

Chapter 9

Namespace Documentation

9.1 io Namespace Reference

contains functions to work with various data types input functions loads and converts them to pcl::PointCloud output functions (doesn't exist right now)

9.1.1 Detailed Description

contains functions to work with various data types input functions loads and converts them to pcl::PointCloud output functions (doesn't exist right now)

9.2 mtds Namespace Reference

9.2.1 Detailed Description

contains all reconstuction methods

9.3 r3d Namespace Reference

Our team dedicated namespace R3D.

Namespaces

- **colorgen**
- **data**
- **exporter**
- **io**
- **mtds**
- **obj**
- **parser**
- **prims**
- **remover**
- **space**
- **visualization**

Classes

- class **CommonUtil**
- struct **HASH**

Variables

- const long double **PI** = 4 * atan(1)

9.3.1 Detailed Description

Our team dedicated namespace R3D.

DXF exporter class.

A namespace.

Contains other subnamespaces to distinct method types in our reconstruction project

A more detailed namespace description.

This class exports data into DXF (a CAD format).

Author

Michal Lffler

Version

1.0

Date

29. 2. 2016

Warning

We are using "trial" version of sgCore. I have no idea how long will this code work.

9.3.2 Variable Documentation

9.3.2.1 const long double r3d::PI = 4 * atan(1)

Definition at line 33 of file CommonUtil.h.

9.4 r3d::colorgen Namespace Reference

Classes

- class **ColorGenerator**

9.5 r3d::data Namespace Reference

Classes

- class **DataStatistics**
Statistics about the segmented pointclouds.

9.6 r3d::exporter Namespace Reference

Classes

- class **Exporter**

9.7 r3d::io Namespace Reference

Functions

- template<typename PointT >
pcl::PointCloud< PointT > * **loadCsvFile** (const std::string fileName)
Loads data coordinates from certain type of CSV file.
- template<typename PointT >
pcl::PointCloud< PointT > * **loadVTKMeshFile** (const std::string fileName)
Loads mesh from VTK file and converts it to pcl::PointCloud.
- template<typename PointT >
pcl::PointCloud< PointT > * **loadPcdFile** (const std::string fileName)
Loads points from PCD file.
- template<typename PointT >
pcl::PointCloud< PointT > * **loadPlyFile** (const std::string fileName)
Loads points from PLY file.
- template<typename PointT >
pcl::PointCloud< PointT > * **loadObjFile** (const std::string fileName)
Loads points from OBJ file.
- template<typename PointT >
pcl::PointCloud< PointT > * **loadFiles** (const std::vector< std::string > files)

9.7.1 Function Documentation

9.7.1.1 template<typename PointT > pcl::PointCloud< PointT > * r3d::io::loadCsvFile (const std::string fileName)

Loads data coordinates from certain type of CSV file.

Parameters

in	<i>fileName</i>	file name to load
----	-----------------	-------------------

Returns

pointcloud of loaded points from file

Definition at line 5 of file DataReader.cpp.

9.7.1.2 `template<typename PointT > pcl::PointCloud<PointT>* r3d::io::loadFiles (const std::vector< std::string > files)`

An template type. PointT template that contains mainly pcl::PointXYZ and pcl::PointXYZRGB data

Parameters

<i>files</i>	an std::vector<std::string> argument of all files to be reader/loaded.
--------------	--

See also

loadObjFile() (p. ??), **loadPlyFile()** (p. ??), **loadPcdFile()** (p. ??), **loadVTKMeshFile()** (p. ??) and **loadCsvFile()** (p. ??)

Returns

cloudData

cloudData pointer.

temporary storage for data that is filled from all selected files, at the end returned

Point Cloud

Definition at line 119 of file DataReader.h.

9.7.1.3 `template<typename PointT > pcl::PointCloud<PointT>* r3d::io::loadObjFile (const std::string fileName)`

Loads points from OBJ file.

Parameters

<i>fileName</i>	Name of file to read data from
-----------------	--------------------------------

Returns

pointcloud of loaded points from file

Definition at line 98 of file DataReader.h.

9.7.1.4 `template<typename PointT > pcl::PointCloud<PointT>* r3d::io::loadPcdFile (const std::string fileName)`

Loads points from PCD file.

Parameters

<i>fileName</i>	Name of file to read data from
-----------------	--------------------------------

Returns

pointcloud of loaded points from file

< Detailed description after the member

Definition at line 60 of file DataReader.h.

9.7.1.5 `template<typename PointT > pcl::PointCloud<PointT>* r3d::io::loadPlyFile (const std::string fileName)`

Loads points from PLY file.

Parameters

<i>fileName</i>	Name of file to read data from
-----------------	--------------------------------

Returns

pointcloud of loaded points from file

Definition at line 79 of file DataReader.h.

9.7.1.6 `template<typename PointT > pcl::PointCloud<PointT>* r3d::io::loadVTKMeshFile (const std::string fileName)`

Loads mesh from VTK file and converts it to pcl::PointCloud.

Parameters

<i>fileName</i>	Name of file to read data from
-----------------	--------------------------------

Returns

pointcloud of loaded points from file

Definition at line 39 of file DataReader.h.

9.8 r3d::mtds Namespace Reference

Classes

- class **BirkysMethodReconstruction**

*This is a derived class from class **Reconstruction** (p. ??). Contains methods and variables for 3D reconstruction using normal estimation, normal histogram, point labeling/coloring, region growing.*

- class **CloudRotationTest**
- class **GrowingRegionReconstruction**

Composite class aggregating reconstruction methods and simple outliers segmentation Uses Birkis segmenation by growing region and color labeling Reconstructs planar objects using RANSAC analytical representation Segments outliers of these planar objects.

- class **NúmeroUnoReconstruction**
- class **PlanePointsExamination**
- class **Reconstruction**

*This is an abstract parent class to all **Reconstruction** (p. ??) methods. Contains original dataset as partial point clouds in aggregated ClosedSpace object m_methodName should be set to dedicated reconstruction method name.*

- class **SlicBasedReconstruction**

*This is a derived class from class **Reconstruction** (p. ??). Contains methods and variables for 3D reconstruction based on SLIC clustering algorithm STILL UNDER CONSTRUCTION.*

9.9 r3d::obj Namespace Reference

Classes

- class **Object**
Class of object.
- class **Wall**
*default shape for simple **Wall** (p. ??) object*

9.10 r3d::parser Namespace Reference

Classes

- class **InputDataParser**
Data parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

9.11 r3d::prims Namespace Reference

Classes

- class **Line**
- class **PlaneSegment**
- class **Polygon**
- class **Primitives**
- class **Rectangle**

9.12 r3d::remover Namespace Reference

Classes

- class **NoiseRemover**
Removes noise from point cloudData parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

9.13 r3d::space Namespace Reference

Classes

- class **ClosedSpace**

This class contains all stored data, reconstructed primitives and objects segmented in scene.

9.14 r3d::visualization Namespace Reference

Classes

- class **CustomInteractor**

9.15 space Namespace Reference

9.15.1 Detailed Description

Contains space and data storing and administrating classes

Chapter 10

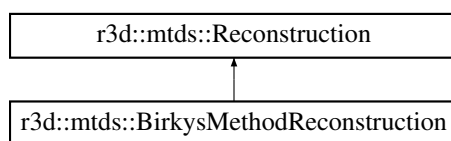
Class Documentation

10.1 r3d::mtds::BirkysMethodReconstruction Class Reference

This is a derived class from class **Reconstruction** (p. ??). Contains methods and variables for 3D reconstruction using normal estimation, normal histogram, point labeling/coloring, region growing.

```
#include "BirkysMethodReconstruction.h"
```

Inheritance diagram for r3d::mtds::BirkysMethodReconstruction:



Public Member Functions

- **BirkysMethodReconstruction** ()
- **~BirkysMethodReconstruction** ()
- unsigned short **getM_OmpNumberOfCores** () const
- void **setM_OmpNumberOfCores** (unsigned short m_omp_number_of_cores)
- unsigned short **getM_NeNumberOfNeighbours** () const
- void **setM_NeNumberOfNeighbours** (unsigned short m_ne_number_of_neighbours)
- unsigned short **getM_NumberOfDominantRegions** () const
- void **setM_NumberOfDominantRegions** (unsigned short m_number_of_dominant_regions)
- unsigned short **getM_HistogramDeletionArea** () const
- void **setM_HistogramDeletionArea** (unsigned short m_histogram_deletion_area)
- unsigned short **getM_GrNumberOfNeighbours** () const
- void **setM_GrNumberOfNeighbours** (unsigned short m_gr_number_of_neighbours)
- unsigned short **getM_PointColorThreshold** () const
- void **setM_PointColorThreshold** (unsigned short m_point_color_threshold)
- unsigned short **getM_MinClusterSize** () const
- void **setM_MinClusterSize** (unsigned short m_min_cluster_size)
- unsigned short **getM_NumberOfPlains** () const
- void **setM_NumberOfPlains** (unsigned short m_number_of_plains)
- unsigned short **getM_PointsPerHalfSphere** () const

- void **setM_PointsPerHalfSphere** (unsigned short m_points_per_half_sphere)
- float **getM_DistanceThreshold** () const
- void **setM_DistanceThreshold** (float m_distance_threshold)
- float **getM_DominantNormalsSearchRadius** () const
- void **setM_DominantNormalsSearchRadius** (float m_dominant_normals_search_radius)
- virtual bool **calculate** () override
virtual function responsible for the whole reconstruction calculations
- virtual void **visualise** (pcl::visualization::PCLVisualizer *viewer) override
virtual function for debug/test visualisation needed during development
- virtual void **setConfiguration** (TiXmlNode *param) override
virtual function for setting up the member variables from the XML config file
- pcl::PointCloud< pcl::PointXYZ >::Ptr **createHalfSphere** (unsigned short pointsPerHalfSphere)
function which generates equidistributed points on the surface of a half sphere The output half sphere is designed for searching dominant normals
- pcl::PointCloud< pcl::PointXYZ >::Ptr **findDominantNormals** (pcl::PointCloud< pcl::PointXYZ >::Ptr normalHistogram, unsigned short numberOfDominantNormals, float dominantNormalsSearchRadius, pcl::PointCloud< pcl::PointXYZ >::Ptr halfSphere)
function which finds a given count of dominant normals in the input histogram of normals
- pcl::PointCloud< pcl::PointXYZRGB >::Ptr **labelPointsAccordingToDominantNormals** (pcl::PointCloud< pcl::PointXYZ >::Ptr cloud, pcl::PointCloud< pcl::Normal >::Ptr normals, pcl::PointCloud< pcl::PointXYZ >::Ptr dominantNormals)
function which label each point of the cloud according to the "nearest" dominant normal
- pcl::PointCloud< pcl::PointXYZRGB >::Ptr **birkysMethodSegmentation** (std::vector< pcl::PointIndices > *clusters)
function which runs the whole segmentation method in sequence

Static Public Member Functions

- static pcl::PointCloud< pcl::Normal >::Ptr **estimateNormals** (pcl::PointCloud< pcl::PointXYZ >::Ptr cloud, XYZ, unsigned short numberOfCores, unsigned short numberOfNeighbours)
function for normal estimation of a point cloud
- static pcl::PointCloud< pcl::PointXYZ >::Ptr **createHistogramOfNormals** (pcl::PointCloud< pcl::Normal >::Ptr normals)
function which creates a histogram of normals from the input normals
- static pcl::RegionGrowingRGB< pcl::PointXYZRGB > **regionGrowingRGB** (pcl::PointCloud< pcl::PointXYZRGB >::Ptr cloudRGB, unsigned short numberOfNeighbours, unsigned short pointColorThreshold, unsigned short minClusterSize, std::vector< pcl::PointIndices > *clusters)
function which apply region growing algorithm to the colored input point cloud, region growing algorithm segments based on color

Additional Inherited Members

10.1.1 Detailed Description

This is a derived class from class **Reconstruction** (p. ??). Contains methods and variables for 3D reconstruction using normal estimation, normal histogram, point labeling/coloring, region growing.

Author

Robert Birkus

Definition at line 23 of file BirkysMethodReconstruction.h.

10.1.2 Constructor & Destructor Documentation

10.1.2.1 r3d::mtds::BirkysMethodReconstruction::BirkysMethodReconstruction ()

Definition at line 5 of file BirkysMethodReconstruction.cpp.

10.1.2.2 r3d::mtds::BirkysMethodReconstruction::~~BirkysMethodReconstruction ()

Definition at line 25 of file BirkysMethodReconstruction.cpp.

10.1.3 Member Function Documentation

10.1.3.1 pcl::PointCloud< pcl::PointXYZRGB >::Ptr r3d::mtds::BirkysMethodReconstruction::birkysMethodSegmentation (std::vector< pcl::PointIndices > * *clusters*)

function which runs the whole segmentation method in sequence

Parameters

out	<i>clusters</i>	- segmented clusters of the point cloud
out	<i>segmented</i>	(colored) point cloud

< Loading needed data and define needed variables

< Calculate normals if they are not available

< Create a half sphere of potencial dominant directions

< searching for dominant normals

< Labeling the points of the cloud according to the found dominant normals

< Region growing

Definition at line 248 of file BirkysMethodReconstruction.cpp.

10.1.3.2 bool r3d::mtds::BirkysMethodReconstruction::calculate () [override], [virtual]

virtual function responsible for the whole reconstruction calculations

< Loading needed data and define needed variables

< Calculate normals if they are not available

< Create a half sphere of potencial dominant directions

< searching for dominant normals

< Labeling the points of the cloud according to the found dominant normals

< Region growing

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 283 of file BirkysMethodReconstruction.cpp.

10.1.3.3 `pcl::PointCloud< pcl::PointXYZ >::Ptr r3d::mtds::BirkysMethodReconstruction::createHalfSphere (unsigned short pointsPerHalfSphere)`

function which generates equidistributed points on the surface of a half sphere The output half sphere is designed for searching dominant normals

Parameters

in	<i>pointsPerHalfSphere</i>	- number of points on the half sphere
out	<i>point</i>	cloud of the half sphere

< setting this to `pointsPerHalfSphere` give us a whole sphere

< the radius of our sphere, we need always a half sphere with radius 1.0 for our method

Definition at line 84 of file `BirkysMethodReconstruction.cpp`.

10.1.3.4 `pcl::PointCloud< pcl::PointXYZ >::Ptr r3d::mtds::BirkysMethodReconstruction::createHistogramOfNormals (pcl::PointCloud< pcl::Normal >::Ptr normals) [static]`

function which creates a histogram of normals from the input normals

Parameters

in	<i>normals</i>	- input normals of point cloud
out	<i>3D</i>	histogram of normals

< Create histogram of normals

We are taking into account only half of the sphere, the opposite directions are the same for us

Definition at line 61 of file `BirkysMethodReconstruction.cpp`.

10.1.3.5 `pcl::PointCloud< pcl::Normal >::Ptr r3d::mtds::BirkysMethodReconstruction::estimateNormals (pcl::PointCloud< pcl::PointXYZ >::Ptr cloudXYZ, unsigned short numberOfCores, unsigned short numberOfNeighbours) [static]`

function for normal estimation of a point cloud

Parameters

in	<i>cloudXYZ</i>	- XYZ point cloud <code>pcl::PointCloud<pcl::PointXYZ>::Ptr</code>
in	<i>numberOfCores</i>	- number of cores for the parallel computation
in	<i>numberOfNeighbours</i>	- number of neighbours considered during normal estimation computation
out	<i>normals</i>	of the input point cloud <code>pcl::PointCloud<pcl::Normal>::Ptr</code>

< Normal estimation

Definition at line 47 of file `BirkysMethodReconstruction.cpp`.

10.1.3.6 `pcl::PointCloud< pcl::PointXYZ >::Ptr r3d::mtds::BirkysMethodReconstruction::findDominantNormals (pcl::PointCloud< pcl::PointXYZ >::Ptr normalHistogram, unsigned short numberOfDominantNormals, float dominantNormalsSearchRadius, pcl::PointCloud< pcl::PointXYZ >::Ptr halfSphere)`

function which finds a given count of dominant normals in the input histogram of normals

Parameters

in	<i>normalHistogram</i>	- 3D histogram of normals
in	<i>numberOfDominantNormals</i>	- the number of dominant normals to find
in	<i>dominantNormalsSearchRadius</i>	- the search radius for neighbour searching
in	<i>halfSphere</i>	- each point of the halfSphere is a potencial dominant normal
out	<i>point</i>	cloud of dominant normals

< Neighbours indeces of the searched point

< Neighbours indeces of the inverse searched point

< Neighbours indeces of the dominant normal

< not used - needed because it is neccessery parameter for a used function

< find the most dominant normal

< search the neighbours of the inverse normal too - overflow

< x coordinate should be always positive

< set the inverse searchPoint

< append the neighbours of the inverse normal to the neighbours of the non-inverse normal

< Delete all points in the neighbourhood of the dominant normal

< True - extract remained indeces instead of removed ones

Definition at line 112 of file BirkysMethodReconstruction.cpp.

10.1.3.7 `float r3d::mtds::BirkysMethodReconstruction::getM_DistanceThreshold () const` `[inline]`

Definition at line 120 of file BirkysMethodReconstruction.h.

10.1.3.8 `float r3d::mtds::BirkysMethodReconstruction::getM_DominantNormalsSearchRadius () const` `[inline]`

Definition at line 130 of file BirkysMethodReconstruction.h.

10.1.3.9 `unsigned short r3d::mtds::BirkysMethodReconstruction::getM_GrNumberOfNeighbours () const` `[inline]`

Definition at line 70 of file BirkysMethodReconstruction.h.

10.1.3.10 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_HistogramDeletionArea () const [inline]

Definition at line 60 of file BirkysMethodReconstruction.h.

10.1.3.11 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_MinClusterSize () const [inline]

Definition at line 90 of file BirkysMethodReconstruction.h.

10.1.3.12 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_NeNumberOfNeighbours () const [inline]

Definition at line 40 of file BirkysMethodReconstruction.h.

10.1.3.13 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_NumberOfDominantRegions () const [inline]

Definition at line 50 of file BirkysMethodReconstruction.h.

10.1.3.14 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_NumberOfPlains () const [inline]

Definition at line 100 of file BirkysMethodReconstruction.h.

10.1.3.15 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_OmpNumberOfCores () const [inline]

Definition at line 30 of file BirkysMethodReconstruction.h.

10.1.3.16 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_PointColorThreshold () const [inline]

Definition at line 80 of file BirkysMethodReconstruction.h.

10.1.3.17 unsigned short r3d::mtds::BirkysMethodReconstruction::getM_PointsPerHalfSphere () const [inline]

Definition at line 110 of file BirkysMethodReconstruction.h.

10.1.3.18 pcl::PointCloud< pcl::PointXYZRGB >::Ptr r3d::mtds::BirkysMethodReconstruction::labelPointsAccordingToDominantNormals (pcl::PointCloud< pcl::PointXYZ >::Ptr *cloud*, pcl::PointCloud< pcl::Normal >::Ptr *normals*, pcl::PointCloud< pcl::PointXYZ >::Ptr *dominantNormals*)

function which label each point of the cloud according to the "nearest" dominant normal

Parameters

in	<i>cloud</i>	- the input point cloud
in	<i>normals</i>	- normals of each point in the input point cloud
in	<i>dominantNormals</i>	- dominant normals of the input point cloud
out	<i>colored</i>	point cloud

< Generating color map

< handle inverse normals

< the sqrt is not neccessery

< handle inverse normals

< the sqrt is not neccessery

Definition at line 179 of file BirkysMethodReconstruction.cpp.

10.1.3.19 `pcl::RegionGrowingRGB< pcl::PointXYZRGB > r3d::mtds::BirkysMethodReconstruction::regionGrowingRGB (pcl::PointCloud< pcl::PointXYZRGB >::Ptr cloudRGB, unsigned short numberOfNeighbours, unsigned short pointColorThreshold, unsigned short minClusterSize, std::vector< pcl::PointIndices > * clusters) [static]`

function which apply region growing algorithm to the colored input point cloud, region growing algorithm segments based on color

Parameters

in	<i>cloudRGB</i>	- colored point cloud
in	<i>numberOfNeighbours</i>	- the number of searched neighbours (input paramater for region growing)
in	<i>pointColorThreshold</i>	- color threshold (input parameter for region growing)
in	<i>minClusterSize</i>	- minimal cluster size (input parameter for region growing), smaller clusters are merged to neighbouring clusters
out	<i>clusters</i>	- segmented clusters of the point cloud
out	<i>instance</i>	of pcl::RegionGrowingRGB class containing the segmented (colored) point cloud

Definition at line 233 of file BirkysMethodReconstruction.cpp.

10.1.3.20 `void r3d::mtds::BirkysMethodReconstruction::setConfiguration (TiXmlNode * param) [override], [virtual]`

virtual function for setting up the member variables from the XML config file

Parameters

in	<i>param</i>	- XML parser class TiXmlNode (p. ??)
----	--------------	---

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 29 of file BirkysMethodReconstruction.cpp.

10.1.3.21 `void r3d::mtds::BirkysMethodReconstruction::setM_DistanceThreshold (float m_distance_threshold) [inline]`

Definition at line 125 of file BirkysMethodReconstruction.h.

10.1.3.22 `void r3d::mtds::BirkysMethodReconstruction::setM_DominantNormalsSearchRadius (float m_dominant_normals_search_radius) [inline]`

Definition at line 135 of file BirkysMethodReconstruction.h.

10.1.3.23 `void r3d::mtds::BirkysMethodReconstruction::setM_GrNumberOfNeighbours (unsigned short m_gr_number_of_neighbours) [inline]`

Definition at line 75 of file BirkysMethodReconstruction.h.

10.1.3.24 `void r3d::mtds::BirkysMethodReconstruction::setM_HistogramDeletionArea (unsigned short m_histogram_deletion_area) [inline]`

Definition at line 65 of file BirkysMethodReconstruction.h.

10.1.3.25 `void r3d::mtds::BirkysMethodReconstruction::setM_MinClusterSize (unsigned short m_min_cluster_size) [inline]`

Definition at line 95 of file BirkysMethodReconstruction.h.

10.1.3.26 `void r3d::mtds::BirkysMethodReconstruction::setM_NeNumberOfNeighbours (unsigned short m_ne_number_of_neighbours) [inline]`

Definition at line 45 of file BirkysMethodReconstruction.h.

10.1.3.27 `void r3d::mtds::BirkysMethodReconstruction::setM_NumberOfDominantRegions (unsigned short m_number_of_dominant_regions) [inline]`

Definition at line 55 of file BirkysMethodReconstruction.h.

10.1.3.28 `void r3d::mtds::BirkysMethodReconstruction::setM_NumberOfPlains (unsigned short m_number_of_plains) [inline]`

Definition at line 105 of file BirkysMethodReconstruction.h.

10.1.3.29 `void r3d::mtds::BirkysMethodReconstruction::setM_OmpNumberOfCores (unsigned short m_omp_number_of_cores) [inline]`

Definition at line 35 of file BirkysMethodReconstruction.h.

10.1.3.30 `void r3d::mtds::BirkysMethodReconstruction::setM_PointColorThreshold (unsigned short m_point_color_threshold) [inline]`

Definition at line 85 of file BirkysMethodReconstruction.h.

10.1.3.31 `void r3d::mtds::BirkysMethodReconstruction::setM_PointsPerHalfSphere (unsigned short m_points_per_half_sphere) [inline]`

Definition at line 115 of file BirkysMethodReconstruction.h.

10.1.3.32 `void r3d::mtds::BirkysMethodReconstruction::visualise (pcl::visualization::PCLVisualizer * viewer) [override], [virtual]`

virtual function for debug/test visualisation needed during development

Parameters

in	<i>viewer</i>	- PCL visualisation class <code>pcl::visualization::PCLVisualizer*</code>
----	---------------	---

Implements `r3d::mtds::Reconstruction` (p. ??).

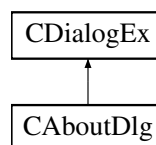
Definition at line 319 of file BirkysMethodReconstruction.cpp.

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

- 3DReconstruction/**BirkysMethodReconstruction.h**
- 3DReconstruction/**BirkysMethodReconstruction.cpp**

10.2 CAboutDlg Class Reference

Inheritance diagram for CAboutDlg:



Public Member Functions

- **CAboutDlg** ()

Protected Member Functions

- virtual void **DoDataExchange** (CDataExchange *pDX)

10.2.1 Detailed Description

Definition at line 32 of file guiDlg.cpp.

10.2.2 Constructor & Destructor Documentation

10.2.2.1 CAboutDlg::CAboutDlg ()

Definition at line 50 of file guiDlg.cpp.

10.2.3 Member Function Documentation

10.2.3.1 void CAboutDlg::DoDataExchange (CDataExchange * *pDX*) [protected],[virtual]

Definition at line 54 of file guiDlg.cpp.

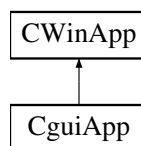
The documentation for this class was generated from the following file:

- gui/guiDlg.cpp

10.3 CguiApp Class Reference

```
#include <gui.h>
```

Inheritance diagram for CguiApp:



Public Member Functions

- **CguiApp** ()
- virtual BOOL **InitInstance** ()

10.3.1 Detailed Description

Definition at line 18 of file gui.h.

10.3.2 Constructor & Destructor Documentation

10.3.2.1 CguiApp::CguiApp ()

Definition at line 22 of file gui.cpp.

10.3.3 Member Function Documentation

10.3.3.1 BOOL CguiApp::InitInstance () [virtual]

Definition at line 39 of file gui.cpp.

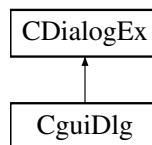
The documentation for this class was generated from the following files:

- gui/**gui.h**
- gui/**gui.cpp**

10.4 CguiDlg Class Reference

```
#include <guiDlg.h>
```

Inheritance diagram for CguiDlg:



Public Member Functions

- **CguiDlg** (CWnd *pParent=NULL)
- afx_msg void **OnBnClickedOk** ()
- afx_msg void **OnBnClickedButton4** ()
- afx_msg void **OnBnClickedCheck1** ()
- afx_msg void **OnCbnSelchangeCombo1** ()
- afx_msg void **OnBnClickedButton2** ()
- afx_msg void **OnBnClickedfile** ()
- afx_msg void **OnBnClickedN** ()
- afx_msg void **OnBnClickedbt** ()
- afx_msg void **OnBnClickedbutton** ()
- afx_msg void **OnBnClickedcalculationbutton** ()
- afx_msg void **OnBnClickedcalculation** ()
- afx_msg void **OnBnClickedvisualizationbutton** ()
- afx_msg void **OnBnClickedxmlfile** ()
- afx_msg void **OnBnClickedSvNorm** ()
- afx_msg void **OnBnClickedXm** ()
- afx_msg void **OnBnClickedXmlFile** ()
- afx_msg void **OnBnClickedOpenConf** ()
- afx_msg void **OnClickedRmButton** ()
- afx_msg void **OnBnClickedRM** ()
- afx_msg void **OnBnClickedButton** ()
- afx_msg void **OnBnClickedBt** ()

Public Attributes

- pcl::visualization::PCLVisualizer * **viewer**
- TiXmlNode * **methodConfig**

Protected Member Functions

- virtual void **DoDataExchange** (CDataExchange *pDX)
- virtual BOOL **OnInitDialog** ()
- afx_msg void **OnSysCommand** (UINT nID, LPARAM lParam)
- afx_msg void **OnPaint** ()
- afx_msg HCURSOR **OnQueryDragIcon** ()

Protected Attributes

- HICON **m_hIcon**

10.4.1 Detailed Description

Definition at line 13 of file guiDlg.h.

10.4.2 Constructor & Destructor Documentation

10.4.2.1 CguiDlg::CguiDlg (CWnd * *pParent* = NULL)

Definition at line 64 of file guiDlg.cpp.

10.4.3 Member Function Documentation

10.4.3.1 void CguiDlg::DoDataExchange (CDataExchange * *pDX*) [protected], [virtual]

Definition at line 84 of file guiDlg.cpp.

10.4.3.2 void CguiDlg::OnBnClickedbt ()

Definition at line 320 of file guiDlg.cpp.

10.4.3.3 void CguiDlg::OnBnClickedBt ()

Definition at line 663 of file guiDlg.cpp.

10.4.3.4 `afx_msg void CguiDlg::OnBnClickedbutton ()`

10.4.3.5 `afx_msg void CguiDlg::OnBnClickedButton ()`

10.4.3.6 `void CguiDlg::OnBnClickedButton2 ()`

Definition at line 258 of file guiDlg.cpp.

10.4.3.7 `void CguiDlg::OnBnClickedButton4 ()`

Definition at line 263 of file guiDlg.cpp.

10.4.3.8 `void CguiDlg::OnBnClickedcalculation ()`

<

Todo For other file types load every data

Definition at line 331 of file guiDlg.cpp.

10.4.3.9 `afx_msg void CguiDlg::OnBnClickedcalculationbutton ()`

10.4.3.10 `void CguiDlg::OnBnClickedCheck1 ()`

Definition at line 246 of file guiDlg.cpp.

10.4.3.11 `void CguiDlg::OnBnClickedfile ()`

Definition at line 266 of file guiDlg.cpp.

10.4.3.12 `void CguiDlg::OnBnClickedN ()`

Definition at line 314 of file guiDlg.cpp.

10.4.3.13 `void CguiDlg::OnBnClickedOk ()`

Definition at line 214 of file guiDlg.cpp.

10.4.3.14 `void CguiDlg::OnBnClickedOpenConf ()`

Definition at line 577 of file guiDlg.cpp.

10.4.3.15 void CguiDlg::OnBnClickedRM ()

Definition at line 619 of file guiDlg.cpp.

10.4.3.16 void CguiDlg::OnBnClickedSvNorm ()

Definition at line 540 of file guiDlg.cpp.

10.4.3.17 void CguiDlg::OnBnClickedvisualizationbutton ()

Definition at line 479 of file guiDlg.cpp.

10.4.3.18 void CguiDlg::OnBnClickedXm ()

Definition at line 570 of file guiDlg.cpp.

10.4.3.19 afx_msg void CguiDlg::OnBnClickedxmlfile ()

10.4.3.20 afx_msg void CguiDlg::OnBnClickedXmlFile ()

10.4.3.21 void CguiDlg::OnCbnSelchangeCombo1 ()

Definition at line 252 of file guiDlg.cpp.

10.4.3.22 afx_msg void CguiDlg::OnClickedRmButton ()

10.4.3.23 BOOL CguiDlg::OnInitDialog () [protected], [virtual]

Definition at line 111 of file guiDlg.cpp.

