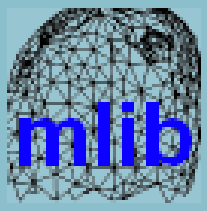


Meshlib

1.8.0.0

Sk. Mohammadul Haque

2013–2021

Table of Contents

Chapter 1

Overview

Introduction

Meshlib is a simple mesh library written in C.

Build

To build the whole project, either Make or Code::Blocks or Visual Studio 2012 (or later) is required.

Contents

- Load/Write ASC, BINv1, COLMAP BIN, NVM, OFF, OBJ, Bundle OUT, PLY, XYZ files.
- Basic Vertex Manipulations.
- Basic Vertex Transformations.
- Basic Vertex Perturbations.
- Basic Face Manipulations.
- Bilateral, Laplacian Geometry Filtering.
- Bilateral, Laplacian, Minimum Intensity, Maximum Intensity Vertex Color Filtering.
- Mesh Cleaning Algorithms.

Author

Sk. Mohammadul Haque

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

Chapter 2

Examples

This chapter shows different examples on how to get started with the Meshlib library. While the examples are particularly simple aiming to demonstrate specific functionalities of the library, one can build upon them to create complex functions.

List of examples

- How to open a mesh file?
- How to compute vertex normals?
- How to draw a mesh in an OpenGL context?
- How to smooth a mesh?

Author

Sk. Mohammadul Haque

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

2.1 How to open a mesh file?

Open a mesh file and display some information about it.

```
#include <meshlib.h>
#include <inttypes.h>

int main(int argc, char* argv[])
{
        MESH m;
        if(argc<2)
        {
                printf("input mesh filename not given\n");
        }
        else
        {
                m = mesh_load_file(argv[1]);
                printf("number of vertices: %" PRId64 "\n", m->num_vertices);
                printf("number of faces: %" PRId64 "\n", m->num_faces);
                mesh_free_mesh(m);
        }
        return 0;
}
```

Author

Sk. Mohammadul Haque

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

2.2 How to compute vertex normals?

Open a mesh file, compute the vertex normals and save it in another mesh file.

```
#include <meshlib.h>

int main(int argc, char* argv[])
{
        MESH m;
        if(argc<3)
        {
                printf("input and output mesh filenames not given\n");
        }
        else
        {
                m = mesh_load_file(argv[1]);
                mesh_calc_vertex_normals(m);
                mesh_save_file(m, argv[2]);
                mesh_free_mesh(m);
        }
        return 0;
}
```

Author

Sk. Mohammadul Haque

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

2.3 How to draw a mesh in an OpenGL context?

Create an ellipsoid mesh and draw it in an OpenGL context.

```
#include <meshlib.h>
#include <GL/glut.h>

MESH m;

void initGL()
{
        glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
        glClearDepth(1.0f);
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glDepthFunc(GL_LEQUAL);
        glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
```

```
}

void display()
{
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        mesh_draw_mesh(m);
        glutSwapBuffers();
}

void reshape(GLsizei width, GLsizei height)
{
        GLfloat aspect = (GLfloat)width/(GLfloat)height;
        glViewport(0, 0, width, height);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(45.0f, aspect, 0.1f, 100.0f);
}

int main(int argc, char* argv[])
{
        mesh_vector3 sz = {1.0, 1.0, 1.0}, pos = {0.0, 0.0, -1.5};
        m = mesh_create_mesh_new_ellipsoid(&sz, &pos);
        char title[] = "Mesh Display";

        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE);
        glutInitWindowSize(640, 480);
        glutCreateWindow(title);
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        initGL();
        glutMainLoop();

        mesh_free_mesh(m);
        return 0;
}
```

Author

Sk. Mohammadul Haque

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

2.4 How to smooth a mesh?

Open a mesh file, smooth and save it in another mesh file.

```
#include <meshlib.h>

int main(int argc, char* argv[])
{
        MESH m;
        if(argc<3)
        {
                printf("input and output mesh filenames not given\n");
        }
        else
        {
                m = mesh_load_file(argv[1]);
                mesh_laplacian_filter(m, 1.0);
                mesh_save_file(m, argv[2]);
                mesh_free_mesh(m);
        }
        return 0;
}
```

Author

Sk. Mohammadul Haque

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

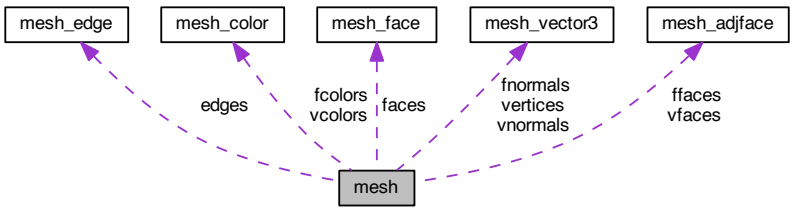
Chapter 5

Data Structure Documentation

5.1 mesh Struct Reference

`#include <meshlib.h>`

Collaboration diagram for mesh:



Data Fields

- uint8_t origin_type
- uint8_t is_loaded
- uint8_t is_vertices
- uint8_t is_faces
- uint8_t is_edges
- uint8_t is_vnormals
- uint8_t is_fnormals
- uint8_t is_vcolors
- uint8_t is_fcolors
- uint8_t is_vfaces
- uint8_t is_ffaces
- uint8_t is_fareas
- uint8_t is_vscalars
- uint8_t is_fscalars
- INTDATA num_vertices
- INTDATA num_faces

- INTDATA num_edges
- MESH_VERTEX vertices
- MESH_FACE faces
- MESH_EDGE edges
- MESH_NORMAL vnormals
- MESH_NORMAL fnormals
- MESH_COLOR vcolors
- MESH_COLOR fcolors
- MESH_VFACE vfaces
- MESH_FFACE ffaces
- MESH_SCALAR fareas
- MESH_SCALAR vscalars
- MESH_SCALAR fscalars
- uint8_t is_trimesh
- uint8_t dummy

5.1.1 Field Documentation

5.1.1.1 dummy

```
uint8_t dummy
```

5.1.1.2 edges

```
MESH_EDGE edges
```

Pointer to edges

5.1.1.3 faces

```
MESH_FACE faces
```

Pointer to faces

5.1.1.4 fareas

```
MESH_SCALAR fareas
```

Pointer to face areas

5.1.1.5 fcolors

`MESH_COLOR fcolors`

Pointer to face colors

5.1.1.6 ffaces

`MESH_FFACE ffaces`

Pointer to face adjacent faces

5.1.1.7 fnormals

`MESH_NORMAL fnormals`

Pointer to face normals

5.1.1.8 fscalars

`MESH_SCALAR fscalars`

Pointer to face scalar field scalars

5.1.1.9 is_edges

`uint8_t is_edges`

Has edges?

5.1.1.10 is_faces

`uint8_t is_faces`

Has faces?

5.1.1.11 is_fareas

`uint8_t is_fareas`

Has face areas?

5.1.1.12 is_fcolors

`uint8_t is_fcolors`

Has face colors?

5.1.1.13 is_ffaces

```
uint8_t is_ffaces
```

Has face adjacent faces?

5.1.1.14 is_fnormals

```
uint8_t is_fnormals
```

Has face normals?

5.1.1.15 is_fscalars

```
uint8_t is_fscalars
```

Has face scalar field scalars?

5.1.1.16 is_loaded

```
uint8_t is_loaded
```

Is loaded?

5.1.1.17 is_trimesh

```
uint8_t is_trimesh
```

Is trimesh?

5.1.1.18 is_vcolors

```
uint8_t is_vcolors
```

Has vertex colors?

5.1.1.19 is_vertices

```
uint8_t is_vertices
```

Has vertices?

5.1.1.20 is_vfaces

```
uint8_t is_vfaces
```

Has vertex adjacent faces?

5.1.1.21 is_vnormals

```
uint8_t is_vnormals
```

Has vertex normals?

5.1.1.22 is_vscalars

```
uint8_t is_vscalars
```

Has vertex scalar field scalars?

5.1.1.23 num_edges

```
INTDATA num_edges
```

Number of edges

5.1.1.24 num_faces

```
INTDATA num_faces
```

Number of faces

5.1.1.25 num_vertices

```
INTDATA num_vertices
```

Number of vertices

5.1.1.26 origin_type

```
uint8_t origin_type
```

Origin type

5.1.1.27 vcolors

```
MESH_COLOR vcolors
```

Pointer to vertex colors

5.1.1.28 vertices

```
MESH_VERTEX vertices
```

Pointer to vertices

5.1.1.29 vfaces

```
MESH_VFACE vfaces
```

Pointer to vertex adjacent faces

5.1.1.30 vnormals

```
MESH_NORMAL vnormals
```

Pointer to vertex normals

5.1.1.31 vscalars

```
MESH_SCALAR vscalars
```

Pointer to vertex scalar field scalars

5.2 mesh_adjface Struct Reference

```
#include <meshlib.h>
```

Data Fields

- INTDATA num_faces
- INTDATA * faces

5.2.1 Field Documentation

5.2.1.1 faces

```
INTDATA* faces
```

Pointer to adjacent face indices

5.2.1.2 num_faces

```
INTDATA num_faces
```

Number of adjacent faces

5.3 mesh_affine Struct Reference

```
#include <meshlib.h>
```

Data Fields

- FLOATDATA data [12]

5.3.1 Field Documentation

5.3.1.1 data

```
FLOATDATA data[12]
```

3X4 Row-major matrix data

5.4 mesh_color Struct Reference

```
#include <meshlib.h>
```

Data Fields

- FLOATDATA r
- FLOATDATA g
- FLOATDATA b
- FLOATDATA a

5.4.1 Field Documentation

5.4.1.1 a

```
FLOATDATA a
```

Alpha channel

5.4.1.2 b

```
FLOATDATA b
```

Green channel

5.4.1.3 g

```
FLOATDATA g
```

Blue channel

5.4.1.4 r

```
FLOATDATA r
```

Red channel

5.5 mesh_edge Struct Reference

```
#include <meshlib.h>
```

Data Fields

- INTDATA vertices [2]
- INTDATA faces [2]

5.5.1 Field Documentation

5.5.1.1 faces

```
INTDATA faces[2]
```

Edge faces

5.5.1.2 vertices

```
INTDATA vertices[2]
```

Edge vertices

5.6 mesh_face Struct Reference

```
#include <meshlib.h>
```

Data Fields

- INTDATA num_vertices
- INTDATA * vertices

5.6.1 Field Documentation

5.6.1.1 num_vertices

```
INTDATA num_vertices
```

Number of vertices

5.6.1.2 vertices

```
INTDATA* vertices
```

Pointer to vertex indices

5.7 mesh_rotation Struct Reference

```
#include <meshlib.h>
```

Data Fields

- FLOATDATA data [9]

5.7.1 Field Documentation

5.7.1.1 data

```
FLOATDATA data[9]
```

3X3 Row-major matrix data

5.8 mesh_struct Struct Reference

```
#include <meshlib.h>
```

Data Fields

- INTDATA num_items
- INTDATA * items

5.8.1 Field Documentation

5.8.1.1 items

```
INTDATA* items
```

Pointer to INTDATA items

5.8.1.2 num_items

```
INTDATA num_items
```

Number of items

5.9 mesh_struct2 Struct Reference

```
#include <meshlib.h>
```

Data Fields

- INTDATA num_items
- INTDATA2 * items

5.9.1 Field Documentation

5.9.1.1 items

```
INTDATA2* items
```

Pointer to INTDATA2 items

5.9.1.2 num_items

```
INTDATA num_items
```

Number of items

5.10 mesh_struct3 Struct Reference

```
#include <meshlib.h>
```

Data Fields

- INTDATA num_items
- INTDATA3 * items

5.10.1 Field Documentation

5.10.1.1 items

```
INTDATA3* items
```

Pointer to INTDATA3 items

5.10.1.2 num_items

```
INTDATA num_items
```

Number of items

5.11 mesh_vector2 Struct Reference

```
#include <meshlib.h>
```

Data Fields

- FLOATDATA x
- FLOATDATA y

5.11.1 Field Documentation

5.11.1.1 x

```
FLOATDATA x
```

x co-ordinate

5.11.1.2 y

```
FLOATDATA y
```

y co-ordinate

5.12 mesh_vector3 Struct Reference

```
#include <meshlib.h>
```

Data Fields

- FLOATDATA x
- FLOATDATA y
- FLOATDATA z

5.12.1 Field Documentation

5.12.1.1 x

```
FLOATDATA x
```

x co-ordinate

5.12.1.2 y

```
FLOATDATA y
```

y co-ordinate

5.12.1.3 z

```
FLOATDATA z
```

z co-ordinate

Chapter 6

File Documentation

6.1 computenormals.md File Reference

6.2 drawmesh.md File Reference

6.3 examples.md File Reference

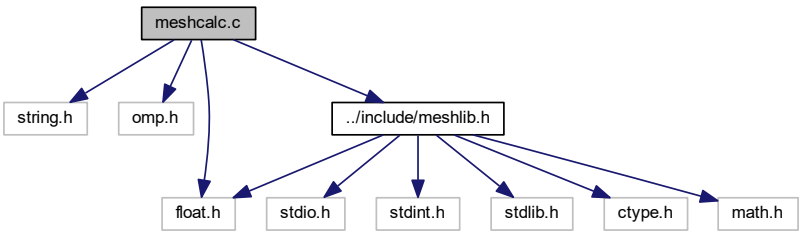
6.4 mainpage.md File Reference

6.5 meshcalc.c File Reference

This file contains functions pertaining to different mesh computations.

```
#include <string.h>
#include <omp.h>
#include <float.h>
#include "../include/meshlib.h"
```

Include dependency graph for meshcalc.c:



Functions

- void mesh_cross_vector3 (MESH_VECTOR3 x, MESH_VECTOR3 y, MESH_VECTOR3 z)

  Computes the cross product of two 3-d vectors.
- void mesh_cross_normal (MESH_NORMAL x, MESH_NORMAL y, MESH_NORMAL z)

  Computes the normalized cross product of two normals.
- void mesh_calc_face_normal (MESH_VERTEX v1, MESH_VERTEX v2, MESH_VERTEX v3, MESH_NORMAL n)

  Computes the face normal given 3 vertices.
- int mesh_calc_vertex_normals (MESH m)

  Computes vertex normals of a given mesh.
- int mesh_calc_face_normals (MESH m)

  Computes face normals of a given mesh.
- int mesh_calc_edges (MESH m)

  Computes edges of a given mesh.
- int mesh_calc_vertex_adjacency (MESH m)

  Computes vertex adjacent faces of a given mesh.
- int mesh_calc_face_adjacency (MESH m)

  Computes face adjacent faces of a given mesh.
- INTDATA mesh_find (MESH_STRUCT s, INTDATA q)

  Finds an item in an INTDATA structure.
- INTDATA mesh_find2 (MESH_STRUCT2 s, INTDATA q)

  Finds an item in an INTDATA2 structure.
- INTDATA mesh_find3 (MESH_STRUCT3 s, INTDATA q)

  Finds an item in an INTDATA3 structure.
- int mesh_upsample (MESH m, int iters)

  Upsamples a given mesh.
- int mesh_upsample_loop (MESH m, int iters)

  Upsamples a given mesh using Loop's algorithm.
- int mesh_upsample_tarea_adaptive (MESH m, int miters, FLOATDATA e)

  Upsamples a given mesh upto a given triangle area threshold.
- FLOATDATA mesh_calc_triangle_area (MESH_VERTEX a, MESH_VERTEX b, MESH_VERTEX c)

  Computes area of a triangle.
- void mesh_calc_aabb (MESH m, MESH_VECTOR3 minv, MESH_VECTOR3 maxv, MESH_VECTOR3 center)

  Computes axis-aligned bounding box.
- int mesh_calc_signed_area (MESH m, MESH_VECTOR3 area)

  Computes signed area of a triangle mesh.
- FLOATDATA mesh_calc_area (MESH m)

  Computes area of a triangle mesh.
- FLOATDATA mesh_calc_volume (MESH m)

  Computes volume of a triangle mesh.

6.5.1 Detailed Description

This file contains functions pertaining to different mesh computations.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.5.2 Function Documentation

6.5.2.1 mesh_calc_aabb()

```
void mesh_calc_aabb (
            MESH m,
            MESH_VECTOR3 minv,
            MESH_VECTOR3 maxv,
            MESH_VECTOR3 center )
```

Computes axis-aligned bounding box.

