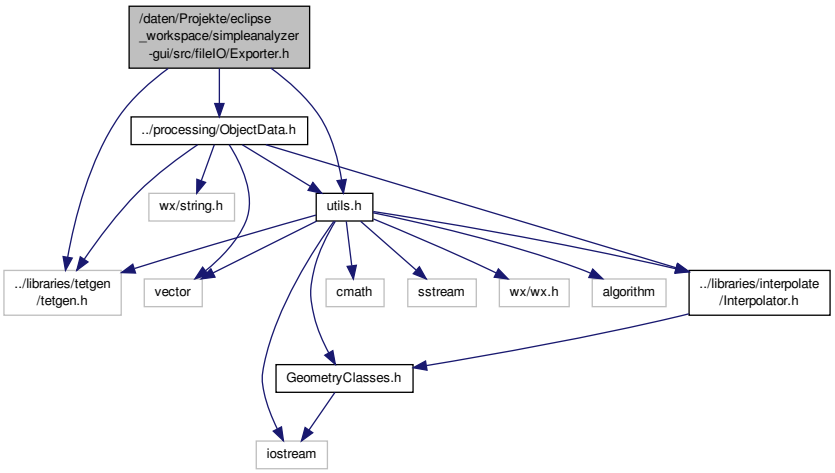
```
= { { 0, 1, 2 }, { 0, 1, 3 }, { 0, 2, 3 }, { 1,
        2, 3 } }
```

Definiert in Zeile 20 der Datei Exporter.cpp.

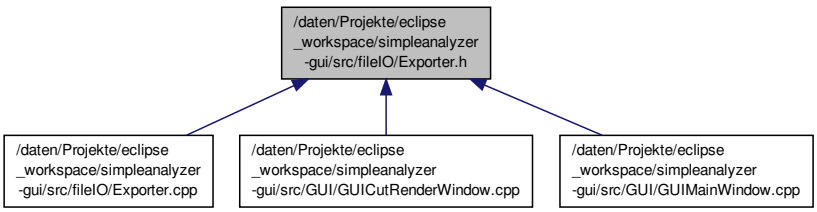
9.7 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Exporter.h-Dateireferenz

```
#include "../libraries/tetgen/tetgen.h"
#include "../processing/ObjectData.h"
#include "../processing/utils.h"
```

Include-Abhängigkeitsdiagramm für Exporter.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

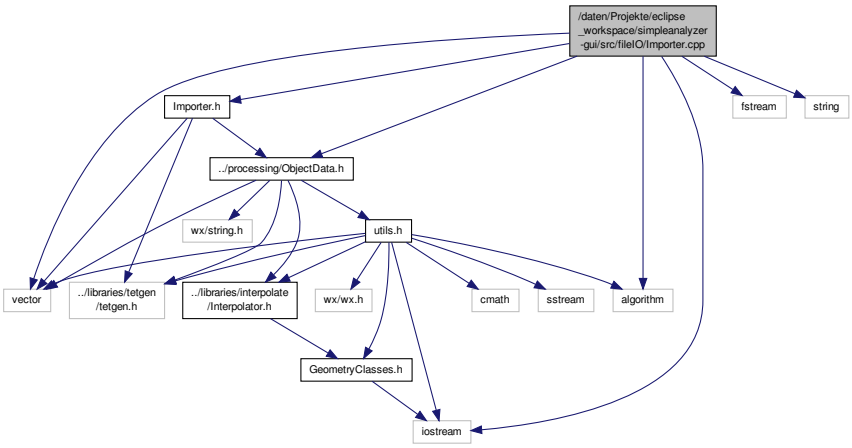
- class Exporter

  Export der gewonnenen Daten.

9.8 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.cpp-Dateireferenz

```
#include "Importer.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <algorithm>
#include "../processing/ObjectData.h"
```

Include-Abhängigkeitsdiagramm für Importer.cpp:



Makrodefinitionen

- #define PATH_SEPARATOR '/'

Funktionen

- int getFaceIndex (string data, bool withUV)

  Extrahiert den Index einer Fläche aus einem Textblock einer Zeile der .obj-Datei.
- string getTextBlock (string data, int n)

  Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

9.8.1 Makro-Dokumentation

9.8.1.1 #define PATH_SEPARATOR '/'

Definiert in Zeile 21 der Datei Importer.cpp.

9.8.2 Dokumentation der Funktionen

9.8.2.1 int getFaceIndex (string *data,* bool *withUV*)

Extrahiert den Index einer Fläche aus einem Textblock einer Zeile der .obj-Datei.

Parameter

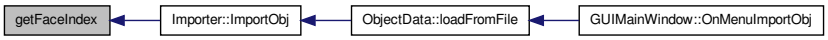
data	Der zu untersuchende Block.
withUV	Enthält die obj-Datei auch Texturdatenindices?

Rückgabe

Der Flächenindex.

Definiert in Zeile 34 der Datei Importer.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



9.8.2.2 string getTextBlock (string *data,* int *n*)

Gibt den n-ten durch Leerzeichen abgetrennten Block aus einem String zurück.

Parameter

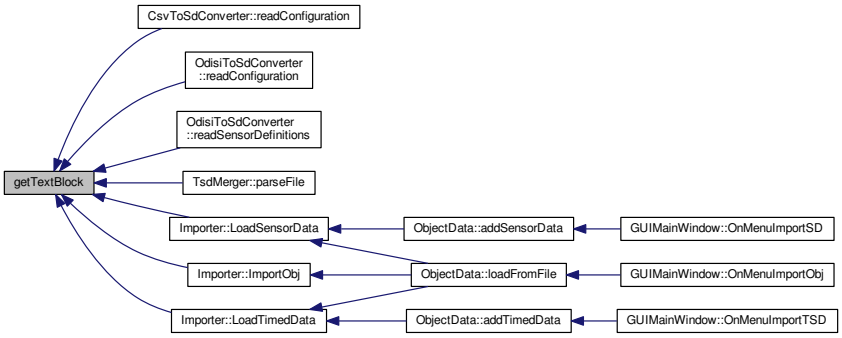
data	Der Ausgansstring.
n	Index des zu findenden Blocks.

Rückgabe

Der n-te durch Leerzeichen getrennte Teilstring. "" Bei ungültigem Index.

Definiert in Zeile 51 der Datei Importer.cpp.

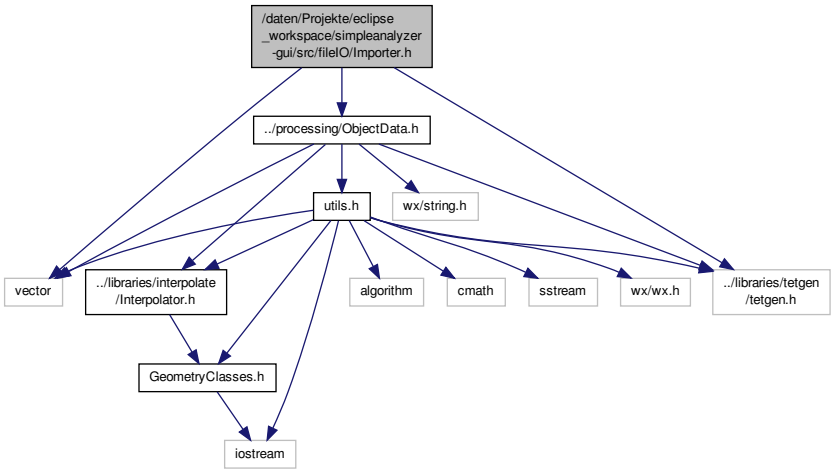
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



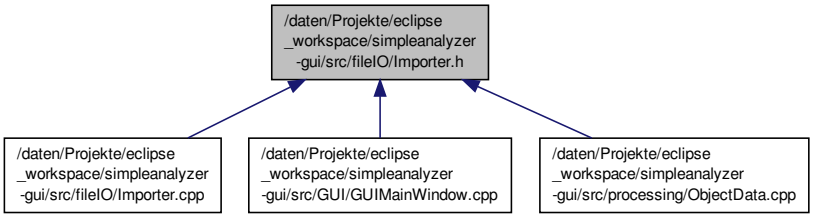
9.9 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/fileIO/Importer.h-Dateireferenz

```
#include "../libraries/tetgen/tetgen.h"
#include "../processing/ObjectData.h"
#include <vector>
```

Include-Abhängigkeitsdiagramm für Importer.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

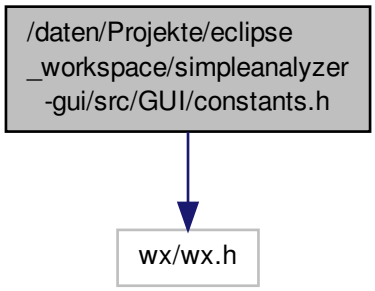
- class Importer

  Importieren von 3D-Modell (.obj) und Sensordaten (.tsd oder .sd).

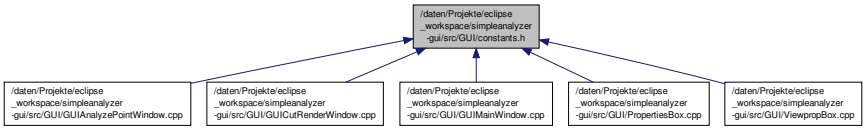
9.10 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/constants.h-Dateireferenz

```
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für constants.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Aufzählungen

- enum EventID {
  ID_ABOUT = 1, ID_TEST, ID_IMPORT_OBJ, ID_IMPORT_SD,
  ID_RECALCBT, ID_MATERIALBOX, ID_ANALYZE, ID_GENERAL_PROP,
  ID_IMMEDIATE_UPDATE_PROP, ID_GENERAL_VIEW_PROP, ID_CHECKLISTBOX_VIEW_PROP, ID_-
  CHANGE_ACTIVE_OBJ,
  ID_ANALYZE_POINT, ID_ANALYZE_POINT_BT, ID_CUT_CANVAS, ID_RENDER_CUT,
  ID_RENDER_CUT_BT, ID_CUT_TRI_EDIT, ID_DELETE_ACTIVE_OBJ, ID_IMPORT_TSD,
  ID_SD_BOX, ID_SD_TIMELINE, ID_ANALYZE_MARKER_CB, ID_CLEAR_MARKER_BT,
  ID_MARKER_NEXT_BT, ID_MARKER_PREV_BT, ID_EXPORT_CUT_IMG_BT, ID_EXPORT_VIEWPORT,
  ID_FIND_MAX_BT, ID_AUTO_UPDATE_CB, ID_EXPORT_VTK, ID_EXPORT_CUT_CSV_BT,
  ID_COLORSCALE_PROP, ID_COLORSCALE_COLORBT, ID_OPEN_MANUAL }

  IDs für die Events der Programmoberfläche.

Variablen

- const int NUMBER_OF_INTERPOLATION_MODES = 2

  Anzahl der verfügbaren Interpolationsmodi.
- const wxString INTERPOLATION_MODE_STRINGS [NUMBER_OF_INTERPOLATION_MODES] = {wx-T("Linear"),wxT("Logarithmisch")}

  Bezeichnungen für die von "Interpolator" verwendeten Interpolationsmodi.

9.10.1 Dokumentation der Aufzählungstypen

9.10.1.1 enum EventID

IDs für die Events der Programmoberfläche.

Müssen kleiner als wxID_LOWEST (wxWidgets 2.8: 4999) sein!

Aufzählungswerte

ID_ABOUT

ID_TEST

ID_IMPORT_OBJ

ID_IMPORT_SD

ID_RECALCBT

ID_MATERIALBOX

ID_ANALYZE

ID_GENERAL_PROP

ID_IMMEDIATE_UPDATE_PROP

ID_GENERAL_VIEW_PROP

ID_CHECKLISTBOX_VIEW_PROP

ID_CHANGE_ACTIVE_OBJ

ID_ANALYZE_POINT

ID_ANALYZE_POINT_BT

ID_CUT_CANVAS

ID_RENDER_CUT

ID_RENDER_CUT_BT

ID_CUT_TRI_EDIT

ID_DELETE_ACTIVE_OBJ

ID_IMPORT_TSD

ID_SD_BOX

ID_SD_TIMELINE

ID_ANALYZE_MARKER_CB

ID_CLEAR_MARKER_BT

ID_MARKER_NEXT_BT

ID_MARKER_PREV_BT

ID_EXPORT_CUT_IMG_BT

ID_EXPORT_VIEWPORT

ID_FIND_MAX_BT

ID_AUTO_UPDATE_CB

ID_EXPORT_VTK

ID_EXPORT_CUT_CSV_BT

ID_COLORSCALE_PROP

ID_COLORSCALE_COLORBT

ID_OPEN_MANUAL

Definiert in Zeile 28 der Datei constants.h.

9.10.2 Variablen-Dokumentation

9.10.2.1 const wxString INTERPOLATION_MODE_STRINGS[NUMBER_OF_INTERPOLATION_MODES] = {wxT("Linear"),wxT("Logarithmisch")}

Bezeichnungen für die von "Interpolator" verwendeten Interpolationsmodi.

Definiert in Zeile 21 der Datei constants.h.

9.10.2.2 const int NUMBER_OF_INTERPOLATION_MODES = 2

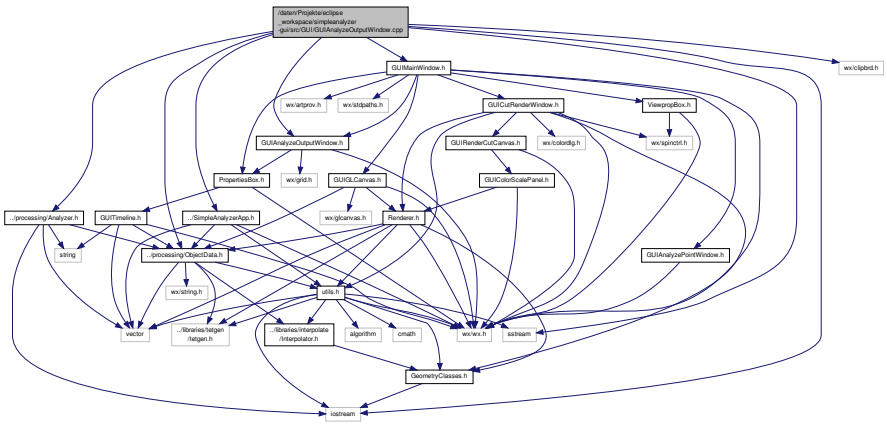
Anzahl der verfügbaren Interpolationsmodi.

Definiert in Zeile 17 der Datei constants.h.

9.11 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.cpp-Dateireferenz

