WOF

v1.11

Generated by Doxygen 1.9.1

1 Main Page

WOF C++ Point Cloud Mesher

- **Point cloud to mesh** - Fast surface reconstruction software
- **Mesh to point cloud** - Quality point cloud creator
- **Mesh to mesh** - Convert a mesh into a point cloud and reconstruct it from there

WOF is as a **C++ library and command line executable** for Windows and Linux development.

Download the WOF library and command line application

There are two WOF builds:

- **WOF PURE** comes without any dependencies so it can be integrated smoothly. It includes a 500 000 points student/eval license.
- **WOF LM** comes with a license manager (LM) and allows an unlimited number of points during the trial period.

Both variants contain a dynamic C++ library, a command line application, C++ examples and documentation.

License

WOF is a commercial project with maintenance and support.

- A free **student license** for non-commercial research and evaluation is included in the PURE build. It allows 500 000 points.
- The LM build includes an **evaluation license** for research and commercial tests. It is unlimited during the trial period. If you want to use WOF in your commercial project, please contact the author and give as much information as possible.

Release Notes and Version History

Version 1.11, May 23rd, 2022:

- Bugfix when no work in progress bar
- New function estimateAvgSpacing() to estimate the density of a point cloud

Version 1.10, May 7th, 2022:

- Progress bar support
- Two separate variants: PURE and LM
- Extended API for detailed control over mesh reduction, mesh smoothing and edge flips.
- C++ example improved

Version 1.09, March 16th, 2022:

- Laplacian smoothing removes noise
- Edge flips fit the triangulation to the thought surface
- Mesh simplification reduces the number of triangles
- Better performance
- Multithreading improved
- Bugfixes
- Enhanced C++-API

Version 1.08, February 17th, 2022:

- Another bugfix

Version 1.07, February 16th, 2022:

- Bugfix in yesterday's version

Version 1.06, February 15th, 2022:

- Mesh melting improved.
- Tiny holes due to poor sampling are better avoided.
- Bugs solved.

- Quality improved.

Version 1.05, July 6th, 2021:

- Improved mesh melting

Version 1.04, April 3rd, 2020:

- Static and dynamic linking is now available *(static linking since v1.10 to reduce the complexity)*
- CMake improved, Visual Studio project, Makefile
- Example data exchanged

Version 1.03, March 23rd, 2020:
First official release of the WOF software:

- Readers and Writers for the ∗.ply, ∗.stl, ∗.asc, ∗.bin, ∗.list file formats exist now
- A Mesh-to-Cloud method has been added
- An API has been made
- The library has been tested for memory leaks
- Documentation pages have been written

Version Beta2, October 2019

- Reconstruction quality improved
- Library versions added

Version Beta1, March 2019

- This software is developed since 2016 when it was a module of the Fade2D software.
- For flexibility reasons the whole point cloud topic has been been moved to the separate WOF project now.

2 Module Index

2.1 Modules

Here is a list of all modules:

3 Namespace Index

3.1 Namespace List

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

4 Hierarchical Index

4.1 Class Hierarchy

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

5 Class Index

5.1 Class List

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

6 File Index

6.1 File List

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

7 Module Documentation

7.1 License related functions

License related functions for "LM" builds.

Macros

- #define WOFLIC_ACTIVATED 10
- #define WOFLIC_GRACE_OK 11
- #define WOFLIC_GRACE_EXPIRED 12
- #define WOFLIC_TRIAL 13
- #define WOFLIC_INVALID 14
- #define WOFLIC_PURE 15

Functions

- int GEOM_WOF::getLicenseState ()

  Check the license state.
- void GEOM_WOF::printLicense ()

  Print license details.
- bool GEOM_WOF::isPure ()

  Check if pure or lm build.
- bool GEOM_WOF::activateWof (const char ∗key, bool bSystemWide)

  Activate WOF license.
- bool GEOM_WOF::deactivateWof ()

  Deactivate WOF license.
- bool GEOM_WOF::extendTrial (const char ∗key)

  Extend Trial.

7.1.1 Detailed Description

Functions related to the license: Activation, deactivation, trial-extension... Not active in "PURE" builds.

7.1.2 Macro Definition Documentation

7.1.2.1 WOFLIC_ACTIVATED `#define WOFLIC_ACTIVATED 10`

WOFLIC_ACTIVATED means the software is activated

7.1.2.2 WOFLIC_GRACE_EXPIRED `#define WOFLIC_GRACE_EXPIRED 12`

WOFLIC_GRACE_EXPIRED means the software is activated but re-verification (no internet) has failed for a long time. Invalid.

7.1.2.3 WOFLIC_GRACE_OK `#define WOFLIC_GRACE_OK 11`

WOFLIC_GRACE_OK means the software is activated but re-verification has failed (no internet, valid for a sufficiently long grace period)

7.1.2.4 WOFLIC_INVALID `#define WOFLIC_INVALID 14`

WOFLIC_INVALID means there is no valid license (trial, product-key)

7.1.2.5 WOFLIC_PURE `#define WOFLIC_PURE 15`

WOFLIC_PURE is the perpetual non-commercial license

7.1.2.6 WOFLIC_TRIAL `#define WOFLIC_TRIAL 13`

WOFLIC_TRIAL means the trial period is still active

7.1.3 Function Documentation

7.1.3.1 activateWof() `bool GEOM_WOF::activateWof ( const char * key, bool bSystemWide )`

This function is used when you have a WOF license key. You can choose to activate system-wide or only for the current user.

Parameters

key	is the purchased software key
bSystemWide	When true then the activation data is stored system-wide. When false the activation is made for the current user.

Note

Activation is only done once. You can use getLicenseState() to find out if the software is already activated.

When the system-wide activation is chosen (`bSystemWide=true`) then the application needs admin-priviledges.

7.1.3.2 deactivateWof() `bool GEOM_WOF::deactivateWof ( )`

Deactivates the WOF license on the present computer so that the key can be used on another machine. This function enables you to replace a computer. Do not use over-frequently, the number of deactivations is limited, it's not a floating license.

Returns

true in case of success, false otherwise

7.1.3.3 extendTrial() `bool GEOM_WOF::extendTrial ( const char * key )`

Parameters

key	is a Trial-Extension-key. You ask for such a key to extend the trial period for your non-commercial research project (see the guidelines) or for an extended commercial test periord.