Parameters

in	*m*	Given mesh
out	*minv*	Minimum point
out	*maxv*	Maximum point
out	*center*	Centre point

6.5.2.2 mesh_calc_area()

```
FLOATDATA mesh_calc_area (
            MESH m )
```

Computes area of a triangle mesh.

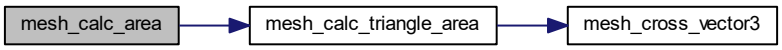
Parameters

in	*m*	Given mesh

Returns

Area (INF, if not trimesh)

Here is the call graph for this function:



6.5.2.3 mesh_calc_edges()

```
int mesh_calc_edges (
            MESH m )
```

Computes edges of a given mesh.

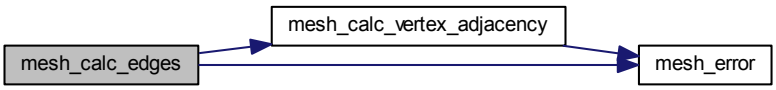
Parameters

in	*m*	Input mesh

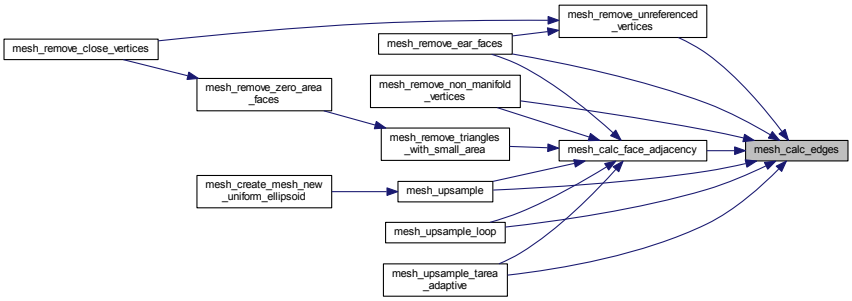
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.4 mesh_calc_face_adjacency()

```
int mesh_calc_face_adjacency (
            MESH m )
```

Computes face adjacent faces of a given mesh.

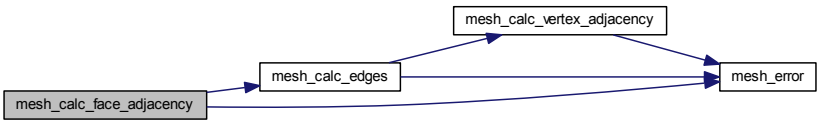
Parameters

in	*m*	Input mesh

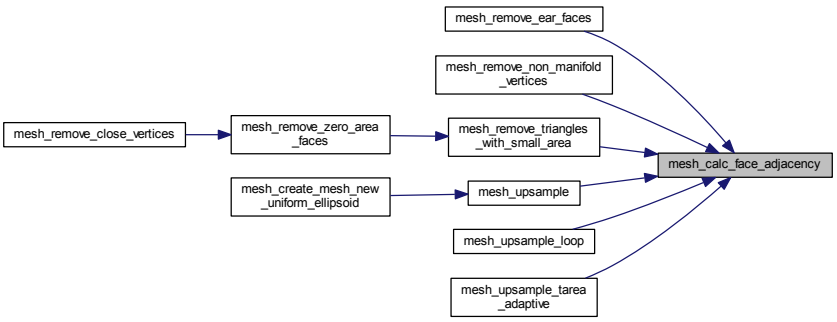
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.5 mesh_calc_face_normal()

```
void mesh_calc_face_normal (
            MESH_VERTEX v1,
            MESH_VERTEX v2,
            MESH_VERTEX v3,
            MESH_NORMAL n )
```

Computes the face normal given 3 vertices.

Parameters

in	*v1*	First vertex
in	*v2*	Second vertex
in	*v3*	Third vertex
out	*n*	Output face normal $\mathbf{n}_f$

Returns

NULL

6.5.2.6 mesh_calc_face_normals()

```
int mesh_calc_face_normals (
            MESH m )
```

Computes face normals of a given mesh.

Parameters

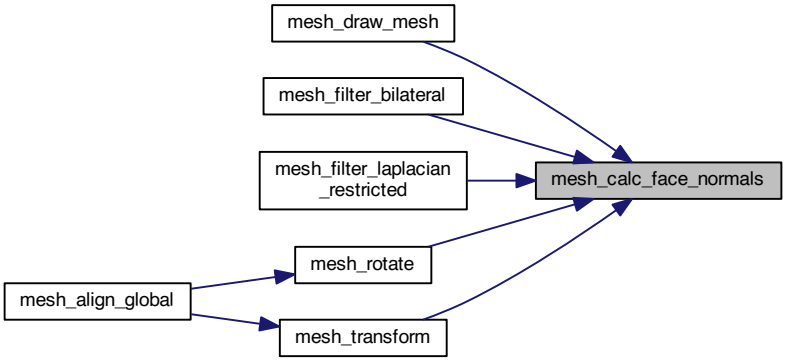
in	*m*	Input mesh

Returns

Error code

Here is the call graph for this function:

Here is the caller graph for this function:



6.5.2.7 mesh_calc_signed_area()

```
int mesh_calc_signed_area (
            MESH m,
            MESH_VECTOR3 area )
```

Computes signed area of a triangle mesh.

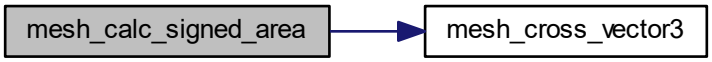
Parameters

in	*m*	Given mesh
in	*area*	Output area

Returns

Error code

Here is the call graph for this function:



6.5.2.8 mesh_calc_triangle_area()

```
FLOATDATA mesh_calc_triangle_area (
              MESH_VERTEX a,
              MESH_VERTEX b,
              MESH_VERTEX c )
```

Computes area of a triangle.

Parameters

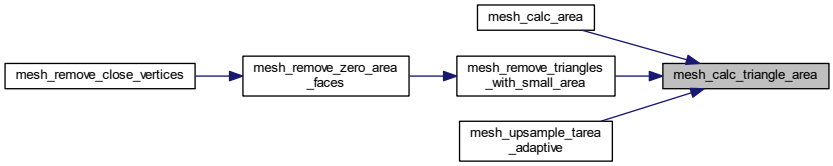
in	*a*	First vertex
in	*b*	Second vertex
in	*c*	Third vertex

Returns

Area

Here is the call graph for this function:

Here is the caller graph for this function:



6.5.2.9 mesh_calc_vertex_adjacency()

```
int mesh_calc_vertex_adjacency (
            MESH m )
```

Computes vertex adjacent faces of a given mesh.

Parameters

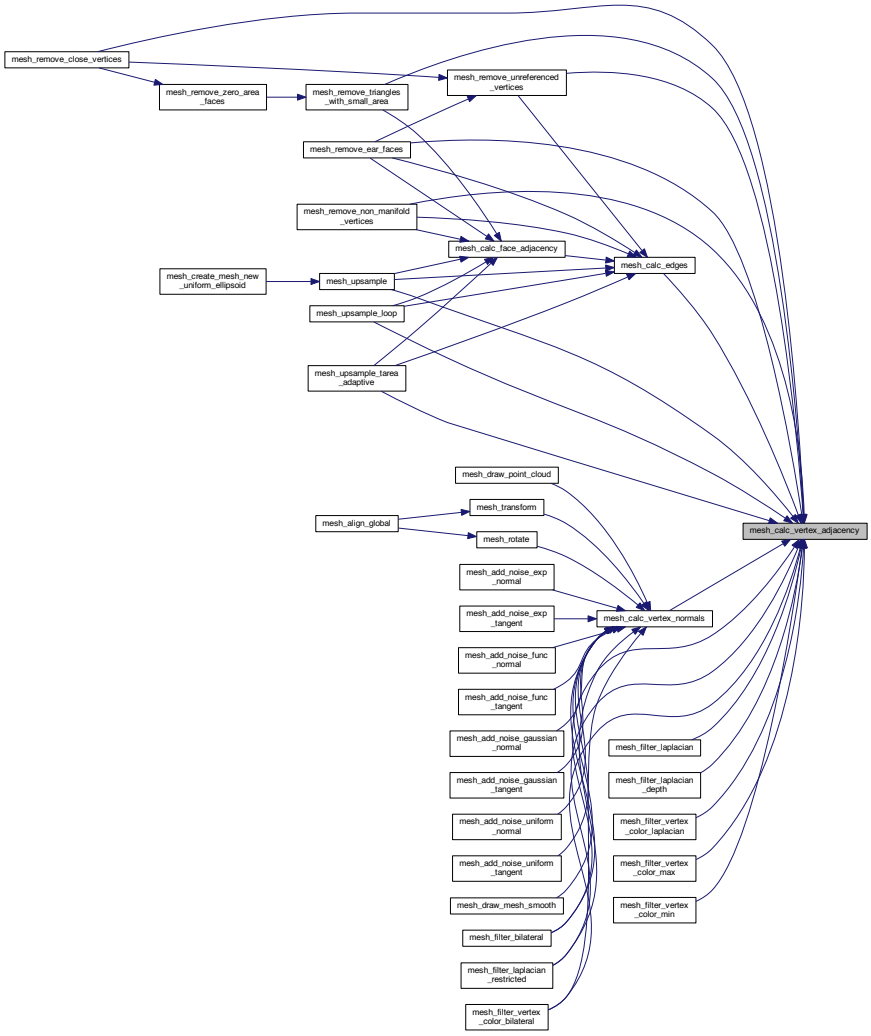
in	*m*	Input mesh

Returns

Error code

Here is the call graph for this function:

Here is the caller graph for this function:



6.5.2.10 mesh_calc_vertex_normals()

```
int mesh_calc_vertex_normals (
            MESH m )
```

Computes vertex normals of a given mesh.

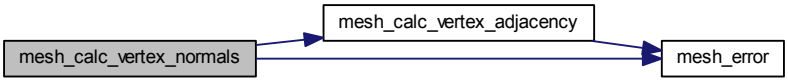
Parameters

in	*m*	Input mesh

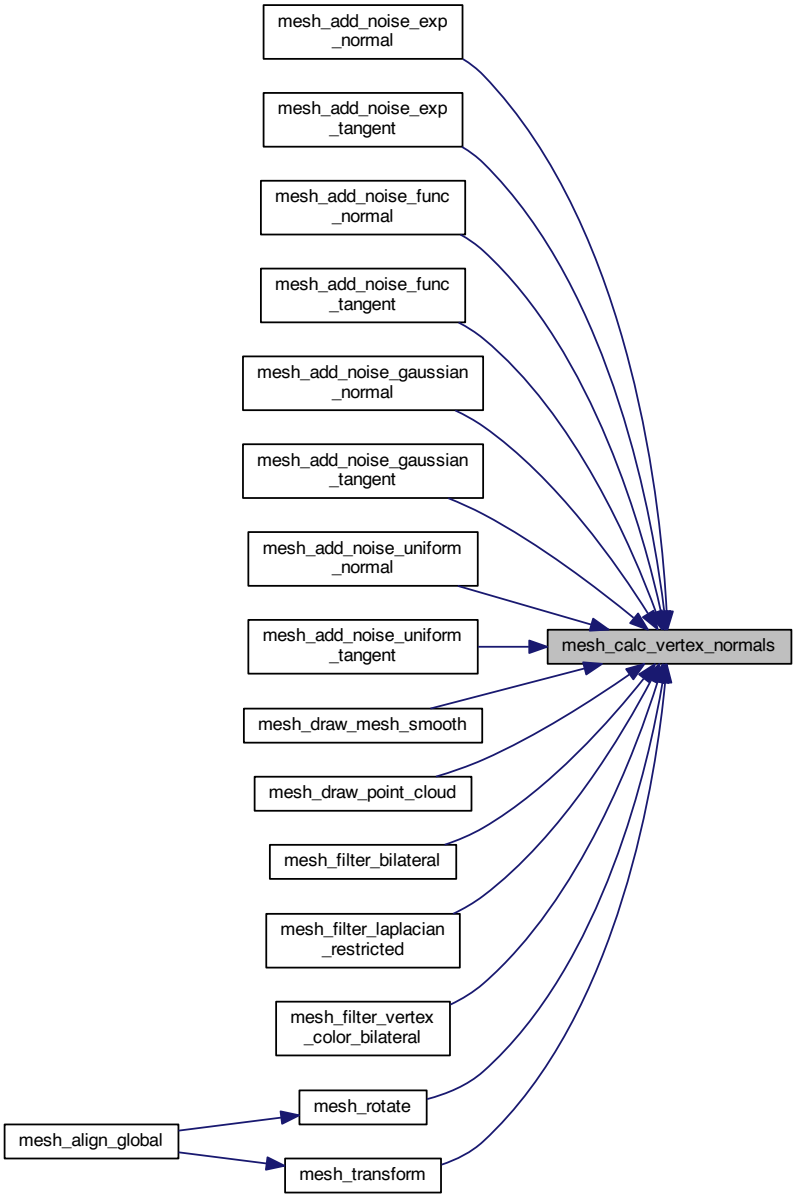
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.11 mesh_calc_volume()

```
FLOATDATA mesh_calc_volume (
            MESH m )
```

Computes volume of a triangle mesh.

Parameters

in	*m*	Given mesh

Returns

Volume (INF, if not trimesh)

Here is the call graph for this function:

Here is the caller graph for this function:

6.5.2.12 mesh_cross_normal()

```
void mesh_cross_normal (
            MESH_NORMAL x,
            MESH_NORMAL y,
            MESH_NORMAL z )
```

Computes the normalized cross product of two normals.

Parameters

in	*x*	First normal
in	*y*	Second normal
out	*z*	Output cross product $\frac{\mathbf{x}\times\mathbf{y}}{\|\mathbf{x}\times\mathbf{y}\|_2}$

Returns

NULL

6.5.2.13 mesh_cross_vector3()

```
void mesh_cross_vector3 (
            MESH_VECTOR3 x,
            MESH_VECTOR3 y,
            MESH_VECTOR3 z )
```

Computes the cross product of two 3-d vectors.

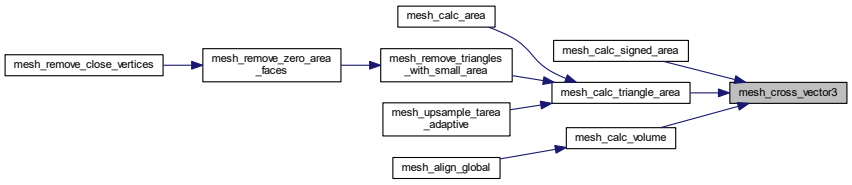
Parameters

in	*x*	First vector
in	*y*	Second vector
out	*z*	Output cross product $\mathbf{x} \times \mathbf{y}$

Returns

NULL

Here is the caller graph for this function:



6.5.2.14 mesh_find()

```
INTDATA mesh_find (
            MESH_STRUCT s,
            INTDATA q )
```

Finds an item in an INTDATA structure.

Parameters

in	*s*	Input INTDATA structure
in	*q*	Query INTDATA

Returns

Index or -1

6.5.2.15 mesh_find2()

```
INTDATA mesh_find2 (
            MESH_STRUCT2 s,
            INTDATA q )
```

Finds an item in an INTDATA2 structure.

Parameters

in	*s*	Input INTDATA2 structure
in	*q*	Query INTDATA2

Returns

Index or -1

6.5.2.16 mesh_find3()

```
INTDATA mesh_find3 (
            MESH_STRUCT3 s,
            INTDATA q )
```

Finds an item in an INTDATA3 structure.

Parameters

in	*s*	Input INTDATA3 structure
in	*q*	Query INTDATA3

Returns

Index or -1

6.5.2.17 mesh_upsample()

```
int mesh_upsample (
            MESH m,
            int iters )
```

Upsamples a given mesh.

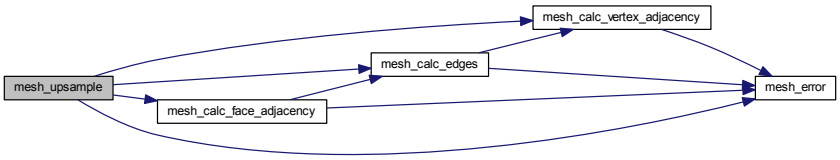
Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

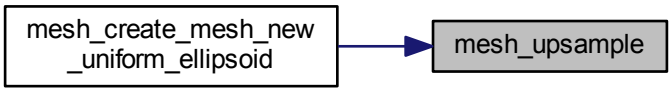
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.18 mesh_upsample_loop()

```
int mesh_upsample_loop (
            MESH m,
            int iters )
```

Upsamples a given mesh using Loop's algorithm.