10.4.3.24 void CguiDlg::OnPaint () [protected]

Definition at line 181 of file guiDlg.cpp.

10.4.3.25 HCURSOR CguiDlg::OnQueryDragIcon () [protected]

Definition at line 208 of file guiDlg.cpp.

10.4.3.26 void CguiDlg::OnSysCommand (UINT *nID*, LPARAM *lParam*) [protected]

Definition at line 164 of file guiDlg.cpp.

10.4.4 Member Data Documentation

10.4.4.1 HICON CguiDlg::m_hlcon [protected]

Definition at line 30 of file guiDlg.h.

10.4.4.2 TiXmlNode* CguiDlg::methodConfig

Definition at line 42 of file guiDlg.h.

10.4.4.3 pcl::visualization::PCLVisualizer* CguiDlg::viewer

Definition at line 41 of file guiDlg.h.

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

- gui/guiDlg.h
- gui/guiDlg.cpp

10.5 r3d::space::ClosedSpace Class Reference

This class contains all stored data, reconstructed primitives and objects segmented in scene.

```
#include "ClosedSpace.h"
```

Public Member Functions

- **ClosedSpace** ()
- **ClosedSpace** (std::vector< std::string > source, std::string name)
Constructor that sets source and space/area name.
- virtual ~**ClosedSpace** ()
- void **setName** (std::string name)
- std::string **getName** () const
- void **setData** (pcl::PointCloud< pcl::PointXYZ > &data) const
- void **setData** (pcl::PointCloud< pcl::RGB > &data) const
- void **setData** (pcl::PointCloud< pcl::Normal > &data) const
- void **clear** ()
- pcl::PointCloud< pcl::PointXYZ > **getXYZData** () const
- pcl::PointCloud< pcl::RGB > **getRGBData** () const
- pcl::PointCloud< pcl::Normal > **getNormalData** () const
- void **setSource** (std::vector< std::string > source)
- std::vector< std::string > **getSource** () const
- void **addObject** (r3d::obj::Object &object)
Adds new segmented Object into object stack counts the reduced data and adds them to reduced PointCloud.
- std::vector< r3d::obj::Object * > **getObjects** ()
- int **addWall** (r3d::obj::Wall &wall)
Adds new segmented/reconstructed Wall object into wall stack.
- std::vector< r3d::obj::Wall > & **getWalls** ()
- void **visualise** (pcl::visualization::PCLVisualizer *viewer)
Loads reconstructed shape data into pcl/vtk viewer in a way they should be represented and viewed.
- void **visualiseOriginalData** (pcl::visualization::PCLVisualizer *viewer) const

10.5.1 Detailed Description

This class contains all stored data, reconstructed primitives and objects segmented in scene.

Author

Lukas Hudec

Definition at line 25 of file ClosedSpace.h.

10.5.2 Constructor & Destructor Documentation

10.5.2.1 `r3d::space::ClosedSpace::ClosedSpace ()`

Definition at line 5 of file ClosedSpace.cpp.

10.5.2.2 `r3d::space::ClosedSpace::ClosedSpace (std::vector< std::string > source, std::string name)`

Constructor that sets source and space/area name.

Parameters

in	<i>source</i>	files names that contains point cloud data
in	<i>name</i>	scene name, in case to hash this "data buffer"

Definition at line 14 of file ClosedSpace.cpp.

10.5.2.3 `r3d::space::ClosedSpace::~~ClosedSpace ()` [virtual]

Definition at line 22 of file ClosedSpace.cpp.

10.5.3 Member Function Documentation

10.5.3.1 `pcl::PointCloud< PointT > r3d::space::ClosedSpace::addObject (r3d::obj::Object & object)`

Adds new segmentated Object into object stack counts the reduced data and adds them to reduced PointCloud.

Parameters

in	<i>object</i>	vector of string file names
----	---------------	-----------------------------

Definition at line 72 of file ClosedSpace.cpp.

10.5.3.2 int r3d::space::ClosedSpace::addWall (r3d::obj::Wall & wall)

Adds new segmented/reconstructed Wall object into wall stack.

Parameters

in	<i>wall</i>	vector of string file names
----	-------------	-----------------------------

Returns

int number of segmented walls in stack

Definition at line 79 of file ClosedSpace.cpp.

10.5.3.3 void r3d::space::ClosedSpace::clear () [inline]

Definition at line 58 of file ClosedSpace.h.

10.5.3.4 std::string r3d::space::ClosedSpace::getName () const [inline]

Definition at line 42 of file ClosedSpace.h.

10.5.3.5 pcl::PointCloud<pcl::Normal> r3d::space::ClosedSpace::getNormalData () const [inline]

Definition at line 91 of file ClosedSpace.h.

10.5.3.6 std::vector<r3d::obj::Object*> r3d::space::ClosedSpace::getObjects () [inline]

Definition at line 111 of file ClosedSpace.h.

10.5.3.7 pcl::PointCloud<pcl::RGB> r3d::space::ClosedSpace::getRGBData () const [inline]

Definition at line 87 of file ClosedSpace.h.

10.5.3.8 std::vector<std::string> r3d::space::ClosedSpace::getSource () const [inline]

Definition at line 100 of file ClosedSpace.h.

10.5.3.9 std::vector<r3d::obj::Wall>& r3d::space::ClosedSpace::getWalls () [inline]

Definition at line 119 of file ClosedSpace.h.

10.5.3.10 `pcl::PointCloud<pcl::PointXYZ> r3d::space::ClosedSpace::getXYZData () const` `[inline]`

Definition at line 83 of file ClosedSpace.h.

10.5.3.11 `void r3d::space::ClosedSpace::setData (pcl::PointCloud< pcl::PointXYZ > & data) const` `[inline]`

Definition at line 46 of file ClosedSpace.h.

10.5.3.12 `void r3d::space::ClosedSpace::setData (pcl::PointCloud< pcl::RGB > & data) const` `[inline]`

Definition at line 50 of file ClosedSpace.h.

10.5.3.13 `void r3d::space::ClosedSpace::setData (pcl::PointCloud< pcl::Normal > & data) const` `[inline]`

Definition at line 54 of file ClosedSpace.h.

10.5.3.14 `r3d::space::ClosedSpace::setName (std::string name)` `[inline]`

Definition at line 38 of file ClosedSpace.h.

10.5.3.15 `void r3d::space::ClosedSpace::setSource (std::vector< std::string > source)` `[inline]`

Definition at line 96 of file ClosedSpace.h.

10.5.3.16 `void r3d::space::ClosedSpace::visualise (pcl::visualization::PCLVisualizer * viewer)`

Loads reconstructed shape data into pcl/vtk viewer in a way they should be represented and viewed.

Parameters

in	<i>viewer</i>	pcl/vtk viewer that will visualise data
----	---------------	---

Returns

bool true - if loading proceeded correctly, false otherwise

Definition at line 31 of file ClosedSpace.cpp.

10.5.3.17 `void r3d::space::ClosedSpace::visualiseOriginalData (pcl::visualization::PCLVisualizer * viewer) const`

Definition at line 45 of file ClosedSpace.cpp.

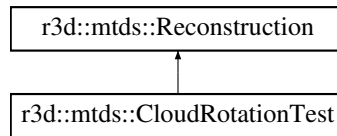
The documentation for this class was generated from the following files:

- 3DReconstruction/**ClosedSpace.h**
- 3DReconstruction/**ClosedSpace.cpp**

10.6 r3d::mtds::CloudRotationTest Class Reference

```
#include <CloudRotationTest.h>
```

Inheritance diagram for r3d::mtds::CloudRotationTest:



Public Member Functions

- **CloudRotationTest** ()
- **~CloudRotationTest** ()
- virtual bool **calculate** () override
virtual function to be implemented by child classes and reconstruction methods
- virtual void **setConfiguration** (TiXmlNode *param) override
- virtual void **visualise** (pcl::visualization::PCLVisualizer *viewer) override
virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation

Public Attributes

- pcl::PointCloud< pcl::PointXYZ >::Ptr **remainingCloud**

Additional Inherited Members

10.6.1 Detailed Description

Definition at line 11 of file CloudRotationTest.h.

10.6.2 Constructor & Destructor Documentation

10.6.2.1 r3d::mtds::CloudRotationTest::CloudRotationTest ()

Definition at line 21 of file CloudRotationTest.cpp.

10.6.2.2 r3d::mtds::CloudRotationTest::~~CloudRotationTest ()

Definition at line 28 of file CloudRotationTest.cpp.

10.6.3 Member Function Documentation

10.6.3.1 `bool r3d::mtds::CloudRotationTest::calculate () [override],[virtual]`

virtual function to be implemented by child classes and reconstruction methods

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 39 of file CloudRotationTest.cpp.

10.6.3.2 `void r3d::mtds::CloudRotationTest::setConfiguration (TiXmlNode * param) [override],[virtual]`

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 32 of file CloudRotationTest.cpp.

10.6.3.3 `void r3d::mtds::CloudRotationTest::visualise (pcl::visualization::PCLVisualizer * viewer) [override],[virtual]`

virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 319 of file CloudRotationTest.cpp.

10.6.4 Member Data Documentation

10.6.4.1 `pcl::PointCloud<pcl::PointXYZ>::Ptr r3d::mtds::CloudRotationTest::remainingCloud`

Definition at line 22 of file CloudRotationTest.h.

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

- 3DReconstruction/**CloudRotationTest.h**
- 3DReconstruction/**CloudRotationTest.cpp**

10.7 r3d::colorgen::ColorGenerator Class Reference

```
#include <ColorGenerator.h>
```

Public Member Functions

- unsigned char * **getNextColor** ()

Static Public Member Functions

- static **ColorGenerator** & **getInstance** ()

10.7.1 Detailed Description

Definition at line 8 of file ColorGenerator.h.

10.7.2 Member Function Documentation

10.7.2.1 static **ColorGenerator**& r3d::colorgen::ColorGenerator::getInstance () [inline],[static]

Definition at line 10 of file ColorGenerator.h.

10.7.2.2 unsigned char * r3d::colorgen::ColorGenerator::getNextColor ()

Definition at line 7 of file ColorGenerator.cpp.

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

- 3DReconstruction/**ColorGenerator.h**
- 3DReconstruction/**ColorGenerator.cpp**

10.8 r3d::CommonUtil Class Reference

```
#include <CommonUtil.h>
```

Public Member Functions

- **CommonUtil** ()
- **~CommonUtil** ()

Static Public Member Functions

- static std::vector< **r3d::HASH** > **computeDistancesHistogram** (std::vector< double > distances, double cut=0)
computes histogram out of all distances in vector use "cut" to group points in distance "cuts" /default is 0, but consider using small values as 0.0001 or 0.001
- static bool **comparePointIndicesDesc** (pcl::PointIndices &a, pcl::PointIndices &b)
- static bool **comparePointIndicesAsc** (pcl::PointIndices &a, pcl::PointIndices &b)
- static bool **verifyPlaneCornerPoints** (const pcl::PointCloud< pcl::PointXYZ >::Ptr &planeCloud, pcl::PointCloud< pcl::PointXYZ > &cornerPoints, pcl::PointCloud< pcl::PointXYZ >::Ptr &planeConcaveHull, double tolerance=2.0, double alpha=0.05)
- static double **getAngle3D** (pcl::PointXYZ A, pcl::PointXYZ B, pcl::PointXYZ C)
- static void **orderPointsInConcaveHull** (pcl::PointCloud< pcl::PointXYZ >::Ptr inputCloud, pcl::PointCloud< pcl::PointXYZ > &outputCloud, double alphaIn=10.0)

10.8.1 Detailed Description

Definition at line 35 of file CommonUtil.h.

10.8.2 Constructor & Destructor Documentation

10.8.2.1 `r3d::CommonUtil::CommonUtil ()`

Definition at line 8 of file CommonUtil.cpp.

10.8.2.2 `r3d::CommonUtil::~~CommonUtil ()`

Definition at line 13 of file CommonUtil.cpp.

10.8.3 Member Function Documentation

10.8.3.1 `bool r3d::CommonUtil::comparePointIndicesAsc (pcl::PointIndices & a, pcl::PointIndices & b) [static]`

Definition at line 51 of file CommonUtil.cpp.

10.8.3.2 `bool r3d::CommonUtil::comparePointIndicesDesc (pcl::PointIndices & a, pcl::PointIndices & b) [static]`

Definition at line 46 of file CommonUtil.cpp.

10.8.3.3 `std::vector< r3d::HASH > r3d::CommonUtil::computeDistancesHistogram (std::vector< double > distances, double cut = 0) [static]`

computes histogram out of all distances in vector use "cut" to group points in distance "cuts" /default is 0, but consider using small values as 0.0001 or 0.001

Definition at line 21 of file CommonUtil.cpp.

10.8.3.4 `double r3d::CommonUtil::getAngle3D (pcl::PointXYZ A, pcl::PointXYZ B, pcl::PointXYZ C) [static]`

Definition at line 128 of file CommonUtil.cpp.

10.8.3.5 `void r3d::CommonUtil::orderPointsInConcaveHull (pcl::PointCloud< pcl::PointXYZ >::Ptr inputCloud, pcl::PointCloud< pcl::PointXYZ > & outputCloud, double alphaIn = 10.0) [static]`

Definition at line 136 of file CommonUtil.cpp.

```
10.8.3.6 bool r3d::CommonUtil::verifyPlaneCornerPoints ( const pcl::PointCloud< pcl::PointXYZ >::Ptr & planeCloud,
pcl::PointCloud< pcl::PointXYZ > & cornerPoints, pcl::PointCloud< pcl::PointXYZ >::Ptr & planeConcaveHull,
double tolerance = 2.0, double alpha = 0.05 ) [static]
```

Definition at line 56 of file CommonUtil.cpp.

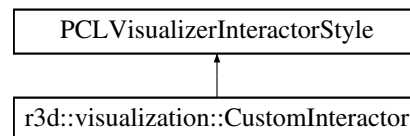
The documentation for this class was generated from the following files:

- 3DReconstruction/**CommonUtil.h**
- 3DReconstruction/**CommonUtil.cpp**

10.9 r3d::visualization::CustomInteractor Class Reference

```
#include <CustomInteractor.h>
```

Inheritance diagram for r3d::visualization::CustomInteractor:



Public Member Functions

- **CustomInteractor** ()
- void **CustomInteractor::OnMouseMove** ()
- void **CustomInteractor::OnLeftButtonDown** ()
- void **CustomInteractor::OnLeftButtonUp** ()
- void **CustomInteractor::OnChar** ()
- void **CustomInteractor::OnKeyPress** ()
- void **CustomInteractor::OnKeyRelease** ()
- void **CustomInteractor::Rotate** ()
- void **CustomInteractor::KeyboardPanLeft** ()
- void **CustomInteractor::KeyboardPanRight** ()
- void **CustomInteractor::MoveForwards** ()
- void **CustomInteractor::MoveBackwards** ()
- void **CustomInteractor::Init** ()

10.9.1 Detailed Description

Definition at line 8 of file CustomInteractor.h.

10.9.2 Constructor & Destructor Documentation

10.9.2.1 r3d::visualization::CustomInteractor::CustomInteractor ()

Definition at line 23 of file CustomInteractor.cpp.

10.9.3 Member Function Documentation

10.9.3.1 void r3d::visualization::CustomInteractor::CustomInteractor::Init ()

10.9.3.2 void r3d::visualization::CustomInteractor::CustomInteractor::KeyboardPanLeft ()

10.9.3.3 void r3d::visualization::CustomInteractor::CustomInteractor::KeyboardPanRight ()

10.9.3.4 void r3d::visualization::CustomInteractor::CustomInteractor::MoveBackwards ()

10.9.3.5 void r3d::visualization::CustomInteractor::CustomInteractor::MoveForwards ()

10.9.3.6 void r3d::visualization::CustomInteractor::CustomInteractor::OnChar ()

10.9.3.7 void r3d::visualization::CustomInteractor::CustomInteractor::OnKeyPress ()

10.9.3.8 void r3d::visualization::CustomInteractor::CustomInteractor::OnKeyRelease ()

10.9.3.9 void r3d::visualization::CustomInteractor::CustomInteractor::OnLeftButtonDown ()

10.9.3.10 void r3d::visualization::CustomInteractor::CustomInteractor::OnLeftButtonUp ()

10.9.3.11 void r3d::visualization::CustomInteractor::CustomInteractor::OnMouseMove ()

IO event handling methods

10.9.3.12 void r3d::visualization::CustomInteractor::CustomInteractor::Rotate ()

Camera movement methods - reused implementation from VTK's vtkInteractorStyleTerrain

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

- Visualization/**CustomInteractor.h**
- Visualization/**CustomInteractor.cpp**

10.10 r3d::data::DataStatistics Class Reference

Statistics about the segmented pointclouds.

```
#include "DataStatistics.h"
```


Public Member Functions

- **DataStatistics** ()
- **~DataStatistics** ()
- bool **isPlane** (pcl::PointCloud< pcl::PointXYZ >::Ptr segment, Eigen::VectorXf coefficients, long double standardDev)

Find whether the point cloud segment is representing a plane object according to RANSAC coefficients and standard deviation of the point cloud segment.
- void **configure** (TiXmlElement &config)

Sets the configuration from XML source.
- long double **computeRoughness** ()

Computes roughness of given cloud represented by standard deviation.
- void **setCloud** (const pcl::PointCloud< pcl::PointXYZ >::Ptr pointCloud)
- pcl::PointCloud< pcl::PointXYZ > **getCloud** () const
- void **setDistanceThreshold** (double distanceThreshold)
- void **setSigma** (double sigma)
- void **setNrThreads** (int nrThreads)
- void **setLineCheckoutStep** (int lineCheckoutStep)
- void **setLinearRepetitions** (int linearRepetitions)
- void **setPlaneIdentificationThreshold** (double planeIdentificationThreshold)
- void **setAngularCoefficientMax** (double angularCoefficientMax)
- double **getPlaneIdentificationThreshold** () const
- long double **getStandardDeviation** () const
- long double **getMaxPointPlaneDistance** () const
- bool **isComputed** () const
- void **setPlaneCoefs** (Eigen::VectorXf coefs)
- Eigen::VectorXf **getPlaneCoefs** () const

returns computed plane coefficients used for standard deviation computing

Static Public Member Functions

- static double **segmentPointDispersion** (pcl::PointCloud< pcl::PointXYZ >::Ptr segment, int iterations)

Computes average mutual distance between points in given cloud represented.

10.10.1 Detailed Description

Statistics about the segmented pointclouds.

Author

Lukas Hudec

Definition at line 10 of file DataStatistics.h.

10.10.2 Constructor & Destructor Documentation

10.10.2.1 r3d::data::DataStatistics::DataStatistics ()

Definition at line 13 of file DataStatistics.cpp.

10.10.2.2 `r3d::data::DataStatistics::~~DataStatistics ()`

Definition at line 17 of file DataStatistics.cpp.

10.10.3 Member Function Documentation

10.10.3.1 `long double r3d::data::DataStatistics::computeRoughness ()`

Computes roughnes of given cloud represented by standard deviation.

< compute plane coefficients

find converging value to set borders for the plane points gaussianBorder = index of the last added points block

< mean == 0, and square makes it positive, so no division by 2 is needed

Definition at line 33 of file DataStatistics.cpp.

10.10.3.2 `void r3d::data::DataStatistics::configure (TiXmlElement & config)`

Sets the configuration from XML source.

Warning

Does not control existence of the config value - may fall

Definition at line 21 of file DataStatistics.cpp.

10.10.3.3 `pcl::PointCloud<pcl::PointXYZ> r3d::data::DataStatistics::getCloud () const [inline]`

Definition at line 36 of file DataStatistics.h.

10.10.3.4 `long double r3d::data::DataStatistics::getMaxPointPlaneDistance () const [inline]`

Definition at line 87 of file DataStatistics.h.

10.10.3.5 `Eigen::VectorXf r3d::data::DataStatistics::getPlaneCoefs () const [inline]`

returns computed plane coefficients used for standard deviation computing

Definition at line 103 of file DataStatistics.h.

10.10.3.6 `double r3d::data::DataStatistics::getPlaneIdentificationThreshold () const [inline]`

Definition at line 76 of file DataStatistics.h.

10.10.3.7 `long double r3d::data::DataStatistics::getStandardDeviation () const` `[inline]`

Definition at line 82 of file DataStatistics.h.

10.10.3.8 `bool r3d::data::DataStatistics::isComputed () const` `[inline]`

Definition at line 92 of file DataStatistics.h.

10.10.3.9 `bool r3d::data::DataStatistics::isPlane (pcl::PointCloud< pcl::PointXYZ >::Ptr segment, Eigen::VectorXf coefficients, long double stDev)`

Find whether the point cloud segment is representing a plane object according to RANSAC coefficients and standard deviation of the point cloud segment.

< find inliers statistically according to the distance from plane

< extract inliers

< extract outliers

< get ratio of inliers and outliers

Definition at line 209 of file DataStatistics.cpp.

10.10.3.10 `double r3d::data::DataStatistics::segmentPointDispersion (pcl::PointCloud< pcl::PointXYZ >::Ptr segment, int iterations)` `[static]`

Computes average mutual distance between points in given cloud represented.

Definition at line 132 of file DataStatistics.cpp.

10.10.3.11 `void r3d::data::DataStatistics::setAngularCoefficientMax (double angularCoefficientMax)` `[inline]`

Definition at line 71 of file DataStatistics.h.

10.10.3.12 `void r3d::data::DataStatistics::setCloud (const pcl::PointCloud< pcl::PointXYZ >::Ptr pointCloud)` `[inline]`

Definition at line 30 of file DataStatistics.h.

10.10.3.13 `void r3d::data::DataStatistics::setDistanceThreshold (double distanceThreshold)` `[inline]`

Definition at line 41 of file DataStatistics.h.

10.10.3.14 `void r3d::data::DataStatistics::setLinearRepetitions (int linearRepetitions)` `[inline]`

Definition at line 61 of file DataStatistics.h.

10.10.3.15 `void r3d::data::DataStatistics::setLineCheckoutStep (int lineCheckoutStep)` `[inline]`

Definition at line 56 of file DataStatistics.h.

10.10.3.16 `void r3d::data::DataStatistics::setNrThreads (int nrThreads)` `[inline]`

Definition at line 51 of file DataStatistics.h.

10.10.3.17 `void r3d::data::DataStatistics::setPlaneCoefs (Eigen::VectorXf coefs)` `[inline]`

Definition at line 97 of file DataStatistics.h.

10.10.3.18 `void r3d::data::DataStatistics::setPlaneIdentificationThreshold (double planeIdentificationThreshold)`
`[inline]`

Definition at line 66 of file DataStatistics.h.

10.10.3.19 `void r3d::data::DataStatistics::setSigma (double sigma)` `[inline]`

Definition at line 46 of file DataStatistics.h.

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

- 3DReconstruction/**DataStatistics.h**
- 3DReconstruction/**DataStatistics.cpp**

10.11 r3d::exporter::Exporter Class Reference

```
#include <Exporter.h>
```

Public Member Functions

- **Exporter** ()
- virtual **~Exporter** ()
- bool **addWalls** (std::vector< **r3d::obj::Wall** > walls)
- bool **addMesh** (pcl::PolygonMesh mesh)
- bool **writeDXF** (const char *filename)
- bool **writeOBJ** (const char *filename)

10.11.1 Detailed Description

Definition at line 20 of file Exporter.h.

10.11.2 Constructor & Destructor Documentation

10.11.2.1 r3d::exporter::Exporter::Exporter ()

Definition at line 15 of file Exporter.cpp.

10.11.2.2 r3d::exporter::Exporter::~~Exporter () [virtual]

Definition at line 20 of file Exporter.cpp.

10.11.3 Member Function Documentation

10.11.3.1 bool r3d::exporter::Exporter::addMesh (pcl::PolygonMesh *mesh*)

Convert each point from mesh to SG_POINT

Take every triangle and convert it to SG_INDEX_TRIANGLE

Definition at line 125 of file Exporter.cpp.

10.11.3.2 bool r3d::exporter::Exporter::addWalls (std::vector< r3d::obj::Wall > *walls*)

Definition at line 40 of file Exporter.cpp.

10.11.3.3 bool r3d::exporter::Exporter::writeDXF (const char * *filename*)

Definition at line 169 of file Exporter.cpp.

10.11.3.4 bool r3d::exporter::Exporter::writeOBJ (const char * *filename*)

Definition at line 174 of file Exporter.cpp.

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

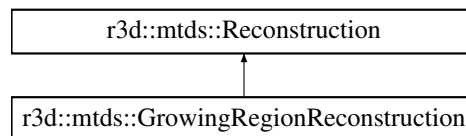
- Exporter/**Exporter.h**
- Exporter/**Exporter.cpp**

10.12 r3d::mtds::GrowingRegionReconstruction Class Reference

Composite class aggregating reconstruction methods and simple outliers segmentation Uses Birkis segmenation by growing region and color labeling Reconstructs planar objects using RANSAC analytical representation Segments outliers of these planar objects.

```
#include "GrowingRegionReconstruction.h"
```

Inheritance diagram for r3d::mtds::GrowingRegionReconstruction:



Public Member Functions

- **GrowingRegionReconstruction** ()
- **~GrowingRegionReconstruction** ()
- virtual bool **calculate** () override
virtual function to be implemented by child classes and reconstruction methods
- virtual void **setConfiguration** (TiXmlNode *param) override
Sets the configuration from XML source.
- virtual void **visualise** (pcl::visualization::PCLVisualizer *viewer) override
virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation

Public Attributes

- **r3d::data::DataStatistics m_dataStatistics**

Additional Inherited Members

10.12.1 Detailed Description

Composite class aggregating reconstruction methods and simple outliers segmentation Uses Birkis segmenation by growing region and color labeling Reconstructs planar objects using RANSAC analytical representation Segments outliers of these planar objects.

Author

Lukas Hudec

Definition at line 21 of file GrowingRegionReconstruction.h.

10.12.2 Constructor & Destructor Documentation

10.12.2.1 r3d::mtds::GrowingRegionReconstruction::GrowingRegionReconstruction ()

Definition at line 17 of file GrowingRegionReconstruction.cpp.

10.12.2.2 r3d::mtds::GrowingRegionReconstruction::~~GrowingRegionReconstruction ()

Definition at line 50 of file GrowingRegionReconstruction.cpp.

10.12.3 Member Function Documentation

10.12.3.1 bool r3d::mtds::GrowingRegionReconstruction::calculate () [override], [virtual]

virtual function to be implemented by child classes and reconstruction methods

Compute points of normals based on environment (kdtree)

< Calculate normals if they are not available

< Create "histogram" of normals from the calculated normals

GR with pcl + param (min/max size of cluster, prahovacie hodnoty..)

< Create a half sphere of potencial dominant directions

< searching for dominant normals

< Labeling the points of the cloud according to the found dominant normals

< Region growing

segmentated pointcloud with segments differed with colors RGB values

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 86 of file GrowingRegionReconstruction.cpp.

10.12.3.2 void r3d::mtds::GrowingRegionReconstruction::setConfiguration (TiXmlNode * param) [override], [virtual]

Sets the configuration from XML source.

Warning

Does not control existence of the config value - may fall

< returns parameter as char*

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 54 of file GrowingRegionReconstruction.cpp.

10.12.3.3 void r3d::mtds::GrowingRegionReconstruction::visualise (pcl::visualization::PCLVisualizer * viewer) [override], [virtual]

virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 292 of file GrowingRegionReconstruction.cpp.

10.12.4 Member Data Documentation

10.12.4.1 `r3d::data::DataStatistics` `r3d::mtds::GrowingRegionReconstruction::m_dataStatistics`

Definition at line 36 of file `GrowingRegionReconstruction.h`.

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

- `3DReconstruction/GrowingRegionReconstruction.h`
- `3DReconstruction/GrowingRegionReconstruction.cpp`

10.13 `r3d::HASH` Struct Reference

```
#include <CommonUtil.h>
```

Public Member Functions

- `HASH()`
- `HASH(double distance, int points)`
- `bool operator==(const HASH &b) const`

Public Attributes

- `double distance`
- `int points`

10.13.1 Detailed Description

Definition at line 10 of file `CommonUtil.h`.

10.13.2 Constructor & Destructor Documentation

10.13.2.1 `r3d::HASH::HASH()` `[inline]`

Definition at line 12 of file `CommonUtil.h`.

10.13.2.2 `r3d::HASH::HASH(double distance, int points)` `[inline]`

Definition at line 18 of file `CommonUtil.h`.

10.13.3 Member Function Documentation

10.13.3.1 `bool r3d::HASH::operator==(const HASH & b) const` `[inline]`

Definition at line 24 of file CommonUtil.h.

10.13.4 Member Data Documentation

10.13.4.1 `double r3d::HASH::distance`

Definition at line 29 of file CommonUtil.h.

10.13.4.2 `int r3d::HASH::points`

Definition at line 30 of file CommonUtil.h.

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

- 3DReconstruction/**CommonUtil.h**

10.14 r3d::parser::InputDataParser Class Reference

Data parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

```
#include <InputDataParser.h>
```

Public Member Functions

- `InputDataParser ()`
- `~InputDataParser ()`

Static Public Member Functions

- `static pcl::PointCloud< pcl::PointXYZ > * parseXYZCsv (const std::string source)`

10.14.1 Detailed Description

Data parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

Author

Lukas Hudec

Definition at line 10 of file InputDataParser.h.

10.14.2 Constructor & Destructor Documentation

10.14.2.1 `r3d::parser::InputDataParser::InputDataParser ()`

Definition at line 5 of file `InputDataParser.cpp`.

10.14.2.2 `r3d::parser::InputDataParser::~~InputDataParser ()`

Definition at line 10 of file `InputDataParser.cpp`.

10.14.3 Member Function Documentation

10.14.3.1 `pcl::PointCloud< pcl::PointXYZ > * r3d::parser::InputDataParser::parseXYZCsv (const std::string source)` [static]

/brief loads and parses CSV file containing XYZ data in columns 1=X, 2=Y, 3=Z (zero based)

Parameters

in	<i>source</i>	file name to be loaded and parsed
----	---------------	-----------------------------------

< Always get the line -> so while advancesr

< void lines could be accientaly present, in case continue

< Tokenize the line

< Assign tokens to a string vector

< Add the point to the cloud

Definition at line 14 of file `InputDataParser.cpp`.