Returns

true in case of success, false otherwise

7.1.3.4 getLicenseState() `int GEOM_WOF::getLicenseState ( )`

Returns

WOFLIC_ACTIVATED when the software is activated

WOFLIC_TRIAL during the trial period

WOFLIC_GRACE_OK when activated but verification has failed (no internet access) which is valid for a grace period

WOFLIC_GRACE_EXPIRED when activated but verification has failed (no internet) for a long time

WOFLIC_INVALID otherwise (trial expired, no license)

7.2 Version related functions

Version related functions.

Functions

- void GEOM_WOF::printVersion ()

  Print version number.
- void GEOM_WOF::getVersion (int &versionMajor, int &versionMinor)

  Get version numbers.
- bool GEOM_WOF::isRelease ()

  Check if the present binary is a Release or Debug build.

7.2.1 Detailed Description

Functions to identify Debug- and Release builds and to fetch the version number.

7.2.2 Function Documentation

7.2.2.1 getVersion() `void GEOM_WOF::getVersion ( int & versionMajor, int & versionMinor )`

Returns the WOF version number

Parameters

versionMajor,versionMinor	[out] are used to return the major and minor version number

7.2.2.2 isRelease() `bool GEOM_WOF::isRelease ( )`

Returns

true when the library has been compiled in release mode or false otherwise

7.2.2.3 printVersion() `void GEOM_WOF::printVersion ( )`

Prints the version number to stdout

7.3 Geometry functions

Geometry operations.

Functions

- MeshPtr GEOM_WOF::melt (std::vector< Point3 > &vInputCorners, double avgLength, double featureThresh=15.0)

  Remesh (melt) a triangle mesh.
- MeshPtr GEOM_WOF::reconstruct_auto (std::vector< Point3 > &vPoints, bool bAllowSmoothing, double sfactor=2.0)

  Reconstruct with an automatic spacing value.
- MeshPtr GEOM_WOF::reconstruct_abs (std::vector< Point3 > &vPoints, bool bAllowSmoothing, double spacingAbs)

  Reconstruct with an absolute spacing value.
- void GEOM_WOF::toCloud (std::vector< Point3 > &vCornersIn, double length, double featureThresh, std::vector< Point3 > &vCloudOut)

  Mesh-to-Cloud.
- void GEOM_WOF::edgeFlips (MeshPtr pMesh)

  Flip edges.
- void GEOM_WOF::edgeFlipsSimA (MeshPtr pMesh)

  Flip edges - simulated annealing.
- void GEOM_WOF::laplacianSmoothing (MeshPtr pMesh, int numIterations)

  Laplacian smoothing.
- int GEOM_WOF::reduce (MeshPtr pMesh, double targetLen=DBL_MAX, double maxDev=2.0)

  Reduce.

7.3.1 Detailed Description

Operations to reconstruct a surface, to sample-and-reconstruct a mesh and to create feature-aligned quality point clouds.

7.3.2 Function Documentation

7.3.2.1 edgeFlips() `void GEOM_WOF::edgeFlips ( MeshPtr pMesh )`

This function uses a simple and fast edge-flip algorithm to fit the mesh better to thought surface.

Parameters

in	*pMesh*	is the mesh to be improved

See also

edgeFlipsSimA()

7.3.2.2 edgeFlipsSimA() `void GEOM_WOF::edgeFlipsSimA ( MeshPtr pMesh )`

Like edgeFlips() the present function flips edges to adapt the mesh better to the thought surface. But with a computationally more expensive simulated annealing algorithm that achieves yet better output quality.

Parameters

in	*pMesh*	is the mesh to be improved

See also

edgeFlips()

7.3.2.3 laplacianSmoothing()

```
void GEOM_WOF::laplacianSmoothing (
        MeshPtr pMesh,
        int numIterations )
```

This function applies weighted laplacian smoothing to create a smoother mesh.

Parameters

in	*pMesh*	is the mesh to be improved
in	*numIterations*	specifies how much the mesh shall be smoothed.

7.3.2.4 melt()

```
MeshPtr GEOM_WOF::melt (
        std::vector< Point3 > & vInputCorners,
        double avgLength,
        double featureThresh = 15.0 )
```

This function converts a mesh into a point cloud and then reconstructs it from there. This way you can get rid of meshing errors and unnecessary complexity. This mesh-melting operation may change the topology of the object i.e., depending on the sample distance it may close holes and create additional bridges.

Parameters

in	*vInputCorners*	contains the input triangles (3 corner points per triangle)
in	*avgLength*	specifies the average distance to be used. Choose this parameter with care to avoid an extreme number of elements.
in	*featureThresh*	is an optional parameter that specifies that edges above this value shall be treated as feature lines on which points shall be placed.

7.3.2.5 reconstruct_abs()

```
MeshPtr GEOM_WOF::reconstruct_abs (
        std::vector< Point3 > & vPoints,
        bool bAllowSmoothing,
        double spacingAbs )
```

This function reconstructs a triangular mesh from a 3D point cloud.

Parameters

in	*vPoints*	contains the input point cloud
in	*spacingAbs*	is an absolute spacing value. Large values
in	*bAllowSmoothing*	specifies if the point cloud shall be smoothed before reconstruction.

Returns

the reconstructed mesh.

If the absolute spacing value is unknown then better use reconstruct_auto() which automatically estimates the point cloud density.

See also

reduce(), laplacianSmoothing(), edgeFlips(), edgeFlipsSimA() which are powerful postprocessing fuctions.

7.3.2.6 reconstruct_auto()

```
MeshPtr GEOM_WOF::reconstruct_auto (
        std::vector< Point3 > & vPoints,
        bool bAllowSmoothing,
        double sfactor = 2.0 )
```

This function reconstructs a triangular mesh from a 3D point cloud.

Parameters

in	*vPoints*	contains the input point cloud
in	*sfactor*	is a factor on the estimated point cloud density. The average spacing in the point cloud is automatically estimated and multiplied by sfactor. Use larger values to avoid holes and to create coarser meshes. By default sfactor=2.0.
in	*bAllowSmoothing*	specifies if the point cloud shall be smoothed before reconstruction.