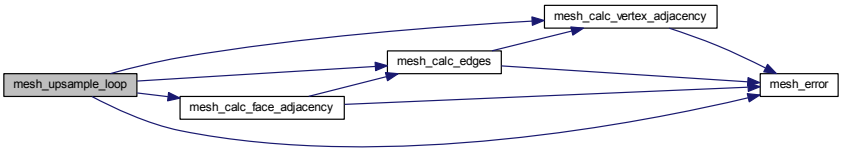
Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.5.2.19 mesh_upsample_tarea_adaptive()

```
int mesh_upsample_tarea_adaptive (
            MESH m,
            int miters,
            FLOATDATA e )
```

Upsamples a given mesh upto a given triangle area threshold.

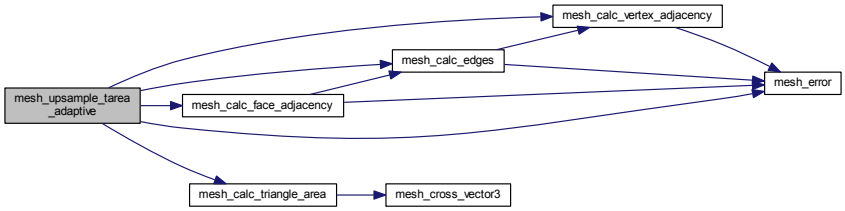
Parameters

in	*m*	Input mesh
in	*miters*	Maximum number of iterations
in	*e*	Triangle area threshold

Returns

Error code

Here is the call graph for this function:

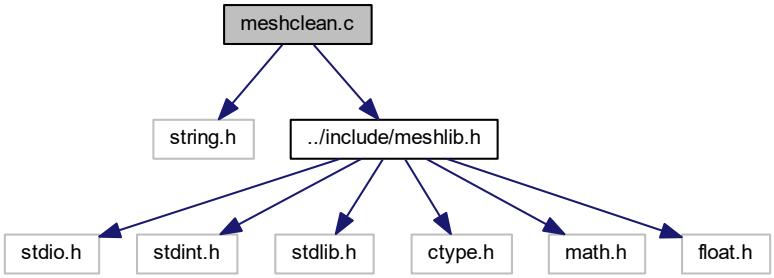


6.6 meshclean.c File Reference

This file contains functions pertaining to different mesh cleaning algorithms.

```
#include <string.h>
#include "../include/meshlib.h"
```

Include dependency graph for meshclean.c:



Functions

- int mesh_remove_boundary_vertices (MESH m, int iters)

  Removes boundary vertices and connecting elements.
- int mesh_remove_boundary_faces (MESH m, int iters)

  Removes boundary faces and connecting elements.
- int mesh_remove_triangles_with_small_area (MESH m, FLOATDATA area)

  Removes triangles with area smaller than a given value.
- int mesh_remove_zero_area_faces (MESH m)

  Removes triangles with zero area.
- int mesh_remove_unreferenced_vertices (MESH m)

  Removes unreferenced vertices.
- int mesh_remove_ear_faces (MESH m, int niters)

  Removes ear faces and connecting vertices.
- int mesh_remove_close_vertices (MESH m, FLOATDATA r)

  Removes close vertices.
- int mesh_remove_non_manifold_vertices (MESH m)

  Removes non-manifold vertices.

6.6.1 Detailed Description

This file contains functions pertaining to different mesh cleaning algorithms.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.6.2 Function Documentation

6.6.2.1 mesh_remove_boundary_faces()

```
int mesh_remove_boundary_faces (
            MESH m,
            int iters )
```

Removes boundary faces and connecting elements.

Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

Returns

Error code

6.6.2.2 mesh_remove_boundary_vertices()

```
int mesh_remove_boundary_vertices (
            MESH m,
            int iters )
```

Removes boundary vertices and connecting elements.

Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

Returns

Error code

6.6.2.3 mesh_remove_close_vertices()

```
int mesh_remove_close_vertices (
            MESH m,
            FLOATDATA r )
```

Removes close vertices.

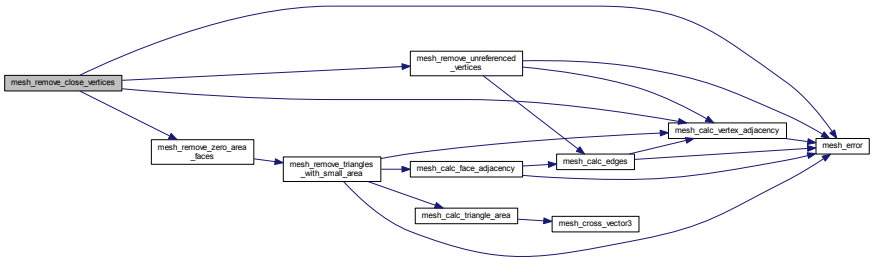
Parameters

in	*m*	Input mesh
in	*r*	Maximum distance between two vertices

Returns

Error code

Here is the call graph for this function:



6.6.2.4 mesh_remove_ear_faces()

```
int mesh_remove_ear_faces (
            MESH m,
            int niters )
```

Removes ear faces and connecting vertices.

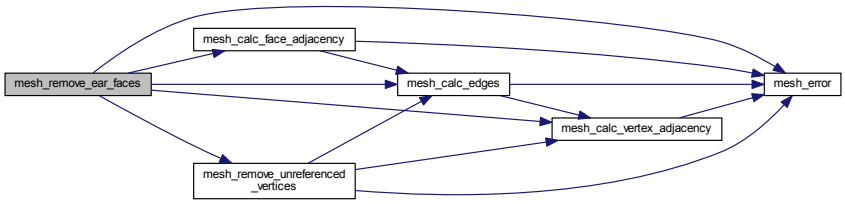
Parameters

in	*m*	Input mesh
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.6.2.5 mesh_remove_non_manifold_vertices()

```
int mesh_remove_non_manifold_vertices (
            MESH m )
```

Removes non-manifold vertices.

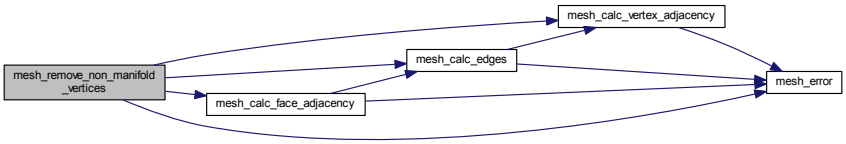
Parameters

in	*m*	Input mesh

Returns

Error code

Here is the call graph for this function:



6.6.2.6 mesh_remove_triangles_with_small_area()

```
int mesh_remove_triangles_with_small_area (
            MESH m,
            FLOATDATA area )
```

Removes triangles with area smaller than a given value.

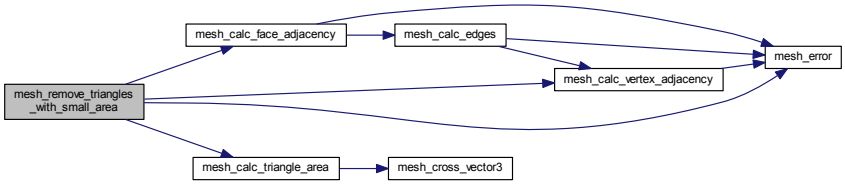
Parameters

in	*m*	Input mesh
in	*area*	Given area

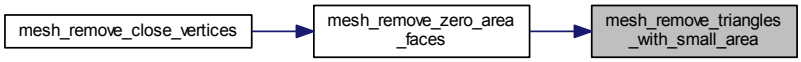
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.7 mesh_remove_unreferenced_vertices()

```
int mesh_remove_unreferenced_vertices (
            MESH m )
```

Removes unreferenced vertices.

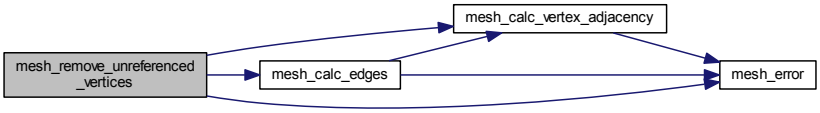
Parameters

in	*m*	Input mesh

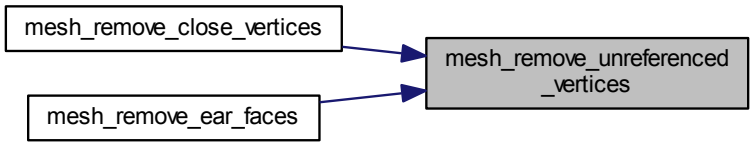
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.8 mesh_remove_zero_area_faces()

```
int mesh_remove_zero_area_faces (
            MESH m )
```

Removes triangles with zero area.

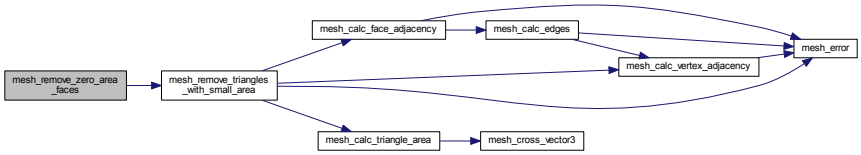
Parameters

in	*m*	Input mesh

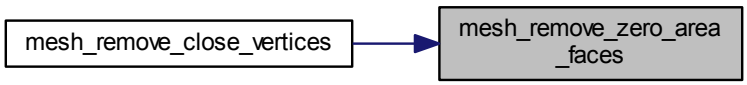
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

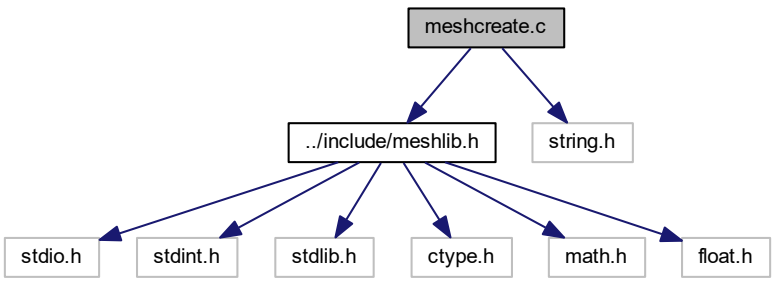


6.7 meshcreate.c File Reference

This file contains functions pertaining to mesh creation and freeing.

```
#include "../include/meshlib.h"
#include <string.h>
```

Include dependency graph for meshcreate.c:



Functions

- MESH mesh_create_mesh_new ()

  Creates a new mesh.
- void mesh_free_mesh (MESH m)

  Frees a mesh.
- MESH mesh_create_mesh_new_grid (MESH_VECTOR3 sz, MESH_VECTOR3 pos, INTDATA m, INTDATA n)

  Creates a grid mesh.
- MESH mesh_create_mesh_new_cuboid (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a cuboid mesh.
- MESH mesh_create_mesh_new_ellipsoid (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates an ellipsoid mesh.
- MESH mesh_create_mesh_new_uniform_ellipsoid (MESH_VECTOR3 sz, MESH_VECTOR3 pos, int n)

  Creates a uniformly sampled ellipsoid mesh.
- MESH mesh_create_mesh_new_cylinder (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a cylinder mesh.
- MESH mesh_create_mesh_new_cone (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a cone mesh.
- MESH mesh_create_mesh_new_rectangle_flat (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a flat rectangle mesh.
- MESH mesh_create_mesh_new_ellipse_flat (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a flat ellipse mesh.

6.7.1 Detailed Description

This file contains functions pertaining to mesh creation and freeing.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.7.2 Function Documentation

6.7.2.1 mesh_create_mesh_new()

```
MESH mesh_create_mesh_new ( )
```

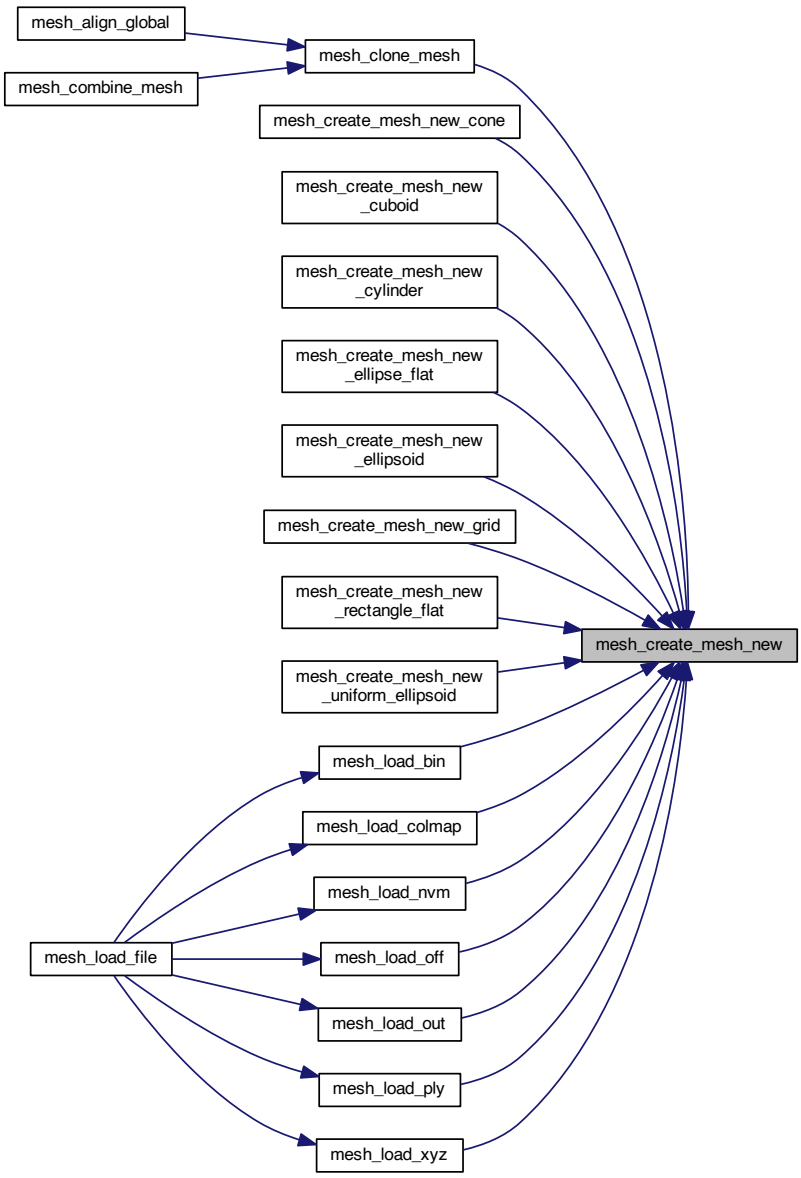
Creates a new mesh.

Returns

Output mesh

Here is the call graph for this function:

Here is the caller graph for this function:



6.7.2.2 mesh_create_mesh_new_cone()

```
MESH mesh_create_mesh_new_cone (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a cone mesh.

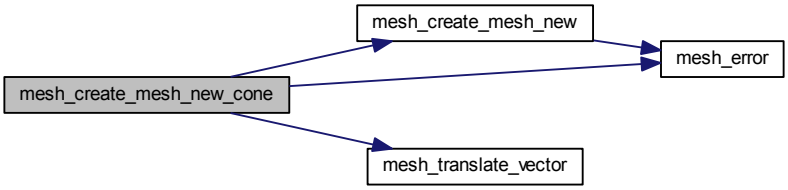
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.7.2.3 mesh_create_mesh_new_cuboid()

```
MESH mesh_create_mesh_new_cuboid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a cuboid mesh.