```
#include "GUIAnalyzeOutputWindow.h"
#include "GUIMainWindow.h"
#include <iostream>
#include <sstream>
#include "../SimpleAnalyzerApp.h"
#include "../processing/ObjectData.h"
#include "../processing/Analyzer.h"
#include <wx/clipbrd.h>
```

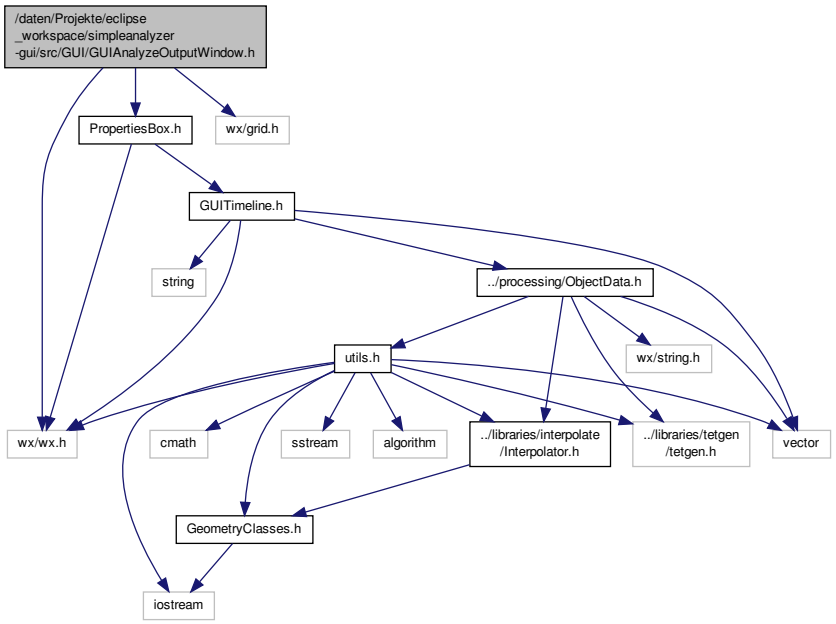
Include-Abhängigkeitsdiagramm für GUIAnalyzeOutputWindow.cpp:



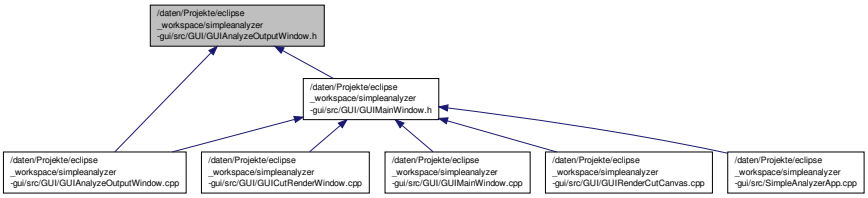
9.12 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzeOutputWindow.h-Dateireferenz

```
#include "PropertiesBox.h"
#include <wx/grid.h>
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für GUIAnalyzeOutputWindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

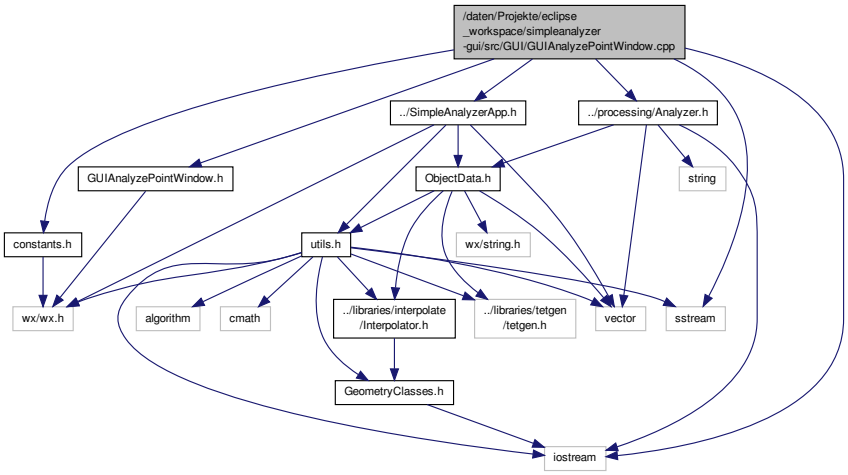
- class GUIAnalyzeOutputWindow

  Übersichtsfenster über die Analysedaten.

9.13 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePoint-Window.cpp-Dateireferenz

```
#include "GUIAnalyzePointWindow.h"
#include <iostream>
#include <sstream>
#include "constants.h"
#include "../processing/Analyzer.h"
#include "../SimpleAnalyzerApp.h"
```

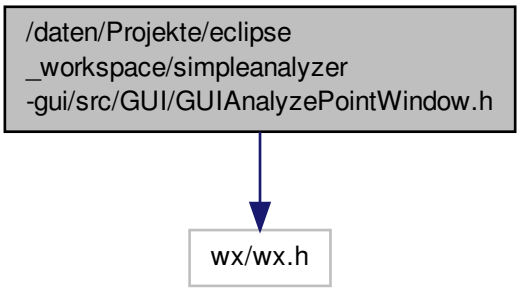
Include-Abhängigkeitsdiagramm für GUIAnalyzePointWindow.cpp:



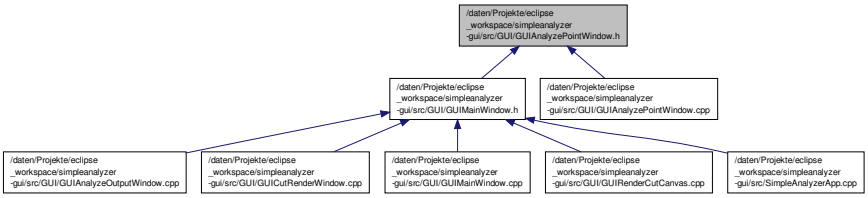
9.14 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIAnalyzePoint-Window.h-Dateireferenz

```
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für GUIAnalyzePointWindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

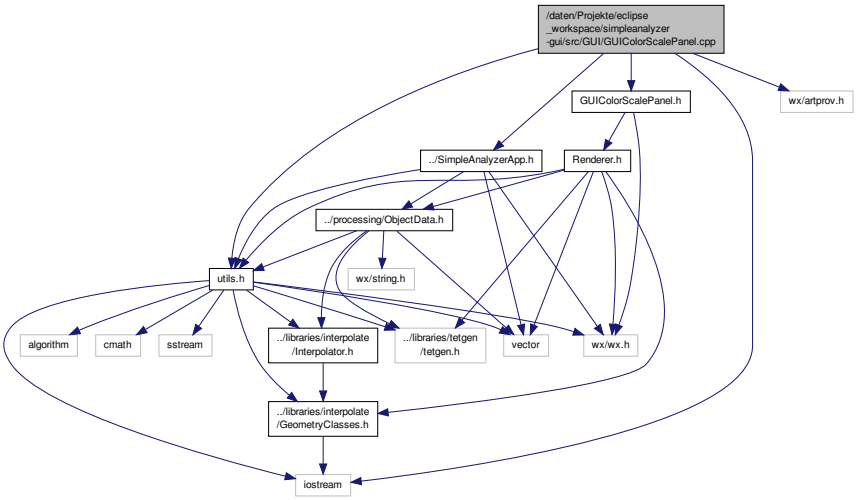
- class GUIAnalyzePointWindow

  Analysefenster für einen Punkt.

9.15 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.cpp-Dateireferenz

```
#include "GUIColorScalePanel.h"
#include <iostream>
#include "../SimpleAnalyzerApp.h"
#include "../processing/utils.h"
#include <wx/artprov.h>
```

Include-Abhängigkeitsdiagramm für GUIColorScalePanel.cpp:



Makrodefinitionen

- #define MIN_WIDTH 5
- #define MIN_HEIGHT 5

9.15.1 Makro-Dokumentation

9.15.1.1 #define MIN_HEIGHT 5

Definiert in Zeile 19 der Datei GUIColorScalePanel.cpp.

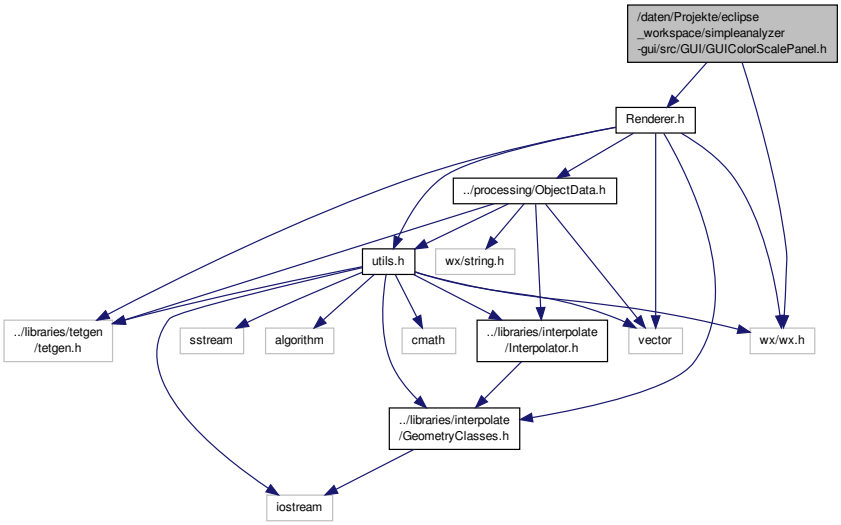
9.15.1.2 #define MIN_WIDTH 5

Definiert in Zeile 18 der Datei GUIColorScalePanel.cpp.

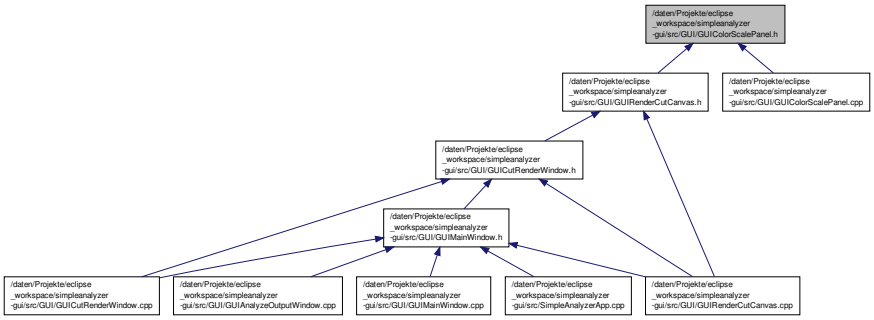
9.16 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIColorScalePanel.h-Dateireferenz

```
#include "Renderer.h"
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für GUIColorScalePanel.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class GUIColorScalePanel

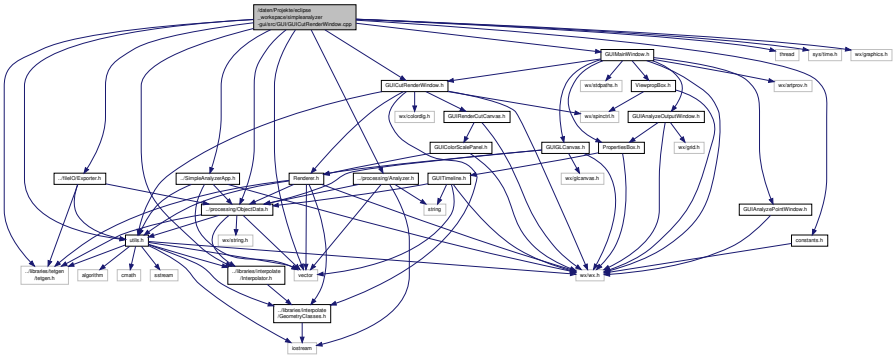
  Farbige Temperaturskala für zweidimensionale Temperaturverteilung.

9.17 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRenderWindow.cpp-Dateireferenz

```
#include "GUICutRenderWindow.h"
```

```
#include "constants.h"
#include <vector>
#include "../SimpleAnalyzerApp.h"
#include "../processing/Analyzer.h"
#include "../processing/ObjectData.h"
#include "../processing/utils.h"
#include "../libraries/interpolate/Interpolator.h"
#include "../libraries/tetgen/tetgen.h"
#include "../fileIO/Exporter.h"
#include "GUIMainWindow.h"
#include <thread>
#include <sys/time.h>
#include <wx/graphics.h>
```

Include-Abhängigkeitsdiagramm für GUICutRenderWindow.cpp:



Funktionen

- void render_thread (bool ∗status_flag, float ∗value_img, wxImage ∗image, int width, int height, int startheight, int delta_h, CutRender_info ∗info, Vector3D ∗xvec, Vector3D ∗yvec, Vector3D ∗v0, vector< tetgenio ∗ > ∗bases, ObjectData ∗obj, vector< SensorPoint > ∗sensor_data, bool use_last_tet)

  Funktion zum verteilten berechnen der 2D-Temperaturverteilung.