Returns

the reconstructed mesh.

See also

reduce(), laplacianSmoothing(), edgeFlips(), edgeFlipsSimA() which are powerful postprocessing fuctions.

7.3.2.7 reduce()

```
int GEOM_WOF::reduce (
        MeshPtr pMesh,
        double targetLen = DBL_MAX,
        double maxDev = 2.0 )
```

This function reduces the number of triangles in `pMesh`.

Parameters

in	*pMesh*	is the mesh to be simplified
in	*targetLen*	is a limit on the edge length (the algorithm will skip larger edges). Default: DBL_MAX
in	*maximum*	angle deviation per simplification step. Default: 2.0

Returns

the number of removed triangles

7.3.2.8 toCloud()

```
void GEOM_WOF::toCloud (
        std::vector< Point3 > & vCornersIn,
        double length,
        double featureThresh,
        std::vector< Point3 > & vCloudOut )
```

This function creates a 3D point cloud from an input mesh.

Parameters

in	*vCornersIn*	contains the input triangles (3 corners per triangle)
in	*length*	specifies the approximate distance to be used
in	*featureThresh*	is an optional parameter. Edges with larger dihedral angles are treated as feature edges and points are preferably placed there.
out	*vCloudOut*	is used to return the point cloud

7.4 File I/O

IO functions.

Enumerations

- enum GEOM_WOF::FileType {
  GEOM_WOF::FT_STL , GEOM_WOF::FT_PLY , GEOM_WOF::FT_XYZ , GEOM_WOF::FT_BIN ,
  GEOM_WOF::FT_LIST , GEOM_WOF::FT_UNKNOWN }

  Filetype.

Functions

- FileType GEOM_WOF::getFileType (const std::string &filename)

  Get File Type.
- bool GEOM_WOF::writePoints_ASCII (const char *filename, const std::vector< Point3 > &vPoints)

  Write points to an ASCII file.
- bool GEOM_WOF::writePoints_BIN (const char *filename, std::vector< Point3 > &vPoints)

  Write points to a binary file.
- bool GEOM_WOF::readPly (const char *filename, bool bReadPoints, std::vector< Point3 > &vPointsOut)

  *Write points to a *.ply file.*
- bool **GEOM_WOF::writePointsPly** (const std::string &filename, std::vector< GEOM_WOF::Point3 > &v↩
  Points, bool bASCII)
- bool GEOM_WOF::readPoints_ASCII (const char *filename, std::vector< Point3 > &vPoints)

  Read points from an ASCII file.
- bool GEOM_WOF::readPoints_BIN (const char *filename, std::vector< Point3 > &vPointsOut)

  Read points from a binary file.
- bool GEOM_WOF::readSTL_ASCII (const char *filename, std::vector< Point3 > &vTriangleCorners)

  Read a mesh from ASCII STL.
- bool GEOM_WOF::readPoints_auto (std::string &inFilename, std::vector< Point3 > &vPoints)

  Read points from a file (automatic detection)
- bool GEOM_WOF::writePoints_auto (std::string &outFilename, std::vector< Point3 > &vPoints, bool bASCII)

  Write points to a file.
- bool GEOM_WOF::writeMesh_auto (const std::string &filename, std::shared_ptr< Mesh > pMesh, bool b↩
  ASCII)

  Write mesh to a file.

7.4.1 Detailed Description

Read/Write functions for point clouds and triangle meshes.

7.4.2 Enumeration Type Documentation

7.4.2.1 FileType `enum GEOM_WOF::FileType`

Enumerator

FT_STL	FileType STL based on the filename extension [.stl].
FT_PLY	FileType PLY based on the filename extension [.ply].
FT_XYZ	FileType XYZ based on the filename extensions [.xyz|.txt|.asc].
FT_BIN	FileType BIN based on the filename extension [.bin].
FT_LIST	FileType LIST based on the filename extension [.list].
FT_UNKNOWN	FileType UNKNOWN for unknown extensions.

7.4.3 Function Documentation

7.4.3.1 getFileType() `FileType GEOM_WOF::getFileType ( const std::string & filename )`

Returns

the file type (FT_STL, FT_PLY, FT_XYZ, FT_BIN, FT_LIST, FT_UNKNOWN) based on the filename extension.

7.4.3.2 readPly() `bool GEOM_WOF::readPly ( const char * filename, bool bReadPoints, std::vector< Point3 > & vPointsOut )`

Parameters

filename	[in] is the input filename
bReadPoints	[in] Use true to get only the points of the ∗.ply file. Otherwise, when you are interested in the triangles then use false to get 3 subsequent corners per triangle.
vPointsOut	[out] is used to return the points

Returns

true when the operation was successful or false otherwise

7.4.3.3 readPoints_ASCII() `bool GEOM_WOF::readPoints_ASCII ( const char * filename, std::vector< Point3 > & vPoints )`

Reads points from a simple ASCII file. Expected file format: Three coordinates (x y z) per line, whitespace separated.

Parameters

filename	[in] is the input filename
vPoints	[out] is used to return the points

Returns

true [in] in case of success or false otherwise

7.4.3.4 readPoints_auto() `bool GEOM_WOF::readPoints_auto ( std::string & inFilename, std::vector< Point3 > & vPoints )`

This function reads points from a ∗.ply-File (ASCII or binary), an ∗.xyz-File (ASCII, 3 coordinates per line), or a ∗.bin-File (simple binary format). The file type is automatically determined from the filename extension.

Parameters

in	*inFilename*	is the input filename
out	*vPoints*	is used to return the points

Returns

true in case of success, false otherwise

7.4.3.5 readPoints_BIN()

```
bool GEOM_WOF::readPoints_BIN (
        const char * filename,
        std::vector< Point3 > & vPointsOut )
```

Parameters

filename	[in] is a binary input file
vPointsOut	[out] is used to return the points

Returns

true in case of success or false otherwise

See also

writePoints_BIN()

7.4.3.6 readSTL_ASCII()

```
bool GEOM_WOF::readSTL_ASCII (
        const char * filename,
        std::vector< Point3 > & vTriangleCorners )
```

Parameters

filename	[in] is the input filename
vTriangleCorners	[out] is used to return three points per triangle

Returns

true when the operation was successful or false otherwise

7.4.3.7 writeMesh_auto()

```
bool GEOM_WOF::writeMesh_auto (
        const std::string & filename,
        std::shared_ptr< Mesh > pMesh,
        bool bASCII )
```

This function writes a Mesh to file. Available formats are ∗.ply (ASCII or binary), ∗.stl (only ASCII) and Geomview-∗.list (ASCII). The file type is automatically determined from the filename extension.