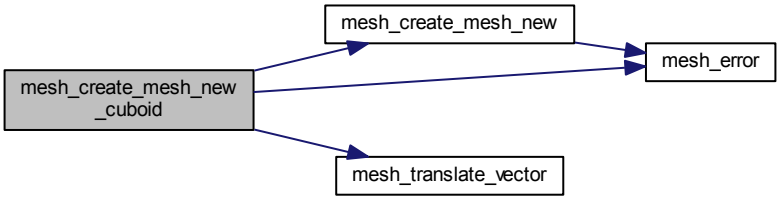
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.7.2.4 mesh_create_mesh_new_cylinder()

```
MESH mesh_create_mesh_new_cylinder (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a cylinder mesh.

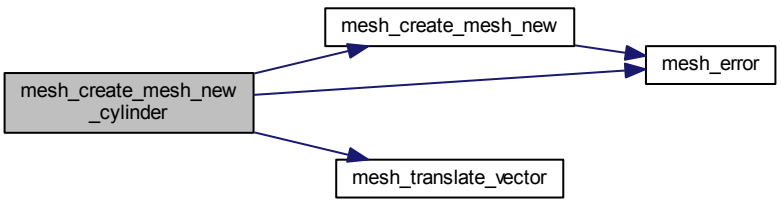
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.7.2.5 mesh_create_mesh_new_ellipse_flat()

```
MESH mesh_create_mesh_new_ellipse_flat (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a flat ellipse mesh.

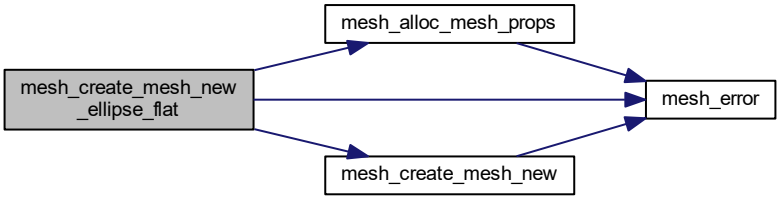
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.7.2.6 mesh_create_mesh_new_ellipsoid()

```
MESH mesh_create_mesh_new_ellipsoid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates an ellipsoid mesh.

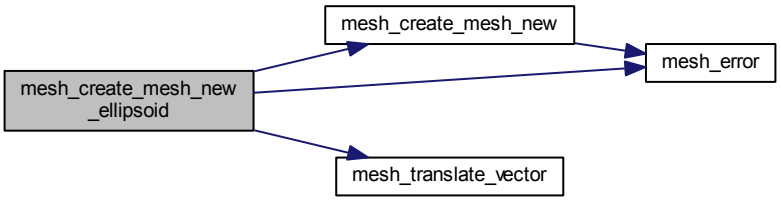
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.7.2.7 mesh_create_mesh_new_grid()

```
MESH mesh_create_mesh_new_grid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos,
            INTDATA m,
            INTDATA n )
```

Creates a grid mesh.

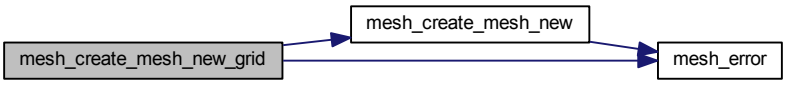
Parameters

in	*sz*	Size vector
in	*pos*	Position vector
in	*m*	Number of x-samples
in	*n*	Number of y-samples

Returns

Output mesh

Here is the call graph for this function:



6.7.2.8 mesh_create_mesh_new_rectangle_flat()

```
MESH mesh_create_mesh_new_rectangle_flat (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a flat rectangle mesh.

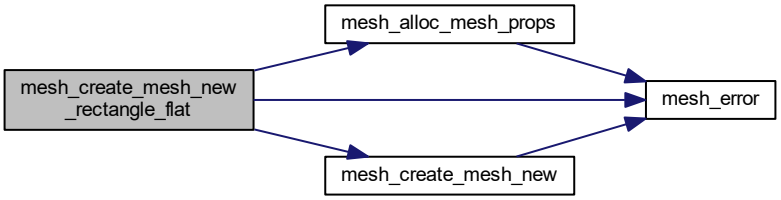
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.7.2.9 mesh_create_mesh_new_uniform_ellipsoid()

```
MESH mesh_create_mesh_new_uniform_ellipsoid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos,
            int n )
```

Creates a uniformly sampled ellipsoid mesh.

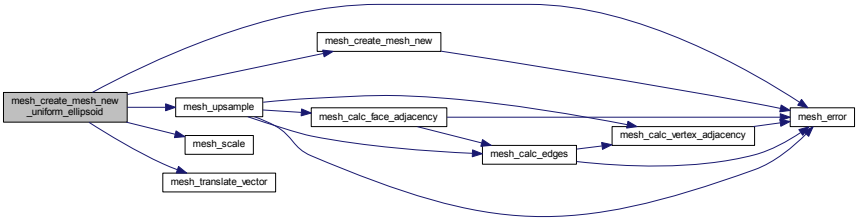
Parameters

in	*sz*	Size vector
in	*pos*	Position vector
in	*n*	Number of upsample ~3

Returns

Output mesh

Here is the call graph for this function:



6.7.2.10 mesh_free_mesh()

```
void mesh_free_mesh (
            MESH m )
```

Frees a mesh.

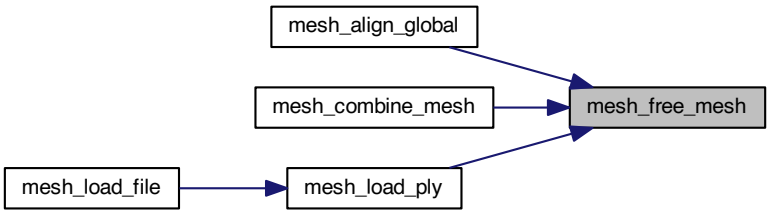
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the caller graph for this function:

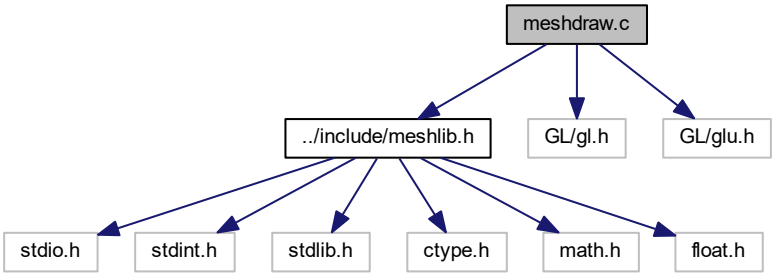


6.8 meshdraw.c File Reference

This file contains functions pertaining to mesh drawing in OpenGL.

```
#include "../include/meshlib.h"
#include <GL/gl.h>
#include <GL/glu.h>
```

Include dependency graph for meshdraw.c:



Functions

- void mesh_draw_mesh (MESH m)

  Draws a given mesh in OpenGL context in flat shading.
- void mesh_draw_mesh_smooth (MESH m)

  Draws a given mesh in OpenGL context in smoothing shading.
- void mesh_draw_point_cloud (MESH m)

  Draws a given mesh in OpenGL context as pointcloud.

6.8.1 Detailed Description

This file contains functions pertaining to mesh drawing in OpenGL.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.8.2 Function Documentation

6.8.2.1 mesh_draw_mesh()

```
void mesh_draw_mesh (
            MESH m )
```

Draws a given mesh in OpenGL context in flat shading.

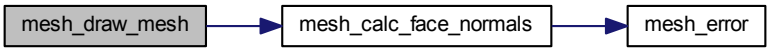
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:



6.8.2.2 mesh_draw_mesh_smooth()

```
void mesh_draw_mesh_smooth (
            MESH m )
```

Draws a given mesh in OpenGL context in smoothing shading.

Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:

6.8.2.3 mesh_draw_point_cloud()

```
void mesh_draw_point_cloud (
            MESH m )
```

Draws a given mesh in OpenGL context as pointcloud.

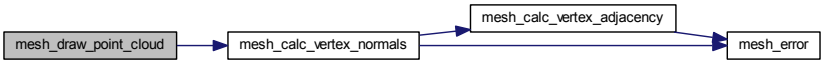
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:

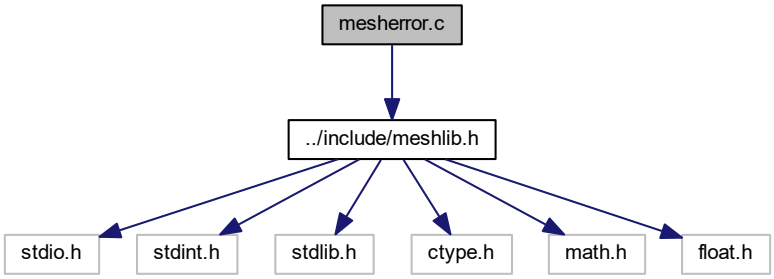


6.9 mesherror.c File Reference

This file contains functions pertaining to handling errors.

```
#include "../include/meshlib.h"
```

Include dependency graph for mesherror.c:



Functions

- void mesh_error (int type)

  Displays error message and exits.

6.9.1 Detailed Description

This file contains functions pertaining to handling errors.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.9.2 Function Documentation

6.9.2.1 mesh_error()

```
void mesh_error (
            int type )
```

Displays error message and exits.

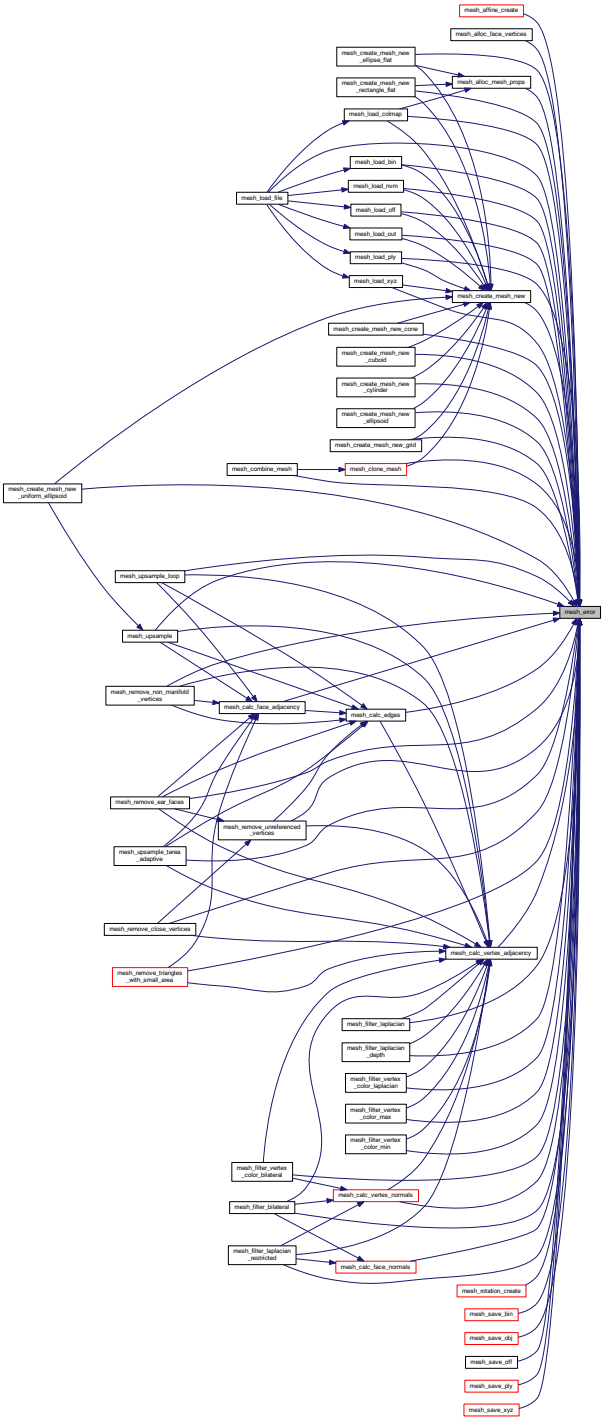
Parameters

in	*type*	Error type (MESH_ERR_MALLOC/MESH_ERR_SIZE_MISMATCH/MESH_ERR_FNOTOPEN)

Returns

NULL

Here is the caller graph for this function:

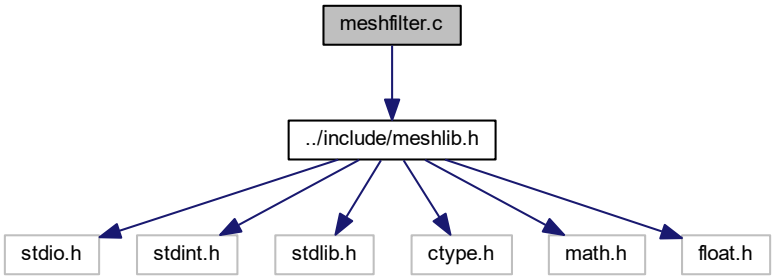


6.10 meshfilter.c File Reference

This file contains functions pertaining to different mesh filtering algorithms.

```
#include "../include/meshlib.h"
```

Include dependency graph for meshfilter.c:



Functions

- int mesh_filter_bilateral (MESH m, FLOATDATA sigma_c, FLOATDATA sigma_s, int niters)

  Mesh bilateral filter.
- int mesh_filter_laplacian (MESH m, FLOATDATA r)

  Mesh Laplacian filter.
- int mesh_filter_laplacian_restricted (MESH m, FLOATDATA r, FLOATDATA ang)

  Restricted Mesh Laplacian filter.
- int mesh_filter_laplacian_depth (MESH m, FLOATDATA r, MESH_VECTOR3 vp)

  Mesh Depth Laplacian filter.
- int mesh_filter_taubin (MESH m, FLOATDATA lambd, FLOATDATA mu, int niters)

  Mesh $\lambda - \mu$ Taubin filter.
- int mesh_filter_vertex_color_bilateral (MESH m, FLOATDATA sigma_k, FLOATDATA sigma_c, FLOATDATA sigma_s, int niters)

  Mesh bilateral vertex color filter.
- int mesh_filter_vertex_color_laplacian (MESH m, FLOATDATA r)

  Mesh Laplacian vertex color filter.
- int mesh_filter_vertex_color_min (MESH m, INTDATA niters)

  Mesh minimum intensity vertex color filter.
- int mesh_filter_vertex_color_max (MESH m, INTDATA niters)

  Mesh maximum intensity vertex color filter.

6.10.1 Detailed Description

This file contains functions pertaining to different mesh filtering algorithms.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.10.2 Function Documentation

6.10.2.1 mesh_filter_bilateral()

```
int mesh_filter_bilateral (
            MESH m,
            FLOATDATA sigma_c,
            FLOATDATA sigma_s,
            int niters )
```

Mesh bilateral filter.

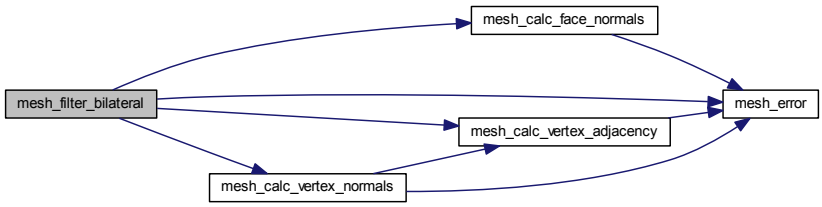
Parameters

in	*m*	Input mesh
in	*sigma_c*	Range standard deviation
in	*sigma_s*	Spatial standard deviation
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.10.2.2 mesh_filter_laplacian()

```
int mesh_filter_laplacian (
            MESH m,
            FLOATDATA r )
```

Mesh Laplacian filter.

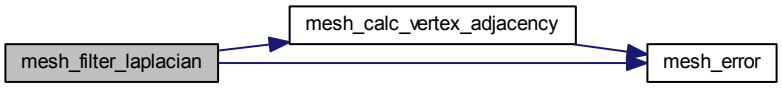
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion

Returns

Error code

Here is the call graph for this function:



6.10.2.3 mesh_filter_laplacian_depth()

```
int mesh_filter_laplacian_depth (
            MESH m,
            FLOATDATA r,
            MESH_VECTOR3 vp )
```

Mesh Depth Laplacian filter.

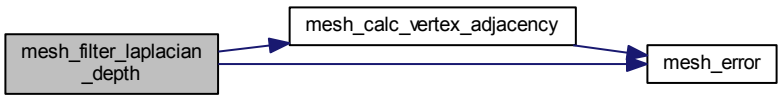
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion
in	*vp*	View-point

Returns

Error code

Here is the call graph for this function:



6.10.2.4 mesh_filter_laplacian_restricted()

```
int mesh_filter_laplacian_restricted (
            MESH m,
            FLOATDATA r,
            FLOATDATA ang )
```

Restricted Mesh Laplacian filter.