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

- `DataHandling/InputDataParser.h`
- `DataHandling/InputDataParser.cpp`

10.15 `r3d::prims::Line` Class Reference

```
#include <Line.h>
```

Public Member Functions

- **Line** ()
- **Line** (Eigen::VectorXf &coeff, pcl::PointCloud< pcl::PointXYZRGB > linePoints)
- **~Line** ()
- void **setCoefficients** (Eigen::VectorXf &coeff)
- void **setCoefficients** (float pointX, float pointY, float pointZ, float directionX, float directionY, float directionZ)
- void **setCloud** (pcl::PointCloud< pcl::PointXYZRGB > linePoints)
- Eigen::VectorXf **getCoefficients** ()
- pcl::PointCloud< pcl::PointXYZRGB > **getPoints** ()

10.15.1 Detailed Description

Definition at line 7 of file Line.h.

10.15.2 Constructor & Destructor Documentation

10.15.2.1 r3d::prims::Line::Line ()

Definition at line 5 of file Line.cpp.

10.15.2.2 r3d::prims::Line::Line (Eigen::VectorXf &coeff, pcl::PointCloud< pcl::PointXYZRGB > linePoints)

Definition at line 11 of file Line.cpp.

10.15.2.3 r3d::prims::Line::~~Line ()

Definition at line 18 of file Line.cpp.

10.15.3 Member Function Documentation

10.15.3.1 Eigen::VectorXf r3d::prims::Line::getCoefficients ()

Definition at line 45 of file Line.cpp.

10.15.3.2 pcl::PointCloud< pcl::PointXYZRGB > r3d::prims::Line::getPoints ()

Definition at line 51 of file Line.cpp.

10.15.3.3 void r3d::prims::Line::setCloud (pcl::PointCloud< pcl::PointXYZRGB > linePoints)

Definition at line 40 of file Line.cpp.

10.15.3.4 void r3d::prims::Line::setCoefficients (Eigen::VectorXf & coeff)

Definition at line 22 of file Line.cpp.

10.15.3.5 void r3d::prims::Line::setCoefficients (float pointX, float pointY, float pointZ, float directionX, float directionY, float directionZ)

Definition at line 30 of file Line.cpp.

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

- 3DReconstruction/**Line.h**
- 3DReconstruction/**Line.cpp**

10.16 r3d::remover::NoiseRemover Class Reference

Removes noise from point cloudData parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

```
#include <NoiseRemover.h>
```

Public Member Functions

- **NoiseRemover** ()
- **~NoiseRemover** ()
- void **setConfiguration** (TiXmlNode *param)
- const char * **getParameter** (TiXmlNode *param, const char *name)
- pcl::PointCloud< pcl::PointXYZ >::Ptr **removeNoise** (pcl::PointCloud< pcl::PointXYZRGBNormal > *cloud)

10.16.1 Detailed Description

Removes noise from point cloudData parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

Author

Martin Jurik

Definition at line 15 of file NoiseRemover.h.

10.16.2 Constructor & Destructor Documentation

10.16.2.1 r3d::remover::NoiseRemover::NoiseRemover ()

Definition at line 4 of file NoiseRemover.cpp.

10.16.2.2 r3d::remover::NoiseRemover::~~NoiseRemover ()

Definition at line 10 of file NoiseRemover.cpp.

10.16.3 Member Function Documentation

10.16.3.1 const char * r3d::remover::NoiseRemover::getParameter (TiXmlNode * param, const char * name)

Definition at line 37 of file NoiseRemover.cpp.

10.16.3.2 pcl::PointCloud< pcl::PointXYZ >::Ptr r3d::remover::NoiseRemover::removeNoise (pcl::PointCloud< pcl::PointXYZRGBNormal > * cloud)

Definition at line 14 of file NoiseRemover.cpp.

10.16.3.3 void r3d::remover::NoiseRemover::setConfiguration (TiXmlNode * param)

Definition at line 29 of file NoiseRemover.cpp.

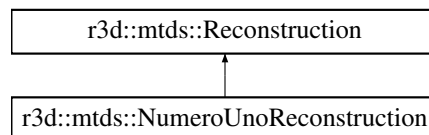
The documentation for this class was generated from the following files:

- DataHandling/**NoiseRemover.h**
- DataHandling/**NoiseRemover.cpp**

10.17 r3d::mtds::NumeroUnoReconstruction Class Reference

```
#include <NumeroUnoReconstruction.h>
```

Inheritance diagram for r3d::mtds::NumeroUnoReconstruction:



Public Member Functions

- **NumeroUnoReconstruction** ()
- **~NumeroUnoReconstruction** ()
- virtual bool **calculate** () override
virtual function to be implemented by child classes and reconstruction methods
- virtual void **setConfiguration** (TiXmlNode *param) override
- virtual void **visualise** (pcl::visualization::PCLVisualizer *viewer) override
virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation

Protected Member Functions

- int **calculate_windows** (float dist, float m_point)
- float **calculate_mean_point** (pcl::PointCloud< pcl::PointXYZ >::Ptr cld)
- float **calculate_total_distance_z** (pcl::PointCloud< pcl::PointXYZ >::Ptr cld, float *min, float *max)
- int **meanshift** (float *cntr, float b_width, std::vector< float > points, int *dir)
- void **calculate_normals** (pcl::PointCloud< pcl::Normal >::Ptr nrmls, pcl::PointCloud< pcl::PointXYZ >::Ptr cld, int k)
- void **calculate_projected_points** (std::vector< float > *proj_points, pcl::PointCloud< pcl::Normal >::Ptr nrmls, pcl::PointCloud< pcl::PointXYZ >::Ptr cld, pcl::PointCloud< pcl::PointXYZ >::Ptr z_cld, float err)
- float **get_centroid** (float *pos, float b_width, std::vector< float > proj_points)

Additional Inherited Members

10.17.1 Detailed Description

Definition at line 10 of file NumeroUnoReconstruction.h.

10.17.2 Constructor & Destructor Documentation

10.17.2.1 r3d::mtds::NumeroUnoReconstruction::NumeroUnoReconstruction ()

Definition at line 5 of file NumeroUnoReconstruction.cpp.

10.17.2.2 r3d::mtds::NumeroUnoReconstruction::~~NumeroUnoReconstruction ()

Definition at line 12 of file NumeroUnoReconstruction.cpp.

10.17.3 Member Function Documentation

10.17.3.1 bool r3d::mtds::NumeroUnoReconstruction::calculate () [override],[virtual]

virtual function to be implemented by child classes and reconstruction methods

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 30 of file NumeroUnoReconstruction.cpp.

10.17.3.2 float r3d::mtds::NumeroUnoReconstruction::calculate_mean_point (pcl::PointCloud< pcl::PointXYZ >::Ptr cld) [protected]

Definition at line 105 of file NumeroUnoReconstruction.cpp.

10.17.3.3 `void r3d::mtds::NumeroUnoReconstruction::calculate_normals (pcl::PointCloud< pcl::Normal >::Ptr nrmls,
pcl::PointCloud< pcl::PointXYZ >::Ptr cld, int k)` [protected]

Definition at line 226 of file NumeroUnoReconstruction.cpp.

10.17.3.4 `void r3d::mtds::NumeroUnoReconstruction::calculate_projected_points (std::vector< float > * proj_points,
pcl::PointCloud< pcl::Normal >::Ptr nrmls, pcl::PointCloud< pcl::PointXYZ >::Ptr cld, pcl::PointCloud<
pcl::PointXYZ >::Ptr z_cld, float err)` [protected]

Definition at line 237 of file NumeroUnoReconstruction.cpp.

10.17.3.5 `float r3d::mtds::NumeroUnoReconstruction::calculate_total_distance_z (pcl::PointCloud< pcl::PointXYZ >::Ptr cld,
float * min, float * max)` [protected]

Definition at line 125 of file NumeroUnoReconstruction.cpp.

10.17.3.6 `int r3d::mtds::NumeroUnoReconstruction::calculate_windows (float dist, float m_point)` [protected]

Definition at line 92 of file NumeroUnoReconstruction.cpp.

10.17.3.7 `float r3d::mtds::NumeroUnoReconstruction::get_centroid (float * pos, float b_width, std::vector< float >
proj_points)` [protected]

Definition at line 252 of file NumeroUnoReconstruction.cpp.

10.17.3.8 `int r3d::mtds::NumeroUnoReconstruction::meanshift (float * cntr, float b_width, std::vector< float > points, int *
dir)` [protected]

Definition at line 141 of file NumeroUnoReconstruction.cpp.

10.17.3.9 `void r3d::mtds::NumeroUnoReconstruction::setConfiguration (TiXmlNode * param)` [override],
[virtual]

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 21 of file NumeroUnoReconstruction.cpp.

10.17.3.10 `void r3d::mtds::NumeroUnoReconstruction::visualise (pcl::visualization::PCLVisualizer * viewer)`
[override], [virtual]

virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 16 of file NumeroUnoReconstruction.cpp.

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

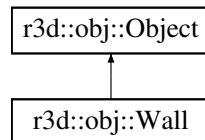
- 3DReconstruction/**NumeroUnoReconstruction.h**
- 3DReconstruction/**NumeroUnoReconstruction.cpp**

10.18 r3d::obj::Object Class Reference

Class of object.

```
#include <Object.h>
```

Inheritance diagram for r3d::obj::Object:



Public Member Functions

- **Object ()**
A constructor.
- **Object** (pcl::PointCloud< pcl::PointXYZ > objectData, std::string name)
- virtual **~Object ()**
- void **setName** (std::string name)
- std::string **getName** () const
- void **setObjectData** (pcl::PointCloud< pcl::PointXYZ > objectData)
- void **setObjectData** (pcl::PointCloud< pcl::PointXYZRGB > objectData)
- pcl::PointCloud< pcl::PointXYZ > & **getObjectData** ()
- void **setPointDispersion** (double dispersion)
- double **getPointDispersion** () const
- void **setStandardDeviation** (long double standardDeviation)
- long double **getStandardDeviation** () const
- unsigned char **getR** ()
- unsigned char **getG** ()
- unsigned char **getB** ()

Protected Attributes

- std::string **m_name**
- pcl::PointCloud< pcl::PointXYZ > **m_originalData**
- long double **m_standardDeviation** = 0.0
Standard Deviation of data and sensor precision.
- double **m_pointDispersion** = 0.0
Mean mutual distance between points in cloud.

10.18.1 Detailed Description

Class of object.

This class is used to demonstrate a...

Author

Lukas Hudec
Martin Jurik

Version

1.0

Date

2015

Precondition

First initialize the...

Bug Not all memory is freed when deleting an object of this class.

Warning

Improper use can crash your application

Copyright

License.

Definition at line 21 of file Object.h.

10.18.2 Constructor & Destructor Documentation

10.18.2.1 r3d::obj::Object::Object ()

A constructor.

A more elaborate description of the constructor.

Definition at line 6 of file Object.cpp.

10.18.2.2 r3d::obj::Object::Object (pcl::PointCloud< pcl::PointXYZ > *objectData*, std::string *name*)

Definition at line 14 of file Object.cpp.

10.18.2.3 `r3d::obj::Object::~~Object () [virtual]`

Definition at line 25 of file Object.cpp.

10.18.3 Member Function Documentation

10.18.3.1 `unsigned char r3d::obj::Object::getB () [inline]`

Definition at line 76 of file Object.h.

10.18.3.2 `unsigned char r3d::obj::Object::getG () [inline]`

Definition at line 72 of file Object.h.

10.18.3.3 `std::string r3d::obj::Object::getName () const [inline]`

Definition at line 36 of file Object.h.

10.18.3.4 `pcl::PointCloud<pcl::PointXYZ>& r3d::obj::Object::getObjectData () [inline]`

Definition at line 43 of file Object.h.

10.18.3.5 `double r3d::obj::Object::getPointDispersion () const [inline]`

Definition at line 53 of file Object.h.

10.18.3.6 `unsigned char r3d::obj::Object::getR () [inline]`

Definition at line 68 of file Object.h.

10.18.3.7 `long double r3d::obj::Object::getStandardDeviation () const [inline]`

Definition at line 63 of file Object.h.

10.18.3.8 `void r3d::obj::Object::setName (std::string name) [inline]`

Definition at line 32 of file Object.h.

10.18.3.9 `void r3d::obj::Object::setObjectData (pcl::PointCloud< pcl::PointXYZ > objectData)`

Definition at line 29 of file Object.cpp.

10.18.3.10 void r3d::obj::Object::setObjectData (pcl::PointCloud< pcl::PointXYZRGB > *objectData*)

Definition at line 34 of file Object.cpp.

10.18.3.11 void r3d::obj::Object::setPointDispersion (double *dispersion*) [inline]

Definition at line 48 of file Object.h.

10.18.3.12 void r3d::obj::Object::setStandardDeviation (long double *standardDeviation*) [inline]

Definition at line 58 of file Object.h.

10.18.4 Member Data Documentation

10.18.4.1 std::string r3d::obj::Object::m_name [protected]

Definition at line 85 of file Object.h.

10.18.4.2 pcl::PointCloud< pcl::PointXYZ > r3d::obj::Object::m_originalData [protected]

Definition at line 86 of file Object.h.

10.18.4.3 double r3d::obj::Object::m_pointDispersion = 0.0 [protected]

Mean mutual distance between points in cloud.

Definition at line 94 of file Object.h.

10.18.4.4 long double r3d::obj::Object::m_standardDeviation = 0.0 [protected]

Standard Deviation of data and sensor precision.

Definition at line 90 of file Object.h.

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

- 3DReconstruction/**Object.h**
- 3DReconstruction/**Object.cpp**

10.19 r3d::mtds::PlanePointsExamination Class Reference

```
#include <PlanePointsExamination.h>
```

Public Member Functions

- **PlanePointsExamination** ()
- **PlanePointsExamination** (pcl::PointCloud< pcl::PointXYZRGB >::Ptr &cloud, std::vector< pcl::PointIndices > &clusters, std::vector< Eigen::Vector4f > &coefficients, std::vector< long double > standardDeviation, double distanceParameter, double radiusParameter)
- **~PlanePointsExamination** ()
- void **computePlanePoints** ()

Computes coefficients and separates inliers and outliers for every plane segment in dataset.
- std::vector< Eigen::Vector4f > **getClusterPlanesCoefficients** ()
- std::vector< **r3d::prims::PlaneSegment** > **getSegmentedClustersPlanes** ()
- std::vector< pcl::PointCloud< pcl::PointXYZRGB > > **getSegmentedOutliers** ()

Public Attributes

- std::vector< **HASH** > **histogram**

10.19.1 Detailed Description

Definition at line 10 of file PlanePointsExamination.h.

10.19.2 Constructor & Destructor Documentation

10.19.2.1 r3d::mtds::PlanePointsExamination::PlanePointsExamination ()

Definition at line 10 of file PlanePointsExamination.cpp.

10.19.2.2 r3d::mtds::PlanePointsExamination::PlanePointsExamination (pcl::PointCloud< pcl::PointXYZRGB >::Ptr & cloud, std::vector< pcl::PointIndices > & clusters, std::vector< Eigen::Vector4f > & coefficients, std::vector< long double > standardDeviation, double distanceParameter, double radiusParameter)

Definition at line 15 of file PlanePointsExamination.cpp.

10.19.2.3 r3d::mtds::PlanePointsExamination::~~PlanePointsExamination ()

Definition at line 26 of file PlanePointsExamination.cpp.

10.19.3 Member Function Documentation

10.19.3.1 void r3d::mtds::PlanePointsExamination::computePlanePoints ()

Computes coefficients and separates inliers and outliers for every plane segment in dataset.

Definition at line 33 of file PlanePointsExamination.cpp.

10.19.3.2 `std::vector<Eigen::Vector4f> r3d::mtds::PlanePointsExamination::getClusterPlanesCoefficients () [inline]`

Definition at line 23 of file PlanePointsExamination.h.

10.19.3.3 `std::vector<r3d::prims::PlaneSegment> r3d::mtds::PlanePointsExamination::getSegmentedClustersPlanes () [inline]`

Definition at line 28 of file PlanePointsExamination.h.

10.19.3.4 `std::vector<pcl::PointCloud<pcl::PointXYZRGB> > r3d::mtds::PlanePointsExamination::getSegmentedOutliers () [inline]`

Definition at line 33 of file PlanePointsExamination.h.

10.19.4 Member Data Documentation

10.19.4.1 `std::vector<HASH> r3d::mtds::PlanePointsExamination::histogram`

Definition at line 38 of file PlanePointsExamination.h.

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

- 3DReconstruction/**PlanePointsExamination.h**
- 3DReconstruction/**PlanePointsExamination.cpp**

10.20 r3d::prims::PlaneSegment Class Reference

```
#include <PlaneSegment.h>
```

Public Member Functions

- **PlaneSegment** ()
- **~PlaneSegment** ()
- bool **computeOutliers** ()

Public Attributes

- Eigen::VectorXf **coefficients**
- pcl::PointCloud< pcl::PointXYZRGB > **cloudData**
- std::vector< int > **inliers**
- pcl::PointCloud< pcl::PointXYZRGB > **inliersPoints**
- pcl::PointCloud< pcl::PointXYZRGB > **outliers**

10.20.1 Detailed Description

Definition at line 4 of file PlaneSegment.h.

10.20.2 Constructor & Destructor Documentation

10.20.2.1 `r3d::prims::PlaneSegment::PlaneSegment ()`

Definition at line 6 of file PlaneSegment.cpp.

10.20.2.2 `r3d::prims::PlaneSegment::~~PlaneSegment ()`

Definition at line 11 of file PlaneSegment.cpp.

10.20.3 Member Function Documentation

10.20.3.1 `bool r3d::prims::PlaneSegment::computeOutliers ()`

< extract inliers = line points

Definition at line 15 of file PlaneSegment.cpp.

10.20.4 Member Data Documentation

10.20.4.1 `pcl::PointCloud<pcl::PointXYZRGB> r3d::prims::PlaneSegment::cloudData`

Definition at line 11 of file PlaneSegment.h.

10.20.4.2 `Eigen::VectorXf r3d::prims::PlaneSegment::coefficients`

Definition at line 10 of file PlaneSegment.h.

10.20.4.3 `std::vector<int> r3d::prims::PlaneSegment::inliers`

Definition at line 12 of file PlaneSegment.h.

10.20.4.4 `pcl::PointCloud<pcl::PointXYZRGB> r3d::prims::PlaneSegment::inliersPoints`

Definition at line 13 of file PlaneSegment.h.

10.20.4.5 pcl::PointCloud<pcl::PointXYZRGB> r3d::prims::PlaneSegment::outliers

Definition at line 14 of file PlaneSegment.h.

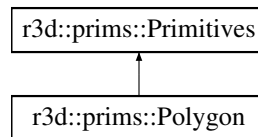
The documentation for this class was generated from the following files:

- 3DReconstruction/**PlaneSegment.h**
- 3DReconstruction/**PlaneSegment.cpp**

10.21 r3d::prims::Polygon Class Reference

```
#include <Polygon.h>
```

Inheritance diagram for r3d::prims::Polygon:



Public Member Functions

- **Polygon** ()
- **Polygon** (Eigen::Vector4f coefficients)
- **Polygon** (pcl::PointCloud< pcl::PointXYZ > coordinates)
- **Polygon** (r3d::prims::Primitives *prims)
- void **addCoordinate** (const pcl::PointXYZ point) override
- void **addCoordinates** (const pcl::PointCloud< pcl::PointXYZ > coordinateCloud)
- void **setCoefficients** (const Eigen::Vector4f coeff)
- Eigen::Vector4f & **getCoefficients** ()
- **~Polygon** ()

Additional Inherited Members

10.21.1 Detailed Description

Definition at line 6 of file Polygon.h.

10.21.2 Constructor & Destructor Documentation

10.21.2.1 r3d::prims::Polygon::Polygon ()

Definition at line 5 of file Polygon.cpp.

10.21.2.2 `r3d::prims::Polygon::Polygon (Eigen::Vector4f coefficients)`

Definition at line 9 of file Polygon.cpp.

10.21.2.3 `r3d::prims::Polygon::Polygon (pcl::PointCloud< pcl::PointXYZ > coordinates)`

Definition at line 14 of file Polygon.cpp.

10.21.2.4 `r3d::prims::Polygon::Polygon (r3d::prims::Primitives * prims)`

Definition at line 19 of file Polygon.cpp.

10.21.2.5 `r3d::prims::Polygon::~~Polygon ()`

Definition at line 24 of file Polygon.cpp.

10.21.3 Member Function Documentation

10.21.3.1 `void r3d::prims::Polygon::addCoordinate (const pcl::PointXYZ point)` `[override],[virtual]`

Implements `r3d::prims::Primitives` (p. ??).

Definition at line 28 of file Polygon.cpp.

10.21.3.2 `void r3d::prims::Polygon::addCoordinates (const pcl::PointCloud< pcl::PointXYZ > coordinateCloud)`

Definition at line 33 of file Polygon.cpp.

10.21.3.3 `Eigen::Vector4f & r3d::prims::Polygon::getCoefficients ()`

Definition at line 43 of file Polygon.cpp.

10.21.3.4 `void r3d::prims::Polygon::setCoefficients (const Eigen::Vector4f coeff)`

Definition at line 38 of file Polygon.cpp.

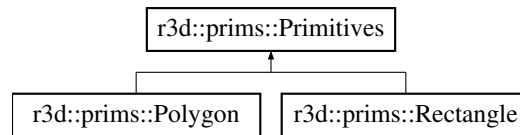
The documentation for this class was generated from the following files:

- 3DReconstruction/**Polygon.h**
- 3DReconstruction/**Polygon.cpp**

10.22 r3d::prims::Primitives Class Reference

```
#include <Primitives.h>
```

Inheritance diagram for r3d::prims::Primitives:



Public Member Functions

- **Primitives** ()
- virtual **~Primitives** ()
- virtual void **addCoordinate** (const pcl::PointXYZ point)=0
- pcl::PointCloud< pcl::PointXYZ > & **getCoordinates** ()
- bool **empty** () const

Protected Attributes

- pcl::PointCloud< pcl::PointXYZ > **m_coordinates**

10.22.1 Detailed Description

Definition at line 8 of file Primitives.h.

10.22.2 Constructor & Destructor Documentation

10.22.2.1 r3d::prims::Primitives::Primitives ()

Definition at line 5 of file Primitives.cpp.

10.22.2.2 r3d::prims::Primitives::~~Primitives () [virtual]

Definition at line 10 of file Primitives.cpp.

10.22.3 Member Function Documentation

10.22.3.1 virtual void r3d::prims::Primitives::addCoordinate (const pcl::PointXYZ *point*) [pure virtual]

Implemented in **r3d::prims::Rectangle** (p. ??), and **r3d::prims::Polygon** (p. ??).

10.22.3.2 `bool r3d::prims::Primitives::empty () const [inline]`

Definition at line 19 of file Primitives.h.

10.22.3.3 `pcl::PointCloud<pcl::PointXYZ> & r3d::prims::Primitives::getCoordinates () [inline]`

Definition at line 15 of file Primitives.h.

10.22.4 Member Data Documentation

10.22.4.1 `pcl::PointCloud<pcl::PointXYZ> r3d::prims::Primitives::m_coordinates [protected]`

Definition at line 24 of file Primitives.h.

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

- 3DReconstruction/**Primitives.h**
- 3DReconstruction/**Primitives.cpp**

10.23 r3d::mtds::Reconstruction Class Reference

This is an abstract parent class to all **Reconstruction** (p. ??) methods. Contains original dataset as partial point clouds in aggregated ClosedSpace object `m_methodName` should be set to dedicated reconstruction method name.

```
#include "Reconstruction.h"
```

Inheritance diagram for `r3d::mtds::Reconstruction`:



Public Member Functions

- **Reconstruction** ()
- **Reconstruction** (std::string name)
- virtual **~Reconstruction** ()
- void **setName** (std::string name)
- std::string **getMethodName** () const
- void **setData** (pcl::PointCloud< pcl::PointXYZ > &data, std::string name)
- void **setData** (pcl::PointCloud< pcl::RGB > &data, std::string name)
- void **setData** (pcl::PointCloud< pcl::Normal > &data, std::string name)
- **r3d::space::ClosedSpace** & **getDataSet** ()
- virtual **r3d::space::ClosedSpace** & **getReconstructedDataset** ()
Virtual in case that some reconstruction method needed to override and change the outcome.
- virtual bool **calculate** ()=0
virtual function to be implemented by child classes and reconstruction methods
- virtual void **visualise** (pcl::visualization::PCLVisualizer *viewer)=0
virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation
- bool **normalEstimation** (std::string saveLocation)
- virtual void **setConfiguration** (TiXmlNode *param)=0
- const char * **getParameter** (TiXmlNode *param, const char *name)

Public Attributes

- const char * **_config_name**

Protected Attributes

- bool **m_hasNewDataset** = false
- r3d::space::ClosedSpace **m_dataSet**
- std::string **m_methodName**

10.23.1 Detailed Description

This is an abstract parent class to all **Reconstruction** (p. ??) methods. Contains original dataset as partial point clouds in aggregated ClosedSpace object **m_methodName** should be set to dedicated reconstruction method name.

Author

Lukas Hudec

Definition at line 24 of file Reconstruction.h.

10.23.2 Constructor & Destructor Documentation

10.23.2.1 r3d::mtds::Reconstruction::Reconstruction ()

Definition at line 34 of file Reconstruction.cpp.

10.23.2.2 r3d::mtds::Reconstruction::Reconstruction (std::string *name*)

Constructor with parameter of **Reconstruction** (p. ??) name

Parameters

in	<i>name</i>	reconstruction method name
----	-------------	----------------------------

Definition at line 38 of file Reconstruction.cpp.

10.23.2.3 r3d::mtds::Reconstruction::~~Reconstruction () [virtual]

Definition at line 43 of file Reconstruction.cpp.

10.23.3 Member Function Documentation

10.23.3.1 `bool r3d::mtds::Reconstruction::calculate () [pure virtual]`

virtual function to be implemented by child classes and reconstruction methods

Implemented in `r3d::mtds::BirkysMethodReconstruction` (p. ??), `r3d::mtds::SlicBasedReconstruction` (p. ??), `r3d::mtds::GrowingRegionReconstruction` (p. ??), `r3d::mtds::CloudRotationTest` (p. ??), and `r3d::mtds::NumeroUnoReconstruction` (p. ??).

Definition at line 52 of file `Reconstruction.cpp`.

10.23.3.2 `r3d::space::ClosedSpace& r3d::mtds::Reconstruction::getDataSet () [inline]`

Definition at line 66 of file `Reconstruction.h`.

10.23.3.3 `std::string r3d::mtds::Reconstruction::getMethodName () const [inline]`

Definition at line 42 of file `Reconstruction.h`.

10.23.3.4 `const char * r3d::mtds::Reconstruction::getParameter (TiXmlNode * param, const char * name)`

Definition at line 71 of file `Reconstruction.cpp`.

10.23.3.5 `r3d::space::ClosedSpace< pcl::PointXYZ > & r3d::mtds::Reconstruction::getReconstructedDataset () [virtual]`

Virtual in case that some reconstruction method needed to override and change the outcome.

Returns

`r3d::space::ClosedSpace< pcl::PointXYZ >` dataset with all segmentated structures and primitives

Definition at line 47 of file `Reconstruction.cpp`.

10.23.3.6 `bool r3d::mtds::Reconstruction::normalEstimation (std::string saveLocation)`

Definition at line 57 of file `Reconstruction.cpp`.

10.23.3.7 `void r3d::mtds::Reconstruction::setConfiguration (TiXmlNode * param) [pure virtual]`

Implemented in `r3d::mtds::BirkysMethodReconstruction` (p. ??), `r3d::mtds::SlicBasedReconstruction` (p. ??), `r3d::mtds::GrowingRegionReconstruction` (p. ??), `r3d::mtds::CloudRotationTest` (p. ??), and `r3d::mtds::NumeroUnoReconstruction` (p. ??).

Definition at line 63 of file `Reconstruction.cpp`.

10.23.3.8 `void r3d::mtds::Reconstruction::setData (pcl::PointCloud< pcl::PointXYZ > & data, std::string name)`
`[inline]`

Definition at line 47 of file Reconstruction.h.

10.23.3.9 `void r3d::mtds::Reconstruction::setData (pcl::PointCloud< pcl::RGB > & data, std::string name)` `[inline]`

Definition at line 53 of file Reconstruction.h.

10.23.3.10 `void r3d::mtds::Reconstruction::setData (pcl::PointCloud< pcl::Normal > & data, std::string name)`
`[inline]`

Definition at line 59 of file Reconstruction.h.

10.23.3.11 `void r3d::mtds::Reconstruction::setName (std::string name)` `[inline]`

Definition at line 37 of file Reconstruction.h.

10.23.3.12 `virtual void r3d::mtds::Reconstruction::visualise (pcl::visualization::PCLVisualizer * viewer)` `[pure virtual]`

virtual function to be implemented by child classes and reconstruction methods as debug/test visualisation

Implemented in `r3d::mtds::BirkysMethodReconstruction` (p. ??), `r3d::mtds::SlicBasedReconstruction` (p. ??), `r3d::mtds::GrowingRegionReconstruction` (p. ??), `r3d::mtds::CloudRotationTest` (p. ??), and `r3d::mtds::NumeroUnoReconstruction` (p. ??).

10.23.4 Member Data Documentation

10.23.4.1 `const char* r3d::mtds::Reconstruction::_config_name`

Definition at line 29 of file Reconstruction.h.

10.23.4.2 `r3d::space::ClosedSpace r3d::mtds::Reconstruction::m_dataSet` `[protected]`

Definition at line 94 of file Reconstruction.h.

10.23.4.3 `bool r3d::mtds::Reconstruction::m_hasNewDataset = false` `[protected]`

Definition at line 92 of file Reconstruction.h.

10.23.4.4 `std::string r3d::mtds::Reconstruction::m_methodName` `[protected]`

Definition at line 96 of file `Reconstruction.h`.

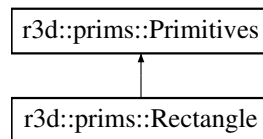
The documentation for this class was generated from the following files:

- 3DReconstruction/**Reconstruction.h**
- 3DReconstruction/**Reconstruction.cpp**

10.24 r3d::prims::Rectangle Class Reference

```
#include <Rectangle.h>
```

Inheritance diagram for `r3d::prims::Rectangle`:



Public Member Functions

- **Rectangle** ()
- **Rectangle** (`r3d::prims::Primitives` *prims)
- **Rectangle** (`Eigen::Vector4f` coefficients)
- **Rectangle** (`pcl::PointXYZ` A, `pcl::PointXYZ` B, `pcl::PointXYZ` C, `pcl::PointXYZ` D)
- **~Rectangle** ()
- void **addCoordinate** (`const` `pcl::PointXYZ` point) override
Adds a rectangle corner coordinate.
- void **setCoordinates** (`pcl::PointXYZ` A, `pcl::PointXYZ` B, `pcl::PointXYZ` C, `pcl::PointXYZ` D)
- void **setCoefficients** (`Eigen::Vector4f` &coeff)
- `Eigen::Vector4f` & **getCoefficients** ()

Additional Inherited Members

10.24.1 Detailed Description

Definition at line 9 of file `Rectangle.h`.