Parameters

in	*filename*	is the output filename
in	*pMesh*	is the mesh to be written
in	*bASCII*	specifies that ASCII mode shall be used when the file can be written in ASCII- or binary mode (as it is the case for ∗.ply)

Returns

true in case of success or false otherwise

7.4.3.8 writePoints_ASCII()

```
bool GEOM_WOF::writePoints_ASCII (
        const char * filename,
        const std::vector< Point3 > & vPoints )
```

Writes points to an ASCII file, three coordinates (x y z) per line, whitespace separated.

Note

Data exchange through ASCII files is easy and convenient but floating point coordinates are not necessarily exact when represented as decimal numbers and ASCII files are big compared to other formats. Thus writing binary files using writePoints_BIN() is recommended.

Parameters

filename	[in] is the output filename
vPoints	[in] contains the points to be written

Returns

true when the operation was successful or false otherwise.

7.4.3.9 writePoints_auto()

```
bool GEOM_WOF::writePoints_auto (
        std::string & outFilename,
        std::vector< Point3 > & vPoints,
        bool bASCII )
```

This function writes points to a ∗.ply-File, ∗.xyz-File (ASCII, 3 coordinates per line), or a ∗.bin-File (simple binary format). The file type is automatically determined from the filename extension.

Parameters

in	*outFilename*	is the output filename
in	*vPoints*	contains the points to be written
in	*bASCII*	specifies that ASCII mode shall be used when the file can be written in ASCII- or binary mode (as it is the case for ∗.ply)

Returns

true in case of success or false otherwise

7.4.3.10 writePoints_BIN()

```
bool GEOM_WOF::writePoints_BIN (
        const char * filename,
        std::vector< Point3 > & vPoints )
```

Writes a binary file, the format is: (int,size_t,double,...,double)
Thereby the first `int` is always 30, the size_t value is vPoints.size() and the double precision values are x0,y0,z0,...,xn,yn,zn.

Parameters

in	*filename*	is the output filename
in	*vPoints*	contains the points to be written

Returns

true when the operation was successful or false otherwise

See also

readPoints_BIN()

8 Namespace Documentation

8.1 GEOM_WOF Namespace Reference

Classes

- class Point3

  3D Point
- class TimerC

  Timer class.
- class Vector3

  3D Vector
- class WofLicenseException

  License-Exception.
- struct WofBugException

  Bug-Exception.
- class ProgressBase

  Base class for progress subscribers.
- class Mesh

  3D Mesh class

Typedefs

- typedef std::shared_ptr< Mesh > MeshPtr

Enumerations

- enum FileType {
  FT_STL , FT_PLY , FT_XYZ , FT_BIN ,
  FT_LIST , FT_UNKNOWN }

  Filetype.

Functions

- std::ostream & **operator**<< (std::ostream &stream, const Point3 &pnt)
- std::istream & **operator**>> (std::istream &stream, Point3 &pnt)
- double sqDistance (const Point3 &p0, const Point3 &p1)

  Get the squared distance between two points.
- double sqDistance (const Point3 *p0, const Point3 *p1)

  Get the squared distance between two points.
- double distance (const Point3 &p0, const Point3 &p1)

  Get the squared distance between two points.
- Point3 center (const Point3 &p0, const Point3 &p1)

  Midpoint of p0 and p1.
- std::ostream & **operator**<< (std::ostream &stream, const Vector3 &vec)
- Vector3 crossProduct (const Vector3 &vec0, const Vector3 &vec1)

  Cross product.
- Vector3 normalize (const Vector3 &other)

Normalize.

- Vector3 **operator-** (const Vector3 &in)
- Vector3 **operator*** (double d, const Vector3 &vec)
- Vector3 **operator+** (const Vector3 &vec0, const Vector3 &vec1)
- Vector3 **operator-** (const Vector3 &vec0, const Vector3 &vec1)
- int getLicenseState ()

  Check the license state.
- void printLicense ()

  Print license details.
- bool isPure ()

  Check if pure or lm build.
- bool activateWof (const char *key, bool bSystemWide)

  Activate WOF license.
- bool deactivateWof ()

  Deactivate WOF license.
- bool extendTrial (const char *key)

  Extend Trial.
- void printVersion ()

  Print version number.
- void getVersion (int &versionMajor, int &versionMinor)

  Get version numbers.
- bool isRelease ()

  Check if the present binary is a Release or Debug build.
- MeshPtr melt (std::vector< Point3 > &vInputCorners, double avgLength, double featureThresh=15.0)

  Remesh (melt) a triangle mesh.
- MeshPtr reconstruct_auto (std::vector< Point3 > &vPoints, bool bAllowSmoothing, double sfactor=2.0)

  Reconstruct with an automatic spacing value.
- MeshPtr reconstruct_abs (std::vector< Point3 > &vPoints, bool bAllowSmoothing, double spacingAbs)

  Reconstruct with an absolute spacing value.
- void toCloud (std::vector< Point3 > &vCornersIn, double length, double featureThresh, std::vector< Point3 > &vCloudOut)

  Mesh-to-Cloud.
- void edgeFlips (MeshPtr pMesh)

  Flip edges.
- void edgeFlipsSimA (MeshPtr pMesh)

  Flip edges - simulated annealing.
- void laplacianSmoothing (MeshPtr pMesh, int numIterations)

  Laplacian smoothing.
- int reduce (MeshPtr pMesh, double targetLen=DBL_MAX, double maxDev=2.0)

  Reduce.
- void subscribe (ProgressBase *pProgressBase)

  Subscribe to progress updates.
- FileType getFileType (const std::string &filename)

  Get File Type.
- bool writePoints_ASCII (const char *filename, const std::vector< Point3 > &vPoints)