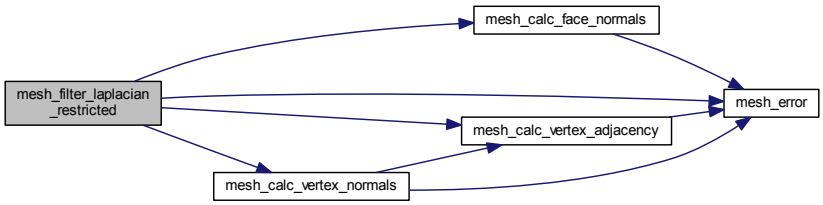
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion
in	*ang*	Minimum angle in degrees to suppress filtering

Returns

Error code

Here is the call graph for this function:



6.10.2.5 mesh_filter_taubin()

```
int mesh_filter_taubin (
            MESH m,
            FLOATDATA lambd,
            FLOATDATA mu,
            int niters )
```

Mesh $\lambda - \mu$ Taubin filter.

Parameters

in	*m*	Input mesh
in	*lambd*	λ value
in	*mu*	μ value
in	*niters*	Number of iterations

Returns

Error code

6.10.2.6 mesh_filter_vertex_color_bilateral()

```
int mesh_filter_vertex_color_bilateral (
            MESH m,
            FLOATDATA sigma_k,
            FLOATDATA sigma_c,
            FLOATDATA sigma_s,
            int niters )
```

Mesh bilateral vertex color filter.

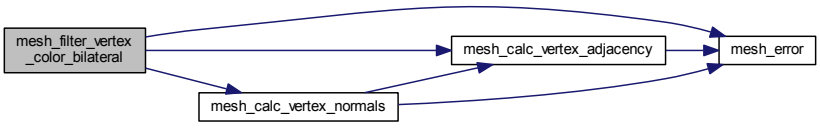
Parameters

in	*m*	Input mesh
in	*sigma_k*	Color standard deviation
in	*sigma_c*	Range standard deviation
in	*sigma_s*	Spatial standard deviation
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.10.2.7 mesh_filter_vertex_color_laplacian()

```
int mesh_filter_vertex_color_laplacian (
            MESH m,
            FLOATDATA r )
```

Mesh Laplacian vertex color filter.

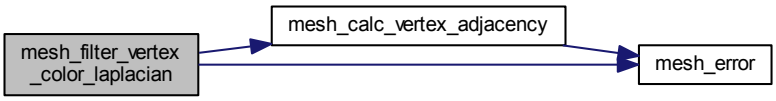
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion

Returns

Error code

Here is the call graph for this function:



6.10.2.8 mesh_filter_vertex_color_max()

```
int mesh_filter_vertex_color_max (
            MESH m,
            INTDATA niters )
```

Mesh maximum intensity vertex color filter.

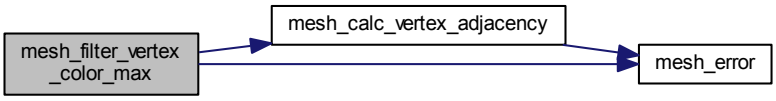
Parameters

in	*m*	Input mesh
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.10.2.9 mesh_filter_vertex_color_min()

```
int mesh_filter_vertex_color_min (
            MESH m,
            INTDATA niters )
```

Mesh minimum intensity vertex color filter.

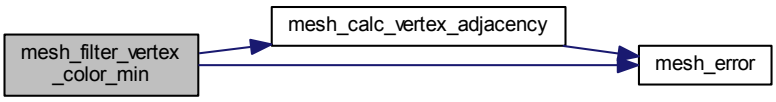
Parameters

in	*m*	Input mesh
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:

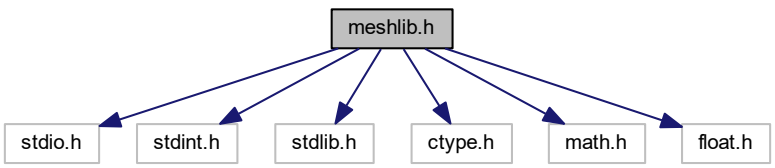


6.11 meshlib.h File Reference

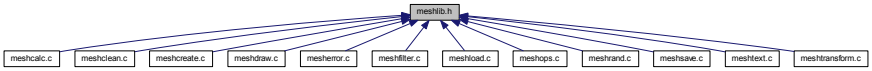
This header file contains declarations of all functions of meshlib.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <float.h>
```

Include dependency graph for meshlib.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct mesh_vector2
- struct mesh_vector3
- struct mesh_color
- struct mesh_struct
- struct mesh_struct2
- struct mesh_struct3
- struct mesh_face
- struct mesh_edge
- struct mesh_adjface
- struct mesh_rotation
- struct mesh_affine
- struct mesh

Macros

- #define __MESHLIB__
- #define _CRT_SECURE_NO_DEPRECATE
- #define MESHLIBAPI extern
- #define MESH_INTDATA_TYPE 1
- #define MESH_FLOATDATA_TYPE 1
- #define INTDATA int64_t /∗ do not change this, careful see meshload fscanf and other functions ∗/
- #define MESH_INTDATA_MIN (INT64_MIN)
- #define MESH_INTDATA_MAX (INT64_MAX)
- #define FLOATDATA double /∗ do not change this, careful see meshload fscanf and other functions ∗/
- #define MESH_FLOATDATA_MIN (DBL_MIN)
- #define MESH_FLOATDATA_MAX (DBL_MAX)
- #define MESH_ORIGIN_TYPE_BUILD 00
- #define MESH_ORIGIN_TYPE_OFF 11
- #define MESH_ORIGIN_TYPE_NOFF 12
- #define MESH_ORIGIN_TYPE_COFF 13
- #define MESH_ORIGIN_TYPE_NCOFF 14
- #define MESH_ORIGIN_TYPE_XYZ 20
- #define MESH_ORIGIN_TYPE_PLY_ASCII 30
- #define MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN 31
- #define MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN 32
- #define MESH_ORIGIN_TYPE_BINV1 40
- #define MESH_ORIGIN_TYPE_BUNDLE_OUT 50
- #define MESH_ORIGIN_TYPE_NVM 60
- #define MESH_ORIGIN_TYPE_BINCOLMAP 70
- #define MESH_ERR_MALLOC 0
- #define MESH_ERR_SIZE_MISMATCH 1
- #define MESH_ERR_FNOTOPEN 2
- #define MESH_ERR_INCOMPATIBLE 3

- #define MESH_ERR_UNKNOWN 4
- #define MESH_PI (3.14159265359)
- #define MESH_TWOPI (6.28318530718)
- #define MESH_PROPS_VERTICES (0x01)
- #define MESH_PROPS_VNORMALS (MESH_PROPS_VERTICES | __MESH_PROPS_VNORMALS)
- #define MESH_PROPS_VCOLORS (MESH_PROPS_VERTICES | __MESH_PROPS_VCOLORS)
- #define MESH_PROPS_VFACES (MESH_PROPS_VERTICES | __MESH_PROPS_VFACES)
- #define MESH_PROPS_VSCALARS (MESH_PROPS_VERTICES | __MESH_PROPS_VSCALARS)
- #define MESH_PROPS_V_ALL_PROPS (0xC00F)
- #define MESH_PROPS_FACES (MESH_PROPS_VERTICES | __MESH_PROPS_FACES)
- #define MESH_PROPS_FNORMALS (MESH_PROPS_FACES | __MESH_PROPS_FNORMALS)
- #define MESH_PROPS_FCOLORS (MESH_PROPS_FACES | __MESH_PROPS_FCOLORS)
- #define MESH_PROPS_FAREAS (MESH_PROPS_FACES | __MESH_PROPS_FAREAS)
- #define MESH_PROPS_FFACES (MESH_PROPS_FACES | __MESH_PROPS_FFACES)
- #define MESH_PROPS_FSCALARS (MESH_PROPS_FACES | __MESH_PROPS_FSCALARS)
- #define MESH_PROPS_F_ALL_PROPS (MESH_PROPS_FACES | __MESH_PROPS_F_ALL_PROPS)
- #define MESH_PROPS_EDGES (MESH_PROPS_VERTICES | __MESH_PROPS_FACES | __MESH_↩ PROPS_EDGES)
- #define MESH_PROPS_ALL_PROPS (0xFFFF)
- #define MESH_EPS20 (1e-20)
- #define MESH_EPS12 (1e-12)
- #define MESH_EPS8 (1e-8)
- #define MESH_EPSM (DBL_EPSILON)
- #define MESH_MIN(a, b) (((a)<(b))?(a):(b))
- #define MESH_MAX(a, b) (((a)>(b))?(a):(b))
- #define MESH_ALIGN_GLOBAL_POSITION (0x01)
- #define MESH_ALIGN_GLOBAL_ORIENTATION (0x02)
- #define MESH_ALIGN_GLOBAL_SCALE (0x04)
- #define MESH_ALIGN_GLOBAL_ALL (0x07)
- #define MESH_ALIGN_GLOBAL_DO_TRANSFORM (0x08)
- #define mesh_bilateral_filter mesh_filter_bilateral
- #define mesh_laplacian_filter mesh_filter_laplacian
- #define mesh_restricted_laplacian_filter mesh_filter_laplacian_restricted
- #define mesh_depth_laplacian_filter mesh_filter_laplacian_depth
- #define mesh_taubin_filter mesh_filter_taubin
- #define mesh_bilateral_vertex_color_filter mesh_filter_vertex_color_bilateral
- #define mesh_laplacian_vertex_color_filter mesh_filter_vertex_color_laplacian
- #define mesh_min_vertex_color_filter mesh_filter_vertex_color_min
- #define mesh_max_vertex_color_filter mesh_filter_vertex_color_max
- #define mesh_write_file mesh_save_file
- #define mesh_write_off mesh_save_off
- #define mesh_write_xyz mesh_save_xyz
- #define mesh_write_ply mesh_save_ply
- #define mesh_write_bin mesh_save_bin
- #define mesh_write_obj mesh_save_obj
- #define mesh_read_file mesh_load_file
- #define mesh_read_off mesh_load_off
- #define mesh_read_xyz mesh_load_xyz
- #define mesh_read_ply mesh_load_ply
- #define mesh_read_bin mesh_load_bin
- #define mesh_read_out mesh_load_out
- #define mesh_read_nvm mesh_load_nvm
- #define mesh_read_colmap mesh_load_colmap

Typedefs

- typedef struct _iobuf * FILEPOINTER
- typedef INTDATA INTDATA2[2]
- typedef INTDATA INTDATA3[3]
- typedef struct mesh_vector2 mesh_vector2
- typedef mesh_vector2 * MESH_VECTOR2
- typedef struct mesh_vector3 mesh_vector3
- typedef mesh_vector3 * MESH_VECTOR3
- typedef mesh_vector3 mesh_vertex
- typedef mesh_vertex * MESH_VERTEX
- typedef mesh_vector3 mesh_normal
- typedef mesh_normal * MESH_NORMAL
- typedef struct mesh_color mesh_color
- typedef mesh_color * MESH_COLOR
- typedef struct mesh_struct mesh_struct
- typedef mesh_struct * MESH_STRUCT
- typedef struct mesh_struct2 mesh_struct2
- typedef mesh_struct2 * MESH_STRUCT2
- typedef struct mesh_struct3 mesh_struct3
- typedef mesh_struct3 * MESH_STRUCT3
- typedef FLOATDATA mesh_scalar
- typedef mesh_scalar * MESH_SCALAR
- typedef struct mesh_face mesh_face
- typedef mesh_face * MESH_FACE
- typedef struct mesh_edge mesh_edge
- typedef struct mesh_edge * MESH_EDGE
- typedef struct mesh_adjface mesh_adjface
- typedef struct mesh_adjface mesh_vface
- typedef mesh_vface * MESH_VFACE
- typedef struct mesh_adjface mesh_fface
- typedef mesh_fface * MESH_FFACE
- typedef struct mesh_rotation mesh_rotation
- typedef mesh_rotation * MESH_ROTATION
- typedef struct mesh_affine mesh_affine
- typedef mesh_affine * MESH_AFFINE
- typedef struct mesh_affine mesh_rigid
- typedef mesh_rigid * MESH_RIGID
- typedef struct mesh mesh
- typedef mesh * MESH

Functions

- MESHLIBAPI void mesh_error (int type)

  Displays error message and exits.
- MESHLIBAPI MESH mesh_create_mesh_new ()

  Creates a new mesh.
- MESHLIBAPI void mesh_free_mesh (MESH m)

  Frees a mesh.
- MESHLIBAPI MESH mesh_create_mesh_new_grid (MESH_VECTOR3 sz, MESH_VECTOR3 pos, INTDATA m, INTDATA n)

  Creates a grid mesh.
- MESHLIBAPI MESH mesh_create_mesh_new_cuboid (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

Creates a cuboid mesh.

- MESHLIBAPI MESH mesh_create_mesh_new_ellipsoid (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates an ellipsoid mesh.
- MESHLIBAPI MESH mesh_create_mesh_new_uniform_ellipsoid (MESH_VECTOR3 sz, MESH_VECTOR3 pos, int n)

  Creates a uniformly sampled ellipsoid mesh.
- MESHLIBAPI MESH mesh_create_mesh_new_cylinder (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a cylinder mesh.
- MESHLIBAPI MESH mesh_create_mesh_new_cone (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a cone mesh.
- MESHLIBAPI MESH mesh_create_mesh_new_rectangle_flat (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a flat rectangle mesh.
- MESHLIBAPI MESH mesh_create_mesh_new_ellipse_flat (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

  Creates a flat ellipse mesh.
- MESHLIBAPI MESH mesh_clone_mesh (MESH m, uint16_t flags)

  Clones a given mesh into another mesh.
- MESHLIBAPI MESH mesh_combine_mesh (MESH m1, MESH m2)

  Combines a given mesh with another given mesh.
- MESHLIBAPI int mesh_alloc_mesh_props (MESH m, INTDATA nv, INTDATA nf, INTDATA ne, uint16_t flags)

  Allocates memory for various properties of a given mesh.
- MESHLIBAPI int mesh_free_mesh_props (MESH m, uint16_t flags)

  Frees memory for various properties of a given mesh.
- MESHLIBAPI int mesh_alloc_face_vertices (MESH_FACE mf, INTDATA nv)

  Allocates memory for vertices of a given mesh face.
- MESHLIBAPI int mesh_free_face_vertices (MESH_FACE mf)

  Frees memory for vertices of a given mesh face.
- MESHLIBAPI MESH mesh_load_file (const char *fname)

  Reads a mesh from an OFF/PLY/ASC/XYZ/BINv1/BundlerOUT/NVM file.
- MESHLIBAPI MESH mesh_load_off (const char *fname)

  Reads a mesh from an OFF file.
- MESHLIBAPI MESH mesh_load_xyz (const char *fname)

  Reads a mesh from an ASC/XYZ file.
- MESHLIBAPI MESH mesh_load_ply (const char *fname)

  Reads a mesh from a PLY file.
- MESHLIBAPI MESH mesh_load_bin (const char *fname)

  Reads a mesh from a CloudCompare BINv1 file.
- MESHLIBAPI MESH mesh_load_out (const char *fname)

  Reads a mesh from a Bundler OUT file.
- MESHLIBAPI MESH mesh_load_nvm (const char *fname)

  Reads a mesh from an NVM file.
- MESHLIBAPI MESH mesh_load_colmap (const char *fname)

  Reads a mesh from a COLMAP BIN file.
- MESHLIBAPI int mesh_save_file (MESH m, const char *fname)

  Saves a mesh to an OFF/PLY/ASC/XYZ/BIN/OBJ file.
- MESHLIBAPI int mesh_save_off (MESH m, const char *fname)

  Saves a mesh to an OFF file.
- MESHLIBAPI int mesh_save_xyz (MESH m, const char *fname)

  Saves a mesh to an XYZ file.
- MESHLIBAPI int mesh_save_ply (MESH m, const char *fname)

  Saves a mesh to a PLY file.
- MESHLIBAPI int mesh_save_bin (MESH m, const char *fname)

Saves a mesh to a BINv1 file.

- MESHLIBAPI int mesh_save_obj (MESH m, const char *fname)

  Saves a mesh to an OBJ file.
- MESHLIBAPI int mesh_calc_vertex_normals (MESH m)

  Computes vertex normals of a given mesh.
- MESHLIBAPI int mesh_calc_face_normals (MESH m)

  Computes face normals of a given mesh.
- MESHLIBAPI int mesh_calc_edges (MESH m)

  Computes edges of a given mesh.
- MESHLIBAPI int mesh_calc_vertex_adjacency (MESH m)

  Computes vertex adjacent faces of a given mesh.
- MESHLIBAPI int mesh_calc_face_adjacency (MESH m)

  Computes face adjacent faces of a given mesh.
- MESHLIBAPI int mesh_upsample (MESH m, int iters)

  Upsamples a given mesh.
- MESHLIBAPI int mesh_upsample_loop (MESH m, int iters)

  Upsamples a given mesh using Loop's algorithm.
- MESHLIBAPI int mesh_upsample_tarea_adaptive (MESH m, int miters, FLOATDATA e)

  Upsamples a given mesh upto a given triangle area threshold.
- MESHLIBAPI void mesh_cross_vector3 (MESH_VECTOR3 x, MESH_VECTOR3 y, MESH_VECTOR3 z)

  Computes the cross product of two 3-d vectors.
- MESHLIBAPI void mesh_cross_normal (MESH_NORMAL x, MESH_NORMAL y, MESH_NORMAL z)

  Computes the normalized cross product of two normals.
- MESHLIBAPI FLOATDATA mesh_calc_triangle_area (MESH_VERTEX a, MESH_VERTEX b, MESH_VERTEX c)

  Computes area of a triangle.
- MESHLIBAPI void mesh_calc_face_normal (MESH_VERTEX v1, MESH_VERTEX v2, MESH_VERTEX v3, MESH_NORMAL n)

  Computes the face normal given 3 vertices.
- MESHLIBAPI void mesh_calc_aabb (MESH m, MESH_VECTOR3 minv, MESH_VECTOR3 maxv, MESH_VECTOR3 center)

  Computes axis-aligned bounding box.
- MESHLIBAPI int mesh_calc_signed_area (MESH m, MESH_VECTOR3 area)

  Computes signed area of a triangle mesh.
- MESHLIBAPI FLOATDATA mesh_calc_area (MESH m)

  Computes area of a triangle mesh.
- MESHLIBAPI FLOATDATA mesh_calc_volume (MESH m)

  Computes volume of a triangle mesh.
- MESHLIBAPI INTDATA mesh_find (MESH_STRUCT s, INTDATA q)

  Finds an item in an INTDATA structure.
- MESHLIBAPI INTDATA mesh_find2 (MESH_STRUCT2 s, INTDATA q)

  Finds an item in an INTDATA2 structure.
- MESHLIBAPI INTDATA mesh_find3 (MESH_STRUCT3 s, INTDATA q)

  Finds an item in an INTDATA3 structure.
- MESHLIBAPI int mesh_remove_boundary_vertices (MESH m, int iters)

  Removes boundary vertices and connecting elements.
- MESHLIBAPI int mesh_remove_boundary_faces (MESH m, int iters)

  Removes boundary faces and connecting elements.
- MESHLIBAPI int mesh_remove_triangles_with_small_area (MESH m, FLOATDATA area)

  Removes triangles with area smaller than a given value.
- MESHLIBAPI int mesh_remove_unreferenced_vertices (MESH m)

Removes unreferenced vertices.

- MESHLIBAPI int mesh_remove_zero_area_faces (MESH m)

  Removes triangles with zero area.
- MESHLIBAPI int mesh_remove_close_vertices (MESH m, FLOATDATA r)

  Removes close vertices.
- MESHLIBAPI int mesh_remove_ear_faces (MESH m, int niters)

  Removes ear faces and connecting vertices.
- MESHLIBAPI int mesh_remove_non_manifold_vertices (MESH m)

  Removes non-manifold vertices.
- MESHLIBAPI int mesh_isnumeric (FILEPOINTER fp)

  Checks if numeric or not.
- MESHLIBAPI int mesh_go_next_word (FILEPOINTER fp)

  Points to the next word.
- MESHLIBAPI int mesh_read_word (FILEPOINTER fp, char ∗c_word, int sz)

  Reads current word and moves to the next word.
- MESHLIBAPI int mesh_read_word_skip_comment (FILEPOINTER fp, char ∗c_word, int sz)

  Reads current word skipping comment with # and moves to the next word.
- MESHLIBAPI int mesh_read_word_only (FILEPOINTER fp, char ∗c_word, int sz)

  Reads current word without moving to the next word.
- MESHLIBAPI int mesh_read_word_only_skip_comment (FILEPOINTER fp, char ∗c_word, int sz)

  Reads current word skipping comment with # without moving to the next word.
- MESHLIBAPI int mesh_count_words_in_line (FILEPOINTER fp, int ∗count)

  Counts number of words in the current line.
- MESHLIBAPI int mesh_skip_line (FILEPOINTER fp)

  Skips to next line.
- MESHLIBAPI int mesh_filter_bilateral (MESH m, FLOATDATA sigma_c, FLOATDATA sigma_s, int niters)

  Mesh bilateral filter.
- MESHLIBAPI int mesh_filter_laplacian (MESH m, FLOATDATA r)

  Mesh Laplacian filter.
- MESHLIBAPI int mesh_filter_laplacian_restricted (MESH m, FLOATDATA r, FLOATDATA ang)

  Restricted Mesh Laplacian filter.
- MESHLIBAPI int mesh_filter_laplacian_depth (MESH m, FLOATDATA r, MESH_VECTOR3 vp)

  Mesh Depth Laplacian filter.
- MESHLIBAPI int mesh_filter_taubin (MESH m, FLOATDATA lambd, FLOATDATA mu, int niters)

  Mesh $\lambda - \mu$ Taubin filter.
- MESHLIBAPI int mesh_filter_vertex_color_bilateral (MESH m, FLOATDATA sigma_k, FLOATDATA sigma_c, FLOATDATA sigma_s, int niters)

  Mesh bilateral vertex color filter.
- MESHLIBAPI int mesh_filter_vertex_color_laplacian (MESH m, FLOATDATA r)

  Mesh Laplacian vertex color filter.
- MESHLIBAPI int mesh_filter_vertex_color_min (MESH m, INTDATA niters)

  Mesh minimum intensity vertex color filter.
- MESHLIBAPI int mesh_filter_vertex_color_max (MESH m, INTDATA niters)

  Mesh maximum intensity vertex color filter.
- MESHLIBAPI MESH_ROTATION mesh_rotation_create ()

  Creates a new rotation.
- MESHLIBAPI void mesh_rotation_free (MESH_ROTATION r)

  Frees a given rotation.
- MESHLIBAPI MESH_ROTATION mesh_rotation_set_matrix (FLOATDATA ∗mat, MESH_ROTATION r)

  Sets rotation from a matrix.

- MESHLIBAPI MESH_ROTATION mesh_rotation_set_angleaxis (FLOATDATA ang, MESH_NORMAL axis, MESH_ROTATION r)

  Sets rotation from angle axis.
- MESHLIBAPI int mesh_translate (MESH m, FLOATDATA x, FLOATDATA y, FLOATDATA z)

  Translates a mesh by x, y and z amounts.
- MESHLIBAPI int mesh_translate_vector (MESH m, MESH_VECTOR3 v)

  Translates a mesh by a given 3-d vector.
- MESHLIBAPI int mesh_scale (MESH m, FLOATDATA sx, FLOATDATA sy, FLOATDATA sz)

  Scales a mesh by x, y and z amounts.
- MESHLIBAPI MESH_VERTEX mesh_vertex_rotate (MESH_VERTEX v, MESH_ROTATION r)

  Rotates a vertex by a given rotation.
- MESHLIBAPI int mesh_rotate (MESH m, MESH_ROTATION r)

  Rotates a mesh by a given rotation.
- MESHLIBAPI MESH_AFFINE mesh_affine_create ()

  Creates a new affine transformation.
- MESHLIBAPI void mesh_affine_free (MESH_AFFINE tx)

  Frees a given affine transformation.
- MESH_AFFINE mesh_affine_set_matrix (FLOATDATA *mat, MESH_AFFINE r)

  Sets affine transformation from a matrix.
- MESHLIBAPI int mesh_transform (MESH m, MESH_AFFINE tx)

  Transforms a mesh by a given affine transformation.
- MESHLIBAPI MESH_AFFINE mesh_affine_set_rotation_translation (MESH_ROTATION r, MESH_VECTOR3 t, MESH_AFFINE tx)
- MESHLIBAPI void mesh_set_seed (int seed)
- MESHLIBAPI FLOATDATA __mesh_rand (void)
- MESHLIBAPI FLOATDATA __mesh_randn (void)
- MESHLIBAPI FLOATDATA __mesh_randexp (void)
- MESHLIBAPI FLOATDATA __mesh_randfun (FLOATDATA(*fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)
- MESHLIBAPI int mesh_add_noise_uniform (MESH m, FLOATDATA sigma)

  Adds uniform random noise to a mesh.
- MESHLIBAPI int mesh_add_noise_gaussian (MESH m, FLOATDATA sigma)

  Adds Gaussian random noise to a mesh.
- MESHLIBAPI int mesh_add_noise_exp (MESH m, FLOATDATA sigma)

  Adds exponential random noise to a mesh.
- MESHLIBAPI int mesh_add_noise_func (MESH m, FLOATDATA sigma, FLOATDATA(*fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)

  Adds random noise given by density function to a mesh.
- MESHLIBAPI int mesh_add_noise_uniform_normal (MESH m, FLOATDATA sigma)

  Adds uniform random noise along normals to a mesh.
- MESHLIBAPI int mesh_add_noise_gaussian_normal (MESH m, FLOATDATA sigma)

  Adds Gaussian random noise along normals to a mesh.
- MESHLIBAPI int mesh_add_noise_exp_normal (MESH m, FLOATDATA sigma)

  Adds exponential random noise along normals to a mesh.
- MESHLIBAPI int mesh_add_noise_func_normal (MESH m, FLOATDATA sigma, FLOATDATA(*fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)

  Adds random noise given by density function along normals to a mesh.
- MESHLIBAPI int mesh_add_noise_uniform_tangent (MESH m, FLOATDATA sigma)

  Adds uniform random noise along tangent planes to a mesh.
- MESHLIBAPI int mesh_add_noise_gaussian_tangent (MESH m, FLOATDATA sigma)

  Adds Gaussian random noise along tangent planes to a mesh.
- MESHLIBAPI int mesh_add_noise_exp_tangent (MESH m, FLOATDATA sigma)

Adds exponential random noise along tangent planes to a mesh.

- MESHLIBAPI int mesh_add_noise_func_tangent (MESH m, FLOATDATA sigma, FLOATDATA(*fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)

  Adds random noise given by density function along tangent planes to a mesh.
- MESHLIBAPI void mesh_draw_mesh (MESH m)

  Draws a given mesh in OpenGL context in flat shading.
- MESHLIBAPI void mesh_draw_mesh_smooth (MESH m)

  Draws a given mesh in OpenGL context in smoothing shading.
- MESHLIBAPI void mesh_draw_point_cloud (MESH m)

  Draws a given mesh in OpenGL context as pointcloud.
- MESHLIBAPI int mesh_align_global (MESH m1, MESH m2, int flags, MESH_AFFINE tx)

  Sets an affine transformation with rotation and translation.

6.11.1 Detailed Description

This header file contains declarations of all functions of meshlib.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.11.2 Macro Definition Documentation

6.11.2.1 __MESHLIB__