10.24.2 Constructor & Destructor Documentation

10.24.2.1 `r3d::prims::Rectangle::Rectangle ()`

Definition at line 6 of file `Rectangle.cpp`.

10.24.2.2 r3d::prims::Rectangle::Rectangle (r3d::prims::Primitives * *prims*)

Definition at line 17 of file Rectangle.cpp.

10.24.2.3 r3d::prims::Rectangle::Rectangle (Eigen::Vector4f *coefficients*)

Definition at line 11 of file Rectangle.cpp.

10.24.2.4 r3d::prims::Rectangle::Rectangle (pcl::PointXYZ *A*, pcl::PointXYZ *B*, pcl::PointXYZ *C*, pcl::PointXYZ *D*)

Definition at line 23 of file Rectangle.cpp.

10.24.2.5 r3d::prims::Rectangle::~~Rectangle ()

Definition at line 29 of file Rectangle.cpp.

10.24.3 Member Function Documentation

10.24.3.1 void r3d::prims::Rectangle::addCoordinate (const pcl::PointXYZ *point*) [override],[virtual]

Adds a rectangle corner coordinate.

As this is a rectangle, if there is already 4 corners, it pops the first and adds this to the end

Implements **r3d::prims::Primitives** (p. ??).

Definition at line 33 of file Rectangle.cpp.

10.24.3.2 Eigen::Vector4f & r3d::prims::Rectangle::getCoefficients ()

Definition at line 58 of file Rectangle.cpp.

10.24.3.3 void r3d::prims::Rectangle::setCoefficients (Eigen::Vector4f & *coeff*)

Definition at line 50 of file Rectangle.cpp.

10.24.3.4 void r3d::prims::Rectangle::setCoordinates (pcl::PointXYZ *A*, pcl::PointXYZ *B*, pcl::PointXYZ *C*, pcl::PointXYZ *D*)

Definition at line 42 of file Rectangle.cpp.

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

- 3DReconstruction/**Rectangle.h**
- 3DReconstruction/**Rectangle.cpp**

10.25 See Class Reference

```
#include <for>
```

10.25.1 Detailed Description

the implementation of this class

Author

Martin Jurik

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

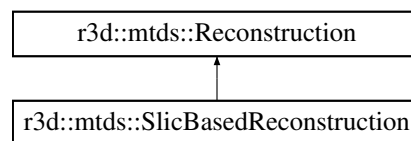
- `gui/gui.h`

10.26 r3d::mtds::SlicBasedReconstruction Class Reference

This is a derived class from class **Reconstruction** (p. ??). Contains methods and variables for 3D reconstruction based on SLIC clustering algorithm STILL UNDER CONSTRUCTION.

```
#include "SlicBasedReconstruction.h"
```

Inheritance diagram for r3d::mtds::SlicBasedReconstruction:



Public Member Functions

- **SlicBasedReconstruction** ()
- **~SlicBasedReconstruction** ()
- virtual void **setConfiguration** (**TiXmlNode** *param) override
virtual function for setting up the member variables from the XML config file
- bool **calculate** () override
virtual function responsible for the whole reconstruction calculations
- virtual void **visualise** (pcl::visualization::PCLVisualizer *viewer) override
virtual function for debug/test visualisation needed during development

Static Public Member Functions

- static std::vector< pcl::PointXYZ > **estimateBoundingBox** (pcl::PointCloud< pcl::PointXYZ >::Ptr cloud)
static function which finds the boundary points of the given point cloud
- static void **calculateMeanAtributesOfClusters** (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloudXYZ, pcl::PointCloud< pcl::Normal >::Ptr &cloudNormals, pcl::PointCloud< pcl::PointXYZ >::Ptr seedsXYZ, pcl::PointCloud< pcl::Normal >::Ptr seedsNormals, std::vector< pcl::PointIndices > &clusters)
TODO.
- static std::vector< pcl::PointIndices > **assignPointsToNearestSeeds** (pcl::PointCloud< pcl::PointXYZ >::Ptr &cloudXYZ, pcl::PointCloud< pcl::Normal >::Ptr &cloudNormals, pcl::PointCloud< pcl::PointXYZ >::Ptr &seedsXYZ, pcl::PointCloud< pcl::Normal >::Ptr &seedsNormals, float gridSize, float compactness)
function for assigning every point of a cloud to its nearest seed according to SLIC distance TODO finish doc

Additional Inherited Members

10.26.1 Detailed Description

This is a derived class from class **Reconstruction** (p. ??). Contains methods and variables for 3D reconstruction based on SLIC clustering algorithm STILL UNDER CONSTRUCTION.

Author

Robert Birkus

Definition at line 20 of file SlicBasedReconstruction.h.

10.26.2 Constructor & Destructor Documentation

10.26.2.1 r3d::mtds::SlicBasedReconstruction::SlicBasedReconstruction ()

Definition at line 5 of file SlicBasedReconstruction.cpp.

10.26.2.2 r3d::mtds::SlicBasedReconstruction::~~SlicBasedReconstruction ()

Definition at line 18 of file SlicBasedReconstruction.cpp.

10.26.3 Member Function Documentation

10.26.3.1 std::vector< pcl::PointIndices > r3d::mtds::SlicBasedReconstruction::assignPointsToNearestSeeds (pcl::PointCloud< pcl::PointXYZ >::Ptr & cloudXYZ, pcl::PointCloud< pcl::Normal >::Ptr & cloudNormals, pcl::PointCloud< pcl::PointXYZ >::Ptr & seedsXYZ, pcl::PointCloud< pcl::Normal >::Ptr & seedsNormals, float gridSize, float compactness) [static]

function for assigning every point of a cloud to its nearest seed according to SLIC distance TODO finish doc

< Loop over 27 neighbour seeds and find the nearest one

< handle inverse normals

Definition at line 145 of file SlicBasedReconstruction.cpp.

10.26.3.2 `bool r3d::mtds::SlicBasedReconstruction::calculate () [override],[virtual]`

virtual function responsible for the whole reconstruction calculations

< Loading needed data and define needed variables

< Calculate normals if they are not available

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 60 of file SlicBasedReconstruction.cpp.

10.26.3.3 `void r3d::mtds::SlicBasedReconstruction::calculateMeanAtributesOfClusters (pcl::PointCloud< pcl::PointXYZ >::Ptr & cloudXYZ, pcl::PointCloud< pcl::Normal >::Ptr & cloudNormals, pcl::PointCloud< pcl::PointXYZ >::Ptr seedsXYZ, pcl::PointCloud< pcl::Normal >::Ptr seedsNormals, std::vector< pcl::PointIndices > & clusters) [static]`

TODO.

Definition at line 180 of file SlicBasedReconstruction.cpp.

10.26.3.4 `std::vector< pcl::PointXYZ > r3d::mtds::SlicBasedReconstruction::estimateBoundingBox (pcl::PointCloud< pcl::PointXYZ >::Ptr cloud) [static]`

static function which finds the boundary points of the given point cloud

Definition at line 22 of file SlicBasedReconstruction.cpp.

10.26.3.5 `void r3d::mtds::SlicBasedReconstruction::setConfiguration (TiXmlNode * param) [override],[virtual]`

virtual function for setting up the member variables from the XML config file

Parameters

in	<i>param</i>	- XML parser class TiXmlNode (p. ??)
----	--------------	---

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 47 of file SlicBasedReconstruction.cpp.

10.26.3.6 `void r3d::mtds::SlicBasedReconstruction::visualise (pcl::visualization::PCLVisualizer * viewer) [override],[virtual]`

virtual function for debug/test visualisation needed during development

Parameters

in	<i>viewer</i>	- PCL visualisation class <code>pcl::visualization::PCLVisualizer*</code>
----	---------------	---

Implements **r3d::mtds::Reconstruction** (p. ??).

Definition at line 138 of file `SlicBasedReconstruction.cpp`.

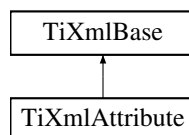
The documentation for this class was generated from the following files:

- 3DReconstruction/**SlicBasedReconstruction.h**
- 3DReconstruction/**SlicBasedReconstruction.cpp**

10.27 TiXmlAttribute Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlAttribute:



Public Member Functions

- **TiXmlAttribute** ()
Construct an empty attribute.
- **TiXmlAttribute** (const char *_name, const char *_value)
Construct an attribute with a name and value.
- const char * **Name** () const
Return the name of this attribute.
- const char * **Value** () const
Return the value of this attribute.
- int **IntValue** () const
Return the value of this attribute, converted to an integer.
- double **DoubleValue** () const
Return the value of this attribute, converted to a double.
- const **TIXML_STRING** & **NameTStr** () const
- int **QueryIntValue** (int *_value) const
- int **QueryDoubleValue** (double *_value) const
*QueryDoubleValue examines the value string. See (p. ??) **QueryIntValue()** (p. ??).*
- void **SetName** (const char *_name)
Set the name of this attribute.
- void **SetValue** (const char *_value)
Set the value.
- void **SetIntValue** (int _value)
Set the value from an integer.
- void **SetDoubleValue** (double _value)
Set the value from a double.
- const **TiXmlAttribute** * **Next** () const
Get the next sibling attribute in the DOM. Returns null at end.

- **TiXmlAttribute * Next ()**
- **const TiXmlAttribute * Previous () const**
Get the previous sibling attribute in the DOM. Returns null at beginning.
- **TiXmlAttribute * Previous ()**
- **bool operator== (const TiXmlAttribute &rhs) const**
- **bool operator< (const TiXmlAttribute &rhs) const**
- **bool operator> (const TiXmlAttribute &rhs) const**
- **virtual const char * Parse (const char *p, TiXmlParsingData *data, TiXmlEncoding encoding)**
- **virtual void Print (FILE *cf, int depth) const**
- **void Print (FILE *cf, int depth, TiXML_STRING *str) const**
- **void SetDocument (TiXmlDocument *doc)**

Friends

- class **TiXmlAttributeSet**

Additional Inherited Members

10.27.1 Detailed Description

An attribute is a name-value pair. Elements have an arbitrary number of attributes, each with a unique name.

Note

The attributes are not TiXmlNode's, since they are not part of the tinyXML document object model. There are other suggested ways to look at this problem.

Definition at line 755 of file tinyxml.h.

10.27.2 Constructor & Destructor Documentation

10.27.2.1 TiXmlAttribute::TiXmlAttribute () [inline]

Construct an empty attribute.

Definition at line 761 of file tinyxml.h.

10.27.2.2 TiXmlAttribute::TiXmlAttribute (const char * _name, const char * _value) [inline]

Construct an attribute with a name and value.

Definition at line 779 of file tinyxml.h.

10.27.3 Member Function Documentation

10.27.3.1 double TiXmlAttribute::DoubleValue () const

Return the value of this attribute, converted to a double.

Definition at line 1254 of file tinyxml.cpp.

10.27.3.2 int TiXmlAttribute::IntValue () const

Return the value of this attribute, converted to an integer.

Definition at line 1249 of file tinyxml.cpp.

10.27.3.3 const char* TiXmlAttribute::Name () const [inline]

Return the name of this attribute.

Definition at line 787 of file tinyxml.h.

10.27.3.4 const TiXML_STRING& TiXmlAttribute::NameTStr () const [inline]

Definition at line 796 of file tinyxml.h.

10.27.3.5 const TiXmlAttribute * TiXmlAttribute::Next () const

Get the next sibling attribute in the DOM. Returns null at end.

Definition at line 1147 of file tinyxml.cpp.

10.27.3.6 TiXmlAttribute* TiXmlAttribute::Next () [inline]

Definition at line 826 of file tinyxml.h.

10.27.3.7 bool TiXmlAttribute::operator< (const TiXmlAttribute & rhs) const [inline]

Definition at line 837 of file tinyxml.h.

10.27.3.8 bool TiXmlAttribute::operator== (const TiXmlAttribute & rhs) const [inline]

Definition at line 836 of file tinyxml.h.

10.27.3.9 `bool TiXmlAttribute::operator> (const TiXmlAttribute & rhs) const` `[inline]`

Definition at line 838 of file tinyxml.h.

10.27.3.10 `const char * TiXmlAttribute::Parse (const char * p, TiXmlParsingData * data, TiXmlEncoding encoding)`
`[virtual]`

Implements **TiXmlBase** (p. ??).

Definition at line 1373 of file tinyxmlparser.cpp.

10.27.3.11 `const TiXmlAttribute * TiXmlAttribute::Previous () const`

Get the previous sibling attribute in the DOM. Returns null at beginning.

Definition at line 1167 of file tinyxml.cpp.

10.27.3.12 `TiXmlAttribute* TiXmlAttribute::Previous ()` `[inline]`

Definition at line 832 of file tinyxml.h.

10.27.3.13 `virtual void TiXmlAttribute::Print (FILE * cfile, int depth) const` `[inline], [virtual]`

All TinyXml classes can print themselves to a filestream or the string class (**TiXmlString** (p. ??) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements **TiXmlBase** (p. ??).

Definition at line 846 of file tinyxml.h.

10.27.3.14 `void TiXmlAttribute::Print (FILE * cfile, int depth, TIXML_STRING * str) const`

Definition at line 1187 of file tinyxml.cpp.

10.27.3.15 `int TiXmlAttribute::QueryDoubleValue (double * _value) const`

`QueryDoubleValue` examines the value string. See (p. ??) `QueryIntValue()` (p. ??).

Definition at line 1220 of file tinyxml.cpp.

10.27.3.16 int TiXmlAttribute::QueryIntValue (int * _value) const

QueryIntValue examines the value string. It is an alternative to the **IntValue()** (p. ??) method with richer error checking. If the value is an integer, it is stored in 'value' and the call returns TIXML_SUCCESS. If it is not an integer, it returns TIXML_WRONG_TYPE.

A specialized but useful call. Note that for success it returns 0, which is the opposite of almost all other TinyXml calls.

Definition at line 1213 of file tinyxml.cpp.

10.27.3.17 void TiXmlAttribute::SetDocument (TiXmlDocument * doc) [inline]

Definition at line 853 of file tinyxml.h.

10.27.3.18 void TiXmlAttribute::SetDoubleValue (double _value)

Set the value from a double.

Definition at line 1238 of file tinyxml.cpp.

10.27.3.19 void TiXmlAttribute::SetIntValue (int _value)

Set the value from an integer.

Definition at line 1227 of file tinyxml.cpp.

10.27.3.20 void TiXmlAttribute::SetName (const char * _name) [inline]

Set the name of this attribute.

Definition at line 811 of file tinyxml.h.

10.27.3.21 void TiXmlAttribute::SetValue (const char * _value) [inline]

Set the value.

Definition at line 812 of file tinyxml.h.

10.27.3.22 const char* TiXmlAttribute::Value () const [inline]

Return the value of this attribute.

Definition at line 788 of file tinyxml.h.

10.27.4 Friends And Related Function Documentation

10.27.4.1 friend class TiXmlAttributeSet [friend]

Definition at line 757 of file tinyxml.h.

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

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**
- DataHandling/**tinyxmlparser.cpp**

10.28 TiXmlAttributeSet Class Reference

```
#include <tinyxml.h>
```

Public Member Functions

- **TiXmlAttributeSet** ()
- **~TiXmlAttributeSet** ()
- void **Add** (TiXmlAttribute *attribute)
- void **Remove** (TiXmlAttribute *attribute)
- const **TiXmlAttribute** * **First** () const
- **TiXmlAttribute** * **First** ()
- const **TiXmlAttribute** * **Last** () const
- **TiXmlAttribute** * **Last** ()
- **TiXmlAttribute** * **Find** (const char *_name) const
- **TiXmlAttribute** * **FindOrCreate** (const char *_name)

10.28.1 Detailed Description

Definition at line 879 of file tinyxml.h.

10.28.2 Constructor & Destructor Documentation

10.28.2.1 TiXmlAttributeSet::TiXmlAttributeSet ()

Definition at line 1477 of file tinyxml.cpp.

10.28.2.2 TiXmlAttributeSet::~TiXmlAttributeSet ()

Definition at line 1484 of file tinyxml.cpp.

10.28.3 Member Function Documentation

10.28.3.1 void TiXmlAttributeSet::Add (TiXmlAttribute * *attribute*)

Definition at line 1491 of file tinyxml.cpp.

10.28.3.2 TiXmlAttribute * TiXmlAttributeSet::Find (const char * *_name*) const

Definition at line 1549 of file tinyxml.cpp.

10.28.3.3 TiXmlAttribute * TiXmlAttributeSet::FindOrCreate (const char * *_name*)

Definition at line 1560 of file tinyxml.cpp.

10.28.3.4 const TiXmlAttribute* TiXmlAttributeSet::First () const [inline]

Definition at line 888 of file tinyxml.h.

10.28.3.5 TiXmlAttribute* TiXmlAttributeSet::First () [inline]

Definition at line 889 of file tinyxml.h.

10.28.3.6 const TiXmlAttribute* TiXmlAttributeSet::Last () const [inline]

Definition at line 890 of file tinyxml.h.

10.28.3.7 TiXmlAttribute* TiXmlAttributeSet::Last () [inline]

Definition at line 891 of file tinyxml.h.

10.28.3.8 void TiXmlAttributeSet::Remove (TiXmlAttribute * *attribute*)

Definition at line 1506 of file tinyxml.cpp.

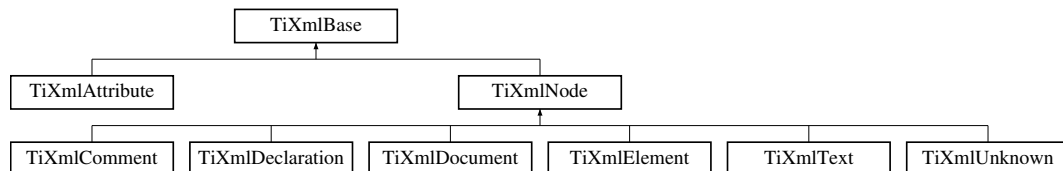
The documentation for this class was generated from the following files:

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**

10.29 TiXmlBase Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlBase:



Public Types

- enum {
TIXML_NO_ERROR = 0, **TIXML_ERROR**, **TIXML_ERROR_OPENING_FILE**, **TIXML_ERROR_PARSING_ELEMENT**,
TIXML_ERROR_FAILED_TO_READ_ELEMENT_NAME, **TIXML_ERROR_READING_ELEMENT_VALUE**,
TIXML_ERROR_READING_ATTRIBUTES, **TIXML_ERROR_PARSING_EMPTY**,
TIXML_ERROR_READING_END_TAG, **TIXML_ERROR_PARSING_UNKNOWN**, **TIXML_ERROR_PARSING_COMMENT**,
TIXML_ERROR_PARSING_DECLARATION,
TIXML_ERROR_DOCUMENT_EMPTY, **TIXML_ERROR_EMBEDDED_NULL**, **TIXML_ERROR_PARSING_CDATA**,
TIXML_ERROR_DOCUMENT_TOP_ONLY,
TIXML_ERROR_STRING_COUNT }

Public Member Functions

- **TiXmlBase** ()
- virtual **~TiXmlBase** ()
- virtual void **Print** (FILE *cfile, int depth) const =0
- int **Row** () const
- int **Column** () const
See (p. ??) Row() (p. ??)
- void **SetUserData** (void *user)
Set a pointer to arbitrary user data.
- void * **GetUserData** ()
Get a pointer to arbitrary user data.
- const void * **GetUserData** () const
Get a pointer to arbitrary user data.
- virtual const char * **Parse** (const char *p, **TiXmlParsingData** *data, **TiXmlEncoding** encoding)=0

Static Public Member Functions

- static void **SetCondenseWhiteSpace** (bool condense)
- static bool **IsWhiteSpaceCondensed** ()
Return the current white space setting.
- static void **EncodeString** (const **TIXML_STRING** &str, **TIXML_STRING** *out)

Static Public Attributes

- static const int **utf8ByteTable** [256]

Static Protected Member Functions

- static const char * **SkipWhiteSpace** (const char *, **TiXmlEncoding** encoding)
- static bool **IsWhiteSpace** (char c)
- static bool **IsWhiteSpace** (int c)
- static const char * **ReadName** (const char *p, **TIXML_STRING** *name, **TiXmlEncoding** encoding)
- static const char * **ReadText** (const char *in, **TIXML_STRING** *text, bool ignoreWhiteSpace, const char *endTag, bool ignoreCase, **TiXmlEncoding** encoding)
- static const char * **GetEntity** (const char *in, char *value, int *length, **TiXmlEncoding** encoding)
- static const char * **GetChar** (const char *p, char *_value, int *length, **TiXmlEncoding** encoding)
- static bool **StringEqual** (const char *p, const char *endTag, bool ignoreCase, **TiXmlEncoding** encoding)
- static int **IsAlpha** (unsigned char anyByte, **TiXmlEncoding** encoding)
- static int **IsAlphaNum** (unsigned char anyByte, **TiXmlEncoding** encoding)
- static int **ToLower** (int v, **TiXmlEncoding** encoding)
- static void **ConvertUTF32ToUTF8** (unsigned long input, char *output, int *length)

Protected Attributes

- **TiXmlCursor** location
- void * **userData**

Field containing a generic user pointer.

Static Protected Attributes

- static const char * **errorString** [**TIXML_ERROR_STRING_COUNT**]

Friends

- class **TiXmlNode**
- class **TiXmlElement**
- class **TiXmlDocument**

10.29.1 Detailed Description

TiXmlBase (p. ??) is a base class for every class in TinyXml. It does little except to establish that TinyXml classes can be printed and provide some utility functions.

In XML, the document and elements can contain other elements and other types of nodes.

```
A Document can contain: Element (container or leaf)
Comment (leaf)
Unknown (leaf)
Declaration( leaf )
```

```
An Element can contain: Element (container or leaf)
Text (leaf)
Attributes (not on tree)
Comment (leaf)
Unknown (leaf)
```

```
A Decleration contains: Attributes (not on tree)
```

Definition at line 170 of file tinyxml.h.

10.29.2 Member Enumeration Documentation

10.29.2.1 anonymous enum

Enumerator

TIXML_NO_ERROR
TIXML_ERROR
TIXML_ERROR_OPENING_FILE
TIXML_ERROR_PARSING_ELEMENT
TIXML_ERROR_FAILED_TO_READ_ELEMENT_NAME
TIXML_ERROR_READING_ELEMENT_VALUE
TIXML_ERROR_READING_ATTRIBUTES
TIXML_ERROR_PARSING_EMPTY
TIXML_ERROR_READING_END_TAG
TIXML_ERROR_PARSING_UNKNOWN
TIXML_ERROR_PARSING_COMMENT
TIXML_ERROR_PARSING_DECLARATION
TIXML_ERROR_DOCUMENT_EMPTY
TIXML_ERROR_EMBEDDED_NULL
TIXML_ERROR_PARSING_CDATA
TIXML_ERROR_DOCUMENT_TOP_ONLY
TIXML_ERROR_STRING_COUNT

Definition at line 240 of file tinyxml.h.

10.29.3 Constructor & Destructor Documentation

10.29.3.1 TiXmlBase::TiXmlBase () [inline]

Definition at line 177 of file tinyxml.h.

10.29.3.2 virtual TiXmlBase::~~TiXmlBase () [inline],[virtual]

Definition at line 178 of file tinyxml.h.

10.29.4 Member Function Documentation

10.29.4.1 int TiXmlBase::Column () const [inline]

See (p. ??) **Row()** (p. ??)

Definition at line 221 of file tinyxml.h.

10.29.4.2 `void TiXmlBase::ConvertUTF32ToUTF8 (unsigned long input, char * output, int * length)` `[static]`,
`[protected]`

Definition at line 65 of file `tinyxmlparser.cpp`.

10.29.4.3 `void TiXmlBase::EncodeString (const TIXML_STRING & str, TIXML_STRING * out)` `[static]`

Expands entities in a string. Note this should not contain the tag's '<', '>', etc, or they will be transformed into entities!

Definition at line 29 of file `tinyxml.cpp`.

10.29.4.4 `static const char* TiXmlBase::GetChar (const char * p, char * _value, int * length, TiXmlEncoding encoding)`
`[inline]`, `[static]`, `[protected]`

Definition at line 303 of file `tinyxml.h`.

10.29.4.5 `const char * TiXmlBase::GetEntity (const char * in, char * value, int * length, TiXmlEncoding encoding)`
`[static]`, `[protected]`

Definition at line 419 of file `tinyxmlparser.cpp`.

10.29.4.6 `void* TiXmlBase::GetUserData ()` `[inline]`

Get a pointer to arbitrary user data.

Definition at line 224 of file `tinyxml.h`.

10.29.4.7 `const void* TiXmlBase::GetUserData () const` `[inline]`

Get a pointer to arbitrary user data.

Definition at line 225 of file `tinyxml.h`.

10.29.4.8 `int TiXmlBase::IsAlpha (unsigned char anyByte, TiXmlEncoding encoding)` `[static]`, `[protected]`

Definition at line 108 of file `tinyxmlparser.cpp`.

10.29.4.9 `int TiXmlBase::IsAlphaNum (unsigned char anyByte, TiXmlEncoding encoding)` `[static]`,
`[protected]`

Definition at line 129 of file `tinyxmlparser.cpp`.

10.29.4.10 `static bool TiXmlBase::IsWhiteSpace (char c)` `[inline], [static], [protected]`

Definition at line 266 of file `tinyxml.h`.

10.29.4.11 `static bool TiXmlBase::IsWhiteSpace (int c)` `[inline], [static], [protected]`

Definition at line 270 of file `tinyxml.h`.

10.29.4.12 `static bool TiXmlBase::IsWhiteSpaceCondensed ()` `[inline], [static]`

Return the current white space setting.

Definition at line 200 of file `tinyxml.h`.

10.29.4.13 `virtual const char* TiXmlBase::Parse (const char * p, TiXmlParsingData * data, TiXmlEncoding encoding)`
`[pure virtual]`

Implemented in **TiXmlDocument** (p. ??), **TiXmlUnknown** (p. ??), **TiXmlDeclaration** (p. ??), **TiXmlText** (p. ??), **TiXmlComment** (p. ??), **TiXmlElement** (p. ??), and **TiXmlAttribute** (p. ??).

10.29.4.14 `virtual void TiXmlBase::Print (FILE * cfile, int depth) const` `[pure virtual]`

All TinyXml classes can print themselves to a filestream or the string class (**TiXmlString** (p. ??) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implemented in **TiXmlDocument** (p. ??), **TiXmlUnknown** (p. ??), **TiXmlDeclaration** (p. ??), **TiXmlText** (p. ??), **TiXmlComment** (p. ??), **TiXmlElement** (p. ??), and **TiXmlAttribute** (p. ??).

10.29.4.15 `const char * TiXmlBase::ReadName (const char * p, TIXML_STRING * name, TiXmlEncoding encoding)`
`[static], [protected]`

Definition at line 382 of file `tinyxmlparser.cpp`.

10.29.4.16 `const char * TiXmlBase::ReadText (const char * in, TIXML_STRING * text, bool ignoreWhiteSpace, const char * endTag, bool ignoreCase, TiXmlEncoding encoding)` `[static], [protected]`

Definition at line 555 of file `tinyxmlparser.cpp`.

10.29.4.17 `int TiXmlBase::Row () const [inline]`

Return the position, in the original source file, of this node or attribute. The row and column are 1-based. (That is the first row and first column is 1,1). If the returns values are 0 or less, then the parser does not have a row and column value.

Generally, the row and column value will be set when the `TiXmlDocument::Load()`, `TiXmlDocument::LoadFile()` (p. ??), or any `TiXmlNode::Parse()` (p. ??) is called. It will NOT be set when the DOM was created from operator>>.

The values reflect the initial load. Once the DOM is modified programmatically (by adding or changing nodes and attributes) the new values will NOT update to reflect changes in the document.

There is a minor performance cost to computing the row and column. Computation can be disabled if `TiXmlDocument::SetTabSize()` (p. ??) is called with 0 as the value.

See also

`TiXmlDocument::SetTabSize()` (p. ??)

Definition at line 220 of file `tinyxml.h`.

10.29.4.18 `static void TiXmlBase::SetCondenseWhiteSpace (bool condense) [inline],[static]`

The world does not agree on whether white space should be kept or not. In order to make everyone happy, these global, static functions are provided to set whether or not TinyXml will condense all white space into a single space or not. The default is to condense. Note changing this value is not thread safe.

Definition at line 197 of file `tinyxml.h`.

10.29.4.19 `void TiXmlBase::SetUserData (void * user) [inline]`

Set a pointer to arbitrary user data.

Definition at line 223 of file `tinyxml.h`.

10.29.4.20 `const char * TiXmlBase::SkipWhiteSpace (const char * p, TiXmlEncoding encoding) [static],
[protected]`

Definition at line 295 of file `tinyxmlparser.cpp`.

10.29.4.21 `bool TiXmlBase::StringEqual (const char * p, const char * endTag, bool ignoreCase, TiXmlEncoding encoding) [static],[protected]`

Definition at line 515 of file `tinyxmlparser.cpp`.

10.29.4.22 `static int TiXmlBase::ToLower (int v, TiXmlEncoding encoding) [inline],[static],
[protected]`

Definition at line 358 of file `tinyxml.h`.

10.29.5 Friends And Related Function Documentation

10.29.5.1 friend class TiXmlDocument [friend]

Definition at line 174 of file tinyxml.h.

10.29.5.2 friend class TiXmlElement [friend]

Definition at line 173 of file tinyxml.h.

10.29.5.3 friend class TiXmlNode [friend]

Definition at line 172 of file tinyxml.h.

10.29.6 Member Data Documentation

10.29.6.1 const char * TiXmlBase::errorString [static],[protected]

Initial value:

```
=
{
    "No error",
    "Error",
    "Failed to open file",
    "Error parsing Element.",
    "Failed to read Element name",
    "Error reading Element value.",
    "Error reading Attributes.",
    "Error: empty tag.",
    "Error reading end tag.",
    "Error parsing Unknown.",
    "Error parsing Comment.",
    "Error parsing Declaration.",
    "Error document empty.",
    "Error null (0) or unexpected EOF found in input stream.",
    "Error parsing CDATA.",
    "Error when TiXmlDocument added to document, because TiXmlDocument can only be at the root.",
}
```

Definition at line 347 of file tinyxml.h.