  Write points to an ASCII file.
- bool writePoints_BIN (const char *filename, std::vector< Point3 > &vPoints)

  Write points to a binary file.
- bool readPly (const char *filename, bool bReadPoints, std::vector< Point3 > &vPointsOut)

  *Write points to a *.ply file.*
- bool **writePointsPly** (const std::string &filename, std::vector< GEOM_WOF::Point3 > &vPoints, bool bASCII)

- bool readPoints_ASCII (const char *filename, std::vector< Point3 > &vPoints)

  Read points from an ASCII file.
- bool readPoints_BIN (const char *filename, std::vector< Point3 > &vPointsOut)

  Read points from a binary file.
- bool readSTL_ASCII (const char *filename, std::vector< Point3 > &vTriangleCorners)

  Read a mesh from ASCII STL.
- bool readPoints_auto (std::string &inFilename, std::vector< Point3 > &vPoints)

  Read points from a file (automatic detection)
- bool writePoints_auto (std::string &outFilename, std::vector< Point3 > &vPoints, bool bASCII)

  Write points to a file.
- bool writeMesh_auto (const std::string &filename, std::shared_ptr< Mesh > pMesh, bool bASCII)

  Write mesh to a file.

8.1.1 Detailed Description

Namespace GEOM_WOF
Namespace of the WOF library

8.1.2 Typedef Documentation

8.1.2.1 MeshPtr `typedef std::shared_ptr<Mesh> GEOM_WOF::MeshPtr`

MeshPtr is a shared pointer to Mesh

8.1.3 Function Documentation

8.1.3.1 subscribe() `void GEOM_WOF::subscribe ( ProgressBase * pProgressBase )`

You can provide your own progress receiver class (e.g. progress bar) deriving from ProgressBase. Whenever the progress state changes its update method ProgressBase::update() will be called.

9 Class Documentation

9.1 GEOM_WOF::Mesh Class Reference

3D Mesh class
`#include <Mesh.h>`

Public Member Functions

- Mesh (RMesh *pRMesh)

  Constructor.
- ~Mesh ()

  Destructor.
- **Mesh** (const Mesh &)=delete
- Mesh & **operator=** (const Mesh &)=delete
- void getTriangles (std::vector< Point3 * > &vTriangleCorners) const

  Get Triangles.
- void getPoints (std::vector< Point3 * > &vPoints) const

  Get Points.
- void getVertexIndexData (std::vector< Point3 * > &vVertices, std::vector< int > &vCornerIndices) const

Get the Mesh as Vertices and Indices.

- bool writePly_BIN (const std::string &name) const

  Write Ply (Binary)
- bool writePly_ASCII (const std::string &name) const

  Write Ply (ASCII)
- bool writeGeomview_ASCII (const std::string &name) const

  Write Geomview.
- bool writeStl_ASCII (const std::string &name) const

  Write STL (ASCII)
- void printStatistics (const std::string &name) const

  Print Statistics.
- double getAverageEdgeLength () const

  Get the average edge length.

9.1.1 Detailed Description

The Mesh is a 3D triangle mesh.

9.1.2 Member Function Documentation

9.1.2.1 getAverageEdgeLength() `double GEOM_WOF::Mesh::getAverageEdgeLength ( ) const`

Computes and returns the average edge length

9.1.2.2 getPoints() `void GEOM_WOF::Mesh::getPoints ( std::vector< Point3 * > & vPoints ) const`

Parameters

out	*vPoints*	is used to return the vertex pointers

9.1.2.3 getTriangles() `void GEOM_WOF::Mesh::getTriangles ( std::vector< Point3 * > & vTriangleCorners ) const`

Parameters

out	*vTriangleCorners*	is used to return the triangles as 3 vertex pointers per triangle. The order of the corners per triangle is counterclockwise.

9.1.2.4 getVertexIndexData() `void GEOM_WOF::Mesh::getVertexIndexData ( std::vector< Point3 * > & vVertices, std::vector< int > & vCornerIndices ) const`

Parameters

vVertices	[out] The vertices of the Mesh
vCornerIndices	[out] Three subsequent indices for each triangle. The indices refer to points in vVertices and thus the index range starts with 0.

Note

There are file formats (e.g., ∗.obj) where the first index must be 1, not 0! Simply increment the indices by 1 then.

9.1.2.5 printStatistics() `void GEOM_WOF::Mesh::printStatistics ( const std::string & name ) const`

Prints mesh statistics to stdout

Parameters

name	serves as arbitrary identifier that is also printed to stdout

9.1.2.6 writeGeomview_ASCII() `bool GEOM_WOF::Mesh::writeGeomview_ASCII ( const std::string & name ) const`

Writes a file for the Geomview viewer

Parameters

name	[in] is the output filename.

9.1.2.7 writePly_ASCII() `bool GEOM_WOF::Mesh::writePly_ASCII ( const std::string & name ) const`

Writes an ASCII PLY file

Parameters

name	[in] is the output filename.

9.1.2.8 writePly_BIN() `bool GEOM_WOF::Mesh::writePly_BIN ( const std::string & name ) const`

Writes a binary PLY file

Parameters

name	[in] is the output filename.

9.1.2.9 writeStl_ASCII() `bool GEOM_WOF::Mesh::writeStl_ASCII ( const std::string & name ) const`

Writes an ASCII STL file

Parameters

name	[in] is the output filename.