```
#define __MESHLIB__
```

6.11.2.2 _CRT_SECURE_NO_DEPRECATE

```
#define _CRT_SECURE_NO_DEPRECATE
```

6.11.2.3 FLOATDATA

```
#define FLOATDATA double /* do not change this, careful see meshload fscanf and other functions
*/
```

Float datatype

6.11.2.4 INTDATA

```
#define INTDATA int64_t /* do not change this, careful see meshload fscanf and other functions
*/
```

Integer datatype

6.11.2.5 MESH_ALIGN_GLOBAL_ALL

```
#define MESH_ALIGN_GLOBAL_ALL (0x07)
```

Mesh alignment all global properties

6.11.2.6 MESH_ALIGN_GLOBAL_DO_TRANSFORM

```
#define MESH_ALIGN_GLOBAL_DO_TRANSFORM (0x08)
```

Mesh alignment global do transform

6.11.2.7 MESH_ALIGN_GLOBAL_ORIENTATION

```
#define MESH_ALIGN_GLOBAL_ORIENTATION (0x02)
```

Mesh alignment global orientation

6.11.2.8 MESH_ALIGN_GLOBAL_POSITION

```
#define MESH_ALIGN_GLOBAL_POSITION (0x01)
```

Mesh alignment global position

6.11.2.9 MESH_ALIGN_GLOBAL_SCALE

```
#define MESH_ALIGN_GLOBAL_SCALE (0x04)
```

Mesh alignment global scale

6.11.2.10 mesh_bilateral_filter

```
#define mesh_bilateral_filter mesh_filter_bilateral
```

6.11.2.11 mesh_bilateral_vertex_color_filter

```
#define mesh_bilateral_vertex_color_filter mesh_filter_vertex_color_bilateral
```

6.11.2.12 mesh_depth_laplacian_filter

```
#define mesh_depth_laplacian_filter mesh_filter_laplacian_depth
```

6.11.2.13 MESH_EPS12

```
#define MESH_EPS12 (1e-12)
```

Medium epsilon

6.11.2.14 MESH_EPS20

```
#define MESH_EPS20 (1e-20)
```

Large epsilon

6.11.2.15 MESH_EPS8

```
#define MESH_EPS8 (1e-8)
```

Small epsilon

6.11.2.16 MESH_EPSM

```
#define MESH_EPSM (DBL_EPSILON)
```

Machine epsilon

6.11.2.17 MESH_ERR_FNOTOPEN

```
#define MESH_ERR_FNOTOPEN 2
```

Mesh error type - file open

6.11.2.18 MESH_ERR_INCOMPATIBLE

```
#define MESH_ERR_INCOMPATIBLE 3
```

Mesh error type - incompatible data

6.11.2.19 MESH_ERR_MALLOC

```
#define MESH_ERR_MALLOC 0
```

Mesh error type - allocation

6.11.2.20 MESH_ERR_SIZE_MISMATCH

```
#define MESH_ERR_SIZE_MISMATCH 1
```

Mesh error type - size mismatch

6.11.2.21 MESH_ERR_UNKNOWN

```
#define MESH_ERR_UNKNOWN 4
```

Mesh error type - unknown

6.11.2.22 MESH_FLOATDATA_MAX

```
#define MESH_FLOATDATA_MAX (DBL_MAX)
```

Float datatype maximum

6.11.2.23 MESH_FLOATDATA_MIN

```
#define MESH_FLOATDATA_MIN (DBL_MIN)
```

Float datatype minimum

6.11.2.24 MESH_FLOATDATA_TYPE

```
#define MESH_FLOATDATA_TYPE 1
```

Float datatype selector

6.11.2.25 MESH_INTDATA_MAX

```
#define MESH_INTDATA_MAX (INT64_MAX)
```

Integer datatype maximum

6.11.2.26 MESH_INTDATA_MIN

```
#define MESH_INTDATA_MIN (INT64_MIN)
```

Integer datatype minimum

6.11.2.27 MESH_INTDATA_TYPE

```
#define MESH_INTDATA_TYPE 1
```

Integer datatype selector

6.11.2.28 mesh_laplacian_filter

```
#define mesh_laplacian_filter mesh_filter_laplacian
```

6.11.2.29 mesh_laplacian_vertex_color_filter

```
#define mesh_laplacian_vertex_color_filter mesh_filter_vertex_color_laplacian
```

6.11.2.30 MESH_MAX

```
#define MESH_MAX(
              a,
              b ) (((a)>(b))?(a):(b))
```

Maximum of two variables

6.11.2.31 mesh_max_vertex_color_filter

```
#define mesh_max_vertex_color_filter mesh_filter_vertex_color_max
```

6.11.2.32 MESH_MIN

```
#define MESH_MIN(
              a,
              b ) (((a)<(b))?(a):(b))
```

Minimum of two variables

6.11.2.33 mesh_min_vertex_color_filter

```
#define mesh_min_vertex_color_filter mesh_filter_vertex_color_min
```

6.11.2.34 MESH_ORIGIN_TYPE_BINCOLMAP

```
#define MESH_ORIGIN_TYPE_BINCOLMAP 70
```

Mesh origin type - COLMAP BIN file

6.11.2.35 MESH_ORIGIN_TYPE_BINV1