10.29.6.2 TiXmlCursor TiXmlBase::location [protected]

Definition at line 349 of file tinyxml.h.

10.29.6.3 void* TiXmlBase::userData [protected]

Field containing a generic user pointer.

Definition at line 352 of file tinyxml.h.

- virtual `~TiXmlComment()`
- virtual `TiXmlNode * Clone()` const
Returns a copy of this Comment.
- virtual void `Print` (FILE *cfile, int depth) const
- virtual const char * `Parse` (const char *p, `TiXmlParsingData` *data, `TiXmlEncoding` encoding)
- virtual const `TiXmlComment * ToComment()` const
Cast to a more defined type. Will return null not of the requested type.
- virtual `TiXmlComment * ToComment()`
Cast to a more defined type. Will return null not of the requested type.
- virtual bool `Accept` (`TiXmlVisitor` *visitor) const

Protected Member Functions

- void `CopyTo` (`TiXmlComment` *target) const

Additional Inherited Members

10.30.1 Detailed Description

An XML comment.

Definition at line 1138 of file `tinyxml.h`.

10.30.2 Constructor & Destructor Documentation

10.30.2.1 `TiXmlComment::TiXmlComment()` `[inline]`

Constructs an empty comment.

Definition at line 1142 of file `tinyxml.h`.

10.30.2.2 `TiXmlComment::TiXmlComment(const char *_value)` `[inline]`

Construct a comment from text.

Definition at line 1144 of file `tinyxml.h`.

10.30.2.3 `TiXmlComment::TiXmlComment(const TiXmlComment ©)`

Definition at line 1260 of file `tinyxml.cpp`.

10.30.2.4 `virtual TiXmlComment::~~TiXmlComment()` `[inline]`, `[virtual]`

Definition at line 1150 of file `tinyxml.h`.

10.30.3 Member Function Documentation

10.30.3.1 `bool TiXmlComment::Accept (TiXmlVisitor * visitor) const` `[virtual]`

Walk the XML tree visiting this node and all of its children.

Implements **TiXmlNode** (p. ??).

Definition at line 1291 of file `tinyxml.cpp`.

10.30.3.2 `TiXmlNode * TiXmlComment::Clone () const` `[virtual]`

Returns a copy of this Comment.

Implements **TiXmlNode** (p. ??).

Definition at line 1297 of file `tinyxml.cpp`.

10.30.3.3 `void TiXmlComment::CopyTo (TiXmlComment * target) const` `[protected]`

Definition at line 1285 of file `tinyxml.cpp`.

10.30.3.4 `TiXmlComment & TiXmlComment::operator=(const TiXmlComment & base)`

Definition at line 1266 of file `tinyxml.cpp`.

10.30.3.5 `const char * TiXmlComment::Parse (const char * p, TiXmlParsingData * data, TiXmlEncoding encoding)`
`[virtual]`

Implements **TiXmlBase** (p. ??).

Definition at line 1318 of file `tinyxmlparser.cpp`.

10.30.3.6 `void TiXmlComment::Print (FILE * cfile, int depth) const` `[virtual]`

All TinyXml classes can print themselves to a filestream or the string class (**TiXmlString** (p. ??) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements **TiXmlBase** (p. ??).

Definition at line 1274 of file `tinyxml.cpp`.

10.30.3.7 `virtual const TiXmlComment* TiXmlComment::ToComment () const` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1162 of file `tinyxml.h`.

10.30.3.8 `virtual TiXmlComment* TiXmlComment::ToComment ()` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1163 of file `tinyxml.h`.

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

- `DataHandling/tinyxml.h`
- `DataHandling/tinyxml.cpp`
- `DataHandling/tinyxmlparser.cpp`

10.31 TiXmlCursor Struct Reference

```
#include <tinyxml.h>
```

Public Member Functions

- **TiXmlCursor** ()
- void **Clear** ()

Public Attributes

- int **row**
- int **col**

10.31.1 Detailed Description

Definition at line 75 of file `tinyxml.h`.

10.31.2 Constructor & Destructor Documentation

10.31.2.1 `TiXmlCursor::TiXmlCursor ()` `[inline]`

Definition at line 77 of file `tinyxml.h`.

10.31.3 Member Function Documentation

10.31.3.1 void TiXmlCursor::Clear () [inline]

Definition at line 78 of file tinyxml.h.

10.31.4 Member Data Documentation

10.31.4.1 int TiXmlCursor::col

Definition at line 81 of file tinyxml.h.

10.31.4.2 int TiXmlCursor::row

Definition at line 80 of file tinyxml.h.

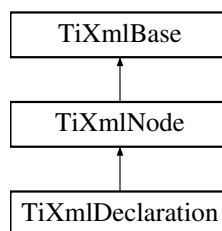
The documentation for this struct was generated from the following file:

- DataHandling/**tinyxml.h**

10.32 TiXmlDeclaration Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlDeclaration:



Public Member Functions

- **TiXmlDeclaration** ()
Construct an empty declaration.
- **TiXmlDeclaration** (const char *_version, const char *_encoding, const char *_standalone)
Construct.
- **TiXmlDeclaration** (const **TiXmlDeclaration** ©)
- **TiXmlDeclaration** & **operator=** (const **TiXmlDeclaration** ©)
- virtual ~**TiXmlDeclaration** ()
- const char * **Version** () const
Version. Will return an empty string if none was found.
- const char * **Encoding** () const

- Encoding. Will return an empty string if none was found.*
- const char * **Standalone** () const
Is this a standalone document?
- virtual TiXmlNode * **Clone** () const
Creates a copy of this Declaration and returns it.
- virtual void **Print** (FILE *cfile, int depth, TIXML_STRING *str) const
- virtual void **Print** (FILE *cfile, int depth) const
- virtual const char * **Parse** (const char *p, TiXmlParsingData *data, TiXmlEncoding encoding)
- virtual const TiXmlDeclaration * **ToDeclaration** () const
Cast to a more defined type. Will return null not of the requested type.
- virtual TiXmlDeclaration * **ToDeclaration** ()
Cast to a more defined type. Will return null not of the requested type.
- virtual bool **Accept** (TiXmlVisitor *visitor) const

Protected Member Functions

- void **CopyTo** (TiXmlDeclaration *target) const

Additional Inherited Members

10.32.1 Detailed Description

In correct XML the declaration is the first entry in the file.

```
<?xml version="1.0" standalone="yes"?>
```

TinyXml will happily read or write files without a declaration, however. There are 3 possible attributes to the declaration: version, encoding, and standalone.

Note: In this version of the code, the attributes are handled as special cases, not generic attributes, simply because there can only be at most 3 and they are always the same.

Definition at line 1261 of file tinyxml.h.

10.32.2 Constructor & Destructor Documentation

10.32.2.1 TiXmlDeclaration::TiXmlDeclaration () [inline]

Construct an empty declaration.

Definition at line 1265 of file tinyxml.h.

10.32.2.2 TiXmlDeclaration::TiXmlDeclaration (const char * _version, const char * _encoding, const char * _standalone)

Construct.

Definition at line 1356 of file tinyxml.cpp.

10.32.2.3 TiXmlDeclaration::TiXmlDeclaration (const TiXmlDeclaration & *copy*)

Definition at line 1380 of file tinyxml.cpp.

10.32.2.4 virtual TiXmlDeclaration::~~TiXmlDeclaration () [inline], [virtual]

Definition at line 1282 of file tinyxml.h.

10.32.3 Member Function Documentation

10.32.3.1 bool TiXmlDeclaration::Accept (TiXmlVisitor * *visitor*) const [virtual]

Walk the XML tree visiting this node and all of its children.

Implements **TiXmlNode** (p. ??).

Definition at line 1427 of file tinyxml.cpp.

10.32.3.2 TiXmlNode * TiXmlDeclaration::Clone () const [virtual]

Creates a copy of this Declaration and returns it.

Implements **TiXmlNode** (p. ??).

Definition at line 1433 of file tinyxml.cpp.

10.32.3.3 void TiXmlDeclaration::CopyTo (TiXmlDeclaration * *target*) const [protected]

Definition at line 1417 of file tinyxml.cpp.

10.32.3.4 const char* TiXmlDeclaration::Encoding () const [inline]

Encoding. Will return an empty string if none was found.

Definition at line 1287 of file tinyxml.h.

10.32.3.5 TiXmlDeclaration & TiXmlDeclaration::operator= (const TiXmlDeclaration & *copy*)

Definition at line 1387 of file tinyxml.cpp.

10.32.3.6 const char * TiXmlDeclaration::Parse (const char * *p*, TiXmlParsingData * *data*, TiXmlEncoding *encoding*) [virtual]

Implements **TiXmlBase** (p. ??).

Definition at line 1553 of file tinyxmlparser.cpp.

10.32.3.7 `void TiXmlDeclaration::Print (FILE * cfile, int depth, TIXML_STRING * str) const` `[virtual]`

Definition at line 1395 of file `tinyxml.cpp`.

10.32.3.8 `virtual void TiXmlDeclaration::Print (FILE * cfile, int depth) const` `[inline],[virtual]`

All TinyXml classes can print themselves to a filestream or the string class (**TiXmlString** (p. ??) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements **TiXmlBase** (p. ??).

Definition at line 1295 of file `tinyxml.h`.

10.32.3.9 `const char* TiXmlDeclaration::Standalone () const` `[inline]`

Is this a standalone document?

Definition at line 1289 of file `tinyxml.h`.

10.32.3.10 `virtual const TiXmlDeclaration* TiXmlDeclaration::ToDeclaration () const` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1301 of file `tinyxml.h`.

10.32.3.11 `virtual TiXmlDeclaration* TiXmlDeclaration::ToDeclaration ()` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1302 of file `tinyxml.h`.

10.32.3.12 `const char* TiXmlDeclaration::Version () const` `[inline]`

Version. Will return an empty string if none was found.

Definition at line 1285 of file `tinyxml.h`.

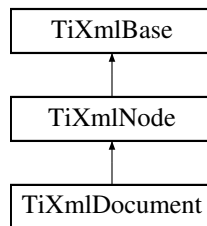
The documentation for this class was generated from the following files:

- `DataHandling/tinyxml.h`
- `DataHandling/tinyxml.cpp`
- `DataHandling/tinyxmlparser.cpp`

10.33 TiXmlDocument Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlDocument:



Public Member Functions

- **TiXmlDocument** ()
Create an empty document, that has no name.
- **TiXmlDocument** (const char *documentName)
Create a document with a name. The name of the document is also the filename of the xml.
- **TiXmlDocument** (const **TiXmlDocument** ©)
- **TiXmlDocument** & **operator=** (const **TiXmlDocument** ©)
- virtual ~**TiXmlDocument** ()
- bool **LoadFile** (**TiXmlEncoding** encoding=**TIXML_DEFAULT_ENCODING**)
- bool **SaveFile** () const
Save a file using the current document value. Returns true if successful.
- bool **LoadFile** (const char *filename, **TiXmlEncoding** encoding=**TIXML_DEFAULT_ENCODING**)
Load a file using the given filename. Returns true if successful.
- bool **SaveFile** (const char *filename) const
Save a file using the given filename. Returns true if successful.
- bool **LoadFile** (FILE *, **TiXmlEncoding** encoding=**TIXML_DEFAULT_ENCODING**)
- bool **SaveFile** (FILE *) const
Save a file using the given FILE. Returns true if successful.*
- virtual const char * **Parse** (const char *p, **TiXmlParsingData** *data=0, **TiXmlEncoding** encoding=**TIXML_DEFAULT_ENCODING**)
- const **TiXmlElement** * **RootElement** () const
- **TiXmlElement** * **RootElement** ()
- bool **Error** () const
- const char * **ErrorDesc** () const
Contains a textual (english) description of the error if one occurs.
- int **ErrorId** () const
- int **ErrorRow** () const
- int **ErrorCol** () const
*The column where the error occurred. See (p. ??) **ErrorRow()** (p. ??)*
- void **SetTabSize** (int _tabsize)
- int **TabSize** () const
- void **ClearError** ()
- void **Print** () const
- virtual void **Print** (FILE *cfile, int depth=0) const
Print this Document to a FILE stream.
- void **SetError** (int err, const char *errorLocation, **TiXmlParsingData** *prevData, **TiXmlEncoding** encoding)
- virtual const **TiXmlDocument** * **ToDocument** () const
Cast to a more defined type. Will return null not of the requested type.
- virtual **TiXmlDocument** * **ToDocument** ()
Cast to a more defined type. Will return null not of the requested type.
- virtual bool **Accept** (**TiXmlVisitor** *content) const

Protected Member Functions

- virtual **TiXmlNode** * **Clone** () const

Additional Inherited Members

10.33.1 Detailed Description

Always the top level node. A document binds together all the XML pieces. It can be saved, loaded, and printed to the screen. The 'value' of a document node is the xml file name.

Definition at line 1369 of file tinyxml.h.

10.33.2 Constructor & Destructor Documentation

10.33.2.1 TiXmlDocument::TiXmlDocument ()

Create an empty document, that has no name.

Definition at line 890 of file tinyxml.cpp.

10.33.2.2 TiXmlDocument::TiXmlDocument (const char * *documentName*)

Create a document with a name. The name of the document is also the filename of the xml.

Definition at line 897 of file tinyxml.cpp.

10.33.2.3 TiXmlDocument::TiXmlDocument (const TiXmlDocument & *copy*)

Definition at line 917 of file tinyxml.cpp.

10.33.2.4 virtual TiXmlDocument::~TiXmlDocument () [inline], [virtual]

Definition at line 1385 of file tinyxml.h.

10.33.3 Member Function Documentation

10.33.3.1 bool TiXmlDocument::Accept (TiXmlVisitor * *content*) const [virtual]

Walk the XML tree visiting this node and all of its children.

Implements **TiXmlNode** (p. ??).

Definition at line 1133 of file tinyxml.cpp.

10.33.3.2 void TiXmlDocument::ClearError () [inline]

If you have handled the error, it can be reset with this call. The error state is automatically cleared if you Parse a new XML block.

Definition at line 1487 of file tinyxml.h.

10.33.3.3 TiXmlNode * TiXmlDocument::Clone () const [protected],[virtual]

Create an exact duplicate of this node and return it. The memory must be deleted by the caller.

Implements **TiXmlNode** (p. ??).

Definition at line 1111 of file tinyxml.cpp.

10.33.3.4 bool TiXmlDocument::Error () const [inline]

If an error occurs, Error will be set to true. Also,

- The **ErrorId()** (p. ??) will contain the integer identifier of the error (not generally useful)
- The **ErrorDesc()** (p. ??) method will return the name of the error. (very useful)
- The **ErrorRow()** (p. ??) and **ErrorCol()** (p. ??) will return the location of the error (if known)

Definition at line 1436 of file tinyxml.h.

10.33.3.5 int TiXmlDocument::ErrorCol () const [inline]

The column where the error occurred. **See** (p. ??) **ErrorRow()** (p. ??)

Definition at line 1454 of file tinyxml.h.

10.33.3.6 const char* TiXmlDocument::ErrorDesc () const [inline]

Contains a textual (english) description of the error if one occurs.

Definition at line 1439 of file tinyxml.h.

10.33.3.7 int TiXmlDocument::ErrorId () const [inline]

Generally, you probably want the error string (**ErrorDesc()** (p. ??)). But if you prefer the ErrorId, this function will fetch it.

Definition at line 1444 of file tinyxml.h.

10.33.3.8 int TiXmlDocument::ErrorRow () const [inline]

Returns the location (if known) of the error. The first column is column 1, and the first row is row 1. A value of 0 means the row and column wasn't applicable (memory errors, for example, have no row/column) or the parser lost the error. (An error in the error reporting, in that case.)

See also

SetTabSize (p. ??), **Row** (p. ??), **Column** (p. ??)

Definition at line 1453 of file tinyxml.h.

10.33.3.9 bool TiXmlDocument::LoadFile (TiXmlEncoding encoding = TIXML_DEFAULT_ENCODING)

Load a file using the current document value. Returns true if successful. Will delete any existing document data before loading.

Definition at line 931 of file tinyxml.cpp.

10.33.3.10 bool TiXmlDocument::LoadFile (const char * filename, TiXmlEncoding encoding = TIXML_DEFAULT_ENCODING)

Load a file using the given filename. Returns true if successful.

Definition at line 942 of file tinyxml.cpp.

10.33.3.11 bool TiXmlDocument::LoadFile (FILE * file, TiXmlEncoding encoding = TIXML_DEFAULT_ENCODING)

Load a file using the given FILE*. Returns true if successful. Note that this method doesn't stream - the entire object pointed at by the FILE* will be interpreted as an XML file. TinyXML doesn't stream in XML from the current file location. Streaming may be added in the future.

Definition at line 963 of file tinyxml.cpp.

10.33.3.12 TiXmlDocument & TiXmlDocument::operator= (const TiXmlDocument & copy)

Definition at line 923 of file tinyxml.cpp.

10.33.3.13 const char * TiXmlDocument::Parse (const char * p, TiXmlParsingData * data = 0, TiXmlEncoding encoding = TIXML_DEFAULT_ENCODING) [virtual]

Parse the given null terminated block of xml data. Passing in an encoding to this method (either TIXML_ENCODING_LEGACY or TIXML_ENCODING_UTF8 will force TinyXml to use that encoding, regardless of what TinyXml might otherwise try to detect.

Implements **TiXmlBase** (p. ??).

Definition at line 685 of file tinyxmlparser.cpp.

10.33.3.14 `void TiXmlDocument::Print () const [inline]`

Write the document to standard out using formatted printing ("pretty print").

Definition at line 1496 of file `tinyxml.h`.

10.33.3.15 `void TiXmlDocument::Print (FILE * cfile, int depth = 0) const [virtual]`

Print this Document to a FILE stream.

Implements **TiXmlBase** (p. ??).

Definition at line 1122 of file `tinyxml.cpp`.

10.33.3.16 `const TiXmlElement* TiXmlDocument::RootElement () const [inline]`

Get the root element – the only top level element – of the document. In well formed XML, there should only be one. TinyXml is tolerant of multiple elements at the document level.

Definition at line 1428 of file `tinyxml.h`.

10.33.3.17 `TiXmlElement* TiXmlDocument::RootElement () [inline]`

Definition at line 1429 of file `tinyxml.h`.

10.33.3.18 `bool TiXmlDocument::SaveFile () const`

Save a file using the current document value. Returns true if successful.

Definition at line 937 of file `tinyxml.cpp`.

10.33.3.19 `bool TiXmlDocument::SaveFile (const char * filename) const`

Save a file using the given filename. Returns true if successful.

Definition at line 1061 of file `tinyxml.cpp`.

10.33.3.20 `bool TiXmlDocument::SaveFile (FILE * fp) const`

Save a file using the given FILE*. Returns true if successful.

Definition at line 1075 of file `tinyxml.cpp`.

10.33.3.21 `void TiXmlDocument::SetError (int err, const char * errorLocation, TiXmlParsingData * prevData, TiXmlEncoding encoding)`

Definition at line 779 of file `tinyxmlparser.cpp`.

10.33.3.22 void TiXmlDocument::SetTabSize (int *tabsize*) [inline]

SetTabSize() (p. ??) allows the error reporting functions (**ErrorRow()** (p. ??) and **ErrorCol()** (p. ??)) to report the correct values for row and column. It does not change the output or input in any way.

By calling this method, with a tab size greater than 0, the row and column of each node and attribute is stored when the file is loaded. Very useful for tracking the DOM back in to the source file.

The tab size is required for calculating the location of nodes. If not set, the default of 4 is used. The tabsize is set per document. Setting the tabsize to 0 disables row/column tracking.

Note that row and column tracking is not supported when using operator>>.

The tab size needs to be enabled before the parse or load. Correct usage:

```
TiXmlDocument doc;
doc.SetTabSize( 8 );
doc.Load( "myfile.xml" );
```

See also

Row (p. ??), **Column** (p. ??)

Definition at line 1480 of file tinyxml.h.

10.33.3.23 int TiXmlDocument::TabSize () const [inline]

Definition at line 1482 of file tinyxml.h.

10.33.3.24 virtual const TiXmlDocument* TiXmlDocument::ToDocument () const [inline], [virtual]

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1509 of file tinyxml.h.

10.33.3.25 virtual TiXmlDocument* TiXmlDocument::ToDocument () [inline], [virtual]

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1510 of file tinyxml.h.

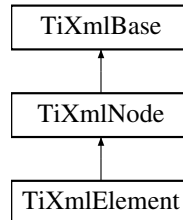
The documentation for this class was generated from the following files:

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**
- DataHandling/**tinyxmlparser.cpp**

10.34 TiXmlElement Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlElement:



Public Member Functions

- **TiXmlElement** (const char *in_value)
Construct an element.
- **TiXmlElement** (const **TiXmlElement** &)
- **TiXmlElement** & **operator=** (const **TiXmlElement** &base)
- virtual ~**TiXmlElement** ()
- const char * **Attribute** (const char *name) const
- const char * **Attribute** (const char *name, int *i) const
- const char * **Attribute** (const char *name, double *d) const
- int **QueryIntAttribute** (const char *name, int *_value) const
- int **QueryUnsignedAttribute** (const char *name, unsigned *_value) const
*QueryUnsignedAttribute examines the attribute - see **QueryIntAttribute**() (p. ??).*
- int **QueryBoolAttribute** (const char *name, bool *_value) const
- int **QueryDoubleAttribute** (const char *name, double *_value) const
*QueryDoubleAttribute examines the attribute - see **QueryIntAttribute**() (p. ??).*
- int **QueryFloatAttribute** (const char *name, float *_value) const
*QueryFloatAttribute examines the attribute - see **QueryIntAttribute**() (p. ??).*
- void **SetAttribute** (const char *name, const char *_value)
- void **SetAttribute** (const char *name, int value)
- void **SetDoubleAttribute** (const char *name, double value)
- void **RemoveAttribute** (const char *name)
- const **TiXmlAttribute** * **FirstAttribute** () const
Access the first attribute in this element.
- **TiXmlAttribute** * **FirstAttribute** ()
- const **TiXmlAttribute** * **LastAttribute** () const
Access the last attribute in this element.
- **TiXmlAttribute** * **LastAttribute** ()
- const char * **GetText** () const
- virtual **TiXmlNode** * **Clone** () const
Creates a new Element and returns it - the returned element is a copy.
- virtual void **Print** (FILE *cfile, int depth) const
- virtual const char * **Parse** (const char *p, **TiXmlParsingData** *data, **TiXmlEncoding** encoding)
- virtual const **TiXmlElement** * **ToElement** () const
Cast to a more defined type. Will return null not of the requested type.
- virtual **TiXmlElement** * **ToElement** ()
Cast to a more defined type. Will return null not of the requested type.
- virtual bool **Accept** (**TiXmlVisitor** *visitor) const

Protected Member Functions

- void **CopyTo** (**TiXmlElement** *target) const
- void **ClearThis** ()
- const char * **ReadValue** (const char *in, **TiXmlParsingData** *prevData, **TiXmlEncoding** encoding)

Additional Inherited Members

10.34.1 Detailed Description

The element is a container class. It has a value, the element name, and can contain other elements, text, comments, and unknowns. Elements also contain an arbitrary number of attributes.

Definition at line 916 of file tinyxml.h.

10.34.2 Constructor & Destructor Documentation

10.34.2.1 **TiXmlElement::TiXmlElement** (const char * *in_value*)

Construct an element.

Definition at line 502 of file tinyxml.cpp.

10.34.2.2 **TiXmlElement::TiXmlElement** (const **TiXmlElement** & *copy*)

Definition at line 520 of file tinyxml.cpp.

10.34.2.3 **TiXmlElement::~~TiXmlElement** () [virtual]

Definition at line 536 of file tinyxml.cpp.

10.34.3 Member Function Documentation

10.34.3.1 **bool TiXmlElement::Accept** (**TiXmlVisitor** * *visitor*) const [virtual]

Walk the XML tree visiting this node and all of its children.

Implements **TiXmlNode** (p. ??).

Definition at line 852 of file tinyxml.cpp.

10.34.3.2 **const char * TiXmlElement::Attribute** (const char * *name*) const

Given an attribute name, **Attribute()** (p. ??) returns the value for the attribute of that name, or null if none exists.

Definition at line 554 of file tinyxml.cpp.

10.34.3.3 `const char * TiXmlElement::Attribute (const char * name, int * i) const`

Given an attribute name, **Attribute()** (p. ??) returns the value for the attribute of that name, or null if none exists. If the attribute exists and can be converted to an integer, the integer value will be put in the return 'i', if 'i' is non-null.

Definition at line 574 of file tinyxml.cpp.

10.34.3.4 `const char * TiXmlElement::Attribute (const char * name, double * d) const`

Given an attribute name, **Attribute()** (p. ??) returns the value for the attribute of that name, or null if none exists. If the attribute exists and can be converted to a double, the double value will be put in the return 'd', if 'd' is non-null.

Definition at line 606 of file tinyxml.cpp.

10.34.3.5 `void TiXmlElement::ClearThis () [protected]`

Definition at line 542 of file tinyxml.cpp.

10.34.3.6 `TiXmlNode * TiXmlElement::Clone () const [virtual]`

Creates a new Element and returns it - the returned element is a copy.

Implements **TiXmlNode** (p. ??).

Definition at line 866 of file tinyxml.cpp.

10.34.3.7 `void TiXmlElement::CopyTo (TiXmlElement * target) const [protected]`

Definition at line 830 of file tinyxml.cpp.

10.34.3.8 `const TiXmlAttribute* TiXmlElement::FirstAttribute () const [inline]`

Access the first attribute in this element.

Definition at line 1060 of file tinyxml.h.

10.34.3.9 `TiXmlAttribute* TiXmlElement::FirstAttribute () [inline]`

Definition at line 1061 of file tinyxml.h.

10.34.3.10 `const char * TiXmlElement::GetText () const`

Convenience function for easy access to the text inside an element. Although easy and concise, **GetText()** (p. ??) is limited compared to getting the **TiXmlText** (p. ??) child and accessing it directly.

If the first child of 'this' is a **TiXmlText** (p. ??), the **GetText()** (p. ??) returns the character string of the Text node, else null is returned.

This is a convenient method for getting the text of simple contained text:

```
<foo>This is text</foo>
const char* str = fooElement->GetText();
```

'str' will be a pointer to "This is text".

Note that this function can be misleading. If the element foo was created from this XML:

```
<foo><b>This is text</b></foo>
```

then the value of str would be null. The first child node isn't a text node, it is another element. From this XML:

```
<foo>This is <b>text</b></foo>
```

GetText() (p. ??) will return "This is ".

WARNING: **GetText()** (p. ??) accesses a child node - don't become confused with the similarly named **TiXmlHandle::Text()** (p. ??) and **TiXmlNode::ToText()** (p. ??) which are safe type casts on the referenced node.

Definition at line 877 of file tinyxml.cpp.

10.34.3.11 `const TiXmlAttribute* TiXmlElement::LastAttribute () const` [inline]

Access the last attribute in this element.

Definition at line 1062 of file tinyxml.h.

10.34.3.12 `TiXmlAttribute* TiXmlElement::LastAttribute ()` [inline]

Definition at line 1063 of file tinyxml.h.

10.34.3.13 `TiXmlElement & TiXmlElement::operator= (const TiXmlElement & base)`

Definition at line 528 of file tinyxml.cpp.

10.34.3.14 `const char * TiXmlElement::Parse (const char * p, TiXmlParsingData * data, TiXmlEncoding encoding)` [virtual]

Implements **TiXmlBase** (p. ??).

Definition at line 1024 of file tinyxmlparser.cpp.

10.34.3.15 void TiXmlElement::Print (FILE * *cfile*, int *depth*) const [virtual]

All TinyXml classes can print themselves to a filestream or the string class (**TiXmlString** (p. ??) in non-STL mode, std::string in STL mode.) Either or both cfile and str can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the << operator.)

Implements **TiXmlBase** (p. ??).

Definition at line 777 of file tinyxml.cpp.

10.34.3.16 int TiXmlElement::QueryBoolAttribute (const char * *name*, bool * *_value*) const

QueryBoolAttribute examines the attribute - see **QueryIntAttribute()** (p. ??). Note that '1', 'true', or 'yes' are considered true, while '0', 'false' and 'no' are considered false.

Definition at line 660 of file tinyxml.cpp.

10.34.3.17 int TiXmlElement::QueryDoubleAttribute (const char * *name*, double * *_value*) const

QueryDoubleAttribute examines the attribute - see **QueryIntAttribute()** (p. ??).

Definition at line 697 of file tinyxml.cpp.

10.34.3.18 int TiXmlElement::QueryFloatAttribute (const char * *name*, float * *_value*) const [inline]

QueryFloatAttribute examines the attribute - see **QueryIntAttribute()** (p. ??).

Definition at line 972 of file tinyxml.h.

10.34.3.19 int TiXmlElement::QueryIntAttribute (const char * *name*, int * *_value*) const

QueryIntAttribute examines the attribute - it is an alternative to the **Attribute()** (p. ??) method with richer error checking. If the attribute is an integer, it is stored in 'value' and the call returns TIXML_SUCCESS. If it is not an integer, it returns TIXML_WRONG_TYPE. If the attribute does not exist, then TIXML_NO_ATTRIBUTE is returned.

Definition at line 638 of file tinyxml.cpp.

10.34.3.20 int TiXmlElement::QueryUnsignedAttribute (const char * *name*, unsigned * *_value*) const

QueryUnsignedAttribute examines the attribute - see **QueryIntAttribute()** (p. ??).

Definition at line 647 of file tinyxml.cpp.

10.34.3.21 `const char * TiXmlElement::ReadValue (const char * in, TiXmlParsingData * prevData, TiXmlEncoding encoding)` `[protected]`

Definition at line 1160 of file `tinyxmlparser.cpp`.

10.34.3.22 `void TiXmlElement::RemoveAttribute (const char * name)`

Deletes an attribute with the given name.

Definition at line 414 of file `tinyxml.cpp`.

10.34.3.23 `void TiXmlElement::SetAttribute (const char * name, const char * _value)`

Sets an attribute of name to a given value. The attribute will be created if it does not exist, or changed if it does.

Definition at line 757 of file `tinyxml.cpp`.

10.34.3.24 `void TiXmlElement::SetAttribute (const char * name, int value)`

Sets an attribute of name to a given value. The attribute will be created if it does not exist, or changed if it does.

Definition at line 717 of file `tinyxml.cpp`.

10.34.3.25 `void TiXmlElement::SetDoubleAttribute (const char * name, double value)`

Sets an attribute of name to a given value. The attribute will be created if it does not exist, or changed if it does.

Definition at line 737 of file `tinyxml.cpp`.