9.17.1 Dokumentation der Funktionen

9.17.1.1 void render_thread (bool ∗ *status_flag,* float ∗ *value_img,* wxImage ∗ *image,* int *width,* int *height,* int *startheight,* int *delta_h,* CutRender_info ∗ *info,* Vector3D ∗ *xvec,* Vector3D ∗ *yvec,* Vector3D ∗ *v0,* vector< tetgenio ∗ > ∗ *bases,* ObjectData ∗ *obj,* vector< SensorPoint > ∗ *sensor_data,* bool *use_last_tet*)

Funktion zum verteilten berechnen der 2D-Temperaturverteilung.

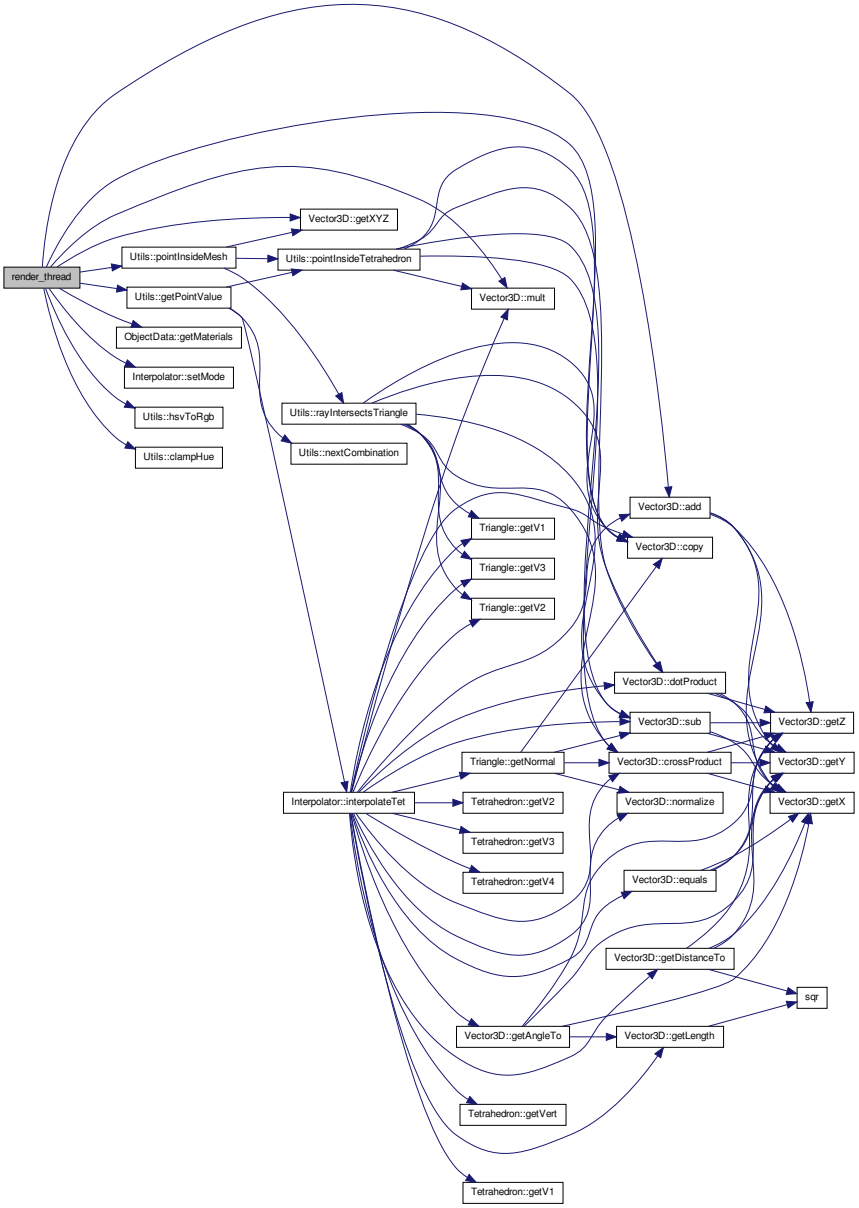
Parameter

status_flag	Zeiger auf Variable, die enthält ob der Thread beendet ist. (0 = Beendet)
value_img	Liste für die Daten der Temperaturverteilung.
image	Grafik für die Temperaturverteilung.
width	Breite der Temperaturverteilungsgrafik.
height	Höhe der Temperaturverteilungsgrafik.
startheight	Starthöhe für diesen Thread in der Grafik.

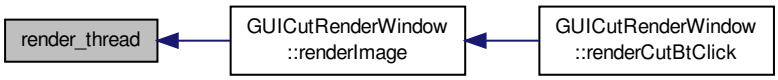
delta_h	Höhe des von diesem Thread zu berechnenden Streifens.
info	Informationen über die Eigenschaften der zu berechnenden Ebene.
xvec	X-Achse der Ebene.
yvec	Y-Achse der Ebene.
v0	Mittelpunkt der Ebene.
bases	Möglichst einfache Geometrien Geometrien der Materialien.
obj	Das aktuelle Objekt.
sensor_data	Die zu verwendenden Sensordaten.
use_last_tet	Versuchen, die Interpolation durch vorgezogenes Testen des zuletzt verwendeten Tetraeders zu beschleunigen. Diese Option ist verursacht Ungenauigkeiten und bietet zumeist wenig Performancegewinn.

Definiert in Zeile 167 der Datei GUICutRenderWindow.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



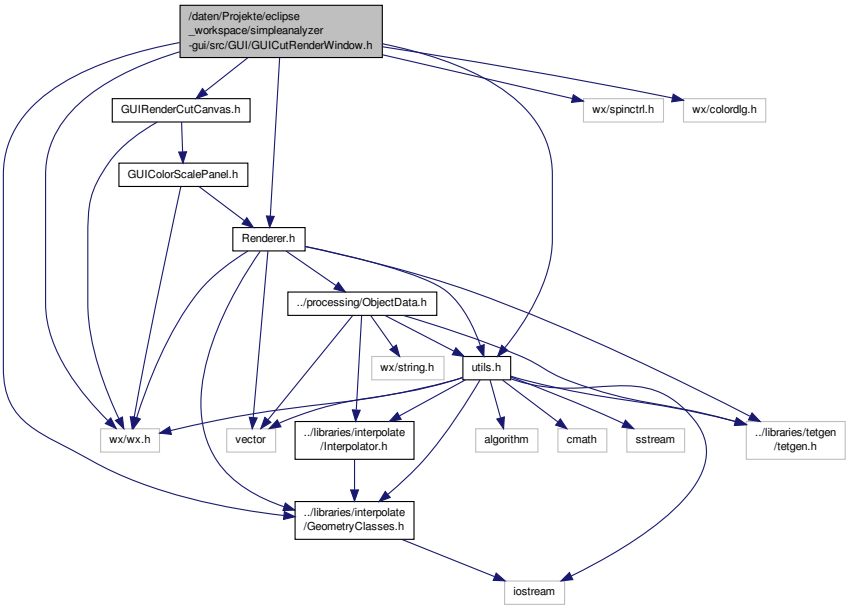
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



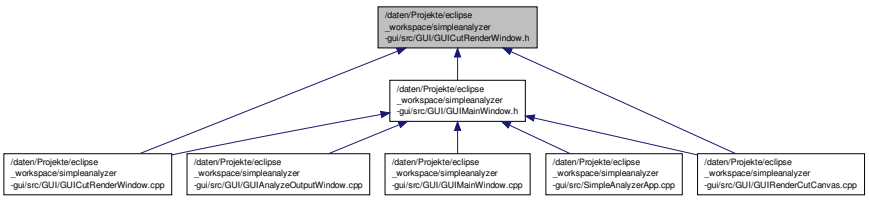
9.18 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUICutRender-Window.h-Dateireferenz

```
#include <wx/wx.h>
#include "GUIRenderCutCanvas.h"
#include "../libraries/interpolate/GeometryClasses.h"
#include "../processing/utils.h"
#include <wx/spinctrl.h>
#include "Renderer.h"
#include <wx/colordlg.h>
```

Include-Abhängigkeitsdiagramm für GUICutRenderWindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

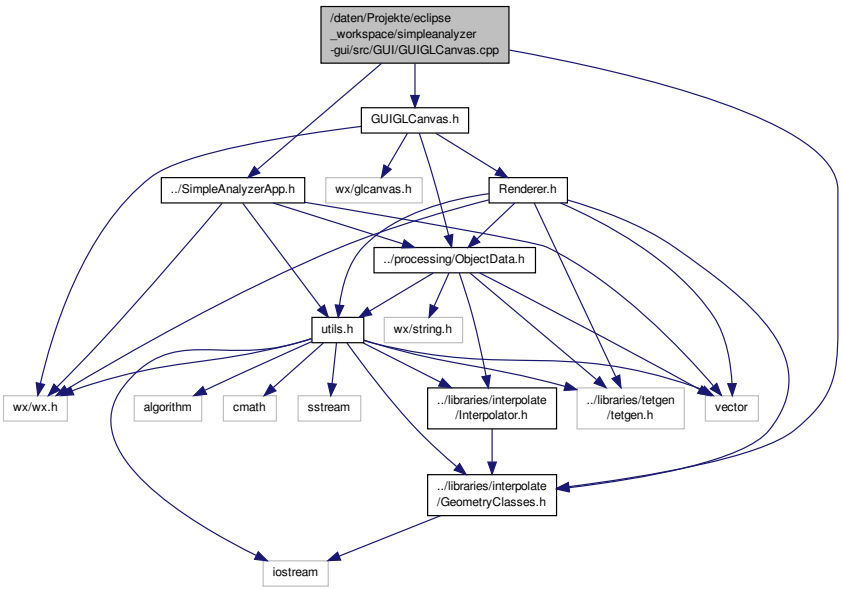
- class GUICutRenderWindow

  Fenster zum erstellen zweidimensionaler Temperaturverteilungen.

9.19 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.cpp-- Dateireferenz

```
#include "GUIGLCanvas.h"
#include "../SimpleAnalyzerApp.h"
#include "../libraries/interpolate/GeometryClasses.h"
```

Include-Abhängigkeitsdiagramm für GUIGLCanvas.cpp:



Variablen

- int attrib_list [] = { WX_GL_RGBA, WX_GL_DOUBLEBUFFER, WX_GL_DEPTH_SIZE, 16, 0 }

9.19.1 Variablen-Dokumentation

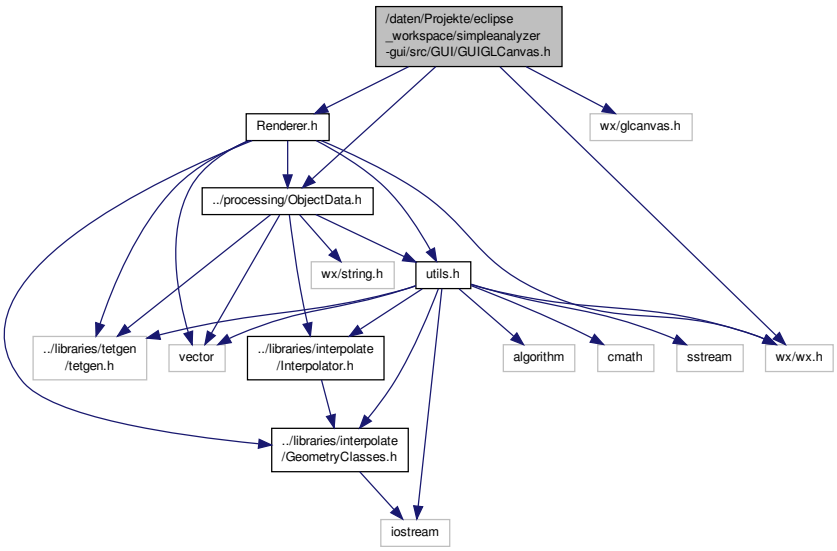
9.19.1.1 int attrib_list[ ] = { WX_GL_RGBA, WX_GL_DOUBLEBUFFER, WX_GL_DEPTH_SIZE, 16, 0 }

Definiert in Zeile 20 der Datei GUIGLCanvas.cpp.

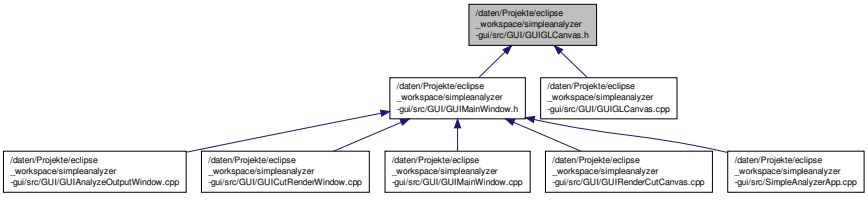
9.20 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIGLCanvas.h-- Dateireferenz

```
#include "Renderer.h"
#include <wx/wx.h>
#include <wx/glcanvas.h>
#include "../processing/ObjectData.h"
```

Include-Abhängigkeitsdiagramm für GUIGLCanvas.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

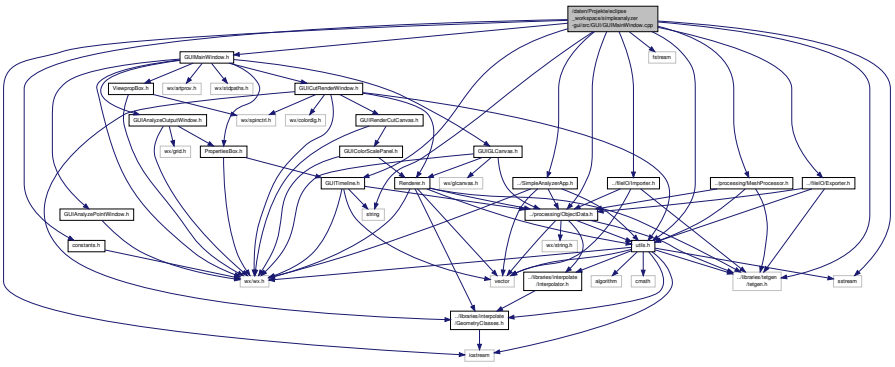
- class GUIGLCanvas

  Zeichenfläche für das 3D-Fenster.