```
#define MESH_ORIGIN_TYPE_BINV1 40
```

Mesh origin type - CloudCompare BINv1file

6.11.2.36 MESH_ORIGIN_TYPE_BUILD

```
#define MESH_ORIGIN_TYPE_BUILD 00
```

Mesh origin type - create new

6.11.2.37 MESH_ORIGIN_TYPE_BUNDLE_OUT

```
#define MESH_ORIGIN_TYPE_BUNDLE_OUT 50
```

Mesh origin type - Bundle OUT file

6.11.2.38 MESH_ORIGIN_TYPE_COFF

```
#define MESH_ORIGIN_TYPE_COFF 13
```

Mesh origin type - COFF file

6.11.2.39 MESH_ORIGIN_TYPE_NCOFF

```
#define MESH_ORIGIN_TYPE_NCOFF 14
```

Mesh origin type - NCOFF file

6.11.2.40 MESH_ORIGIN_TYPE_NOFF

```
#define MESH_ORIGIN_TYPE_NOFF 12
```

Mesh origin type - NOFF file

6.11.2.41 MESH_ORIGIN_TYPE_NVM

```
#define MESH_ORIGIN_TYPE_NVM 60
```

Mesh origin type - NVM file

6.11.2.42 MESH_ORIGIN_TYPE_OFF

```
#define MESH_ORIGIN_TYPE_OFF 11
```

Mesh origin type - OFF file

6.11.2.43 MESH_ORIGIN_TYPE_PLY_ASCII

```
#define MESH_ORIGIN_TYPE_PLY_ASCII 30
```

Mesh origin type - PLY ascii file

6.11.2.44 MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN

```
#define MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN 32
```

Mesh origin type - PLY binary BE file

6.11.2.45 MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN

```
#define MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN 31
```

Mesh origin type - PLY binary LE file

6.11.2.46 MESH_ORIGIN_TYPE_XYZ

```
#define MESH_ORIGIN_TYPE_XYZ 20
```

Mesh origin type - XYZ file

6.11.2.47 MESH_PI

```
#define MESH_PI (3.14159265359)
```

π

6.11.2.48 MESH_PROPS_ALL_PROPS

```
#define MESH_PROPS_ALL_PROPS (0xFFFF)
```

Mesh all properties

6.11.2.49 MESH_PROPS_EDGES

```
#define MESH_PROPS_EDGES (MESH_PROPS_VERTICES | __MESH_PROPS_FACES | __MESH_PROPS_EDGES)
```

Mesh edges

6.11.2.50 MESH_PROPS_F_ALL_PROPS

```
#define MESH_PROPS_F_ALL_PROPS (MESH_PROPS_FACES | __MESH_PROPS_F_ALL_PROPS)
```

Mesh all face properties

6.11.2.51 MESH_PROPS_FACES

```
#define MESH_PROPS_FACES (MESH_PROPS_VERTICES | __MESH_PROPS_FACES)
```

Mesh faces

6.11.2.52 MESH_PROPS_FAREAS

```
#define MESH_PROPS_FAREAS (MESH_PROPS_FACES | __MESH_PROPS_FAREAS)
```

Mesh faces and face areas

6.11.2.53 MESH_PROPS_FCOLORS

```
#define MESH_PROPS_FCOLORS (MESH_PROPS_FACES | __MESH_PROPS_FCOLORS)
```

Mesh faces and face colors

6.11.2.54 MESH_PROPS_FFACES

```
#define MESH_PROPS_FFACES (MESH_PROPS_FACES | __MESH_PROPS_FFACES)
```

Mesh faces and face face adjacency

6.11.2.55 MESH_PROPS_FNORMALS

```
#define MESH_PROPS_FNORMALS (MESH_PROPS_FACES | __MESH_PROPS_FNORMALS)
```

Mesh faces and face normals

6.11.2.56 MESH_PROPS_FSCALARS

```
#define MESH_PROPS_FSCALARS (MESH_PROPS_FACES | __MESH_PROPS_FSCALARS)
```

Mesh vertices and face scalars

6.11.2.57 MESH_PROPS_V_ALL_PROPS

```
#define MESH_PROPS_V_ALL_PROPS (0xC00F)
```

Mesh all vertex properties

6.11.2.58 MESH_PROPS_VCOLORS

```
#define MESH_PROPS_VCOLORS (MESH_PROPS_VERTICES | __MESH_PROPS_VCOLORS)
```

Mesh vertices and vertex colors

6.11.2.59 MESH_PROPS_VERTICES

```
#define MESH_PROPS_VERTICES (0x01)
```

Mesh vertices

6.11.2.60 MESH_PROPS_VFACES

```
#define MESH_PROPS_VFACES (MESH_PROPS_VERTICES | __MESH_PROPS_VFACES)
```

Mesh vertices and vertex face adjacency

6.11.2.61 MESH_PROPS_VNORMALS

```
#define MESH_PROPS_VNORMALS (MESH_PROPS_VERTICES | __MESH_PROPS_VNORMALS)
```

Mesh vertices and vertex normals

6.11.2.62 MESH_PROPS_VSCALARS

```
#define MESH_PROPS_VSCALARS (MESH_PROPS_VERTICES | __MESH_PROPS_VSCALARS)
```

Mesh vertices and vertex scalars

6.11.2.63 mesh_read_bin

```
#define mesh_read_bin mesh_load_bin
```

6.11.2.64 mesh_read_colmap

```
#define mesh_read_colmap mesh_load_colmap
```

6.11.2.65 mesh_read_file

```
#define mesh_read_file mesh_load_file
```

6.11.2.66 mesh_read_nvm

```
#define mesh_read_nvm mesh_load_nvm
```

6.11.2.67 mesh_read_off

```
#define mesh_read_off mesh_load_off
```

6.11.2.68 mesh_read_out

```
#define mesh_read_out mesh_load_out
```

6.11.2.69 mesh_read_ply

```
#define mesh_read_ply mesh_load_ply
```

6.11.2.70 mesh_read_xyz

```
#define mesh_read_xyz mesh_load_xyz
```

6.11.2.71 mesh_restricted_laplacian_filter

```
#define mesh_restricted_laplacian_filter mesh_filter_laplacian_restricted
```

6.11.2.72 mesh_taubin_filter

```
#define mesh_taubin_filter mesh_filter_taubin
```

6.11.2.73 MESH_TWOPI

```
#define MESH_TWOPI (6.28318530718)
```

2π

6.11.2.74 mesh_write_bin

```
#define mesh_write_bin mesh_save_bin
```

6.11.2.75 mesh_write_file

```
#define mesh_write_file mesh_save_file
```

6.11.2.76 mesh_write_obj

```
#define mesh_write_obj mesh_save_obj
```

6.11.2.77 mesh_write_off

```
#define mesh_write_off mesh_save_off
```

6.11.2.78 mesh_write_ply

```
#define mesh_write_ply mesh_save_ply
```

6.11.2.79 mesh_write_xyz

```
#define mesh_write_xyz mesh_save_xyz
```

6.11.2.80 MESHLIBAPI

```
#define MESHLIBAPI extern
```

6.11.3 Typedef Documentation

6.11.3.1 FILEPOINTER

```
typedef struct _iobuf* FILEPOINTER
```

File pointer

6.11.3.2 INTDATA2

```
typedef INTDATA INTDATA2[2]
```

2- element INTDATA

6.11.3.3 INTDATA3

```
typedef INTDATA INTDATA3[3]
```

3- element INTDATA

6.11.3.4 mesh

```
typedef struct mesh mesh
```

Mesh

6.11.3.5 MESH

```
typedef mesh* MESH
```

Pointer to mesh

6.11.3.6 mesh_adjface

```
typedef struct mesh_adjface mesh_adjface
```

Adjacent face structure

6.11.3.7 mesh_affine

```
typedef struct mesh_affine mesh_affine
```

Affine transformation

6.11.3.8 MESH_AFFINE

```
typedef mesh_affine* MESH_AFFINE
```

Pointer to affine transformation

6.11.3.9 mesh_color

```
typedef struct mesh_color mesh_color
```

6.11.3.10 MESH_COLOR

```
typedef mesh_color* MESH_COLOR
```

Color

6.11.3.11 mesh_edge

```
typedef struct mesh_edge mesh_edge
```

Edge

6.11.3.12 MESH_EDGE

```
typedef struct mesh_edge* MESH_EDGE
```

Pointer to edge

6.11.3.13 mesh_face

```
typedef struct mesh_face mesh_face
```

Face

6.11.3.14 MESH_FACE

```
typedef mesh_face* MESH_FACE
```

Pointer to face

6.11.3.15 mesh_fface

```
typedef struct mesh_adjface mesh_fface
```

Face adjacent faces

6.11.3.16 MESH_FFACE

```
typedef mesh_fface* MESH_FFACE
```

Pointer to face adjacent faces

6.11.3.17 mesh_normal

```
typedef mesh_vector3 mesh_normal
```

Normal

6.11.3.18 MESH_NORMAL

```
typedef mesh_normal* MESH_NORMAL
```

Normal pointer

6.11.3.19 mesh_rigid

```
typedef struct mesh_affine mesh_rigid
```

Rigid transformation

6.11.3.20 MESH_RIGID

```
typedef mesh_rigid* MESH_RIGID
```

Pointer to rigid transformation

6.11.3.21 mesh_rotation

```
typedef struct mesh_rotation mesh_rotation
```

Rotation

6.11.3.22 MESH_ROTATION

```
typedef mesh_rotation* MESH_ROTATION
```

Pointer to rotation

6.11.3.23 mesh_scalar

```
typedef FLOATDATA mesh_scalar
```

Scalar

6.11.3.24 MESH_SCALAR

```
typedef mesh_scalar* MESH_SCALAR
```

Pointer to Scalar

6.11.3.25 mesh_struct

```
typedef struct mesh_struct mesh_struct
```

INTDATA Structure

6.11.3.26 MESH_STRUCT

```
typedef mesh_struct* MESH_STRUCT
```

INTDATA Structure pointer

6.11.3.27 mesh_struct2

```
typedef struct mesh_struct2 mesh_struct2
```

INTDATA2 Structure

6.11.3.28 MESH_STRUCT2

```
typedef mesh_struct2* MESH_STRUCT2
```

INTDATA2 Structure pointer

6.11.3.29 mesh_struct3

```
typedef struct mesh_struct3 mesh_struct3
```

INTDATA3 Structure

6.11.3.30 MESH_STRUCT3

```
typedef mesh_struct3* MESH_STRUCT3
```

INTDATA3 Structure pointer

6.11.3.31 mesh_vector2

```
typedef struct mesh_vector2 mesh_vector2
```

Generic 2-d vector

6.11.3.32 MESH_VECTOR2

```
typedef mesh_vector2* MESH_VECTOR2
```

Generic 2-d vector pointer

6.11.3.33 mesh_vector3

```
typedef struct mesh_vector3 mesh_vector3
```

Generic 3-d vector

6.11.3.34 MESH_VECTOR3

```
typedef mesh_vector3* MESH_VECTOR3
```

Generic 3-d vector pointer

6.11.3.35 mesh_vertex

```
typedef mesh_vector3 mesh_vertex
```

Vertex

6.11.3.36 MESH_VERTEX

```
typedef mesh_vertex* MESH_VERTEX
```

Vertex pointer

6.11.3.37 mesh_vface

```
typedef struct mesh_adjface mesh_vface
```

Vertex adjacent faces

6.11.3.38 MESH_VFACE

```
typedef mesh_vface* MESH_VFACE
```

Pointer to vertex adjacent faces

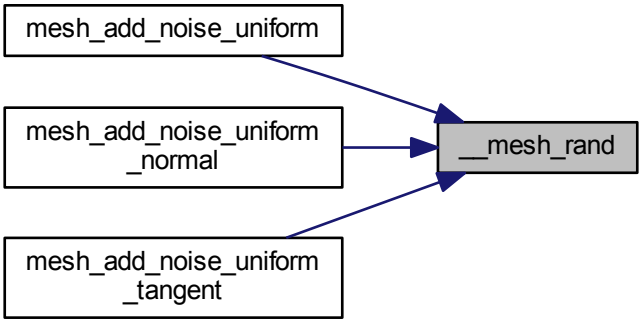
6.11.4 Function Documentation

6.11.4.1 __mesh_rand()

```
MESHLIBAPI FLOATDATA __mesh_rand (
            void  )
```

Here is the call graph for this function:

Here is the caller graph for this function:

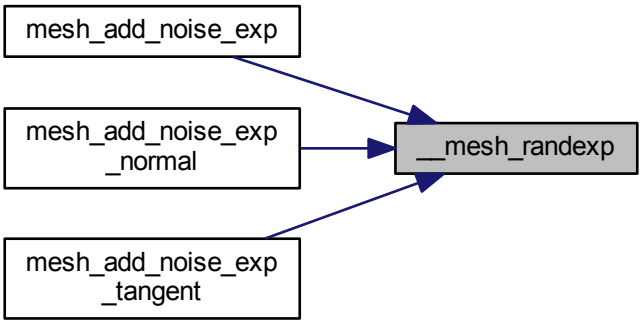


6.11.4.2 __mesh_randexp()

```
MESHLIBAPI FLOATDATA __mesh_randexp (
            void  )
```

Here is the call graph for this function:

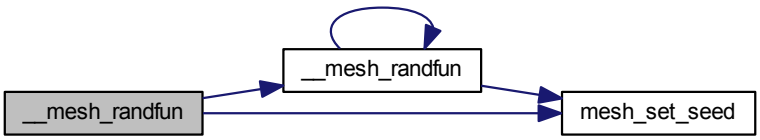
Here is the caller graph for this function:



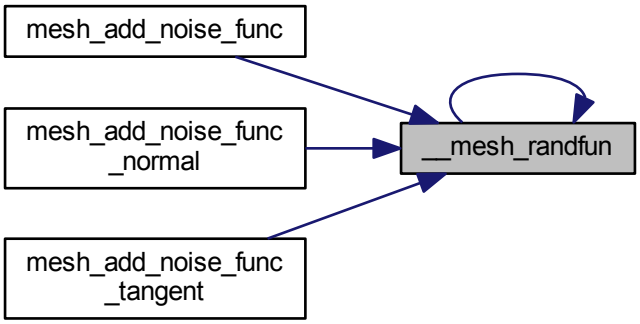
6.11.4.3 __mesh_randfun()

```
MESHLIBAPI FLOATDATA __mesh_randfun (
             FLOATDATA(*)(FLOATDATA) fun,
             FLOATDATA xmin,
             FLOATDATA xmax )
```

Here is the call graph for this function:



Here is the caller graph for this function:

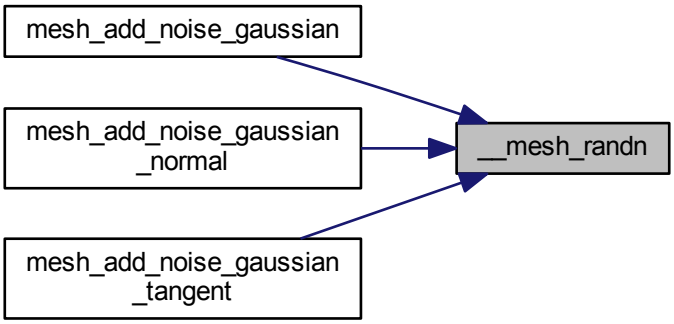


6.11.4.4 __mesh_randn()

```
MESHLIBAPI FLOATDATA __mesh_randn (
            void  )
```

Here is the call graph for this function:

Here is the caller graph for this function:



6.11.4.5 mesh_add_noise_exp()

```
MESHLIBAPI int mesh_add_noise_exp (
            MESH m,
            FLOATDATA sigma )
```

Adds exponential random noise to a mesh.

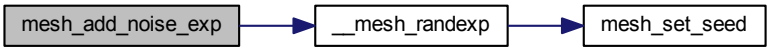
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.6 mesh_add_noise_exp_normal()

```
MESHLIBAPI int mesh_add_noise_exp_normal (
            MESH m,
            FLOATDATA sigma )
```

Adds exponential random noise along normals to a mesh.

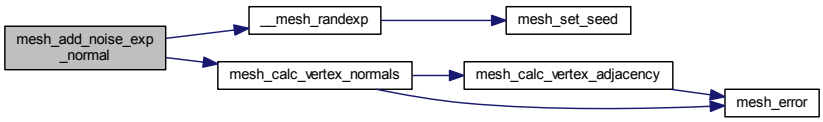
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.7 mesh_add_noise_exp_tangent()