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

- Mesh.h

9.2 GEOM_WOF::Point3 Class Reference

3D Point

`#include <Point3.h>`

Public Member Functions

- Point3 (const double x_, const double y_, const double z_)

  Constructor.
- Point3 ()

  Default constructor.
- Point3 (const Point3 &p_)

  Copy constructor.
- Point3 & operator= (const Point3 &other)

  operator=
- ~Point3 ()

  Destructor.
- double x () const

  Get the x-coordinate.
- double y () const

  Get the y-coordinate.
- double z () const

  Get the z-coordinate.
- void xyz (double &x_, double &y_, double &z_) const

  Get the x-, y- and z-coordinate.
- void addOwnCoords (double &x, double &y, double &z) const

  Add the point's coordinates to x,y,z.
- void addWeightedOwnCoords (double weight, double &x, double &y, double &z) const

  Add the point's weighted coordinates to x,y,z.
- bool operator< (const Point3 &p) const

  Less than operator.
- bool operator> (const Point3 &p) const

  Greater than operator.
- bool operator== (const Point3 &p) const

  Equality operator.
- bool operator!= (const Point3 &p) const

  Inequality operator.
- void set (const double x_, const double y_, const double z_)

  Set the coordiantes.
- void set (const Point3 &pnt)

  Set the coordiantes.
- Vector3 operator- (const Point3 &other) const

  operator-
- Point3 operator+ (const Vector3 &vec) const

  operator+
- Point3 operator- (const Vector3 &vec) const

  operator-

Protected Attributes

- double **coordX**
- double **coordY**
- double **coordZ**

Friends

- std::ostream & **operator**<< (std::ostream &stream, const Point3 &pnt)
- std::istream & **operator**>> (std::istream &stream, Point3 &pnt)

9.2.1 Constructor & Destructor Documentation

9.2.1.1 Point3() [1/3]

```
GEOM_WOF::Point3::Point3 (
        const double x_,
        const double y_,
        const double z_ )  [inline]
```

Parameters

*x_,y_,z_*	[in] coordinates

9.2.1.2 Point3() [2/3]

```
GEOM_WOF::Point3::Point3 ( )  [inline]
```

Coordinates are initialized to 0.

9.2.1.3 Point3() [3/3]

```
GEOM_WOF::Point3::Point3 (
        const Point3 & p_ )  [inline]
```

Copies the coordinates of p_

9.2.2 Member Function Documentation

9.2.2.1 addOwnCoords()

```
void GEOM_WOF::Point3::addOwnCoords (
        double & x,
        double & y,
        double & z ) const  [inline]
```

Parameters

x,y,z	[inout] are used to accumulate the point's coordinates

9.2.2.2 addWeightedOwnCoords()

```
void GEOM_WOF::Point3::addWeightedOwnCoords (
        double weight,
        double & x,
        double & y,
        double & z ) const  [inline]
```

Parameters

weight	is a factor on x,y,z
x,y,z	[inout] are used to accumulate the point's coordinates

9.2.2.3 operator"!=()

```
bool GEOM_WOF::Point3::operator!= (
```

const Point3 & *p*) const [inline]

Parameters

p	[in] The point whose coordinates are compared with the ones of the present point

9.2.2.4 operator+() Point3 GEOM_WOF::Point3::operator+ (

const Vector3 & *vec*) const [inline]

Returns

a point that corresponds to the present point moved by `vec`

9.2.2.5 operator-() [1/2] Vector3 GEOM_WOF::Point3::operator- (

const Point3 & *other*) const [inline]

Returns

the difference vector (`*this` - `other`) i.e., a vector pointing from the point `other` to `*this`.

9.2.2.6 operator-() [2/2] Point3 GEOM_WOF::Point3::operator- (

const Vector3 & *vec*) const [inline]

Returns

a point that corresponds to the present point moved by `vec`

9.2.2.7 operator<() bool GEOM_WOF::Point3::operator< (

const Point3 & *p*) const [inline]

Parameters

p	[in] is compared with *this

Returns

true if the coordinates of the present point are lexicographically smaller than the ones of `p` or false otherwise

9.2.2.8 operator=() Point3& GEOM_WOF::Point3::operator= (

const Point3 & *other*) [inline]

Assigns `other`

9.2.2.9 operator==() bool GEOM_WOF::Point3::operator== (

const Point3 & *p*) const [inline]

Parameters

p	[in] The point whose coordinates are compared with the ones of the present point

9.2.2.10 operator>() bool GEOM_WOF::Point3::operator> (const Point3 & p) const [inline]

Parameters

p	[in] The point whose coordinates are compared with the ones of the present point

Returns

true if the coordinates of the present point are lexicographically greater than the ones of p or false otherwise

9.2.2.11 set() [1/2] void GEOM_WOF::Point3::set (const double x_, const double y_, const double z_) [inline]

Set the coordinates of the present point to x_,y_,z_.

Parameters

*x_,y_↩ ,z_*	[in] are the coordinates to be assigned

9.2.2.12 set() [2/2] void GEOM_WOF::Point3::set (const Point3 & pnt) [inline]

Set the coordinates of the present point to the ones of pnt

Parameters

pnt	carries the coordinates to be assigned

9.2.2.13 x() double GEOM_WOF::Point3::x ( ) const [inline]

Returns

the x-coordinate

9.2.2.14 xyz() void GEOM_WOF::Point3::xyz (double & x_, double & y_, double & z_) const [inline]

Parameters

*x_,y_↩ ,z_*	[out] x,y,z-coordinates

Returns

all 3 coordinates at once

9.2.2.15 y() double GEOM_WOF::Point3::y ( ) const [inline]

Returns

the y-coordinate

9.2.2.16 z() double GEOM_WOF::Point3::z ( ) const [inline]

Returns

the z-coordinate

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

- Point3.h

9.3 GEOM_WOF::ProgressBase Class Reference

Base class for progress subscribers.
`#include <wof_api_definitions.h>`

Public Member Functions

- virtual void update (const std::string &s, double d)=0

  update

9.3.1 Detailed Description

A progress subscriber class can be derived from ProgressBase to receive progress updates from the WOF library.
A simple terminal progress bar could be derived like this:

```
class MyProgressBar:public GEOM_WOF::ProgressBase
{
public:
    // WOF calls the update method with d={0.0,...,1.0}
    void update(const std::string& s,double d)
    {
        if(s!=lastMessage)
        {
            cout«"\n"; // New message, line feed
            lastMessage=s;
        }
        cout«("Progress(\""+s+"\"): ")«d*100.0«" %                                    \r"«flush;
        if(d>=1.0) cout«endl;
    }
protected:
    std::string lastMessage;
};
```

9.3.2 Member Function Documentation

9.3.2.1 update() virtual void GEOM_WOF::ProgressBase::update (const std::string & *s*, double *d*) [pure virtual]

This method must be defined in the derived class. It is called whenever the progress changes and thus it should be computationally inexpensive.
The documentation for this class was generated from the following file:

- wof_api_definitions.h

9.4 GEOM_WOF::TimerC Class Reference

Timer class.
`#include <TimerC.h>`

Public Member Functions

- TimerC ()

  Constructor.
- double stop ()

  Timer stop.
- double get () const

  Get the elapsed time.
- void report (const std::string &s)

  Report.