9.21 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.cpp-Dateireferenz

```
#include "GUIMainWindow.h"
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
#include "../SimpleAnalyzerApp.h"
#include "../libraries/tetgen/tetgen.h"
#include "../processing/MeshProcessor.h"
#include "../fileIO/Importer.h"
#include "../processing/ObjectData.h"
#include "constants.h"
#include "../fileIO/Exporter.h"
#include "GUITimeline.h"
#include "../processing/utils.h"
```

Include-Abhängigkeitsdiagramm für GUIMainWindow.cpp:



Makrodefinitionen

- #define PATH_SEPARATOR "/"
- #define PROPBOXWIDTH 300
- #define VIEWBOXWIDTH 300

9.21.1 Makro-Dokumentation

9.21.1.1 #define PATH_SEPARATOR "/"

Definiert in Zeile 31 der Datei GUIMainWindow.cpp.

9.21.1.2 #define PROPBOXWIDTH 300

Definiert in Zeile 37 der Datei GUIMainWindow.cpp.

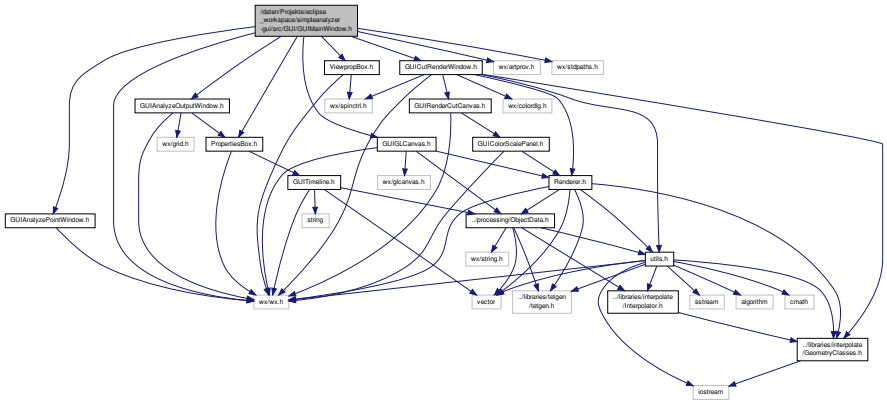
9.21.1.3 #define VIEWBOXWIDTH 300

Definiert in Zeile 38 der Datei GUIMainWindow.cpp.

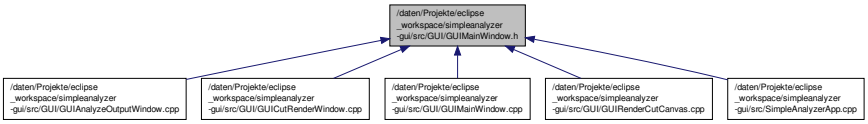
9.22 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIMainWindow.h-- Dateireferenz

```
#include "GUIGLCanvas.h"
#include "PropertiesBox.h"
#include "ViewpropBox.h"
#include "GUIAnalyzeOutputWindow.h"
#include "GUIAnalyzePointWindow.h"
#include "GUICutRenderWindow.h"
#include <wx/wx.h>
#include <wx/artprov.h>
#include <wx/stdpaths.h>
```

Include-Abhängigkeitsdiagramm für GUIMainWindow.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class GUIMainWindow

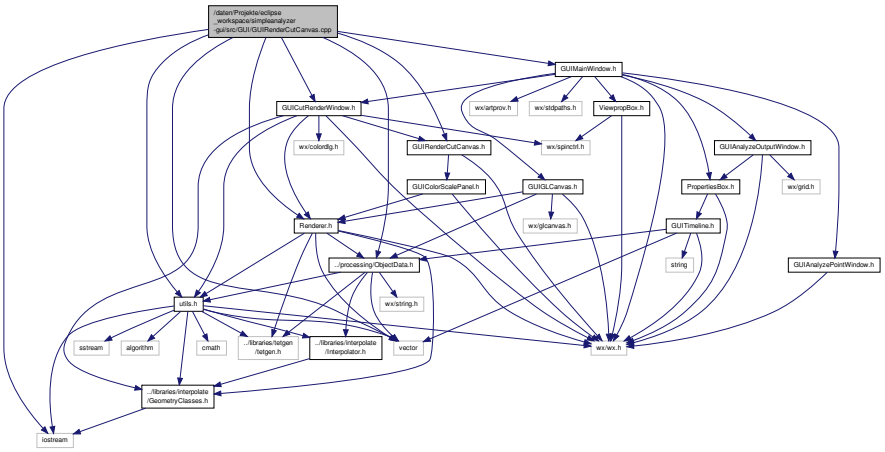
  Hauptfenster mit Hauptmenü und Zugriff auf die einzelnen Programmfunktionen.

9.23 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCut-Canvas.cpp-Dateireferenz

```
#include "GUIRenderCutCanvas.h"
```

```
#include "GUICutRenderWindow.h"
#include <vector>
#include "../processing/utils.h"
#include "../processing/ObjectData.h"
#include "GUIMainWindow.h"
#include "Renderer.h"
#include <iostream>
```

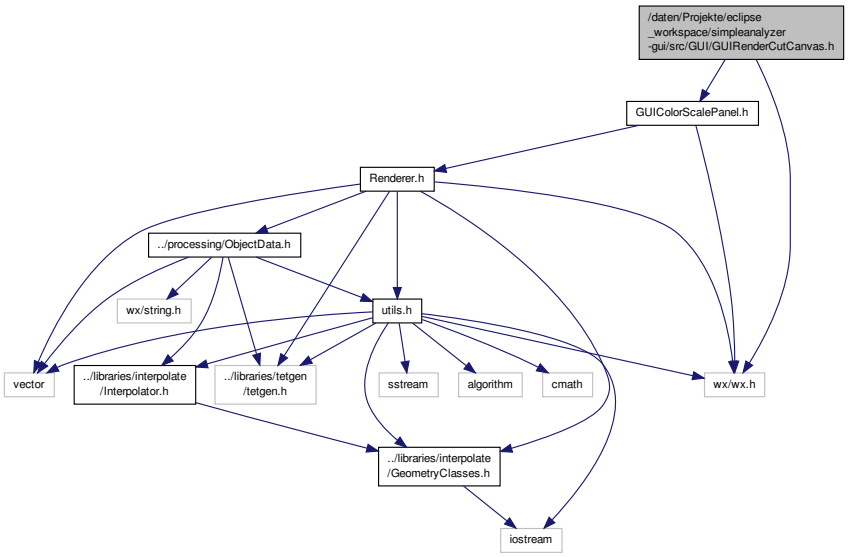
Include-Abhängigkeitsdiagramm für GUIRenderCutCanvas.cpp:



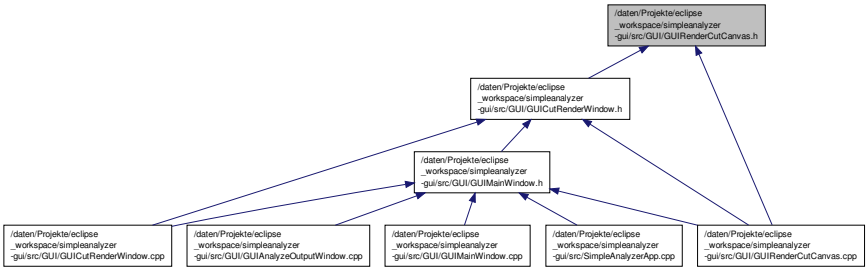
9.24 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUIRenderCut-Canvas.h-Dateireferenz

```
#include <wx/wx.h>
#include "GUIColorScalePanel.h"
```

Include-Abhängigkeitsdiagramm für GUIRenderCutCanvas.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

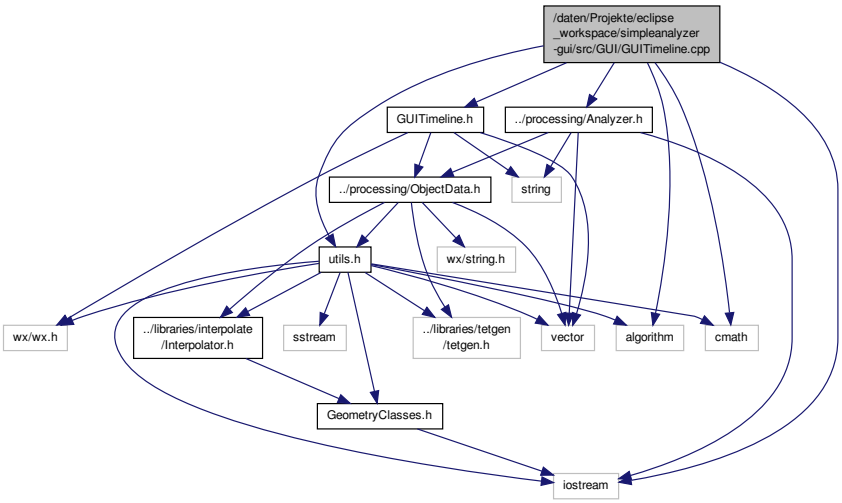
- class GUIRenderCutCanvas

  Zeichenfläche für die 2D-Temperaturverteilung.

9.25 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.cpp-- Dateireferenz

```
#include "GUITimeline.h"
#include <iostream>
#include "../processing/utils.h"
#include <cmath>
#include <algorithm>
#include "../processing/Analyzer.h"
```

Include-Abhängigkeitsdiagramm für GUITimeline.cpp:



Makrodefinitionen

- #define SCALE_REFINE_STEPS 3

Variablen

- const wxEventType wxEVT_TIMELINE_CHANGE = wxNewEventType()

  Typ wxEVT_TIMELINE_CHANGE zum Auslösen eines Events bei Veränderung der Zeitleiste.
- const float refine_factors [SCALE_REFINE_STEPS] = { .5, .2, .1 }

9.25.1 Makro-Dokumentation

9.25.1.1 #define SCALE_REFINE_STEPS 3

Definiert in Zeile 31 der Datei GUITimeline.cpp.

9.25.2 Variablen-Dokumentation

9.25.2.1 const float refine_factors[SCALE_REFINE_STEPS] = { .5, .2, .1 }

Definiert in Zeile 34 der Datei GUITimeline.cpp.

9.25.2.2 const wxEventType wxEVT_TIMELINE_CHANGE = wxNewEventType()

Typ wxEVT_TIMELINE_CHANGE zum Auslösen eines Events bei Veränderung der Zeitleiste.

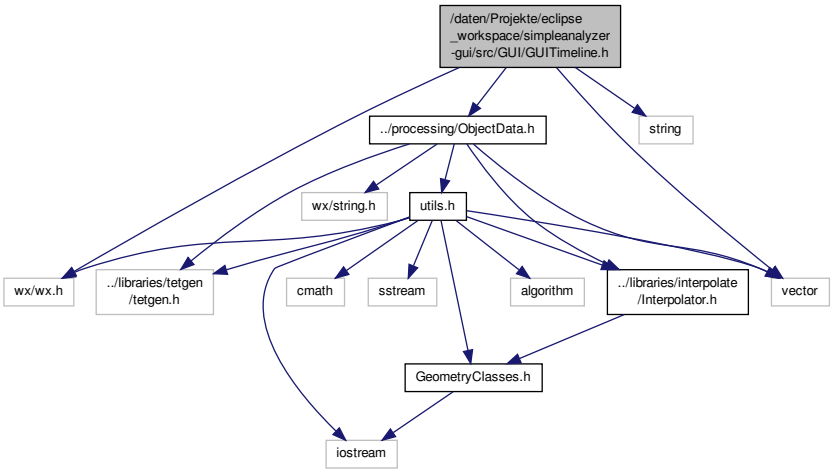
Die Definition als globale Konstante ist durch das GUI-System vorgegeben.

Definiert in Zeile 18 der Datei GUITimeline.cpp.

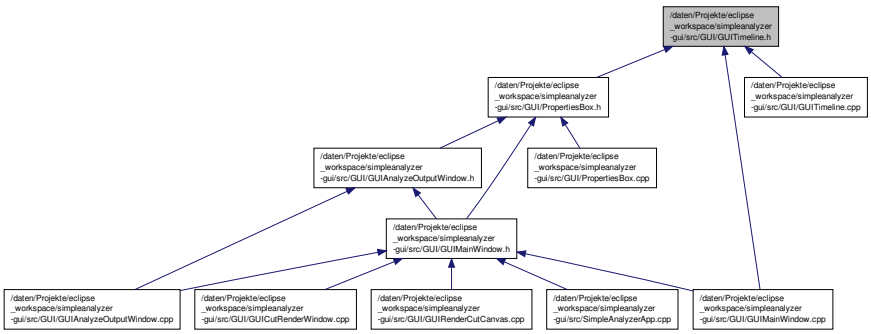
9.26 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/GUITimeline.h-- Dateireferenz

```
#include <wx/wx.h>
#include <string>
#include <vector>
#include "../processing/ObjectData.h"
```

Include-Abhängigkeitsdiagramm für GUITimeline.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class GUITimeline

  Eine Zeitleistenkomponente.