10.34.3.26 `virtual const TiXmlElement* TiXmlElement::ToElement () const` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1109 of file `tinyxml.h`.

10.34.3.27 `virtual TiXmlElement* TiXmlElement::ToElement ()` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1110 of file `tinyxml.h`.

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

- `DataHandling/tinyxml.h`
- `DataHandling/tinyxml.cpp`
- `DataHandling/tinyxmlparser.cpp`

10.35 TiXmlHandle Class Reference

```
#include <tinyxml.h>
```

Public Member Functions

- **TiXmlHandle** (**TiXmlNode** *_node)
Create a handle from any node (at any depth of the tree.) This can be a null pointer.
- **TiXmlHandle** (const **TiXmlHandle** &ref)
Copy constructor.
- **TiXmlHandle operator=** (const **TiXmlHandle** &ref)
- **TiXmlHandle FirstChild** () const
Return a handle to the first child node.
- **TiXmlHandle FirstChild** (const char *value) const
Return a handle to the first child node with the given name.
- **TiXmlHandle FirstChildElement** () const
Return a handle to the first child element.
- **TiXmlHandle FirstChildElement** (const char *value) const
Return a handle to the first child element with the given name.
- **TiXmlHandle Child** (const char *value, int index) const
- **TiXmlHandle Child** (int index) const
- **TiXmlHandle ChildElement** (const char *value, int index) const
- **TiXmlHandle ChildElement** (int index) const
- **TiXmlNode * ToNode** () const
- **TiXmlElement * ToElement** () const
- **TiXmlText * ToText** () const
- **TiXmlUnknown * ToUnknown** () const
- **TiXmlNode * Node** () const
- **TiXmlElement * Element** () const
- **TiXmlText * Text** () const
- **TiXmlUnknown * Unknown** () const

10.35.1 Detailed Description

A **TiXmlHandle** (p. ??) is a class that wraps a node pointer with null checks; this is an incredibly useful thing. Note that **TiXmlHandle** (p. ??) is not part of the TinyXml DOM structure. It is a separate utility class.

Take an example:

```
<Document>
<Element attributeA = "valueA">
<Child attributeB = "value1" />
<Child attributeB = "value2" />
</Element>
</Document>
```

Assuming you want the value of "attributeB" in the 2nd "Child" element, it's very easy to write a *lot* of code that looks like:

```

TiXmlElement* root = document.FirstChildElement( "Document" );
if ( root )
{
    TiXmlElement* element = root->FirstChildElement( "Element" );
    if ( element )
    {
        TiXmlElement* child = element->FirstChildElement( "Child" );
        if ( child )
        {
            TiXmlElement* child2 = child->NextSiblingElement( "Child" );
            if ( child2 )
            {
                // Finally do something useful.
            }
        }
    }
}

```

And that doesn't even cover "else" cases. **TiXmlHandle** (p. ??) addresses the verbosity of such code. A **TiXmlHandle** (p. ??) checks for null pointers so it is perfectly safe and correct to use:

```

TiXmlHandle docHandle( &document );
TiXmlElement* child2 = docHandle.FirstChild( "Document" ).FirstChild( "Element" ).Child( "Child", 1 ).ToElement();
if ( child2 )
{
    // do something useful
}

```

Which is MUCH more concise and useful.

It is also safe to copy handles - internally they are nothing more than node pointers.

```

TiXmlHandle handleCopy = handle;

```

What they should not be used for is iteration:

```

int i=0;
while ( true )
{
    TiXmlElement* child = docHandle.FirstChild( "Document" ).FirstChild( "Element" ).Child( "Child", i ).ToElement();
    if ( !child )
        break;
    // do something
    ++i;
}

```

It seems reasonable, but it is in fact two embedded while loops. The Child method is a linear walk to find the element, so this code would iterate much more than it needs to. Instead, prefer:

```

TiXmlElement* child = docHandle.FirstChild( "Document" ).FirstChild( "Element" ).FirstChild( "Child" ).ToElement();

for( child; child; child=child->NextSiblingElement() )
{
    // do something
}

```

Definition at line 1615 of file tinyxml.h.

10.35.2 Constructor & Destructor Documentation

10.35.2.1 TiXmlHandle::TiXmlHandle (TiXmlNode **_node*) [inline]

Create a handle from any node (at any depth of the tree.) This can be a null pointer.

Definition at line 1619 of file tinyxml.h.

10.35.2.2 TiXmlHandle::TiXmlHandle (const TiXmlHandle & ref) [inline]

Copy constructor.

Definition at line 1621 of file tinyxml.h.

10.35.3 Member Function Documentation

10.35.3.1 TiXmlHandle TiXmlHandle::Child (const char * value, int index) const

Return a handle to the "index" child with the given name. The first child is 0, the second 1, etc.

Definition at line 1676 of file tinyxml.cpp.

10.35.3.2 TiXmlHandle TiXmlHandle::Child (int index) const

Return a handle to the "index" child. The first child is 0, the second 1, etc.

Definition at line 1657 of file tinyxml.cpp.

10.35.3.3 TiXmlHandle TiXmlHandle::ChildElement (const char * value, int index) const

Return a handle to the "index" child element with the given name. The first child element is 0, the second 1, etc. Note that only TiXmlElements are indexed: other types are not counted.

Definition at line 1714 of file tinyxml.cpp.

10.35.3.4 TiXmlHandle TiXmlHandle::ChildElement (int index) const

Return a handle to the "index" child element. The first child element is 0, the second 1, etc. Note that only TiXmlElement↵s are indexed: other types are not counted.

Definition at line 1695 of file tinyxml.cpp.

10.35.3.5 TiXmlElement* TiXmlHandle::Element () const [inline]

Deprecated use ToElement. Return the handle as a **TiXmlElement** (p. ??). This may return null.

Definition at line 1680 of file tinyxml.h.

10.35.3.6 TiXmlHandle TiXmlHandle::FirstChild () const

Return a handle to the first child node.

Definition at line 1609 of file tinyxml.cpp.

10.35.3.7 **TiXmlHandle** TiXmlHandle::FirstChild (const char * *value*) const

Return a handle to the first child node with the given name.

Definition at line 1621 of file tinyxml.cpp.

10.35.3.8 **TiXmlHandle** TiXmlHandle::FirstChildElement () const

Return a handle to the first child element.

Definition at line 1633 of file tinyxml.cpp.

10.35.3.9 **TiXmlHandle** TiXmlHandle::FirstChildElement (const char * *value*) const

Return a handle to the first child element with the given name.

Definition at line 1645 of file tinyxml.cpp.

10.35.3.10 **TiXmlNode*** TiXmlHandle::Node () const [inline]

Deprecated use ToNode. Return the handle as a **TiXmlNode** (p. ??). This may return null.

Definition at line 1676 of file tinyxml.h.

10.35.3.11 **TiXmlHandle** TiXmlHandle::operator= (const **TiXmlHandle** & *ref*) [inline]

Definition at line 1622 of file tinyxml.h.

10.35.3.12 **TiXmlText*** TiXmlHandle::Text () const [inline]

Deprecated use **ToText()** (p. ??) Return the handle as a **TiXmlText** (p. ??). This may return null.

Definition at line 1684 of file tinyxml.h.

10.35.3.13 **TiXmlElement*** TiXmlHandle::ToElement () const [inline]

Return the handle as a **TiXmlElement** (p. ??). This may return null.

Definition at line 1665 of file tinyxml.h.

10.35.3.14 **TiXmlNode*** TiXmlHandle::ToNode () const [inline]

Return the handle as a **TiXmlNode** (p. ??). This may return null.

Definition at line 1662 of file tinyxml.h.

10.35.3.15 **TiXmlText*** TiXmlHandle::ToText () const [inline]

Return the handle as a **TiXmlText** (p. ??). This may return null.

Definition at line 1668 of file tinyxml.h.

10.35.3.16 **TiXmlUnknown*** TiXmlHandle::ToUnknown () const [inline]

Return the handle as a **TiXmlUnknown** (p. ??). This may return null.

Definition at line 1671 of file tinyxml.h.

10.35.3.17 **TiXmlUnknown*** TiXmlHandle::Unknown () const [inline]

Deprecated use **ToUnknown()** (p. ??) Return the handle as a **TiXmlUnknown** (p. ??). This may return null.

Definition at line 1688 of file tinyxml.h.

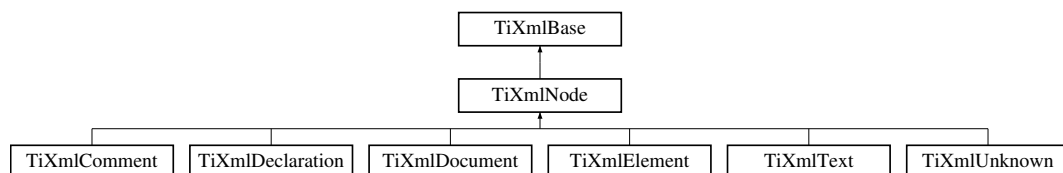
The documentation for this class was generated from the following files:

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**

10.36 TiXmlNode Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlNode:



Public Types

- enum **NodeType** {
TINYXML_DOCUMENT, **TINYXML_ELEMENT**, **TINYXML_COMMENT**, **TINYXML_UNKNOWN**,
TINYXML_TEXT, **TINYXML_DECLARATION**, **TINYXML_TYPECOUNT** }

Public Member Functions

- virtual **~TiXmlNode** ()
- const char * **Value** () const
- const **TIXML_STRING** & **ValueTStr** () const
- void **SetValue** (const char *_value)
- void **Clear** ()
 - Delete all the children of this node. Does not affect 'this'.*
- **TiXmlNode** * **Parent** ()
 - One step up the DOM.*
- const **TiXmlNode** * **Parent** () const
- const **TiXmlNode** * **FirstChild** () const
 - The first child of this node. Will be null if there are no children.*
- **TiXmlNode** * **FirstChild** ()
- const **TiXmlNode** * **FirstChild** (const char ***value**) const
- **TiXmlNode** * **FirstChild** (const char *_value)
 - The first child of this node with the matching 'value'. Will be null if none found.*
- const **TiXmlNode** * **LastChild** () const
- **TiXmlNode** * **LastChild** ()
 - The last child of this node. Will be null if there are no children.*
- const **TiXmlNode** * **LastChild** (const char ***value**) const
- **TiXmlNode** * **LastChild** (const char *_value)
 - The last child of this node matching 'value'. Will be null if there are no children.*
- const **TiXmlNode** * **IterateChildren** (const **TiXmlNode** *previous) const
- **TiXmlNode** * **IterateChildren** (const **TiXmlNode** *previous)
- const **TiXmlNode** * **IterateChildren** (const char ***value**, const **TiXmlNode** *previous) const
 - This flavor of IterateChildren searches for children with a particular 'value'.*
- **TiXmlNode** * **IterateChildren** (const char *_value, const **TiXmlNode** *previous)
- **TiXmlNode** * **InsertEndChild** (const **TiXmlNode** &addThis)
- **TiXmlNode** * **LinkEndChild** (**TiXmlNode** *addThis)
- **TiXmlNode** * **InsertBeforeChild** (**TiXmlNode** *beforeThis, const **TiXmlNode** &addThis)
- **TiXmlNode** * **InsertAfterChild** (**TiXmlNode** *afterThis, const **TiXmlNode** &addThis)
- **TiXmlNode** * **ReplaceChild** (**TiXmlNode** *replaceThis, const **TiXmlNode** &withThis)
- bool **RemoveChild** (**TiXmlNode** *removeThis)
 - Delete a child of this node.*
- const **TiXmlNode** * **PreviousSibling** () const
 - Navigate to a sibling node.*
- **TiXmlNode** * **PreviousSibling** ()
- const **TiXmlNode** * **PreviousSibling** (const char *) const
 - Navigate to a sibling node.*
- **TiXmlNode** * **PreviousSibling** (const char *_prev)
- const **TiXmlNode** * **NextSibling** () const
 - Navigate to a sibling node.*
- **TiXmlNode** * **NextSibling** ()
- const **TiXmlNode** * **NextSibling** (const char *) const
 - Navigate to a sibling node with the given 'value'.*
- **TiXmlNode** * **NextSibling** (const char *_next)
- const **TiXmlElement** * **NextSiblingElement** () const
- **TiXmlElement** * **NextSiblingElement** ()
- const **TiXmlElement** * **NextSiblingElement** (const char *) const
- **TiXmlElement** * **NextSiblingElement** (const char *_next)
- const **TiXmlElement** * **FirstChildElement** () const
 - Convenience function to get through elements.*

- **TiXmlElement * FirstChildElement ()**
- **const TiXmlElement * FirstChildElement (const char *_value) const**
Convenience function to get through elements.
- **TiXmlElement * FirstChildElement (const char *_value)**
- **int Type () const**
- **const TiXmlDocument * GetDocument () const**
- **TiXmlDocument * GetDocument ()**
- **bool NoChildren () const**
Returns true if this node has no children.
- **virtual const TiXmlDocument * ToDocument () const**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual const TiXmlElement * ToElement () const**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual const TiXmlComment * ToComment () const**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual const TiXmlUnknown * ToUnknown () const**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual const TiXmlText * ToText () const**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual const TiXmlDeclaration * ToDeclaration () const**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual TiXmlDocument * ToDocument ()**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual TiXmlElement * ToElement ()**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual TiXmlComment * ToComment ()**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual TiXmlUnknown * ToUnknown ()**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual TiXmlText * ToText ()**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual TiXmlDeclaration * ToDeclaration ()**
Cast to a more defined type. Will return null if not of the requested type.
- **virtual TiXmlNode * Clone () const =0**
- **virtual bool Accept (TiXmlVisitor *visitor) const =0**

Protected Member Functions

- **TiXmlNode (NodeType _type)**
- **void CopyTo (TiXmlNode *target) const**
- **TiXmlNode * Identify (const char *start, TiXmlEncoding encoding)**

Protected Attributes

- **TiXmlNode * parent**
- **NodeType type**
- **TiXmlNode * firstChild**
- **TiXmlNode * lastChild**
- **TIXML_STRING value**
- **TiXmlNode * prev**
- **TiXmlNode * next**

Friends

- class **TiXmlDocument**
- class **TiXmlElement**

Additional Inherited Members

10.36.1 Detailed Description

The parent class for everything in the Document Object Model. (Except for attributes). Nodes have siblings, a parent, and children. A node can be in a document, or stand on its own. The type of a **TiXmlNode** (p. ??) can be queried, and it can be cast to its more defined type.

Definition at line 399 of file `tinyxml.h`.

10.36.2 Member Enumeration Documentation

10.36.2.1 enum **TiXmlNode::NodeType**

The types of XML nodes supported by TinyXml. (All the unsupported types are picked up by UNKNOWN.)

Enumerator

TINYXML_DOCUMENT
TINYXML_ELEMENT
TINYXML_COMMENT
TINYXML_UNKNOWN
TINYXML_TEXT
TINYXML_DECLARATION
TINYXML_TYPECOUNT

Definition at line 438 of file `tinyxml.h`.

10.36.3 Constructor & Destructor Documentation

10.36.3.1 **TiXmlNode::~~TiXmlNode**() [virtual]

Definition at line 124 of file `tinyxml.cpp`.

10.36.3.2 **TiXmlNode::TiXmlNode**(**NodeType_type**) [protected]

Definition at line 113 of file `tinyxml.cpp`.

10.36.4 Member Function Documentation

10.36.4.1 virtual bool TiXmlNode::Accept (TiXmlVisitor * visitor) const [pure virtual]

Accept a hierarchical visit the nodes in the TinyXML DOM. Every node in the XML tree will be conditionally visited and the host will be called back via the **TiXmlVisitor** (p. ??) interface.

This is essentially a SAX interface for TinyXML. (Note however it doesn't re-parse the XML for the callbacks, so the performance of TinyXML is unchanged by using this interface versus any other.)

The interface has been based on ideas from:

- <http://www.saxproject.org/>
- <http://c2.com/cgi/wiki?HierarchicalVisitorPattern>

Which are both good references for "visiting".

An example of using **Accept()** (p. ??):

```
TiXmlPrinter printer;
tinyxmlDoc.Accept( &printer );
const char* xmlcstr = printer.CStr();
```

Implemented in **TiXmlDocument** (p. ??), **TiXmlUnknown** (p. ??), **TiXmlDeclaration** (p. ??), **TiXmlText** (p. ??), **TiXmlComment** (p. ??), and **TiXmlElement** (p. ??).

10.36.4.2 void TiXmlNode::Clear ()

Delete all the children of this node. Does not affect 'this'.

Definition at line 146 of file tinyxml.cpp.

10.36.4.3 virtual TiXmlNode* TiXmlNode::Clone () const [pure virtual]

Create an exact duplicate of this node and return it. The memory must be deleted by the caller.

Implemented in **TiXmlDocument** (p. ??), **TiXmlUnknown** (p. ??), **TiXmlDeclaration** (p. ??), **TiXmlText** (p. ??), **TiXmlComment** (p. ??), and **TiXmlElement** (p. ??).

10.36.4.4 void TiXmlNode::CopyTo (TiXmlNode * target) const [protected]

Definition at line 138 of file tinyxml.cpp.

10.36.4.5 const TiXmlNode* TiXmlNode::FirstChild () const [inline]

The first child of this node. Will be null if there are no children.

Definition at line 498 of file tinyxml.h.

10.36.4.6 **TiXmlNode*** **TiXmlNode::FirstChild** () [inline]

Definition at line 499 of file tinyxml.h.

10.36.4.7 **const TiXmlNode *** **TiXmlNode::FirstChild** (**const char *** *value*) **const**

The first child of this node with the matching 'value'. Will be null if none found.

Definition at line 338 of file tinyxml.cpp.

10.36.4.8 **TiXmlNode*** **TiXmlNode::FirstChild** (**const char *** *_value*) [inline]

The first child of this node with the matching 'value'. Will be null if none found.

Definition at line 502 of file tinyxml.h.

10.36.4.9 **const TiXmlElement *** **TiXmlNode::FirstChildElement** () **const**

Convenience function to get through elements.

Definition at line 429 of file tinyxml.cpp.

10.36.4.10 **TiXmlElement*** **TiXmlNode::FirstChildElement** () [inline]

Definition at line 641 of file tinyxml.h.

10.36.4.11 **const TiXmlElement *** **TiXmlNode::FirstChildElement** (**const char *** *_value*) **const**

Convenience function to get through elements.

Definition at line 444 of file tinyxml.cpp.

10.36.4.12 **TiXmlElement*** **TiXmlNode::FirstChildElement** (**const char *** *_value*) [inline]

Definition at line 647 of file tinyxml.h.

10.36.4.13 **const TiXmlDocument *** **TiXmlNode::GetDocument** () **const**

Return a pointer to the Document this node lives in. Returns null if not in a document.

Definition at line 489 of file tinyxml.cpp.

10.36.4.14 `TiXmlDocument* TiXmlNode::GetDocument ()` `[inline]`

Definition at line 666 of file tinyxml.h.

10.36.4.15 `TiXmlNode * TiXmlNode::Identify (const char * start, TiXmlEncoding encoding)` `[protected]`

Definition at line 799 of file tinyxmlparser.cpp.

10.36.4.16 `TiXmlNode * TiXmlNode::InsertAfterChild (TiXmlNode * afterThis, const TiXmlNode & addThis)`

Add a new node related to this. Adds a child after the specified child. Returns a pointer to the new object or NULL if an error occurred.

Definition at line 240 of file tinyxml.cpp.

10.36.4.17 `TiXmlNode * TiXmlNode::InsertBeforeChild (TiXmlNode * beforeThis, const TiXmlNode & addThis)`

Add a new node related to this. Adds a child before the specified child. Returns a pointer to the new object or NULL if an error occurred.

Definition at line 207 of file tinyxml.cpp.

10.36.4.18 `TiXmlNode * TiXmlNode::InsertEndChild (const TiXmlNode & addThis)`

Add a new node related to this. Adds a child past the LastChild. Returns a pointer to the new object or NULL if an error occurred.

Definition at line 191 of file tinyxml.cpp.

10.36.4.19 `const TiXmlNode * TiXmlNode::IterateChildren (const TiXmlNode * previous) const`

An alternate way to walk the children of a node. One way to iterate over nodes is:

```
for( child = parent->FirstChild(); child; child = child->NextSibling() )
```

IterateChildren does the same thing with the syntax:

```
child = 0;  
while( child = parent->IterateChildren( child ) )
```

IterateChildren takes the previous child as input and finds the next one. If the previous child is null, it returns the first. IterateChildren will return null when done.

Definition at line 362 of file tinyxml.cpp.

10.36.4.20 `TiXmlNode* TiXmlNode::IterateChildren (const TiXmlNode * previous)` `[inline]`

Definition at line 539 of file tinyxml.h.

10.36.4.21 `const TiXmlNode * TiXmlNode::IterateChildren (const char * value, const TiXmlNode * previous) const`

This flavor of IterateChildren searches for children with a particular 'value'.

Definition at line 376 of file tinyxml.cpp.

10.36.4.22 `TiXmlNode* TiXmlNode::IterateChildren (const char * _value, const TiXmlNode * previous)` `[inline]`

Definition at line 545 of file tinyxml.h.

10.36.4.23 `const TiXmlNode* TiXmlNode::LastChild () const` `[inline]`

Definition at line 507 of file tinyxml.h.

10.36.4.24 `TiXmlNode* TiXmlNode::LastChild ()` `[inline]`

The last child of this node. Will be null if there are no children.

Definition at line 508 of file tinyxml.h.

10.36.4.25 `const TiXmlNode * TiXmlNode::LastChild (const char * value) const`

Definition at line 350 of file tinyxml.cpp.

10.36.4.26 `TiXmlNode* TiXmlNode::LastChild (const char * _value)` `[inline]`

The last child of this node matching 'value'. Will be null if there are no children.

Definition at line 511 of file tinyxml.h.

10.36.4.27 `TiXmlNode * TiXmlNode::LinkEndChild (TiXmlNode * addThis)`

Add a new node related to this. Adds a child past the LastChild.

NOTE: the node to be added is passed by pointer, and will be henceforth owned (and deleted) by tinyXml. This method is efficient and avoids an extra copy, but should be used with care as it uses a different memory model than the other insert functions.

See also

InsertEndChild (p. ??)

Definition at line 163 of file tinyxml.cpp.

10.36.4.28 `const TiXmlNode* TiXmlNode::NextSibling () const` `[inline]`

Navigate to a sibling node.

Definition at line 607 of file tinyxml.h.

10.36.4.29 `TiXmlNode* TiXmlNode::NextSibling ()` `[inline]`

Definition at line 608 of file tinyxml.h.

10.36.4.30 `const TiXmlNode * TiXmlNode::NextSibling (const char * _value) const`

Navigate to a sibling node with the given 'value'.

Definition at line 390 of file tinyxml.cpp.

10.36.4.31 `TiXmlNode* TiXmlNode::NextSibling (const char * _next)` `[inline]`

Definition at line 612 of file tinyxml.h.

10.36.4.32 `const TiXmlElement * TiXmlNode::NextSiblingElement () const`

Convenience function to get through elements. Calls NextSibling and ToElement. Will skip all non-Element nodes. Returns 0 if there is not another element.

Definition at line 459 of file tinyxml.cpp.

10.36.4.33 `TiXmlElement* TiXmlNode::NextSiblingElement ()` `[inline]`

Definition at line 621 of file tinyxml.h.

10.36.4.34 `const TiXmlElement * TiXmlNode::NextSiblingElement (const char * _value) const`

Convenience function to get through elements. Calls NextSibling and ToElement. Will skip all non-Element nodes. Returns 0 if there is not another element.

Definition at line 474 of file tinyxml.cpp.

10.36.4.35 `TiXmlElement* TiXmlNode::NextSiblingElement (const char * _next)` `[inline]`

Definition at line 630 of file tinyxml.h.

10.36.4.36 `bool TiXmlNode::NoChildren () const [inline]`

Returns true if this node has no children.

Definition at line 671 of file tinyxml.h.

10.36.4.37 `TiXmlNode* TiXmlNode::Parent () [inline]`

One step up the DOM.

Definition at line 495 of file tinyxml.h.

10.36.4.38 `const TiXmlNode* TiXmlNode::Parent () const [inline]`

Definition at line 496 of file tinyxml.h.

10.36.4.39 `const TiXmlNode* TiXmlNode::PreviousSibling () const [inline]`

Navigate to a sibling node.

Definition at line 590 of file tinyxml.h.

10.36.4.40 `TiXmlNode* TiXmlNode::PreviousSibling () [inline]`

Definition at line 591 of file tinyxml.h.

10.36.4.41 `const TiXmlNode * TiXmlNode::PreviousSibling (const char * _value) const`

Navigate to a sibling node.

Definition at line 402 of file tinyxml.cpp.

10.36.4.42 `TiXmlNode* TiXmlNode::PreviousSibling (const char * _prev) [inline]`

Definition at line 595 of file tinyxml.h.

10.36.4.43 `bool TiXmlNode::RemoveChild (TiXmlNode * removeThis)`

Delete a child of this node.

Definition at line 312 of file tinyxml.cpp.

10.36.4.44 `TiXmlNode * TiXmlNode::ReplaceChild (TiXmlNode * replaceThis, const TiXmlNode & withThis)`

Replace a child of this node. Returns a pointer to the new object or NULL if an error occurred.

Definition at line 273 of file `tinyxml.cpp`.

10.36.4.45 `void TiXmlNode::SetValue (const char * _value) [inline]`

Changes the value of the node. Defined as:

```
Document:  filename of the xml file
Element:   name of the element
Comment:   the comment text
Unknown:   the tag contents
Text:      the text string
```

Definition at line 484 of file `tinyxml.h`.

10.36.4.46 `virtual const TiXmlComment* TiXmlNode::ToComment () const [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlComment** (p. ??).

Definition at line 675 of file `tinyxml.h`.

10.36.4.47 `virtual TiXmlComment* TiXmlNode::ToComment () [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlComment** (p. ??).

Definition at line 682 of file `tinyxml.h`.

10.36.4.48 `virtual const TiXmlDeclaration* TiXmlNode::ToDeclaration () const [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlDeclaration** (p. ??).

Definition at line 678 of file `tinyxml.h`.

10.36.4.49 `virtual TiXmlDeclaration* TiXmlNode::ToDeclaration () [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlDeclaration** (p. ??).

Definition at line 685 of file `tinyxml.h`.

10.36.4.50 `virtual const TiXmlDocument* TiXmlNode::ToDocument () const [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlDocument** (p. ??).

Definition at line 673 of file tinyxml.h.

10.36.4.51 `virtual TiXmlDocument* TiXmlNode::ToDocument () [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlDocument** (p. ??).

Definition at line 680 of file tinyxml.h.

10.36.4.52 `virtual const TiXmlElement* TiXmlNode::ToElement () const [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlElement** (p. ??).

Definition at line 674 of file tinyxml.h.

10.36.4.53 `virtual TiXmlElement* TiXmlNode::ToElement () [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlElement** (p. ??).

Definition at line 681 of file tinyxml.h.

10.36.4.54 `virtual const TiXmlText* TiXmlNode::ToText () const [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlText** (p. ??).

Definition at line 677 of file tinyxml.h.

10.36.4.55 `virtual TiXmlText* TiXmlNode::ToText () [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlText** (p. ??).

Definition at line 684 of file tinyxml.h.

10.36.4.56 `virtual const TiXmlNode* TiXmlNode::ToUnknown () const [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlNode** (p. ??).

Definition at line 676 of file tinyxml.h.

10.36.4.57 `virtual TiXmlNode* TiXmlNode::ToUnknown () [inline],[virtual]`

Cast to a more defined type. Will return null if not of the requested type.

Reimplemented in **TiXmlNode** (p. ??).

Definition at line 683 of file tinyxml.h.

10.36.4.58 `int TiXmlNode::Type () const [inline]`

Query the type (as an enumerated value, above) of this node. The possible types are: TINYXML_DOCUMENT, TINYXML_ELEMENT, TINYXML_COMMENT, TINYXML_UNKNOWN, TINYXML_TEXT, and TINYXML_DECLARATION.

Definition at line 660 of file tinyxml.h.

10.36.4.59 `const char* TiXmlNode::Value () const [inline]`

The meaning of 'value' changes for the specific type of **TiXmlNode** (p. ??).

```
Document:  filename of the xml file
Element:   name of the element
Comment:   the comment text
Unknown:   the tag contents
Text:      the text string
```

The subclasses will wrap this function.

Definition at line 463 of file tinyxml.h.

10.36.4.60 `const TIXML_STRING& TiXmlNode::ValueTStr () const [inline]`

Definition at line 473 of file tinyxml.h.

10.36.5 Friends And Related Function Documentation

10.36.5.1 `friend class TiXmlDocument [friend]`

Definition at line 401 of file tinyxml.h.

10.36.5.2 friend class **TiXmlElement** [friend]

Definition at line 402 of file tinyxml.h.

10.36.6 Member Data Documentation

10.36.6.1 **TiXmlNode*** **TiXmlNode::firstChild** [protected]

Definition at line 734 of file tinyxml.h.

10.36.6.2 **TiXmlNode*** **TiXmlNode::lastChild** [protected]

Definition at line 735 of file tinyxml.h.

10.36.6.3 **TiXmlNode*** **TiXmlNode::next** [protected]

Definition at line 740 of file tinyxml.h.

10.36.6.4 **TiXmlNode*** **TiXmlNode::parent** [protected]

Definition at line 731 of file tinyxml.h.

10.36.6.5 **TiXmlNode*** **TiXmlNode::prev** [protected]

Definition at line 739 of file tinyxml.h.

10.36.6.6 **NodeType** **TiXmlNode::type** [protected]

Definition at line 732 of file tinyxml.h.

10.36.6.7 **TIXML_STRING** **TiXmlNode::value** [protected]

Definition at line 737 of file tinyxml.h.

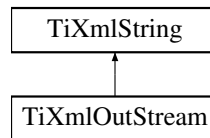
The documentation for this class was generated from the following files:

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**
- DataHandling/**tinyxmlparser.cpp**

10.37 TiXmlOutputStream Class Reference

```
#include <tinyst.h>
```

Inheritance diagram for TiXmlOutputStream:



Public Member Functions

- **TiXmlOutputStream** & **operator**<< (const **TiXmlString** &in)
- **TiXmlOutputStream** & **operator**<< (const char *in)

Additional Inherited Members

10.37.1 Detailed Description

Definition at line 261 of file tinyst.h.

10.37.2 Member Function Documentation

10.37.2.1 **TiXmlOutputStream**& **TiXmlOutputStream::operator**<< (const **TiXmlString** & *in*) [inline]

Definition at line 266 of file tinyst.h.