9.4.1 Detailed Description

TimerC measures the time consumption

9.4.2 Constructor & Destructor Documentation

9.4.2.1 TimerC() `GEOM_WOF::TimerC::TimerC ( ) [inline]`

At construction TimerC stores the current time

9.4.3 Member Function Documentation

9.4.3.1 get() `double GEOM_WOF::TimerC::get ( ) const [inline]`

Returns

the elapsed time in seconds between TimerC construction and the first call to TimerC::stop(). When the timer has not been stopped then the time since construction is returned.

9.4.3.2 report() `void GEOM_WOF::TimerC::report ( const std::string & s ) [inline]`

Prints the time since construction or since last report. This command is intended to measure successive intervals

9.4.3.3 stop() `double GEOM_WOF::TimerC::stop ( ) [inline]`

Returns

the elapsed time since TimerC construction in seconds

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

- TimerC.h

9.5 GEOM_WOF::Vector3 Class Reference

3D Vector

`#include <Vector3.h>`

Public Member Functions

- Vector3 (const double x_, const double y_, const double z_)

  Constructor.
- Vector3 ()

  Default constructor.
- Vector3 (const Vector3 &v_)

  Copy constructor.
- bool isDegenerate () const

  isDegenerate
- void xyz (double &x_, double &y_, double &z_) const

  Get x,y,z.
- double x () const

  Get the x-value.
- double y () const

  Get the y-value.
- double z () const

  Get the z-value.
- void set (const double x_, const double y_, const double z_)

  Set x,y,z.
- void add (const Vector3 &other)

  Add a Vector3 to the present one.
- void **sub** (const Vector3 &other)
- void **div** (double div)
- void **mul** (double mul)
- double sqLength () const

  Get the squared length of the vector.
- int getMaxAbsIndex () const

  Get max index.
- double getMaxComponent () const

  Get max component.
- double getMaxAbsComponent () const

  Get max absolute component.
- double getCartesian (int i) const

  Get component i.
- double length () const

  Get the length of the vector.
- double operator∗ (const Vector3 &other) const

  Scalar product.
- Vector3 operator∗ (double val) const

  Multiply by a scalar value.
- Vector3 operator/ (double val) const

  Divide by a scalar value.
- Vector3 & operator= (const Vector3 &other)

  Equality operator.

Protected Attributes

- double **valX**
- double **valY**
- double **valZ**

9.5.1 Constructor & Destructor Documentation

9.5.1.1 Vector3() [1/3] GEOM_WOF::Vector3::Vector3 (

```
const double x_,
const double y_,
const double z_ )
```

Parameters

*x_,y_↩ ,z_*	Values to initialize the Vector

9.5.1.2 Vector3() [2/3] GEOM_WOF::Vector3::Vector3 ( )

The vector is initialized to (0,0,0)

9.5.1.3 Vector3() [3/3] GEOM_WOF::Vector3::Vector3 (

```
const Vector3 & v_ )
```

Copies `v_`

9.5.2 Member Function Documentation

9.5.2.1 add() void GEOM_WOF::Vector3::add (

```
const Vector3 & other ) [inline]
```

Parameters

other	is added to the present Vector3

9.5.2.2 getCartesian() double GEOM_WOF::Vector3::getCartesian (

```
int i ) const
```

Returns

the `i-th` component

9.5.2.3 getMaxAbsComponent() double GEOM_WOF::Vector3::getMaxAbsComponent ( ) const

Returns

the maximum absolute component

9.5.2.4 getMaxAbsIndex() int GEOM_WOF::Vector3::getMaxAbsIndex ( ) const

Returns

the index of the largest absolute component (0,1 or 2)

9.5.2.5 getMaxComponent() double GEOM_WOF::Vector3::getMaxComponent ( ) const [inline]

Returns

the maximum component

9.5.2.6 isDegenerate() bool GEOM_WOF::Vector3::isDegenerate ( ) const

Returns

true if the vector length is 0, false otherwise.

9.5.2.7 length() double GEOM_WOF::Vector3::length ( ) const

Returns

the length of the vector

9.5.2.8 operator*() [1/2] double GEOM_WOF::Vector3::operator* (const Vector3 & *other*) const

Returns

the scalar product of the present Vector3 and `other`

9.5.2.9 operator*() [2/2] Vector3 GEOM_WOF::Vector3::operator* (double *val*) const

Returns

the present Vector3 multiplied by `val`

9.5.2.10 operator/() Vector3 GEOM_WOF::Vector3::operator/ (double *val*) const

Returns

the present Vector3 divided by `val`

9.5.2.11 operator=() Vector3& GEOM_WOF::Vector3::operator= (const Vector3 & *other*)

Returns

true when the present Vector3 has the same x,y,z-values as `other`

9.5.2.12 set() void GEOM_WOF::Vector3::set (const double *x_*, const double *y_*, const double *z_*)

Assigns values to the present Vector3

Parameters

*x_,y_↩ ,z_*	Values to assign

9.5.2.13 sqLength() `double GEOM_WOF::Vector3::sqLength ( ) const`

Returns

the squared length of the Vector3

9.5.2.14 x() `double GEOM_WOF::Vector3::x ( ) const [inline]`

Returns

x

9.5.2.15 xyz() `void GEOM_WOF::Vector3::xyz ( double & x_, double & y_, double & z_ ) const [inline]`

Parameters

*x_,y_↩ ,z_*	[out] Used to return the x,y,z-values of the Vector

9.5.2.16 y() `double GEOM_WOF::Vector3::y ( ) const [inline]`

Returns

y

9.5.2.17 z() `double GEOM_WOF::Vector3::z ( ) const [inline]`

Returns

z

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

- Vector3.h

9.6 GEOM_WOF::WofBugException Struct Reference

Bug-Exception.
`#include <wof_api_definitions.h>`
Inherits std::exception.