Variablen

- const wxEventType wxEVT_TIMELINE_CHANGE

  Typ wxEVT_TIMELINE_CHANGE zum Auslösen eines Events bei Veränderung der Zeitleiste.

9.26.1 Variablen-Dokumentation

9.26.1.1 const wxEventType wxEVT_TIMELINE_CHANGE

Typ wxEVT_TIMELINE_CHANGE zum Auslösen eines Events bei Veränderung der Zeitleiste.

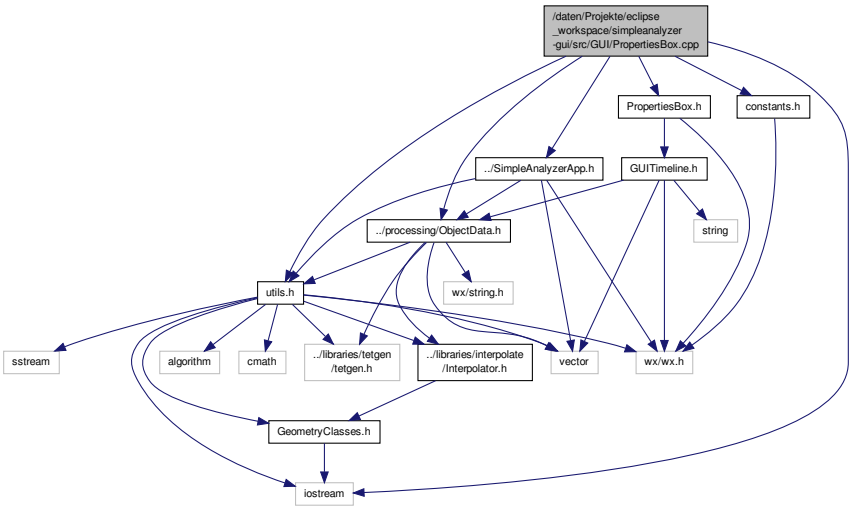
Die Definition als globale Konstante ist durch das GUI-System vorgegeben.

Definiert in Zeile 18 der Datei GUITimeline.cpp.

9.27 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.cpp-- Dateireferenz

```
#include "PropertiesBox.h"
#include "constants.h"
#include <iostream>
#include "../SimpleAnalyzerApp.h"
#include "../processing/utils.h"
#include "../processing/ObjectData.h"
```

Include-Abhängigkeitsdiagramm für PropertiesBox.cpp:



Variablen

- wxString sdfilestring [] = { wxT("") }

9.27.1 Variablen-Dokumentation

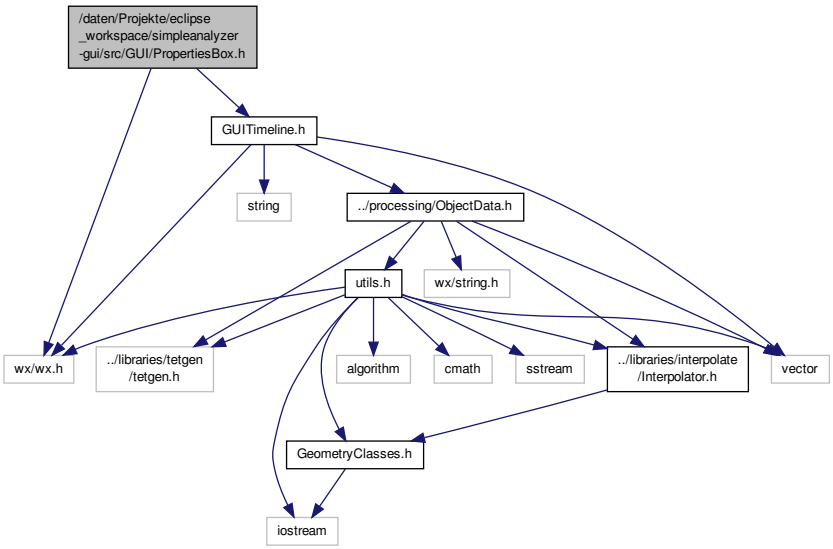
9.27.1.1 wxString sdfilestring[ ] = { wxT("") }

Definiert in Zeile 17 der Datei PropertiesBox.cpp.

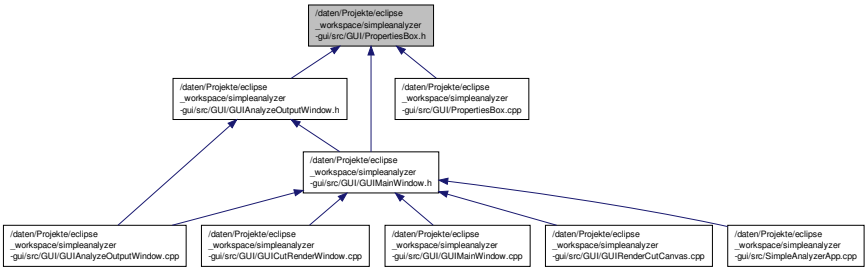
9.28 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/PropertiesBox.h-- Dateireferenz

```
#include <wx/wx.h>
#include "GUITimeline.h"
```

Include-Abhängigkeitsdiagramm für PropertiesBox.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

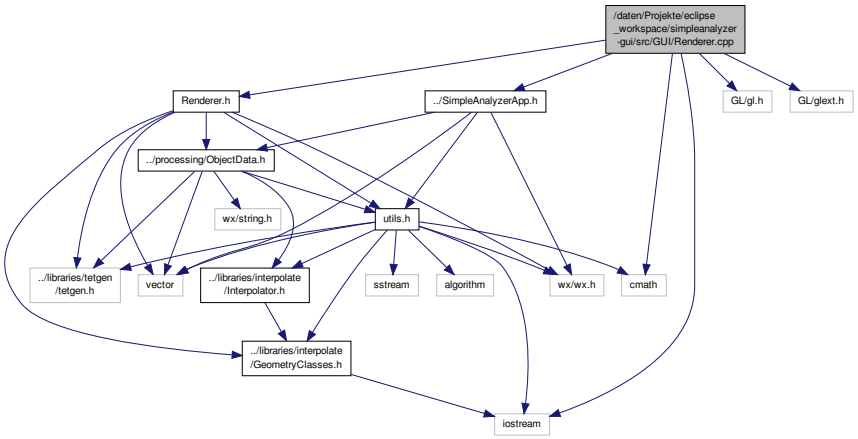
- class PropertiesBox

  Oberfläche zum Verändern/Anzeigen der Eigenschaften eines Objekts.

9.29 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.cpp-- Dateireferenz

```
#include "Renderer.h"
#include <iostream>
#include <cmath>
#include "../SimpleAnalyzerApp.h"
#include <GL/gl.h>
#include <GL/glext.h>
```

Include-Abhängigkeitsdiagramm für Renderer.cpp:



Funktionen

- bool pointBehindCut (Vector3D ∗point, Triangle ∗cut)
- void drawVector (Vector3D ∗pos, Vector3D ∗dir)

  Zeichnet einen Vektor als Pfeil.
- void renderGrid ()

  Zeichnet markante Linien zum leichteren Erfassen des Koordinatensystems.
- void drawCutRenderInfo (CutRender_info ∗info)

  Visualisiert Informationen über eine 2D-Temperaturverteilung.

9.29.1 Dokumentation der Funktionen

9.29.1.1 void drawCutRenderInfo (CutRender_info ∗ *info*)

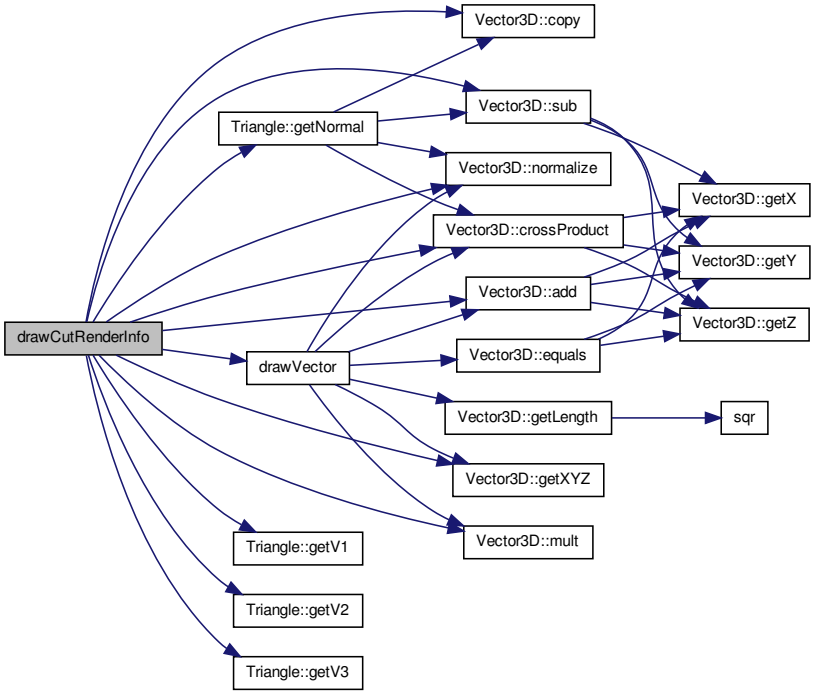
Visualisiert Informationen über eine 2D-Temperaturverteilung.

Parameter

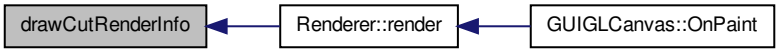
info	Die zu visualisierenden Informationen.

Definiert in Zeile 576 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



9.29.1.2 void drawVector (Vector3D * *pos*, Vector3D * *dir*)

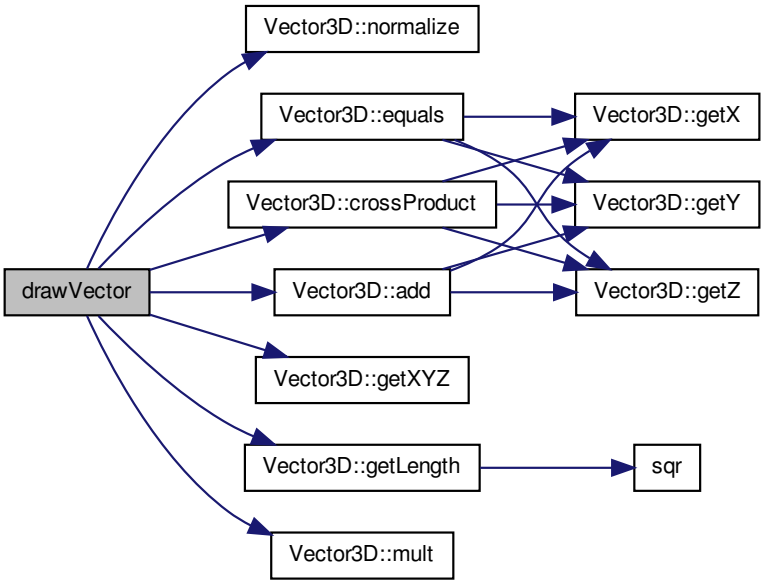
Zeichnet einen Vektor als Pfeil.

Parameter

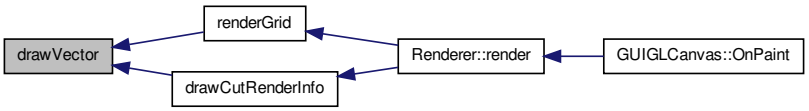
pos	Startpunkt des Pfeils.
dir	Richtung und Länge des Pfeils.

Definiert in Zeile 373 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



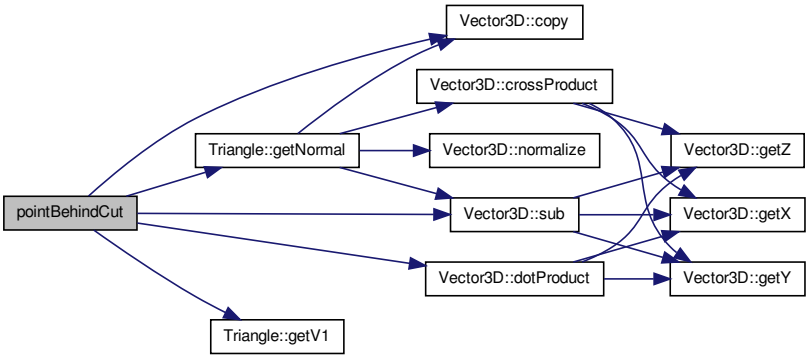
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



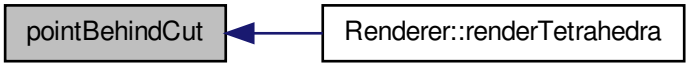
9.29.1.3 bool pointBehindCut (Vector3D * *point,* Triangle * *cut*)

Definiert in Zeile 46 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

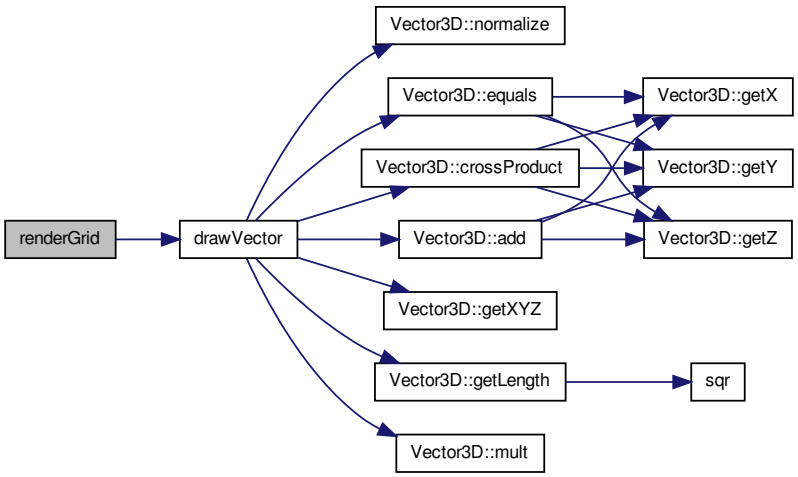


9.29.1.4 void renderGrid ( )

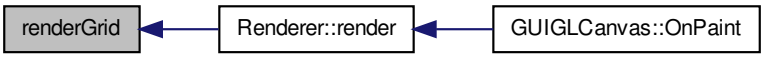
Zeichnet markante Linien zum leichteren Erfassen des Koordinatensystems.