10.37.2.2 **TiXmlOutputStream**& **TiXmlOutputStream::operator**<< (const char * *in*) [inline]

Definition at line 273 of file tinyst.h.

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

- DataHandling/**tinyst.h**

10.38 TiXmlParsingData Class Reference

Public Member Functions

- void **Stamp** (const char *now, **TiXmlEncoding** encoding)
- const **TiXmlCursor** & **Cursor** () const

Friends

- class **TiXmlDocument**

10.38.1 Detailed Description

Definition at line 150 of file `tinyxmlparser.cpp`.

10.38.2 Member Function Documentation

10.38.2.1 `const TiXmlCursor& TiXmlParsingData::Cursor () const` `[inline]`

Definition at line 156 of file `tinyxmlparser.cpp`.

10.38.2.2 `void TiXmlParsingData::Stamp (const char * now, TiXmlEncoding encoding)`

Definition at line 175 of file `tinyxmlparser.cpp`.

10.38.3 Friends And Related Function Documentation

10.38.3.1 `friend class TiXmlDocument` `[friend]`

Definition at line 152 of file `tinyxmlparser.cpp`.

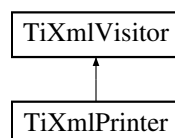
The documentation for this class was generated from the following file:

- `DataHandling/tinyxmlparser.cpp`

10.39 TiXmlPrinter Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for `TiXmlPrinter`:



Public Member Functions

- **TiXmlPrinter** ()
- virtual bool **VisitEnter** (const **TiXmlDocument** &doc)
Visit a document.
- virtual bool **VisitExit** (const **TiXmlDocument** &doc)
Visit a document.
- virtual bool **VisitEnter** (const **TiXmlElement** &element, const **TiXmlAttribute** *firstAttribute)
Visit an element.
- virtual bool **VisitExit** (const **TiXmlElement** &element)
Visit an element.
- virtual bool **Visit** (const **TiXmlDeclaration** &declaration)
Visit a declaration.
- virtual bool **Visit** (const **TiXmlText** &text)
Visit a text node.
- virtual bool **Visit** (const **TiXmlComment** &comment)
Visit a comment node.
- virtual bool **Visit** (const **TiXmlUnknown** &unknown)
Visit an unknown node.
- void **SetIndent** (const char *_indent)
- const char * **Indent** ()
Query the indentation string.
- void **SetLineBreak** (const char *_lineBreak)
- const char * **LineBreak** ()
Query the current line breaking string.
- void **SetStreamPrinting** ()
- const char * **CStr** ()
Return the result.
- size_t **Size** ()
Return the length of the result string.

10.39.1 Detailed Description

Print to memory functionality. The **TiXmlPrinter** (p. ??) is useful when you need to:

1. Print to memory (especially in non-STL mode)
2. Control formatting (line endings, etc.)

When constructed, the **TiXmlPrinter** (p. ??) is in its default "pretty printing" mode. Before calling **Accept()** you can call methods to control the printing of the XML document. After **TiXmlNode::Accept()** (p. ??) is called, the printed document can be accessed via the **CStr()** (p. ??), **Str()**, and **Size()** (p. ??) methods.

TiXmlPrinter (p. ??) uses the Visitor API.

```
TiXmlPrinter printer;
printer.SetIndent( "\\t" );

doc.Accept( &printer );
fprintf( stdout, "%s", printer.CStr() );
```

Definition at line 1714 of file tinyxml.h.

10.39.2 Constructor & Destructor Documentation

10.39.2.1 TiXmlPrinter::TiXmlPrinter () [inline]

Definition at line 1717 of file tinyxml.h.

10.39.3 Member Function Documentation

10.39.3.1 const char* TiXmlPrinter::CStr () [inline]

Return the result.

Definition at line 1753 of file tinyxml.h.

10.39.3.2 const char* TiXmlPrinter::Indent () [inline]

Query the indentation string.

Definition at line 1736 of file tinyxml.h.

10.39.3.3 const char* TiXmlPrinter::LineBreak () [inline]

Query the current line breaking string.

Definition at line 1743 of file tinyxml.h.

10.39.3.4 void TiXmlPrinter::SetIndent (const char *_indent) [inline]

Set the indent characters for printing. By default 4 spaces but tab () is also useful, or null/empty string for no indentation.

Definition at line 1734 of file tinyxml.h.

10.39.3.5 void TiXmlPrinter::SetLineBreak (const char *_lineBreak) [inline]

Set the line breaking string. By default set to newline (). Some operating systems prefer other characters, or can be set to the null/empty string for no indention.

Definition at line 1741 of file tinyxml.h.

10.39.3.6 void TiXmlPrinter::SetStreamPrinting () [inline]

Switch over to "stream printing" which is the most dense formatting without linebreaks. Common when the XML is needed for network transmission.

Definition at line 1748 of file tinyxml.h.

10.39.3.7 `size_t TiXmlPrinter::Size () [inline]`

Return the length of the result string.

Definition at line 1755 of file tinyxml.h.

10.39.3.8 `bool TiXmlPrinter::Visit (const TiXmlDeclaration &) [virtual]`

Visit a declaration.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1834 of file tinyxml.cpp.

10.39.3.9 `bool TiXmlPrinter::Visit (const TiXmlText &) [virtual]`

Visit a text node.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1806 of file tinyxml.cpp.

10.39.3.10 `bool TiXmlPrinter::Visit (const TiXmlComment &) [virtual]`

Visit a comment node.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1843 of file tinyxml.cpp.

10.39.3.11 `bool TiXmlPrinter::Visit (const TiXmlUnknown &) [virtual]`

Visit an unknown node.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1854 of file tinyxml.cpp.

10.39.3.12 `bool TiXmlPrinter::VisitEnter (const TiXmlDocument &) [virtual]`

Visit a document.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1733 of file tinyxml.cpp.

10.39.3.13 `bool TiXmlPrinter::VisitEnter (const TiXmlElement &, const TiXmlAttribute *)` [virtual]

Visit an element.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1743 of file tinyxml.cpp.

10.39.3.14 `bool TiXmlPrinter::VisitExit (const TiXmlDocument &)` [virtual]

Visit a document.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1738 of file tinyxml.cpp.

10.39.3.15 `bool TiXmlPrinter::VisitExit (const TiXmlElement &)` [virtual]

Visit an element.

Reimplemented from **TiXmlVisitor** (p. ??).

Definition at line 1780 of file tinyxml.cpp.

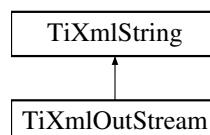
The documentation for this class was generated from the following files:

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**

10.40 TiXmlString Class Reference

```
#include <tinystr.h>
```

Inheritance diagram for TiXmlString:



Public Types

- typedef size_t **size_type**

Public Member Functions

- **TiXmlString** ()
- **TiXmlString** (const **TiXmlString** ©)
- **TIXML_EXPLICIT** **TiXmlString** (const char *copy)
- **TIXML_EXPLICIT** **TiXmlString** (const char *str, **size_type** len)
- **~TiXmlString** ()
- **TiXmlString** & **operator=** (const char *copy)
- **TiXmlString** & **operator=** (const **TiXmlString** ©)
- **TiXmlString** & **operator+=** (const char *suffix)
- **TiXmlString** & **operator+=** (char single)
- **TiXmlString** & **operator+=** (const **TiXmlString** &suffix)
- const char * **c_str** () const
- const char * **data** () const
- **size_type** **length** () const
- **size_type** **size** () const
- bool **empty** () const
- **size_type** **capacity** () const
- const char & **at** (**size_type** index) const
- char & **operator[]** (**size_type** index) const
- **size_type** **find** (char lookup) const
- **size_type** **find** (char tofind, **size_type** offset) const
- void **clear** ()
- void **reserve** (**size_type** cap)
- **TiXmlString** & **assign** (const char *str, **size_type** len)
- **TiXmlString** & **append** (const char *str, **size_type** len)
- void **swap** (**TiXmlString** &other)

Static Public Attributes

- static const **size_type** **npos** = static_cast< **TiXmlString::size_type** >(-1)

10.40.1 Detailed Description

Definition at line 32 of file `tinystl.h`.

10.40.2 Member Typedef Documentation

10.40.2.1 typedef size_t TiXmlString::size_type

Definition at line 36 of file `tinystl.h`.

10.40.3 Constructor & Destructor Documentation

10.40.3.1 TiXmlString::TiXmlString () [inline]

Definition at line 43 of file `tinystl.h`.

10.40.3.2 `TiXmlString::TiXmlString (const TiXmlString & copy)` `[inline]`

Definition at line 48 of file `tinyst.h`.

10.40.3.3 `TIXML_EXPLICIT TiXmlString::TiXmlString (const char * copy)` `[inline]`

Definition at line 55 of file `tinyst.h`.

10.40.3.4 `TIXML_EXPLICIT TiXmlString::TiXmlString (const char * str, size_type len)` `[inline]`

Definition at line 62 of file `tinyst.h`.

10.40.3.5 `TiXmlString::~~TiXmlString ()` `[inline]`

Definition at line 69 of file `tinyst.h`.

10.40.4 Member Function Documentation

10.40.4.1 `TiXmlString & TiXmlString::append (const char * str, size_type len)`

Definition at line 45 of file `tinyst.cpp`.

10.40.4.2 `TiXmlString & TiXmlString::assign (const char * str, size_type len)`

Definition at line 26 of file `tinyst.cpp`.

10.40.4.3 `const char& TiXmlString::at (size_type index) const` `[inline]`

Definition at line 124 of file `tinyst.h`.

10.40.4.4 `const char* TiXmlString::c_str () const` `[inline]`

Definition at line 105 of file `tinyst.h`.

10.40.4.5 `size_type TiXmlString::capacity () const` `[inline]`

Definition at line 120 of file `tinyst.h`.

10.40.4.6 `void TiXmlString::clear ()` `[inline]`

Definition at line 155 of file `tinyst.h`.

10.40.4.7 `const char* TiXmlString::data () const` `[inline]`

Definition at line 108 of file `tinystl.h`.

10.40.4.8 `bool TiXmlString::empty () const` `[inline]`

Definition at line 117 of file `tinystl.h`.

10.40.4.9 `size_type TiXmlString::find (char lookup) const` `[inline]`

Definition at line 138 of file `tinystl.h`.

10.40.4.10 `size_type TiXmlString::find (char tofind, size_type offset) const` `[inline]`

Definition at line 144 of file `tinystl.h`.

10.40.4.11 `size_type TiXmlString::length () const` `[inline]`

Definition at line 111 of file `tinystl.h`.

10.40.4.12 `TiXmlString& TiXmlString::operator+=(const char * suffix)` `[inline]`

Definition at line 86 of file `tinystl.h`.

10.40.4.13 `TiXmlString& TiXmlString::operator+=(char single)` `[inline]`

Definition at line 92 of file `tinystl.h`.

10.40.4.14 `TiXmlString& TiXmlString::operator+=(const TiXmlString & suffix)` `[inline]`

Definition at line 98 of file `tinystl.h`.

10.40.4.15 `TiXmlString& TiXmlString::operator=(const char * copy)` `[inline]`

Definition at line 74 of file `tinystl.h`.

10.40.4.16 `TiXmlString& TiXmlString::operator=(const TiXmlString & copy)` `[inline]`

Definition at line 79 of file `tinystl.h`.

10.40.4.17 `char& TiXmlString::operator[] (size_type index) const` `[inline]`

Definition at line 131 of file `tinyst.h`.

10.40.4.18 `void TiXmlString::reserve (size_type cap)`

Definition at line 14 of file `tinyst.cpp`.

10.40.4.19 `size_type TiXmlString::size () const` `[inline]`

Definition at line 114 of file `tinyst.h`.

10.40.4.20 `void TiXmlString::swap (TiXmlString & other)` `[inline]`

Definition at line 174 of file `tinyst.h`.

10.40.5 Member Data Documentation

10.40.5.1 `const TiXmlString::size_type TiXmlString::npos = static_cast< TiXmlString::size_type >(-1)` `[static]`

Definition at line 39 of file `tinyst.h`.

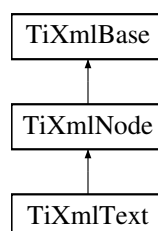
The documentation for this class was generated from the following files:

- DataHandling/**`tinyst.h`**
- DataHandling/**`tinyst.cpp`**

10.41 TiXmlText Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for `TiXmlText`:



Public Member Functions

- **TiXmlText** (const char *initValue)
- virtual ~**TiXmlText** ()
- **TiXmlText** (const **TiXmlText** ©)
- **TiXmlText** & **operator=** (const **TiXmlText** &base)
- virtual void **Print** (FILE *cfile, int depth) const
- bool **CDATA** () const
Queries whether this represents text using a CDATA section.
- void **SetCDATA** (bool _cdata)
Turns on or off a CDATA representation of text.
- virtual const char * **Parse** (const char *p, **TiXmlParsingData** *data, **TiXmlEncoding** encoding)
- virtual const **TiXmlText** * **Text** () const
Cast to a more defined type. Will return null not of the requested type.
- virtual **TiXmlText** * **Text** ()
Cast to a more defined type. Will return null not of the requested type.
- virtual bool **Accept** (**TiXmlVisitor** *content) const

Protected Member Functions

- virtual **TiXmlNode** * **Clone** () const
[internal use] Creates a new Element and returns it.
- void **CopyTo** (**TiXmlText** *target) const
- bool **Blank** () const

Friends

- class **TiXmlElement**

Additional Inherited Members

10.41.1 Detailed Description

XML text. A text node can have 2 ways to output the next. "normal" output and CDATA. It will default to the mode it was parsed from the XML file and you generally want to leave it alone, but you can change the output mode with **SetCDATA()** (p. ??) and query it with **CDATA()** (p. ??).

Definition at line 1188 of file tinyxml.h.

10.41.2 Constructor & Destructor Documentation

10.41.2.1 TiXmlText::TiXmlText (const char * initValue) [inline]

Constructor for text element. By default, it is treated as normal, encoded text. If you want it be output as a CDATA text element, set the parameter _cdata to 'true'

Definition at line 1196 of file tinyxml.h.

10.41.2.2 `virtual TiXmlText::~TiXmlText () [inline], [virtual]`

Definition at line 1201 of file `tinyxml.h`.

10.41.2.3 `TiXmlText::TiXmlText (const TiXmlText & copy) [inline]`

Definition at line 1212 of file `tinyxml.h`.

10.41.3 Member Function Documentation

10.41.3.1 `bool TiXmlText::Accept (TiXmlVisitor * content) const [virtual]`

Walk the XML tree visiting this node and all of its children.

Implements **TiXmlNode** (p. ??).

Definition at line 1337 of file `tinyxml.cpp`.

10.41.3.2 `bool TiXmlText::Blank () const [protected]`

Definition at line 1612 of file `tinyxmlparser.cpp`.

10.41.3.3 `bool TiXmlText::CDATA () const [inline]`

Queries whether this represents text using a CDATA section.

Definition at line 1219 of file `tinyxml.h`.

10.41.3.4 `TiXmlNode * TiXmlText::Clone () const [protected], [virtual]`

[internal use] Creates a new Element and returns it.

Implements **TiXmlNode** (p. ??).

Definition at line 1343 of file `tinyxml.cpp`.

10.41.3.5 `void TiXmlText::CopyTo (TiXmlText * target) const [protected]`

Definition at line 1330 of file `tinyxml.cpp`.

10.41.3.6 `TiXmlText& TiXmlText::operator= (const TiXmlText & base) [inline]`

Definition at line 1213 of file `tinyxml.h`.

10.41.3.7 `const char * TiXmlText::Parse (const char * p, TiXmlParsingData * data, TiXmlEncoding encoding)`
[virtual]

Implements **TiXmlBase** (p. ??).

Definition at line 1478 of file `tinyxmlparser.cpp`.

10.41.3.8 `void TiXmlText::Print (FILE * cfile, int depth) const` [virtual]

All TinyXml classes can print themselves to a filestream or the string class (**TiXmlString** (p. ??) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements **TiXmlBase** (p. ??).

Definition at line 1309 of file `tinyxml.cpp`.

10.41.3.9 `void TiXmlText::SetCDATA (bool _cdata)` [inline]

Turns on or off a CDATA representation of text.

Definition at line 1221 of file `tinyxml.h`.

10.41.3.10 `virtual const TiXmlText* TiXmlText::ToText () const` [inline], [virtual]

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1225 of file `tinyxml.h`.

10.41.3.11 `virtual TiXmlText* TiXmlText::ToText ()` [inline], [virtual]

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1226 of file `tinyxml.h`.

10.41.4 Friends And Related Function Documentation

10.41.4.1 `friend class TiXmlElement` [friend]

Definition at line 1190 of file `tinyxml.h`.

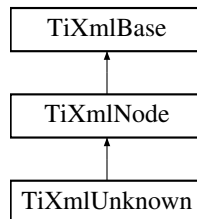
The documentation for this class was generated from the following files:

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**
- DataHandling/**tinyxmlparser.cpp**

10.42 TiXmlUnknown Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlUnknown:



Public Member Functions

- **TiXmlUnknown** ()
- virtual **~TiXmlUnknown** ()
- **TiXmlUnknown** (const **TiXmlUnknown** ©)
- **TiXmlUnknown** & **operator=** (const **TiXmlUnknown** ©)
- virtual **TiXmlNode** * **Clone** () const
Creates a copy of this Unknown and returns it.
- virtual void **Print** (FILE *cfile, int depth) const
- virtual const char * **Parse** (const char *p, **TiXmlParsingData** *data, **TiXmlEncoding** encoding)
- virtual const **TiXmlUnknown** * **ToUnknown** () const
Cast to a more defined type. Will return null not of the requested type.
- virtual **TiXmlUnknown** * **ToUnknown** ()
Cast to a more defined type. Will return null not of the requested type.
- virtual bool **Accept** (**TiXmlVisitor** *content) const

Protected Member Functions

- void **CopyTo** (**TiXmlUnknown** *target) const

Additional Inherited Members

10.42.1 Detailed Description

Any tag that tinyXml doesn't recognize is saved as an unknown. It is a tag of text, but should not be modified. It will be written back to the XML, unchanged, when the file is saved.

DTD tags get thrown into TiXmlUnknowns.

Definition at line 1330 of file tinyxml.h.

10.42.2 Constructor & Destructor Documentation

10.42.2.1 TiXmlUnknown::TiXmlUnknown () [inline]

Definition at line 1333 of file tinyxml.h.

10.42.2.2 `virtual TiXmlUnknown::~~TiXmlUnknown () [inline], [virtual]`

Definition at line 1334 of file `tinyxml.h`.

10.42.2.3 `TiXmlUnknown::TiXmlUnknown (const TiXmlUnknown & copy) [inline]`

Definition at line 1336 of file `tinyxml.h`.

10.42.3 Member Function Documentation

10.42.3.1 `bool TiXmlUnknown::Accept (TiXmlVisitor * content) const [virtual]`

Walk the XML tree visiting this node and all of its children.

Implements **TiXmlNode** (p. ??).

Definition at line 1459 of file `tinyxml.cpp`.

10.42.3.2 `TiXmlNode * TiXmlUnknown::Clone () const [virtual]`

Creates a copy of this Unknown and returns it.

Implements **TiXmlNode** (p. ??).

Definition at line 1465 of file `tinyxml.cpp`.

10.42.3.3 `void TiXmlUnknown::CopyTo (TiXmlUnknown * target) const [protected]`

Definition at line 1453 of file `tinyxml.cpp`.

10.42.3.4 `TiXmlUnknown& TiXmlUnknown::operator= (const TiXmlUnknown & copy) [inline]`

Definition at line 1337 of file `tinyxml.h`.

10.42.3.5 `const char * TiXmlUnknown::Parse (const char * p, TiXmlParsingData * data, TiXmlEncoding encoding) [virtual]`

Implements **TiXmlBase** (p. ??).

Definition at line 1256 of file `tinyxmlparser.cpp`.

10.42.3.6 `void TiXmlUnknown::Print (FILE * cfile, int depth) const` `[virtual]`

All TinyXml classes can print themselves to a filestream or the string class (**TiXmlString** (p. ??) in non-STL mode, `std::string` in STL mode.) Either or both `cfile` and `str` can be null.

This is a formatted print, and will insert tabs and newlines.

(For an unformatted stream, use the `<<` operator.)

Implements **TiXmlBase** (p. ??).

Definition at line 1445 of file `tinyxml.cpp`.

10.42.3.7 `virtual const TiXmlUnknown* TiXmlUnknown::ToUnknown () const` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1346 of file `tinyxml.h`.

10.42.3.8 `virtual TiXmlUnknown* TiXmlUnknown::ToUnknown ()` `[inline],[virtual]`

Cast to a more defined type. Will return null not of the requested type.

Reimplemented from **TiXmlNode** (p. ??).

Definition at line 1347 of file `tinyxml.h`.

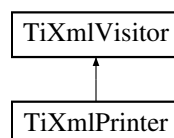
The documentation for this class was generated from the following files:

- DataHandling/**tinyxml.h**
- DataHandling/**tinyxml.cpp**
- DataHandling/**tinyxmlparser.cpp**

10.43 TiXmlVisitor Class Reference

```
#include <tinyxml.h>
```

Inheritance diagram for TiXmlVisitor:



Public Member Functions

- virtual `~TiXmlVisitor()`
- virtual bool **VisitEnter** (const **TiXmlDocument** &)
Visit a document.
- virtual bool **VisitExit** (const **TiXmlDocument** &)
Visit a document.
- virtual bool **VisitEnter** (const **TiXmlElement** &, const **TiXmlAttribute** *)
Visit an element.
- virtual bool **VisitExit** (const **TiXmlElement** &)
Visit an element.
- virtual bool **Visit** (const **TiXmlDeclaration** &)
Visit a declaration.
- virtual bool **Visit** (const **TiXmlText** &)
Visit a text node.
- virtual bool **Visit** (const **TiXmlComment** &)
Visit a comment node.
- virtual bool **Visit** (const **TiXmlUnknown** &)
Visit an unknown node.

10.43.1 Detailed Description

Implements the interface to the "Visitor pattern" (see the `Accept()` method.) If you call the `Accept()` method, it requires being passed a **TiXmlVisitor** (p. ??) class to handle callbacks. For nodes that contain other nodes (Document, Element) you will get called with a `VisitEnter/VisitExit` pair. Nodes that are always leaves are simply called with `Visit()` (p. ??).

If you return 'true' from a Visit method, recursive parsing will continue. If you return false, **no children of this node or its siblings** will be Visited.

All flavors of Visit methods have a default implementation that returns 'true' (continue visiting). You need to only override methods that are interesting to you.

Generally `Accept()` is called on the **TiXmlDocument** (p. ??), although all nodes support Visiting.

You should never change the document from a callback.

See also

TiXmlNode::Accept() (p. ??)

Definition at line 104 of file `tinyxml.h`.

10.43.2 Constructor & Destructor Documentation

10.43.2.1 virtual **TiXmlVisitor::~TiXmlVisitor** () `[inline], [virtual]`

Definition at line 107 of file `tinyxml.h`.

10.43.3 Member Function Documentation

10.43.3.1 `virtual bool TiXmlVisitor::Visit (const TiXmlDeclaration &) [inline],[virtual]`

Visit a declaration.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 120 of file tinyxml.h.

10.43.3.2 `virtual bool TiXmlVisitor::Visit (const TiXmlText &) [inline],[virtual]`

Visit a text node.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 122 of file tinyxml.h.

10.43.3.3 `virtual bool TiXmlVisitor::Visit (const TiXmlComment &) [inline],[virtual]`

Visit a comment node.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 124 of file tinyxml.h.

10.43.3.4 `virtual bool TiXmlVisitor::Visit (const TiXmlUnknown &) [inline],[virtual]`

Visit an unknown node.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 126 of file tinyxml.h.

10.43.3.5 `virtual bool TiXmlVisitor::VisitEnter (const TiXmlDocument &) [inline],[virtual]`

Visit a document.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 110 of file tinyxml.h.

10.43.3.6 `virtual bool TiXmlVisitor::VisitEnter (const TiXmlElement & , const TiXmlAttribute *) [inline],[virtual]`

Visit an element.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 115 of file tinyxml.h.

10.43.3.7 `virtual bool TiXmlVisitor::VisitExit (const TiXmlDocument &) [inline],[virtual]`

Visit a document.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 112 of file tinyxml.h.

10.43.3.8 `virtual bool TiXmlVisitor::VisitExit (const TiXmlElement &) [inline],[virtual]`

Visit an element.

Reimplemented in **TiXmlPrinter** (p. ??).

Definition at line 117 of file tinyxml.h.

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

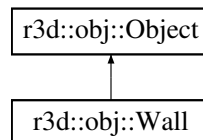
- DataHandling/**tinyxml.h**

10.44 r3d::obj::Wall Class Reference

default shape for simple **Wall** (p. ??) object

```
#include <Wall.h>
```

Inheritance diagram for r3d::obj::Wall:



Public Member Functions

- **Wall** ()
- **Wall** (pcl::PointCloud< pcl::PointXYZ > coordinates)
Constructor for wall object,.
- **~Wall** ()
- void **setShape** (r3d::prims::Polygon shape)
Only one shape per wall allowed.
- r3d::prims::Polygon & **getShape** ()
- void **addCornerPoint** (const pcl::PointXYZ corner)
- void **addCornerPoint** (const Eigen::Vector3f corner)
- pcl::PointCloud< pcl::PointXYZ > & **getCornerPoints** ()
*Sets cloud with corner points of the **Wall** (p. ??) and returns the number of set corners in cloud.*
- int **coordinatesCount** ()
- bool **isValidWall** ()
- void **setIsValid** (bool isValid)

Additional Inherited Members

10.44.1 Detailed Description

default shape for simple **Wall** (p. ??) object

Author

Lukas Hudec

Definition at line 14 of file Wall.h.

10.44.2 Constructor & Destructor Documentation

10.44.2.1 r3d::obj::Wall::Wall ()

Definition at line 5 of file Wall.cpp.

10.44.2.2 r3d::obj::Wall::Wall (pcl::PointCloud< pcl::PointXYZ > *coordinates*)

Constructor for wall object,.

points does not have to be oriented by this order, but have to follow the same pattern

Parameters

<i>coordinates</i>	Wall (p. ??) coordinates in a pointcloud
--------------------	---

Definition at line 9 of file Wall.cpp.

10.44.2.3 r3d::obj::Wall::~~Wall ()

Definition at line 14 of file Wall.cpp.

10.44.3 Member Function Documentation

10.44.3.1 void r3d::obj::Wall::addCornerPoint (const pcl::PointXYZ *corner*)

Definition at line 23 of file Wall.cpp.

10.44.3.2 void r3d::obj::Wall::addCornerPoint (const Eigen::Vector3f *corner*)

Definition at line 28 of file Wall.cpp.

10.44.3.3 int r3d::obj::Wall::coordinatesCount ()

Definition at line 38 of file Wall.cpp.

10.44.3.4 pcl::PointCloud< pcl::PointXYZ > & r3d::obj::Wall::getCornerPoints ()

Sets cloud with corner points of the **Wall** (p. ??) and returns the number of set corners in cloud.

Definition at line 33 of file Wall.cpp.

10.44.3.5 r3d::prims::Polygon& r3d::obj::Wall::getShape () [inline]

Definition at line 30 of file Wall.h.

10.44.3.6 bool r3d::obj::Wall::isValidWall ()

Definition at line 47 of file Wall.cpp.

10.44.3.7 void r3d::obj::Wall::setIsValid (bool *isValid*)

Definition at line 43 of file Wall.cpp.

10.44.3.8 void r3d::obj::Wall::setShape (r3d::prims::Polygon *shape*)

Only one shape per wall allowed.

Definition at line 18 of file Wall.cpp.

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

- 3DReconstruction/**Wall.h**
- 3DReconstruction/**Wall.cpp**

Chapter 11

File Documentation

11.1 3DReconstruction/BirkysMethodReconstruction.cpp File Reference

```
#include "stdafx.h"
#include "BirkysMethodReconstruction.h"
```

11.2 3DReconstruction/BirkysMethodReconstruction.h File Reference

```
#include "Reconstruction.h"
#include <pcl/segmentation/region_growing_rgb.h>
```

Classes

- class **r3d::mtds::BirkysMethodReconstruction**

*This is a derived class from class **Reconstruction** (p.??). Contains methods and variables for 3D reconstruction using normal estimation, normal histogram, point labeling/coloring, region growing.*

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **mtds**
- **r3d::mtds**

Macros

- #define **__BIRKYSMETHODRECONSTRUCTION__**

11.2.1 Macro Definition Documentation

11.2.1.1 `#define __BIRKYSMETHODRECONSTRUCTION__`

Definition at line 4 of file `BirkysMethodReconstruction.h`.

11.3 3DReconstruction/ClosedSpace.cpp File Reference

```
#include "stdafx.h"
#include "ClosedSpace.h"
```

11.4 3DReconstruction/ClosedSpace.h File Reference

```
#include <pcl/visualization/pcl_visualizer.h>
#include "Wall.h"
```

Classes

- class `r3d::space::ClosedSpace`

This class contains all stored data, reconstructed primitives and objects segmented in scene.

Namespaces

- `r3d`
Our team dedicated namespace R3D.
- `space`
- `r3d::space`

11.5 3DReconstruction/CloudRotationTest.cpp File Reference

```
#include "stdafx.h"
#include "CloudRotationTest.h"
#include "Wall.h"
#include "Rectangle.h"
#include <pcl/common/transforms.h>
#include <pcl/filters/voxel_grid.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/sample_consensus/ransac.h>
#include <pcl/sample_consensus/sac_model_plane.h>
#include <pcl/sample_consensus/sac_model_sphere.h>
#include <pcl/sample_consensus/sac_model_perpendicular_plane.h>
#include <pcl/segmentation/sac_segmentation.h>
#include <pcl/segmentation/extract_clusters.h>
#include <pcl/common/intersections.h>
#include <pcl/io/pcd_io.h>
#include <vector>
#include <math.h>
#include <random>
#include <pcl/surface/concave_hull.h>
```

11.6 3DReconstruction/CloudRotationTest.h File Reference

```
#include "Reconstruction.h"
```

Classes

- class **r3d::mtds::CloudRotationTest**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::mtds**

Macros

- #define **__CLOUDROTATIONTEST__**

11.6.1 Macro Definition Documentation

11.6.1.1 #define __CLOUDROTATIONTEST__

Definition at line 4 of file CloudRotationTest.h.