Public Member Functions

- virtual const char * **what** () const throw ()

9.6.1 Detailed Description

The WofBugException is thrown in case of unexpected states caused by invalid input **OR** a bug. Software quality is very important for this project. If you find a bug, send a bug report and it will be processed immediately.
The documentation for this struct was generated from the following file:

- wof_api_definitions.h

9.7 GEOM_WOF::WofLicenseException Class Reference

License-Exception.
`#include <wof_api_definitions.h>`
Inherits std::exception.

9.7.1 Detailed Description

The WofLicenseException is thrown in case of an invalid license state. If your trial has expired but you still need the software for your non-commercial research, contact Geom Software: `bkorn@geom.at` for a trial extension code. Include information about your project.
The documentation for this class was generated from the following file:

- wof_api_definitions.h

10 File Documentation

10.1 wof_api.h File Reference

```
#include <vector>
#include "License.h"
#include "wof_api_io.h"
#include "wof_api_functions.h"
#include "wof_api_definitions.h"
#include "TimerC.h"
```

Namespace

- GEOM_WOF

Macros

- #define WOFLIC_ACTIVATED 10
- #define WOFLIC_GRACE_OK 11
- #define WOFLIC_GRACE_EXPIRED 12
- #define WOFLIC_TRIAL 13
- #define WOFLIC_INVALID 14
- #define WOFLIC_PURE 15

Functions

- int GEOM_WOF::getLicenseState ()

  Check the license state.
- void GEOM_WOF::printLicense ()

  Print license details.
- bool GEOM_WOF::isPure ()

  Check if pure or lm build.
- bool GEOM_WOF::activateWof (const char ∗key, bool bSystemWide)

  Activate WOF license.

- bool GEOM_WOF::deactivateWof ()

  Deactivate WOF license.
- bool GEOM_WOF::extendTrial (const char *key)

  Extend Trial.
- void GEOM_WOF::printVersion ()

  Print version number.
- void GEOM_WOF::getVersion (int &versionMajor, int &versionMinor)

  Get version numbers.
- bool GEOM_WOF::isRelease ()

  Check if the present binary is a Release or Debug build.

10.2 wof_api_definitions.h File Reference

Classes

- class GEOM_WOF::WofLicenseException

  License-Exception.
- struct GEOM_WOF::WofBugException

  Bug-Exception.
- class GEOM_WOF::ProgressBase

  Base class for progress subscribers.

Namespace

- GEOM_WOF

Macros

- #define **WOF_VER_MAJOR** 1
- #define **WOF_VER_MINOR** 11
- #define **CLASS_DECLSPEC**

10.3 wof_api_functions.h File Reference

```
#include <vector>
#include <float.h>
#include "Point3.h"
#include "Mesh.h"
```

Namespace

- GEOM_WOF

Functions

- MeshPtr GEOM_WOF::melt (std::vector< Point3 > &vInputCorners, double avgLength, double featureThresh=15.0)

  Remesh (melt) a triangle mesh.
- MeshPtr GEOM_WOF::reconstruct_auto (std::vector< Point3 > &vPoints, bool bAllowSmoothing, double sfactor=2.0)

  Reconstruct with an automatic spacing value.
- MeshPtr GEOM_WOF::reconstruct_abs (std::vector< Point3 > &vPoints, bool bAllowSmoothing, double spacingAbs)

  Reconstruct with an absolute spacing value.

- void GEOM_WOF::toCloud (std::vector< Point3 > &vCornersIn, double length, double featureThresh, std::vector< Point3 > &vCloudOut)

  Mesh-to-Cloud.
- void GEOM_WOF::edgeFlips (MeshPtr pMesh)

  Flip edges.
- void GEOM_WOF::edgeFlipsSimA (MeshPtr pMesh)

  Flip edges - simulated annealing.
- void GEOM_WOF::laplacianSmoothing (MeshPtr pMesh, int numIterations)

  Laplacian smoothing.
- int GEOM_WOF::reduce (MeshPtr pMesh, double targetLen=DBL_MAX, double maxDev=2.0)

  Reduce.
- void GEOM_WOF::subscribe (ProgressBase *pProgressBase)

  Subscribe to progress updates.

10.4 wof_api_io.h File Reference

```
#include <vector>
#include "Point3.h"
#include "Mesh.h"
```

Namespace

- GEOM_WOF

Enumerations

- enum GEOM_WOF::FileType { GEOM_WOF::FT_STL , GEOM_WOF::FT_PLY , GEOM_WOF::FT_XYZ , GEOM_WOF::FT_BIN , GEOM_WOF::FT_LIST , GEOM_WOF::FT_UNKNOWN }

  Filetype.

Functions

- FileType GEOM_WOF::getFileType (const std::string &filename)

  Get File Type.
- bool GEOM_WOF::writePoints_ASCII (const char *filename, const std::vector< Point3 > &vPoints)

  Write points to an ASCII file.
- bool GEOM_WOF::writePoints_BIN (const char *filename, std::vector< Point3 > &vPoints)

  Write points to a binary file.
- bool GEOM_WOF::readPly (const char *filename, bool bReadPoints, std::vector< Point3 > &vPointsOut)

  *Write points to a *.ply file.*
- bool **GEOM_WOF::writePointsPly** (const std::string &filename, std::vector< GEOM_WOF::Point3 > &vPoints, bool bASCII)
- bool GEOM_WOF::readPoints_ASCII (const char *filename, std::vector< Point3 > &vPoints)

  Read points from an ASCII file.
- bool GEOM_WOF::readPoints_BIN (const char *filename, std::vector< Point3 > &vPointsOut)

  Read points from a binary file.
- bool GEOM_WOF::readSTL_ASCII (const char *filename, std::vector< Point3 > &vTriangleCorners)

  Read a mesh from ASCII STL.
- bool GEOM_WOF::readPoints_auto (std::string &inFilename, std::vector< Point3 > &vPoints)

  Read points from a file (automatic detection)
- bool GEOM_WOF::writePoints_auto (std::string &outFilename, std::vector< Point3 > &vPoints, bool bASCII)

  Write points to a file.

- bool GEOM_WOF::writeMesh_auto (const std::string &filename, std::shared_ptr< Mesh > pMesh, bool b↩ ASCII)

  Write mesh to a file.

Index