Definiert in Zeile 455 der Datei Renderer.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



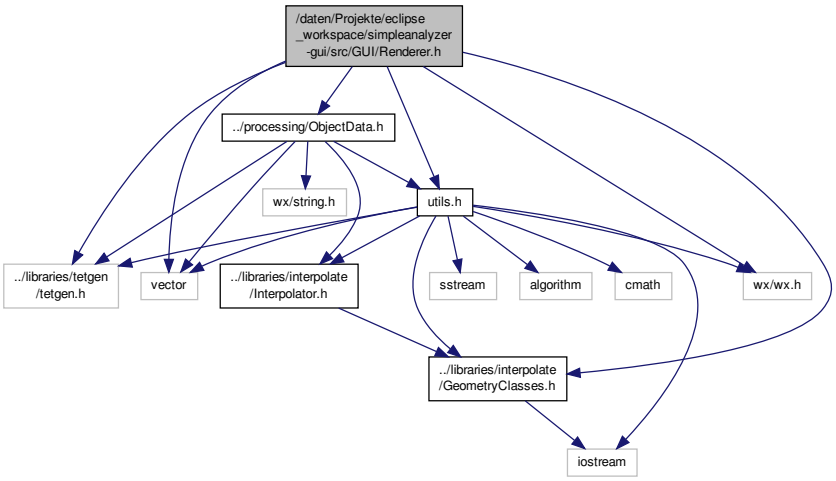
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



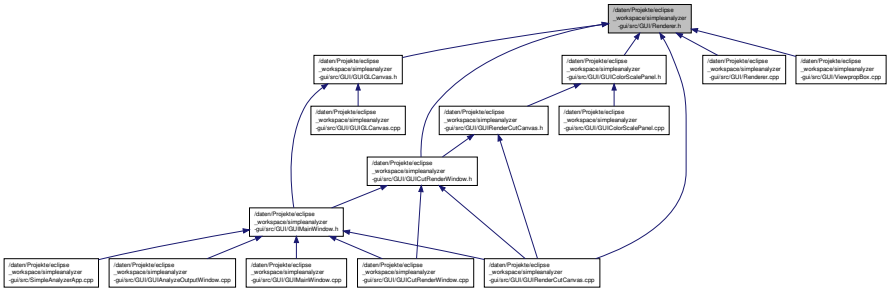
9.30 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/Renderer.h-Dateireferenz

```
#include "../libraries/tetgen/tetgen.h"
#include <vector>
#include "../libraries/interpolate/GeometryClasses.h"
#include "../processing/ObjectData.h"
#include "../processing/utils.h"
#include <wx/wx.h>
```

Include-Abhängigkeitsdiagramm für Renderer.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Renderer

  Zeichnet den Inhalt der 3D-Fensters.

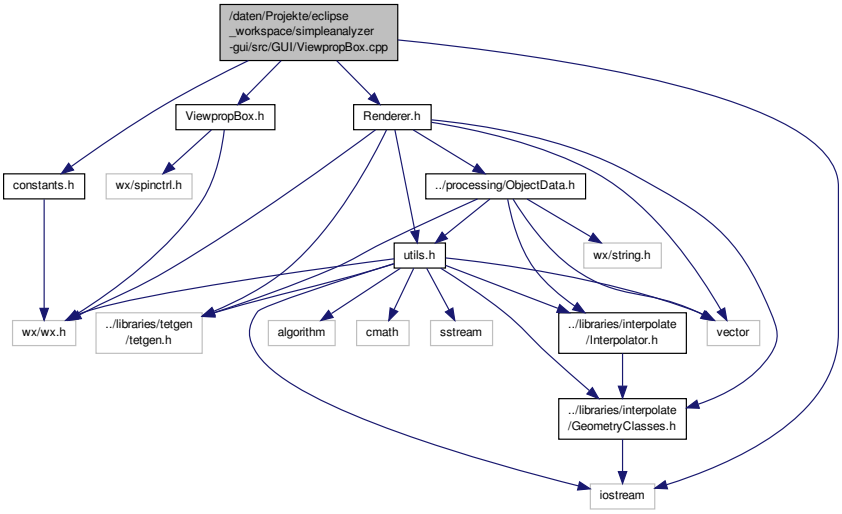
- struct Renderer::Viewport_info

  Informationen über die Ansicht des Modells (Virtuelle Kamera) und welche Elemente dargestellt werden.

9.31 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.cpp-- Dateireferenz

```
#include "ViewpropBox.h"
#include "constants.h"
#include "Renderer.h"
#include <iostream>
```

Include-Abhängigkeitsdiagramm für ViewpropBox.cpp:



Variablen

- wxString renderchoices [ ] = { wxT("Kein"), wxT("Material"), wxT("Wert") }

9.31.1 Variablen-Dokumentation

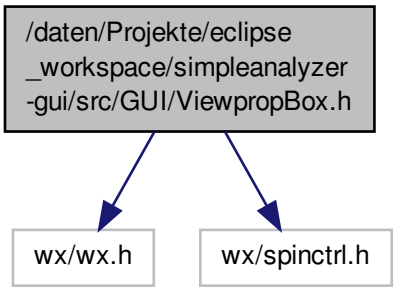
9.31.1.1 wxString renderchoices[ ] = { wxT("Kein"), wxT("Material"), wxT("Wert") }

Definiert in Zeile 15 der Datei ViewpropBox.cpp.

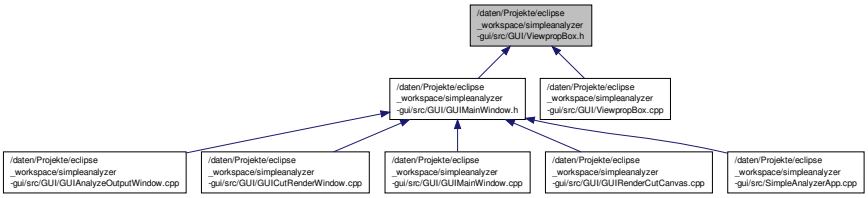
9.32 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/GUI/ViewpropBox.h-- Dateireferenz

```
#include <wx/wx.h>
#include <wx/spinctrl.h>
```

Include-Abhängigkeitsdiagramm für ViewpropBox.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

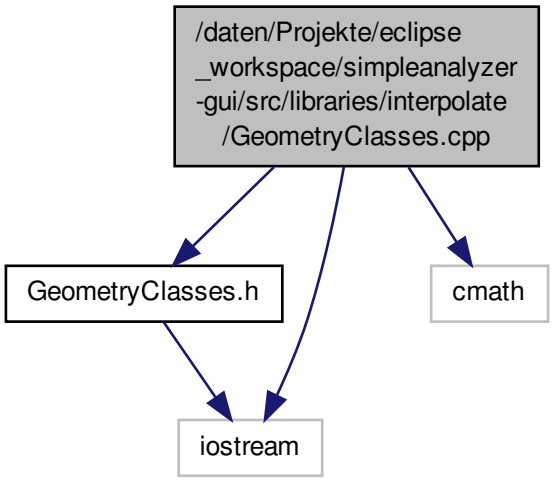
- class ViewpropBox

  Oberfläche zum Verändern/Anzeigen der Visualisierungsoptionen.

9.33 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/-GeometryClasses.cpp-Dateireferenz

```
#include "GeometryClasses.h"
#include <iostream>
#include <cmath>
```

Include-Abhängigkeitsdiagramm für GeometryClasses.cpp:



Makrodefinitionen

- #define EPSILON 0.0000001

Funktionen

- double sqr (double v)
- std::ostream & operator<< (std::ostream &out, const Vector3D &vec)

  Definition des <<-Operators für die Ausgabe eines Vektors.

9.33.1 Makro-Dokumentation

9.33.1.1 #define EPSILON 0.0000001

Definiert in Zeile 13 der Datei GeometryClasses.cpp.

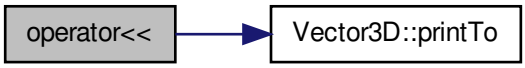
9.33.2 Dokumentation der Funktionen

9.33.2.1 std::ostream& operator<< (std::ostream & *out,* const Vector3D & *vec*)

Definition des <<-Operators für die Ausgabe eines Vektors.

Definiert in Zeile 132 der Datei GeometryClasses.cpp.

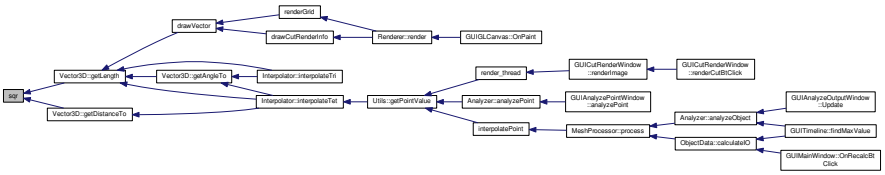
Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



9.33.2.2 double sqr (double *v*) [inline]

Definiert in Zeile 14 der Datei GeometryClasses.cpp.

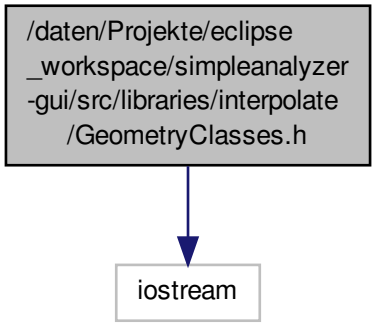
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



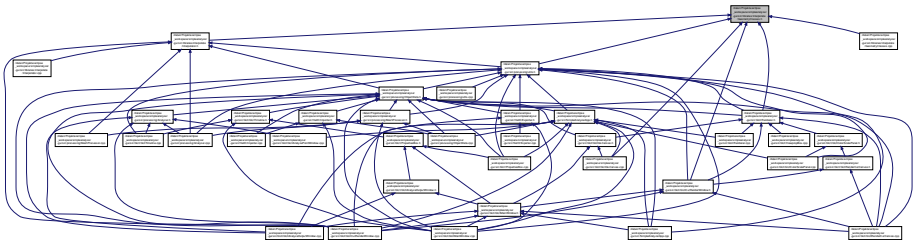
9.34 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/-GeometryClasses.h-Dateireferenz

```
#include <iostream>
```

Include-Abhängigkeitsdiagramm für GeometryClasses.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Vector3D

  3D-Vektorklasse mit nützlichen Operationen.
- class Matrix3D

  3x3-Matrixklasse mit Operationen.
- class Triangle

  Ein durch 3 Ortsvektoren beschriebenes Dreieck.
- class Tetrahedron

  Ein durch 4 Ortsvektoren beschriebener Tetraeder.

Funktionen

- std::ostream & operator<< (std::ostream &out, const Vector3D &vec)

  Definition des <<-Operators für die Ausgabe eines Vektors.

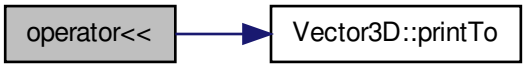
9.34.1 Dokumentation der Funktionen

9.34.1.1 std::ostream& operator<< (std::ostream & *out,* const Vector3D & *vec*)

Definition des <<-Operators für die Ausgabe eines Vektors.

Definiert in Zeile 132 der Datei GeometryClasses.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:

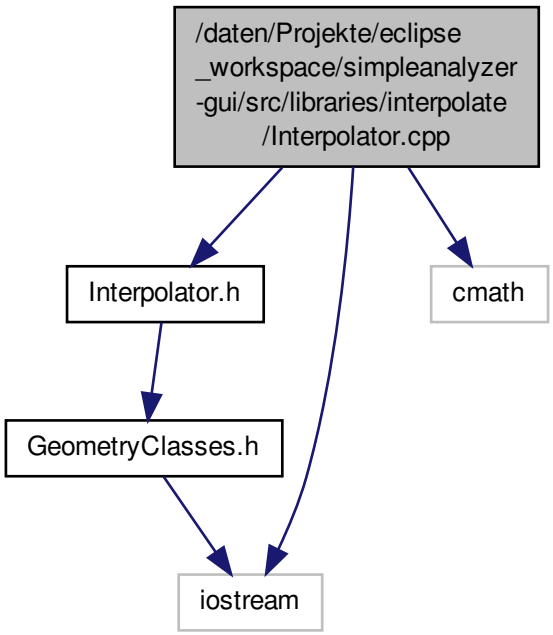


9.35 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/-Interpolator.cpp-Dateireferenz

```
#include "Interpolator.h"
```

```
#include <iostream>
#include <cmath>
```

Include-Abhängigkeitsdiagramm für Interpolator.cpp:



Makrodefinitionen

- #define PI 3.14159265358979323846

Funktionen

- double sqr (double v)
- double getSign (double x)

9.35.1 Makro-Dokumentation

9.35.1.1 #define PI 3.14159265358979323846

Definiert in Zeile 4 der Datei Interpolator.cpp.