11.7 3DReconstruction/ColorGenerator.cpp File Reference

```
#include "stdafx.h"  
#include "ColorGenerator.h"
```

11.8 3DReconstruction/ColorGenerator.h File Reference

Classes

- class **r3d::colorgen::ColorGenerator**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::colorgen**

Macros

- `#define _COLORGENERATOR_`

11.8.1 Macro Definition Documentation

11.8.1.1 `#define _COLORGENERATOR_`

Definition at line 4 of file ColorGenerator.h.

11.9 3DReconstruction/CommonUtil.cpp File Reference

```
#include "stdafx.h"
#include "CommonUtil.h"
#include <pcl/common/intersections.h>
#include <pcl/segmentation/extract_clusters.h>
#include <pcl/surface/concave_hull.h>
```

11.10 3DReconstruction/CommonUtil.h File Reference

```
#include <boost/thread/thread.hpp>
#include <pcl/PointIndices.h>
#include <pcl/filters/voxel_grid.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/common/common_headers.h>
```

Classes

- struct **r3d::HASH**
- class **r3d::CommonUtil**

Namespaces

- **r3d**

Our team dedicated namespace R3D.

Variables

- const long double **r3d::PI** = 4 * atan(1)

11.11 3DReconstruction/DataStatistics.cpp File Reference

```
#include "stdafx.h"
#include <pcl/sample_consensus/sac_model_plane.h>
#include <pcl/filters/extract_indices.h>
#include <pcl/features/normal_3d_omp.h>
#include <random>
#include "tinysql.h"
#include "CommonUtil.h"
#include "DataStatistics.h"
```

11.12 3DReconstruction/DataStatistics.h File Reference

```
#include "tinysql.h"
```

Classes

- **class r3d::data::DataStatistics**
Statistics about the segmented pointclouds.

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::data**

11.13 3DReconstruction/GrowingRegionReconstruction.cpp File Reference

```
#include "stdafx.h"
#include <pcl/common/intersections.h>
#include <pcl/visualization/point_cloud_color_handlers.h>
#include <pcl/surface/concave_hull.h>
#include "GrowingRegionReconstruction.h"
#include "BirkysMethodReconstruction.h"
#include "PlanePointsExamination.h"
#include "DataStatistics.h"
#include "PlaneSegment.h"
#include "Wall.h"
#include <NoiseRemover.h>
```

11.14 3DReconstruction/GrowingRegionReconstruction.h File Reference

```
#include "ClosedSpace.h"
#include "Reconstruction.h"
#include "PlaneSegment.h"
#include "Line.h"
#include "DataStatistics.h"
```

Classes

- class **r3d::mtds::GrowingRegionReconstruction**

Composite class aggregating reconstruction methods and simple outliers segmentation Uses Birkis segmenation by growing region and color labeling Reconstructs planar objects using RANSAC analytical representation Segments outliers of these planar objects.

Namespaces

- **r3d**

Our team dedicated namespace R3D.

- **r3d::mtds**

Macros

- `#define __GROWING_REGION_RECONSTRUCTION__`

11.14.1 Macro Definition Documentation

11.14.1.1 `#define __GROWING_REGION_RECONSTRUCTION__`

Definition at line 4 of file GrowingRegionReconstruction.h.

11.15 3DReconstruction/Line.cpp File Reference

```
#include "stdafx.h"
#include "Line.h"
```

11.16 3DReconstruction/Line.h File Reference

Classes

- class **r3d::prims::Line**

Namespaces

- **r3d**

Our team dedicated namespace R3D.

- **r3d::prims**

11.17 3DReconstruction/NumeroUnoReconstruction.cpp File Reference

```
#include "stdafx.h"
#include "NumeroUnoReconstruction.h"
```

11.18 3DReconstruction/NumeroUnoReconstruction.h File Reference

```
#include "Reconstruction.h"
```

Classes

- class **r3d::mtds::NumeroUnoReconstruction**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::mtds**

Macros

- #define **__NUMEROUNORECONSTRUCTION__**

11.18.1 Macro Definition Documentation

11.18.1.1 #define __NUMEROUNORECONSTRUCTION__

Definition at line 4 of file NumeroUnoReconstruction.h.

11.19 3DReconstruction/Object.cpp File Reference

```
#include "stdafx.h"
#include "ColorGenerator.h"
#include "Object.h"
```

11.20 3DReconstruction/Object.h File Reference

Classes

- class **r3d::obj::Object**
Class of object.

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::obj**

11.21 3DReconstruction/PlanePointsExamination.cpp File Reference

```
#include "stdafx.h"
#include <pcl/sample_consensus/sac_model_plane.h>
#include <pcl/filters/extract_indices.h>
#include <boost/filesystem.hpp>
#include "CommonUtil.h"
#include "PlaneSegment.h"
#include "PlanePointsExamination.h"
```

11.22 3DReconstruction/PlanePointsExamination.h File Reference

```
#include "PlaneSegment.h"
#include "CommonUtil.h"
#include "PlanePointsExamination.h"
```

Classes

- class **r3d::mtds::PlanePointsExamination**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::mtds**

11.23 3DReconstruction/PlaneSegment.cpp File Reference

```
#include "stdafx.h"
#include <pcl/filters/extract_indices.h>
#include "PlaneSegment.h"
```

11.24 3DReconstruction/PlaneSegment.h File Reference

Classes

- class **r3d::prims::PlaneSegment**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::prims**

11.25 3DReconstruction/Polygon.cpp File Reference

```
#include "stdafx.h"  
#include "Polygon.h"
```

11.26 3DReconstruction/Polygon.h File Reference

```
#include "Primitives.h"
```

Classes

- class **r3d::prims::Polygon**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::prims**

11.27 3DReconstruction/Primitives.cpp File Reference

```
#include "stdafx.h"  
#include "Primitives.h"
```

11.28 3DReconstruction/Primitives.h File Reference

Classes

- class **r3d::prims::Primitives**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::prims**

11.29 3DReconstruction/Reconstruction.cpp File Reference

```
#include "stdafx.h"
#include "Reconstruction.h"
```

11.30 3DReconstruction/Reconstruction.h File Reference

```
#include "CommonUtil.h"
#include "ClosedSpace.h"
#include "tinyxml.h"
#include <pcl/io/pcd_io.h>
#include <pcl/filters/statistical_outlier_removal.h>
```

Classes

- class **r3d::mtds::Reconstruction**

*This is an abstract parent class to all **Reconstruction** (p. ??) methods. Contains original dataset as partial point clouds in aggregated **ClosedSpace** object `m_methodName` should be set to dedicated reconstruction method name.*

Namespaces

- **r3d**

Our team dedicated namespace R3D.

- **mtds**
- **r3d::mtds**

Macros

- `#define __RECONSTRUCTION__`

11.30.1 Macro Definition Documentation

11.30.1.1 `#define __RECONSTRUCTION__`

Definition at line 4 of file `Reconstruction.h`.

11.31 3DReconstruction/Rectangle.cpp File Reference

```
#include "stdafx.h"
#include "Rectangle.h"
```

11.32 3DReconstruction/Rectangle.h File Reference

```
#include "Primitives.h"
```

Classes

- class **r3d::prims::Rectangle**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::prims**

11.33 3DReconstruction/SlicBasedReconstruction.cpp File Reference

```
#include "stdafx.h"  
#include "SlicBasedReconstruction.h"  
#include "BirkysMethodReconstruction.h"
```

11.34 3DReconstruction/SlicBasedReconstruction.h File Reference

```
#include "Reconstruction.h"
```

Classes

- class **r3d::mtds::SlicBasedReconstruction**
*This is a derived class from class **Reconstruction** (p. ??). Contains methods and variables for 3D reconstruction based on SLIC clustering algorithm STILL UNDER CONSTRUCTION.*

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **mtds**
- **r3d::mtds**

Macros

- **#define __SLICBASEDRECONSTRUCTION__**

11.34.1 Macro Definition Documentation

11.34.1.1 `#define __SLICBASEDRECONSTRUCTION__`

Definition at line 4 of file `SlicBasedReconstruction.h`.

11.35 3DReconstruction/stdafx.cpp File Reference

```
#include "stdafx.h"
```

11.36 DataHandling/stdafx.cpp File Reference

```
#include "stdafx.h"
```

11.37 gui/stdafx.cpp File Reference

```
#include "stdafx.h"  
#include <stdio.h>
```

11.38 3DReconstruction/stdafx.h File Reference

```
#include "targetver.h"  
#include <iostream>  
#include <stdio.h>  
#include <stdlib.h>  
#include <string>  
#include <fstream>  
#include <pcl/common/common_headers.h>  
#include <pcl/common/file_io.h>  
#include <pcl/features/normal_3d.h>  
#include <pcl/console/parse.h>  
#include <pcl/search/search.h>  
#include <pcl/search/kdtree.h>  
#include <pcl/point_traits.h>  
#include <pcl/point_types.h>  
#include <pcl/point_cloud.h>  
#include <boost/thread/thread.hpp>  
#include <pcl/segmentation/region_growing_rgb.h>  
#include <pcl/sample_consensus/ransac.h>  
#include <pcl/sample_consensus/sac_model_plane.h>  
#include <pcl/sample_consensus/sac_model_line.h>  
#include <pcl/kdtree/kdtree_flann.h>  
#include <pcl/filters/extract_indices.h>  
#include <pcl/features/normal_3d_omp.h>
```


11.39 DataHandling/stdafx.h File Reference

```
#include "targetver.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>
#include <pcl/common/file_io.h>
#include <pcl/io/ply_io.h>
#include <pcl/io/pcd_io.h>
#include <pcl/io/obj_io.h>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>
#include <pcl/common/common_headers.h>
```

11.40 gui/stdafx.h File Reference

```
#include "targetver.h"
#include <afxwin.h>
#include <afxext.h>
#include <afxdisp.h>
#include <afxdtctl.h>
#include <afxcmn.h>
#include <afxcontrolbars.h>
#include "afxdialogex.h"
#include <string>
#include <vector>
#include <pcl/common/common_headers.h>
#include <pcl/point_types.h>
#include <pcl/point_cloud.h>
#include <pcl/features/normal_3d_omp.h>
```

Macros

- `#define VC_EXTRALEAN`
- `#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS`
- `#define _AFX_ALL_WARNINGS`

11.40.1 Macro Definition Documentation

11.40.1.1 `#define _AFX_ALL_WARNINGS`

Definition at line 17 of file `stdafx.h`.

11.40.1.2 `#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS`

Definition at line 14 of file `stdafx.h`.

11.40.1.3 `#define VC_EXTRALEAN`

Definition at line 9 of file `stdafx.h`.

11.41 3DReconstruction/targetver.h File Reference

```
#include <SDKDDKVer.h>
```

11.42 DataHandling/targetver.h File Reference

```
#include <SDKDDKVer.h>
```

11.43 gui/targetver.h File Reference

```
#include <SDKDDKVer.h>
```

11.44 3DReconstruction/Wall.cpp File Reference

```
#include "stdafx.h"  
#include "Wall.h"
```

11.45 3DReconstruction/Wall.h File Reference

```
#include "Object.h"  
#include "Polygon.h"
```

Classes

- class **r3d::obj::Wall**
*default shape for simple **Wall** (p. ??) object*

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::obj**

11.46 DataHandling/DataReader.cpp File Reference

```
#include "stdafx.h"
#include "DataReader.h"
```

Macros

- `#define WIN32_LEAN_AND_MEAN`

11.46.1 Macro Definition Documentation

11.46.1.1 `#define WIN32_LEAN_AND_MEAN`

11.47 DataHandling/DataReader.h File Reference

```
#include "InputDataParser.h"
#include <pcl/io/ply_io.h>
#include <pcl/io/pcd_io.h>
#include <pcl/io/obj_io.h>
#include <pcl/io/vtk_lib_io.h>
#include <pcl/conversions.h>
#include <pcl/common/file_io.h>
```

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **io**
contains functions to work with various data types input functions loads and converts them to pcl::PointCloud output functions (doesn't exist right now)
- **r3d::io**

Functions

- `template<typename PointT >`
`pcl::PointCloud< PointT > * r3d::io::loadCsvFile (const std::string fileName)`
Loads data coordinates from certain type of CSV file.
- `template<typename PointT >`
`pcl::PointCloud< PointT > * r3d::io::loadVTKMeshFile (const std::string fileName)`
Loads mesh from VTK file and converts it to pcl::PointCloud.
- `template<typename PointT >`
`pcl::PointCloud< PointT > * r3d::io::loadPcdFile (const std::string fileName)`
Loads points from PCD file.
- `template<typename PointT >`
`pcl::PointCloud< PointT > * r3d::io::loadPlyFile (const std::string fileName)`
Loads points from PLY file.
- `template<typename PointT >`
`pcl::PointCloud< PointT > * r3d::io::loadObjFile (const std::string fileName)`
Loads points from OBJ file.
- `template<typename PointT >`
`pcl::PointCloud< PointT > * r3d::io::loadFiles (const std::vector< std::string > files)`

11.48 DataHandling/InputDataParser.cpp File Reference

```
#include "stdafx.h"  
#include "InputDataParser.h"
```

11.49 DataHandling/InputDataParser.h File Reference

Classes

- class **r3d::parser::InputDataParser**
Data parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::parser**

11.50 DataHandling/NoiseRemover.cpp File Reference

```
#include "stdafx.h"  
#include "NoiseRemover.h"
```

11.51 DataHandling/NoiseRemover.h File Reference

```
#include <pcl/io/pcd_io.h>  
#include <pcl/point_types.h>  
#include <pcl/filters/statistical_outlier_removal.h>  
#include "tinyxml.h"
```

Classes

- class **r3d::remover::NoiseRemover**
Removes noise from point cloudData parser, whatever data type from ASCII to pointcloud XYZ coordinate system.

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::remover**

11.52 DataHandling/tinystl.cpp File Reference

```
#include "stdafx.h"  
#include "tinystl.h"
```

Functions

- **TiXmlString operator+** (const **TiXmlString** &a, const **TiXmlString** &b)
- **TiXmlString operator+** (const **TiXmlString** &a, const char *b)
- **TiXmlString operator+** (const char *a, const **TiXmlString** &b)

11.52.1 Function Documentation

11.52.1.1 TiXmlString operator+ (const TiXmlString & a, const TiXmlString & b)

Definition at line 58 of file tinystl.cpp.

11.52.1.2 TiXmlString operator+ (const TiXmlString & a, const char * b)

Definition at line 67 of file tinystl.cpp.

11.52.1.3 TiXmlString operator+ (const char * a, const TiXmlString & b)

Definition at line 77 of file tinystl.cpp.

11.53 DataHandling/tinystl.h File Reference

```
#include <assert.h>  
#include <string.h>
```

Classes

- class **TiXmlString**
- class **TiXmlOutputStream**

Macros

- **#define TIXML_EXPLICIT**

Functions

- `bool operator== (const TiXmlString &a, const TiXmlString &b)`
- `bool operator< (const TiXmlString &a, const TiXmlString &b)`
- `bool operator!= (const TiXmlString &a, const TiXmlString &b)`
- `bool operator> (const TiXmlString &a, const TiXmlString &b)`
- `bool operator<= (const TiXmlString &a, const TiXmlString &b)`
- `bool operator>= (const TiXmlString &a, const TiXmlString &b)`
- `bool operator== (const TiXmlString &a, const char *b)`
- `bool operator== (const char *a, const TiXmlString &b)`
- `bool operator!= (const TiXmlString &a, const char *b)`
- `bool operator!= (const char *a, const TiXmlString &b)`
- `TiXmlString operator+ (const TiXmlString &a, const TiXmlString &b)`
- `TiXmlString operator+ (const TiXmlString &a, const char *b)`
- `TiXmlString operator+ (const char *a, const TiXmlString &b)`

11.53.1 Macro Definition Documentation

11.53.1.1 `#define TIXML_EXPLICIT`

Definition at line 21 of file `tinyst.h`.

11.53.2 Function Documentation

11.53.2.1 `bool operator!= (const TiXmlString & a, const TiXmlString & b)` `[inline]`

Definition at line 242 of file `tinyst.h`.

11.53.2.2 `bool operator!= (const TiXmlString & a, const char * b)` `[inline]`

Definition at line 249 of file `tinyst.h`.

11.53.2.3 `bool operator!= (const char * a, const TiXmlString & b)` `[inline]`

Definition at line 250 of file `tinyst.h`.

11.53.2.4 `TiXmlString operator+ (const TiXmlString & a, const TiXmlString & b)`

Definition at line 58 of file `tinyst.cpp`.

11.53.2.5 `TiXmlString operator+ (const TiXmlString & a, const char * b)`

Definition at line 67 of file `tinyst.cpp`.

11.53.2.6 TiXmlString operator+ (const char * *a*, const TiXmlString & *b*)

Definition at line 77 of file tinystl.cpp.

11.53.2.7 bool operator< (const TiXmlString & *a*, const TiXmlString & *b*) [inline]

Definition at line 237 of file tinystl.h.

11.53.2.8 bool operator<= (const TiXmlString & *a*, const TiXmlString & *b*) [inline]

Definition at line 244 of file tinystl.h.

11.53.2.9 bool operator== (const TiXmlString & *a*, const TiXmlString & *b*) [inline]

Definition at line 232 of file tinystl.h.

11.53.2.10 bool operator== (const TiXmlString & *a*, const char * *b*) [inline]

Definition at line 247 of file tinystl.h.

11.53.2.11 bool operator== (const char * *a*, const TiXmlString & *b*) [inline]

Definition at line 248 of file tinystl.h.

11.53.2.12 bool operator> (const TiXmlString & *a*, const TiXmlString & *b*) [inline]

Definition at line 243 of file tinystl.h.

11.53.2.13 bool operator>= (const TiXmlString & *a*, const TiXmlString & *b*) [inline]

Definition at line 245 of file tinystl.h.

11.54 DataHandling/tinyxml.cpp File Reference

```
#include "stdafx.h"
#include <ctype.h>
#include "tinyxml.h"
```

Functions

- FILE * **TiXmlFOpen** (const char *filename, const char *mode)

11.54.1 Function Documentation

11.54.1.1 FILE * TiXmlFOpen (const char * *filename*, const char * *mode*)

Definition at line 16 of file tinyxml.cpp.

11.55 DataHandling/tinyxml.h File Reference

```
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "tinystr.h"
```

Classes

- struct **TiXmlCursor**
- class **TiXmlVisitor**
- class **TiXmlBase**
- class **TiXmlNode**
- class **TiXmlAttribute**
- class **TiXmlAttributeSet**
- class **TiXmlElement**
- class **TiXmlComment**
- class **TiXmlText**
- class **TiXmlDeclaration**
- class **TiXmlUnknown**
- class **TiXmlDocument**
- class **TiXmlHandle**
- class **TiXmlPrinter**

Macros

- #define **TIXML_STRING** TiXmlString
- #define **TIXML_SAFE**
- #define **TIXML_SNPRINTF** snprintf
- #define **TIXML_SSCANF** sscanf

Enumerations

- enum { **TIXML_SUCCESS**, **TIXML_NO_ATTRIBUTE**, **TIXML_WRONG_TYPE** }
- enum **TiXmlEncoding** { **TIXML_ENCODING_UNKNOWN**, **TIXML_ENCODING_UTF8**, **TIXML_ENCODING_**↵
NG_LEGACY }

Variables

- const int **TIXML_MAJOR_VERSION** = 2
- const int **TIXML_MINOR_VERSION** = 6
- const int **TIXML_PATCH_VERSION** = 2
- const **TiXmlEncoding** **TIXML_DEFAULT_ENCODING** = **TIXML_ENCODING_UNKNOWN**

11.55.1 Macro Definition Documentation

11.55.1.1 #define TIXML_SAFE

Definition at line 36 of file tinyxml.h.

11.55.1.2 #define TIXML_SNPRINTF snprintf

Definition at line 54 of file tinyxml.h.

11.55.1.3 #define TIXML_SSCANF sscanf

Definition at line 55 of file tinyxml.h.

11.55.1.4 #define TIXML_STRING TiXmlString

Definition at line 29 of file tinyxml.h.

11.55.2 Enumeration Type Documentation

11.55.2.1 anonymous enum

Enumerator

TIXML_SUCCESS
TIXML_NO_ATTRIBUTE
TIXML_WRONG_TYPE

Definition at line 130 of file tinyxml.h.

11.55.2.2 enum TiXmlEncoding

Enumerator

TIXML_ENCODING_UNKNOWN
TIXML_ENCODING_UTF8
TIXML_ENCODING_LEGACY

Definition at line 139 of file tinyxml.h.

11.55.3 Variable Documentation

11.55.3.1 `const TiXmlEncoding TIXML_DEFAULT_ENCODING = TIXML_ENCODING_UNKNOWN`

Definition at line 146 of file `tinyxml.h`.

11.55.3.2 `const int TIXML_MAJOR_VERSION = 2`

Definition at line 68 of file `tinyxml.h`.

11.55.3.3 `const int TIXML_MINOR_VERSION = 6`

Definition at line 69 of file `tinyxml.h`.

11.55.3.4 `const int TIXML_PATCH_VERSION = 2`

Definition at line 70 of file `tinyxml.h`.

11.56 DataHandling/tinyxmlerror.cpp File Reference

```
#include "stdafx.h"  
#include "tinyxml.h"
```

11.57 DataHandling/tinyxmlparser.cpp File Reference

```
#include "stdafx.h"  
#include <ctype.h>  
#include <stddef.h>  
#include "tinyxml.h"
```

Classes

- class **TiXmlParsingData**

Variables

- const unsigned char **TIXML_UTF_LEAD_0** = 0xefU
- const unsigned char **TIXML_UTF_LEAD_1** = 0xbbU
- const unsigned char **TIXML_UTF_LEAD_2** = 0xbfU

11.57.1 Variable Documentation

11.57.1.1 `const unsigned char TIXML_UTF_LEAD_0 = 0xefU`

Definition at line 39 of file `tinyxmlparser.cpp`.

11.57.1.2 `const unsigned char TIXML_UTF_LEAD_1 = 0xbbU`

Definition at line 40 of file `tinyxmlparser.cpp`.

11.57.1.3 `const unsigned char TIXML_UTF_LEAD_2 = 0xbfU`

Definition at line 41 of file `tinyxmlparser.cpp`.

11.58 Exporter/Exporter.cpp File Reference

```
#include <pcl/point_cloud.h>
#include <pcl/point_types.h>
#include <pcl/PCLPointCloud2.h>
#include <pcl/common/io.h>
#include <pcl/io/pcd_io.h>
#include "Exporter.h"
#include "Polygon.h"
#include "sgCore.h"
#include <vector>
#include <iostream>
#include <fstream>
#include <string>
```

Functions

- `bool comparePoint` (`pcl::PointXYZ p1`, `pcl::PointXYZ p2`)
- `bool equalPoint` (`pcl::PointXYZ p1`, `pcl::PointXYZ p2`)

11.58.1 Function Documentation

11.58.1.1 `bool comparePoint (pcl::PointXYZ p1, pcl::PointXYZ p2)`

Definition at line 25 of file `Exporter.cpp`.

11.58.1.2 `bool equalPoint (pcl::PointXYZ p1, pcl::PointXYZ p2)`

Definition at line 34 of file `Exporter.cpp`.

11.59 Exporter/Exporter.h File Reference

```
#include <vector>
#include <pcl/point_types.h>
#include <pcl/PolygonMesh.h>
#include "Wall.h"
```

Classes

- class **r3d::exporter::Exporter**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::exporter**

11.60 Exporter/ExporterTest.cpp File Reference

```
#include <pcl/common/common_headers.h>
#include "stdafx.h"
#include "Exporter.h"
#include "Polygon.h"
#include "Wall.h"
#include <iostream>
#include <pcl/io/pcd_io.h>
#include <pcl/console/parse.h>
#include <pcl/io/ply_io.h>
#include <pcl/io/vtk_lib_io.h>
```

Functions

- int **main** ()

11.60.1 Function Documentation

11.60.1.1 int main ()

Definition at line 16 of file ExporterTest.cpp.

11.61 gui/gui.cpp File Reference

```
#include "stdafx.h"
#include "gui.h"
#include "guiDlg.h"
```

Variables

- **CguiApp theApp**

11.61.1 Variable Documentation

11.61.1.1 CguiApp theApp

Definition at line 34 of file gui.cpp.

11.62 gui/gui.h File Reference

```
#include "stdafx.h"
#include "resource.h"
```

Classes

- class **CguiApp**

Variables

- **CguiApp theApp**

11.62.1 Variable Documentation

11.62.1.1 CguiApp theApp

Definition at line 34 of file gui.cpp.

11.63 gui/guiDlg.cpp File Reference

```
#include "stdafx.h"
#include "NumeroUnoReconstruction.h"
#include "BirkysMethodReconstruction.h"
#include "GrowingRegionReconstruction.h"
#include "CloudRotationTest.h"
#include "SlicBasedReconstruction.h"
#include "CustomInteractor.h"
#include "DataReader.h"
#include "gui.h"
#include "guiDlg.h"
#include "afxdialogex.h"
#include <string>
#include <vector>
#include "Exporter.h"
#include "NoiseRemover.h"
#include <pcl/io/pcd_io.h>
```

Classes

- class **CAboutDlg**

11.64 gui/guiDlg.h File Reference

```
#include "Reconstruction.h"
#include "tinyxml.h"
#include "tinystl.h"
```

Classes

- class **CguiDlg**

11.65 gui/resource.h File Reference

Macros

- #define **IDM_ABOUTBOX** 0x0010
- #define **IDD_ABOUTBOX** 100
- #define **IDS_ABOUTBOX** 101
- #define **IDD_GUI_DIALOG** 102
- #define **IDR_MAINFRAME** 128
- #define **IDC_COMBO1** 1000
- #define **IDC_BUTTON4** 1009
- #define **stop** 1009
- #define **IDC_BUTTON1** 1010
- #define **IDC_BUTTON_SAVE** 1010
- #define **BUTTON_SV_NORM** 1010
- #define **IDC_RADIO1** 1012
- #define **IDC_RADIO2** 1013
- #define **IDC_RADIO3** 1014
- #define **IDC_RADIO4** 1015
- #define **IDC_BUTTON2** 1016
- #define **vypocet_N** 1016
- #define **open_file** 1018
- #define **stop_bt** 1023
- #define **calculation** 1024
- #define **visualizationbutton** 1025
- #define **IDC_BUTTON5** 1027
- #define **BTN_OPEN_CONF** 1027
- #define **IDC_BUTTON3** 1028
- #define **noise_bt** 1028
- #define **IDC_BUTTON6** 1029
- #define **EXP_BT** 1029

11.65.1 Macro Definition Documentation

11.65.1.1 `#define BTN_OPEN_CONF 1027`

Definition at line 27 of file resource.h.

11.65.1.2 `#define BUTTON_SV_NORM 1010`

Definition at line 15 of file resource.h.

11.65.1.3 `#define calculation 1024`

Definition at line 24 of file resource.h.

11.65.1.4 `#define EXP_BT 1029`

Definition at line 31 of file resource.h.

11.65.1.5 `#define IDC_BUTTON1 1010`

Definition at line 13 of file resource.h.

11.65.1.6 `#define IDC_BUTTON2 1016`

Definition at line 20 of file resource.h.

11.65.1.7 `#define IDC_BUTTON3 1028`

Definition at line 28 of file resource.h.

11.65.1.8 `#define IDC_BUTTON4 1009`

Definition at line 11 of file resource.h.

11.65.1.9 `#define IDC_BUTTON5 1027`

Definition at line 26 of file resource.h.

11.65.1.10 `#define IDC_BUTTON6 1029`

Definition at line 30 of file resource.h.

11.65.1.11 `#define IDC_BUTTON_SAVE 1010`

Definition at line 14 of file resource.h.

11.65.1.12 `#define IDC_COMBO1 1000`

Definition at line 10 of file resource.h.

11.65.1.13 `#define IDC_RADIO1 1012`

Definition at line 16 of file resource.h.

11.65.1.14 `#define IDC_RADIO2 1013`

Definition at line 17 of file resource.h.

11.65.1.15 `#define IDC_RADIO3 1014`

Definition at line 18 of file resource.h.

11.65.1.16 `#define IDC_RADIO4 1015`

Definition at line 19 of file resource.h.

11.65.1.17 `#define IDD_ABOUTBOX 100`

Definition at line 6 of file resource.h.

11.65.1.18 `#define IDD_GUI_DIALOG 102`

Definition at line 8 of file resource.h.

11.65.1.19 `#define IDM_ABOUTBOX 0x0010`

Definition at line 5 of file resource.h.

11.65.1.20 `#define IDR_MAINFRAME 128`

Definition at line 9 of file resource.h.

11.65.1.21 #define IDS_ABOUTBOX 101

Definition at line 7 of file resource.h.

11.65.1.22 #define noise_bt 1028

Definition at line 29 of file resource.h.

11.65.1.23 #define open_file 1018

Definition at line 22 of file resource.h.

11.65.1.24 #define stop 1009

Definition at line 12 of file resource.h.

11.65.1.25 #define stop_bt 1023

Definition at line 23 of file resource.h.

11.65.1.26 #define visualizationbutton 1025

Definition at line 25 of file resource.h.

11.65.1.27 #define vycocet_N 1016

Definition at line 21 of file resource.h.

11.66 Visualization/CustomInteractor.cpp File Reference

```
#include <iostream>
#include <iomanip>
#include "CustomInteractor.h"
#include "vtkInteractorStyleTerrain.h"
#include "vtkRenderWindowInteractor.h"
#include "vtkRenderWindow.h"
#include "vtkRenderer.h"
#include "vtkActor.h"
#include "vtkCamera.h"
#include "vtkCallbackCommand.h"
#include "vtkExtractEdges.h"
#include "vtkMath.h"
#include "vtkObjectFactory.h"
#include "vtkPolyData.h"
#include "vtkPolyDataMapper.h"
#include "vtkSphereSource.h"
```

11.67 Visualization/CustomInteractor.h File Reference

```
#include <pcl/visualization/interactor_style.h>
```

Classes

- class **r3d::visualization::CustomInteractor**

Namespaces

- **r3d**
Our team dedicated namespace R3D.
- **r3d::visualization**

11.68 Visualization/testCustomInteractor.cpp File Reference

```
#include <iostream>
#include <boost/thread/thread.hpp>
#include <pcl/common/common_headers.h>
#include <pcl/features/normal_3d.h>
#include <pcl/io/pcd_io.h>
#include <pcl/visualization/pcl_visualizer.h>
#include <pcl/console/parse.h>
#include "CustomInteractor.h"
```

Functions

- int **testmain** (void)

11.68.1 Function Documentation

11.68.1.1 int testmain (void)

main loop

Definition at line 19 of file testCustomInteractor.cpp.