9.35.2 Dokumentation der Funktionen

9.35.2.1 double getSign (double *x*) [inline]

Definiert in Zeile 20 der Datei Interpolator.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

9.35.2.2 double sqr (double *v*) [inline]

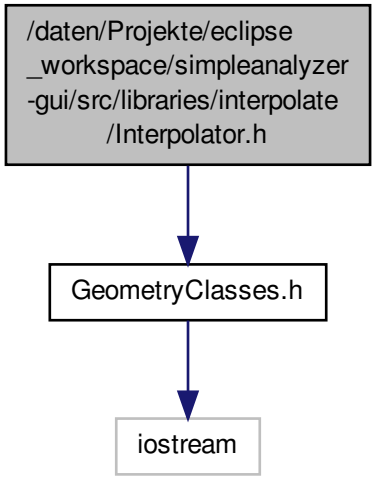
Definiert in Zeile 7 der Datei Interpolator.cpp.

Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:

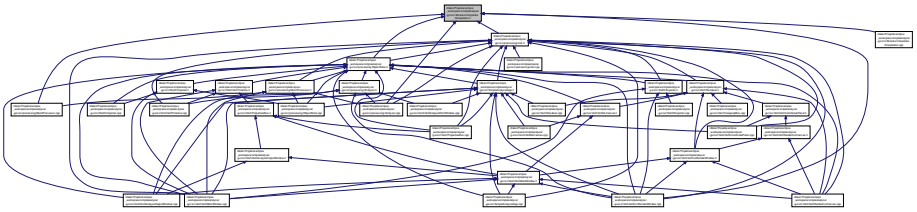
9.36 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/libraries/interpolate/-Interpolator.h-Dateireferenz

```
#include "GeometryClasses.h"
```

Include-Abhängigkeitsdiagramm für Interpolator.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

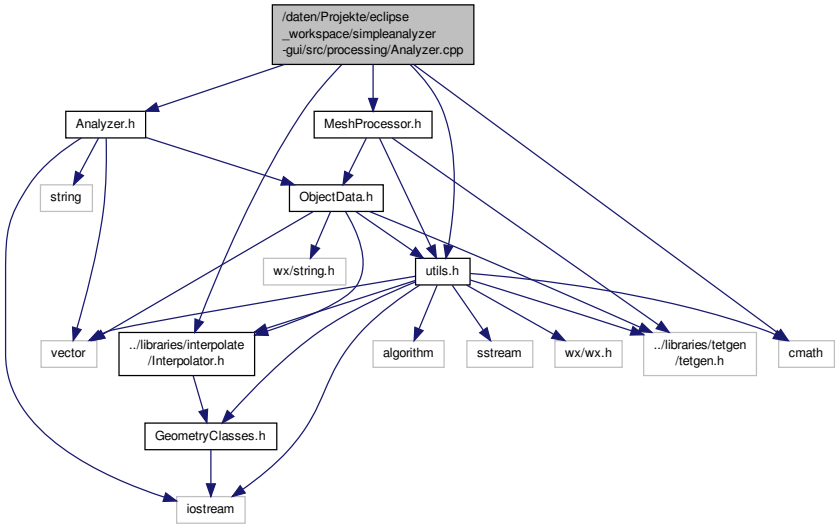
- class Interpolator

  2- und 3-dimensionale Inter-/Extrapolation

9.37 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.cpp-Dateireferenz

```
#include "Analyzer.h"
#include <cmath>
#include "MeshProcessor.h"
#include "../libraries/interpolate/Interpolator.h"
#include "utils.h"
```

Include-Abhängigkeitsdiagramm für Analyzer.cpp:



Funktionen

- std::ostream & operator<< (std::ostream &out, const Analyzer::AnalyzerData_object &data)

9.37.1 Dokumentation der Funktionen

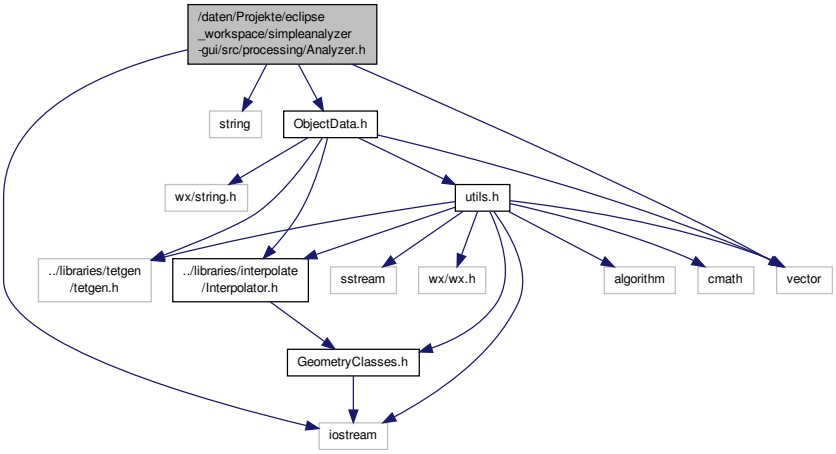
9.37.1.1 std::ostream& operator<< (std::ostream & *out,* const Analyzer::AnalyzerData_object & *data*)

Definiert in Zeile 164 der Datei Analyzer.cpp.

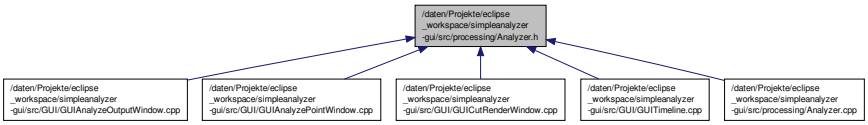
9.38 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/Analyzer.h-- Dateireferenz

```
#include <vector>
#include <string>
#include <iostream>
#include "ObjectData.h"
```

Include-Abhängigkeitsdiagramm für Analyzer.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class Analyzer

  Ermittelt Daten aus der Temperaturverteilung.
- struct Analyzer::AnalyzerData_material

  Analyseergebnisse für ein Material.
- struct Analyzer::AnalyzerData_dataset

  Analyseergebnisse für einen Sensordatensatz.
- struct Analyzer::AnalyzerData_object

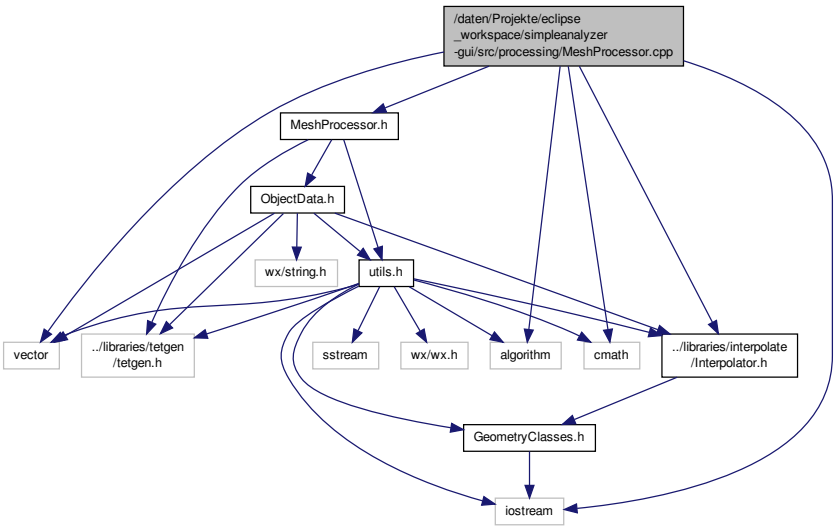
  Analyseergebnisse für ein Objekt.
- struct Analyzer::AnalyzerData_point

  Analyseergebnisse für einen Punkt.

9.39 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.cpp-Dateireferenz

```
#include "MeshProcessor.h"
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include "../libraries/interpolate/Interpolator.h"
```

Include-Abhängigkeitsdiagramm für MeshProcessor.cpp:



Funktionen

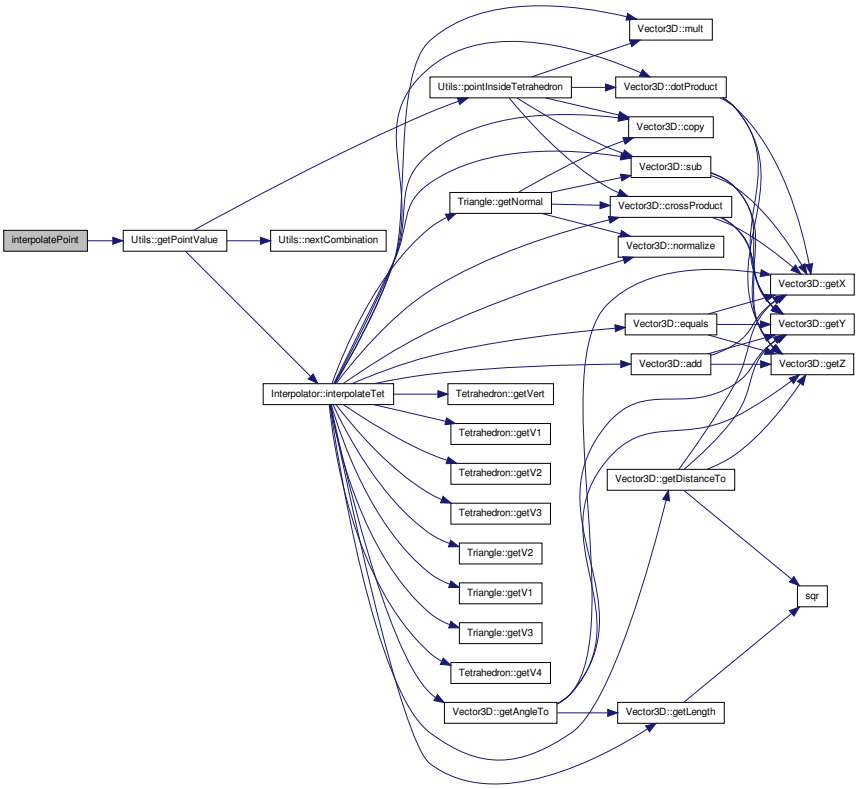
- void interpolatePoint (ObjectData::MaterialData ∗data, vector< SensorPoint > ∗sensorpoints, int pointIndex, Interpolator ∗interpolator)

9.39.1 Dokumentation der Funktionen

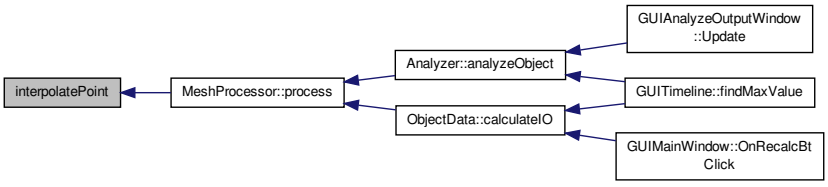
9.39.1.1 void interpolatePoint (ObjectData::MaterialData ∗ *data,* vector< SensorPoint > ∗ *sensorpoints,* int *pointIndex,* Interpolator ∗ *interpolator*)

Definiert in Zeile 20 der Datei MeshProcessor.cpp.

Hier ist ein Graph, der zeigt, was diese Funktion aufruft:



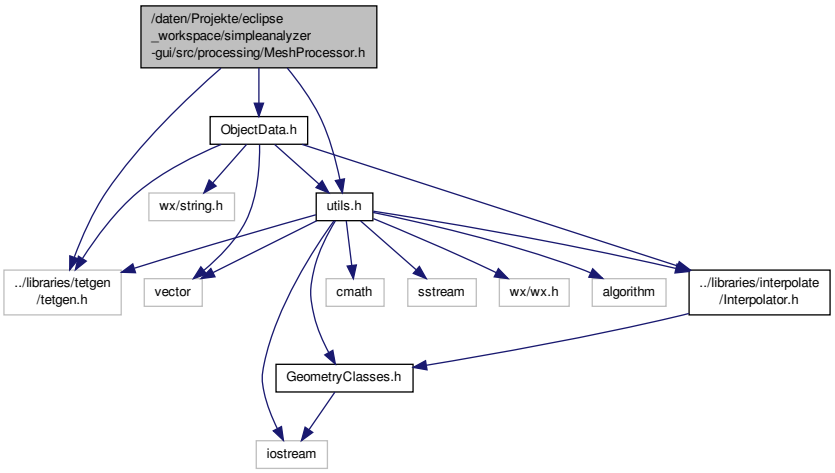
Hier ist ein Graph der zeigt, wo diese Funktion aufgerufen wird:



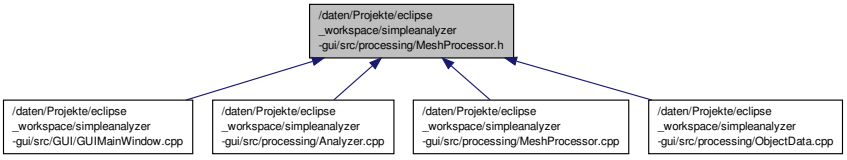
9.40 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/MeshProcessor.h-Dateireferenz

```
#include "../libraries/tetgen/tetgen.h"
#include "ObjectData.h"
#include "utils.h"
```

Include-Abhängigkeitsdiagramm für MeshProcessor.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

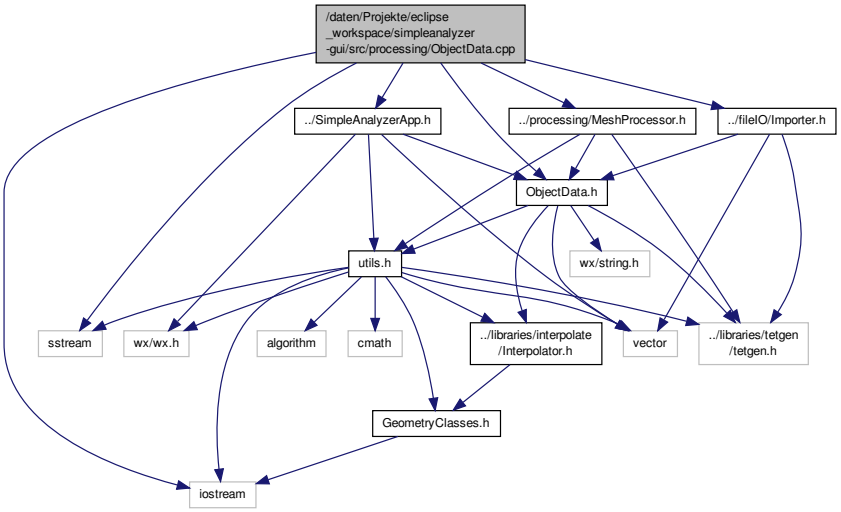
- class MeshProcessor

  Errechnet die Temperaturverteilung für ein Objekt.

9.41 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.cpp-Dateireferenz

```
#include "ObjectData.h"
#include "../SimpleAnalyzerApp.h"
#include "../fileIO/Importer.h"
#include "../processing/MeshProcessor.h"
#include <iostream>
#include <sstream>
```

Include-Abhängigkeitsdiagramm für ObjectData.cpp:



Makrodefinitionen

- #define PATH_SEPARATOR '/'

9.41.1 Makro-Dokumentation

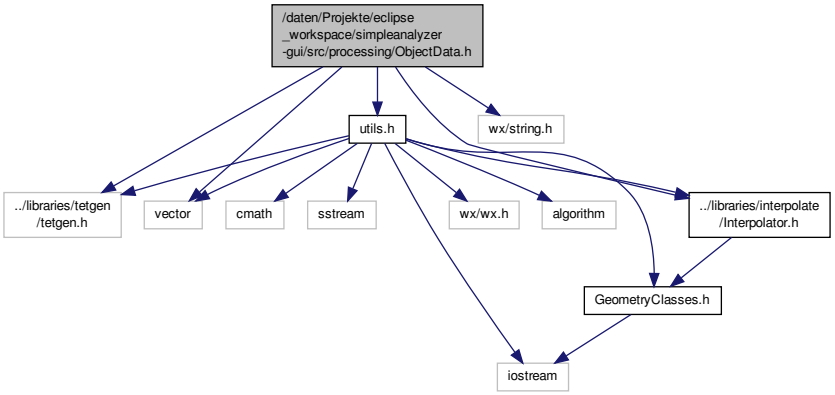
9.41.1.1 #define PATH_SEPARATOR '/'

Definiert in Zeile 20 der Datei ObjectData.cpp.

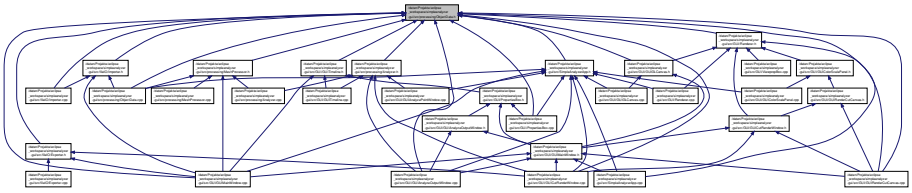
9.42 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/ObjectData.h-Dateireferenz

```
#include "../libraries/tetgen/tetgen.h"
#include <vector>
#include "../libraries/interpolate/Interpolator.h"
#include <wx/string.h>
#include "utils.h"
```

Include-Abhängigkeitsdiagramm für ObjectData.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- class ObjectData

  Die Daten eines Versuchsobjekts.
- struct ObjectData::MaterialData

  Die Daten eines Materials.

Makrodefinitionen

- #define NUMBEROFSENSORATTRIBUTES 1

9.42.1 Makro-Dokumentation

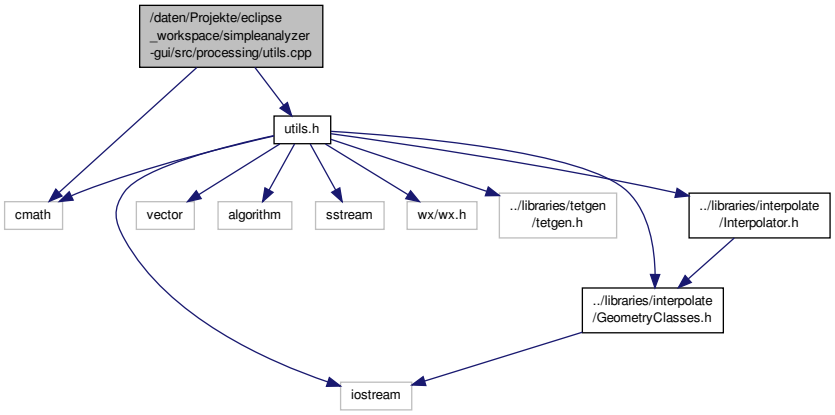
9.42.1.1 #define NUMBEROFSENSORATTRIBUTES 1

Definiert in Zeile 16 der Datei ObjectData.h.

9.43 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.cpp-- Dateireferenz

```
#include "utils.h"
#include <cmath>
```

Include-Abhängigkeitsdiagramm für utils.cpp:



Makrodefinitionen

- #define EPSILON 0.000001

9.43.1 Makro-Dokumentation

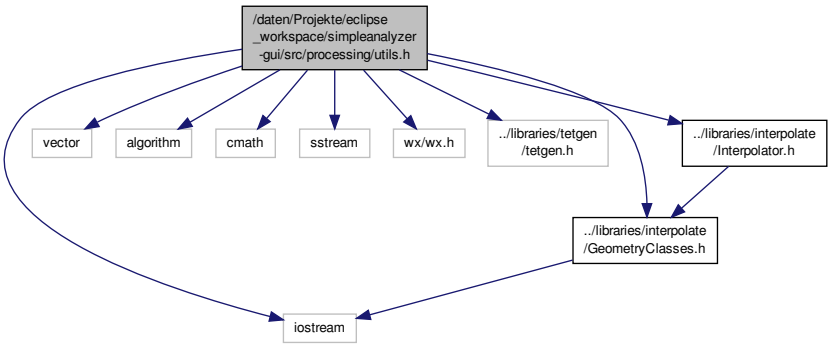
9.43.1.1 #define EPSILON 0.000001

Definiert in Zeile 14 der Datei utils.cpp.

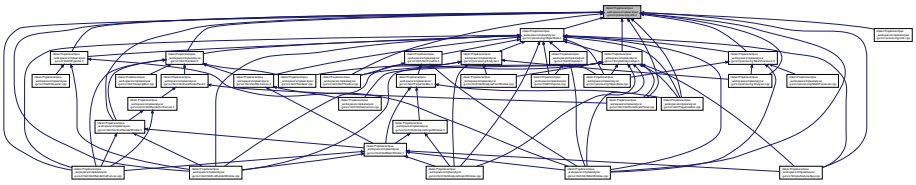
9.44 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/processing/utils.h-- Dateireferenz

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>
#include <sstream>
#include <wx/wx.h>
#include "../libraries/tetgen/tetgen.h"
#include "../libraries/interpolate/GeometryClasses.h"
#include "../libraries/interpolate/Interpolator.h"
```

Include-Abhängigkeitsdiagramm für utils.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:



Klassen

- struct Utils::Visualization_info

  Informationen über die Farbgebung bei der Visualisierung.
- struct Utils::SortStruct

  Hilfsstruktur zum Sortieren von Punkten nach dem Abstand zu einem anderen Punkt.
- struct Utils::SensorPoint

  Daten eines Sensordatenpunktes.
- struct Utils::CutRender_info

  Daten zur Darstellung einer 2D-Temperaturverteilungs-Ebene.
- struct Utils::SensorData

  Ein Sensordatensatz.
- struct Utils::SensorPointComparator

  Hilfsstruktur zum Vergleichen des Abstands von Messpunkten.

Namensbereiche

- Utils

  allgemeine Funktionen und Typen.

Aufzählungen

- enum Utils::PIM_algorithm { Utils::ALGORITHM_TETRAHEDRONS = 0, Utils::ALGORITHM_RAY }

  Zum Punkt-in-Volumen Testen verwendeter Algorithmus.

Funktionen

- double Utils::sqr (double d)

  Quadriert eine Zahl.

- float Utils::clampHue (float h)

  Begrenzt einen Wert auf den Bereich 0..1.

- string Utils::floattostr (double val)

  Hilfsfunktion zur Umwandlung einer Zahl in einen String.

- wxString Utils::floattowxstr (double val)

  Wandelt eine Fließkommazahl in einen wxWidgets-String um.

- wxString Utils::floattowxstr (double val, int digits)

  Wandelt eine Fließkommazahl in einen wxWidgets-String um.

- int Utils::rayIntersectsTriangle (Vector3D *p, Vector3D *direction, Triangle *tri, double *depth)

  Testet, ob ein Strahl ein Dreieck schneidet.

- int Utils::pointInsideMesh (Vector3D *p, tetgenio *io, PIM_algorithm algorithm)

  Testet, ob sich ein Punkt innerhalb eines Körpers befindet.

- int Utils::pointInsideTetrahedron (Vector3D *pges, Vector3D *v1, Vector3D *v2, Vector3D *v3, Vector3D *v4)

  Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

- int Utils::pointInsideTetrahedron (double *pges, double *v1, double *v2, double *v3, double *v4)

  Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

- int Utils::pointInsideTetrahedron (double *p, vector< SensorPoint * > *tet)

  Testet, ob sich ein Punkt innerhalb eines Tetraeders befindet.

- void Utils::nextCombination (vector< int > *indices, int depth, int dataPointCount)

  Ermöglicht das generieren aller möglichen Verteilungen von 4 Elementen auf dataPointCount Plätze.

- double Utils::getPointValue (int &status, vector< SensorPoint > *sensorpoints, double *p, Interpolator *interpolator, vector< SensorPoint * > *prev_tet=NULL, vector< SensorPoint * > *current_tet=NULL)

  Gibt den inter/extrapolierten Wert eines Punktes zurück.

- float * Utils::hsvToRgb (float h, float s, float v)

  Wandelt eine Farbe im HSV-Format ins RGB-Format um.

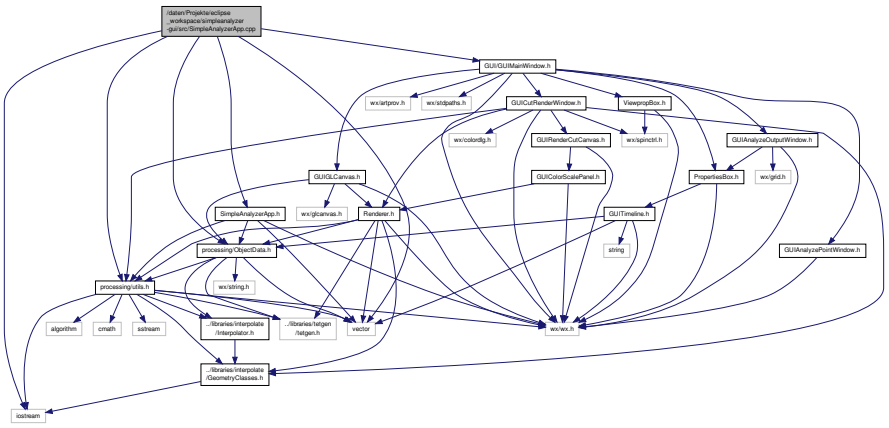
- void Utils::copySensorPoint (SensorPoint *from, SensorPoint *to)

  Kopiert die Eigenschaften eines Sensorpunktes in einen Anderen.

9.45 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.cpp-Dateireferenz

```
#include <iostream>
#include "SimpleAnalyzerApp.h"
#include "GUI/GUIMainWindow.h"
#include "processing/ObjectData.h"
#include "processing/utils.h"
#include <vector>
```

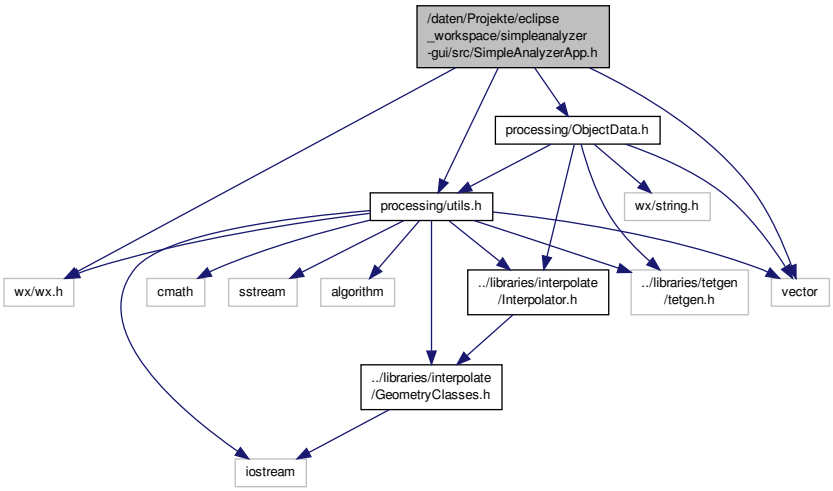
Include-Abhängigkeitsdiagramm für SimpleAnalyzerApp.cpp:



9.46 /daten/Projekte/eclipse_workspace/simpleanalyzer-gui/src/SimpleAnalyzerApp.h-- Dateireferenz

```
#include <wx/wx.h>
#include <vector>
#include "processing/utils.h"
#include "processing/ObjectData.h"
```

Include-Abhängigkeitsdiagramm für SimpleAnalyzerApp.h:



Dieser Graph zeigt, welche Datei direkt oder indirekt diese Datei enthält:

Klassen

- class SimpleAnalyzerApp

  Regelt den allgemeinen Ablauf des Programms.

Index