```
MESHLIBAPI int mesh_add_noise_exp_tangent (
            MESH m,
            FLOATDATA sigma )
```

Adds exponential random noise along tangent planes to a mesh.

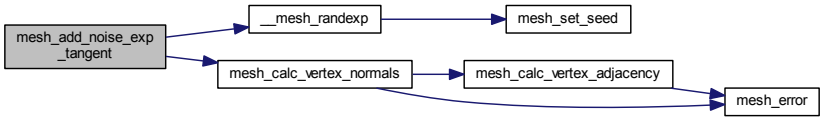
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.8 mesh_add_noise_func()

```
MESHLIBAPI int mesh_add_noise_func (
            MESH m,
            FLOATDATA sigma,
            FLOATDATA(*)(FLOATDATA) fun,
            FLOATDATA xmin,
            FLOATDATA xmax )
```

Adds random noise given by density function to a mesh.

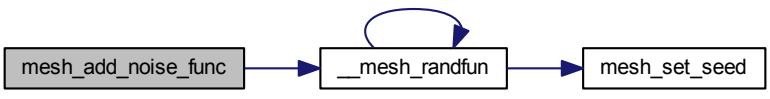
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation
in	*fun*	Density function
in	*xmin*	Lower limit
in	*xmax*	Upper limit

Returns

Error code

Here is the call graph for this function:



6.11.4.9 mesh_add_noise_func_normal()

```
MESHLIBAPI int mesh_add_noise_func_normal (
            MESH m,
            FLOATDATA sigma,
            FLOATDATA(*)(FLOATDATA) fun,
            FLOATDATA xmin,
            FLOATDATA xmax )
```

Adds random noise given by density function along normals to a mesh.

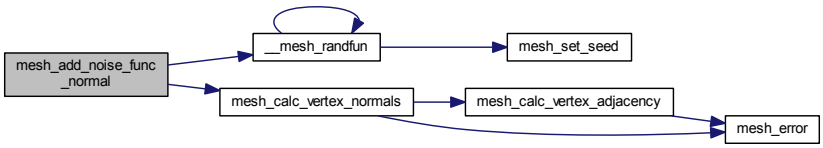
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation
in	*fun*	Density function
in	*xmin*	Lower limit
in	*xmax*	Upper limit

Returns

Error code

Here is the call graph for this function:



6.11.4.10 mesh_add_noise_func_tangent()

```
MESHLIBAPI int mesh_add_noise_func_tangent (
            MESH m,
            FLOATDATA sigma,
            FLOATDATA(*)(FLOATDATA) fun,
            FLOATDATA xmin,
            FLOATDATA xmax )
```

Adds random noise given by density function along tangent planes to a mesh.

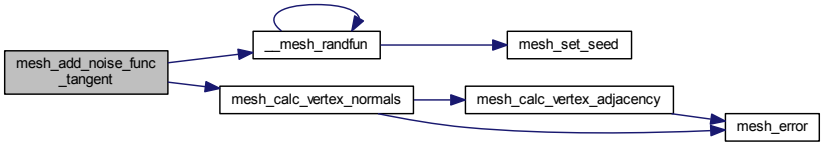
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation
in	*fun*	Density function
in	*xmin*	Lower limit
in	*xmax*	Upper limit

Returns

Error code

Here is the call graph for this function:



6.11.4.11 mesh_add_noise_gaussian()

```
MESHLIBAPI int mesh_add_noise_gaussian (
            MESH m,
            FLOATDATA sigma )
```

Adds Gaussian random noise to a mesh.

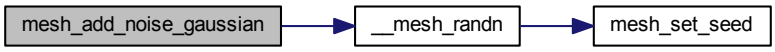
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.12 mesh_add_noise_gaussian_normal()

```
MESHLIBAPI int mesh_add_noise_gaussian_normal (
            MESH m,
            FLOATDATA sigma )
```

Adds Gaussian random noise along normals to a mesh.

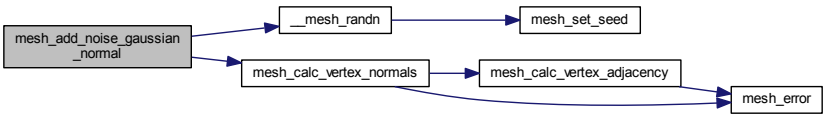
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.13 mesh_add_noise_gaussian_tangent()

```
MESHLIBAPI int mesh_add_noise_gaussian_tangent (
            MESH m,
            FLOATDATA sigma )
```

Adds Gaussian random noise along tangent planes to a mesh.

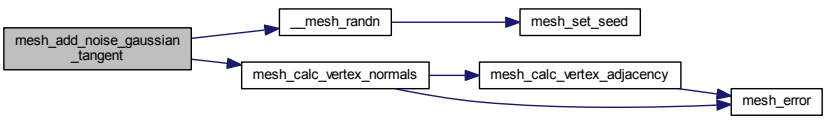
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.14 mesh_add_noise_uniform()

```
MESHLIBAPI int mesh_add_noise_uniform (
            MESH m,
            FLOATDATA sigma )
```

Adds uniform random noise to a mesh.

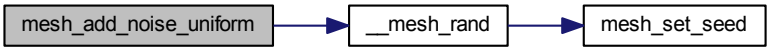
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.15 mesh_add_noise_uniform_normal()

```
MESHLIBAPI int mesh_add_noise_uniform_normal (
            MESH m,
            FLOATDATA sigma )
```

Adds uniform random noise along normals to a mesh.

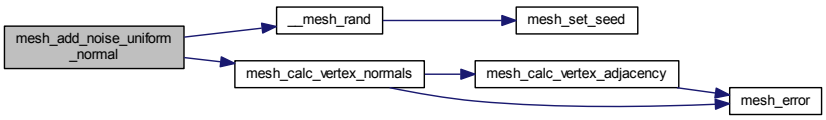
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.16 mesh_add_noise_uniform_tangent()

```
MESHLIBAPI int mesh_add_noise_uniform_tangent (
            MESH m,
            FLOATDATA sigma )
```

Adds uniform random noise along tangent planes to a mesh.

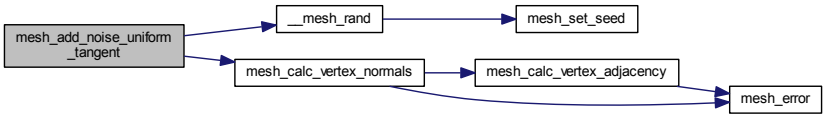
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.11.4.17 mesh_affine_create()

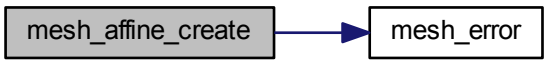
```
MESHLIBAPI MESH_AFFINE mesh_affine_create ( )
```

Creates a new affine transformation.

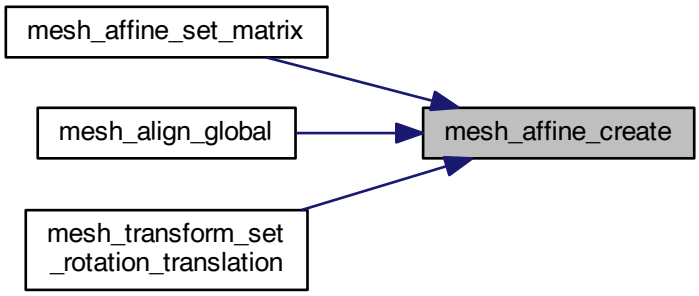
Returns

Output affine transformation

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.18 mesh_affine_free()

```
MESHLIBAPI void mesh_affine_free (
            MESH_AFFINE tx )
```

Frees a given affine transformation.

Parameters

tx	Input affine transformation

Returns

NULL

6.11.4.19 mesh_affine_set_matrix()

```
MESH_AFFINE mesh_affine_set_matrix (
            FLOATDATA * mat,
            MESH_AFFINE r )
```

Sets affine transformation from a matrix.

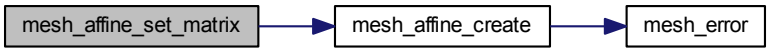
Parameters

in	*mat*	Input matrix
out	*r*	Input affine transformation

Returns

Output affine transformation

Here is the call graph for this function:



6.11.4.20 mesh_affine_set_rotation_translation()

```
MESHLIBAPI MESH_AFFINE mesh_affine_set_rotation_translation (
            MESH_ROTATION r,
            MESH_VECTOR3 t,
            MESH_AFFINE tx )
```

6.11.4.21 mesh_align_global()

```
MESHLIBAPI int mesh_align_global (
            MESH m1,
            MESH m2,
            int flags,
            MESH_AFFINE tx )
```

Sets an affine transformation with rotation and translation.

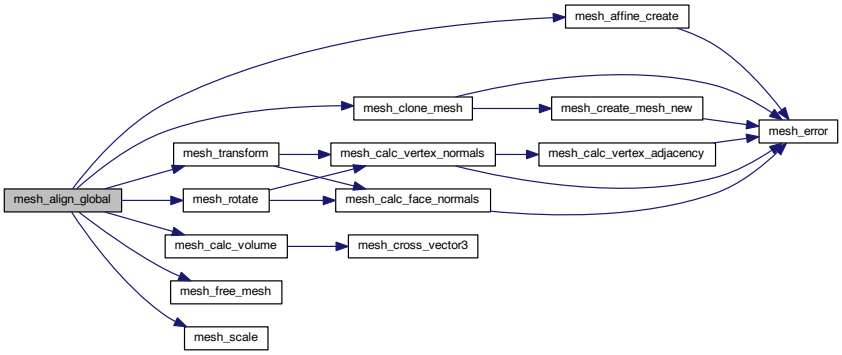
Parameters

in	*m1*	Input mesh
out	*m2*	Input mesh to align
in	*flags*	(MESH_ALIGN_GLOBAL_POSITION/MESH_ALIGN_GLOBAL_ORIENTATION/MESH_↩ ALIGN_GLOBAL_SCALE/MESH_ALIGN_GLOBAL_ALL/MESH_ALIGN_GLOBAL_DO_↩ TRANSFORM)
out	*tx*	Output affine transformation, if not null

Returns

Error code

Here is the call graph for this function:



6.11.4.22 mesh_alloc_face_vertices()

```
MESHLIBAPI int mesh_alloc_face_vertices (
            MESH_FACE mf,
            INTDATA nv )
```

Allocates memory for vertices of a given mesh face.

Parameters

in	*mf*	Input mesh face
in	*nv*	New number of vertices

Returns

Error code

Here is the call graph for this function:

6.11.4.23 mesh_alloc_mesh_props()

```
MESHLIBAPI int mesh_alloc_mesh_props (
            MESH m,
            INTDATA nv,
            INTDATA nf,
            INTDATA ne,
            uint16_t flags )
```

Allocates memory for various properties of a given mesh.

Parameters

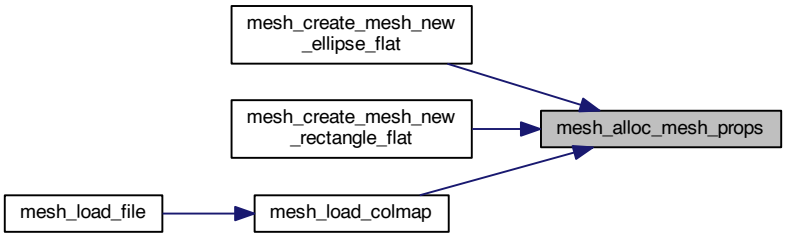
in	*m*	Input mesh
in	*nv*	New number of vertices
in	*nf*	New number of faces
in	*ne*	New number of edges
in	*flags*	Flags to allocate which properties (MESH_PROPS_VERTICES/MESH_PROPS_↩ VNORMALS/MESH_PROPS_VCOLORS/MESH_PROPS_VFACES/MESH_PROPS_↩ VSCALARS/MESH_PROPS_V_ALL_PROPS/MESH_PROPS_FACES/MESH_PROPS_↩ FNORMALS/MESH_PROPS_FCOLORS/MESH_PROPS_FAREAS/MESH_PROPS_↩ FSCALARS/MESH_PROPS_F_ALL_PROPS/MESH_PROPS_ALL_PROPS)

Returns

Error code

Here is the call graph for this function:

Here is the caller graph for this function:



6.11.4.24 mesh_calc_aabb()

```
MESHLIBAPI void mesh_calc_aabb (
            MESH m,
            MESH_VECTOR3 minv,
            MESH_VECTOR3 maxv,
            MESH_VECTOR3 center )
```

Computes axis-aligned bounding box.

Parameters

in	*m*	Given mesh
out	*minv*	Minimum point
out	*maxv*	Maximum point
out	*center*	Centre point

6.11.4.25 mesh_calc_area()

```
MESHLIBAPI FLOATDATA mesh_calc_area (
            MESH m )
```

Computes area of a triangle mesh.

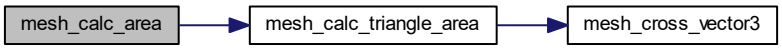
Parameters

in	*m*	Given mesh

Returns

Area (INF, if not trimesh)

Here is the call graph for this function:



6.11.4.26 mesh_calc_edges()

```
MESHLIBAPI int mesh_calc_edges (
            MESH m )
```

Computes edges of a given mesh.

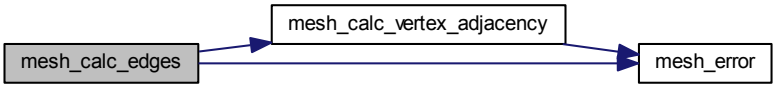
Parameters

in	*m*	Input mesh

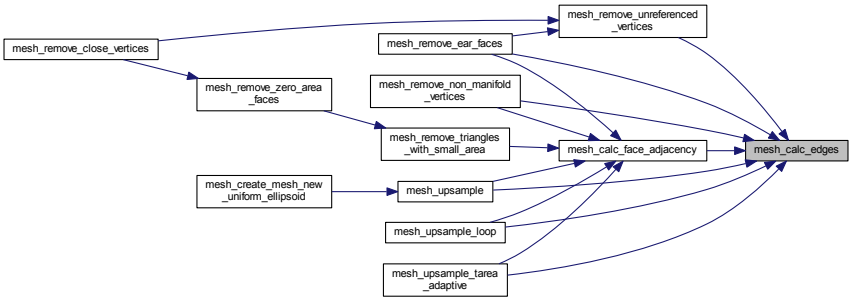
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.27 mesh_calc_face_adjacency()

```
MESHLIBAPI int mesh_calc_face_adjacency (
            MESH m )
```

Computes face adjacent faces of a given mesh.

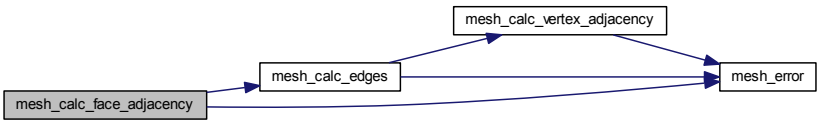
Parameters

in	*m*	Input mesh

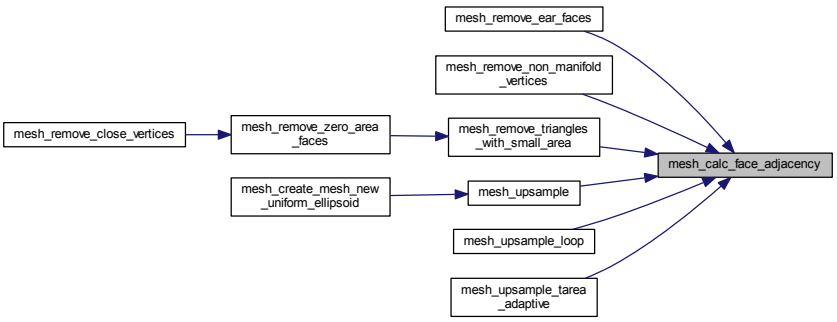
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.28 mesh_calc_face_normal()

```
MESHLIBAPI void mesh_calc_face_normal (
            MESH_VERTEX v1,
            MESH_VERTEX v2,
            MESH_VERTEX v3,
            MESH_NORMAL n )
```

Computes the face normal given 3 vertices.

Parameters

in	*v1*	First vertex
in	*v2*	Second vertex
in	*v3*	Third vertex
out	*n*	Output face normal $\mathbf{n}_f$

Returns

NULL

6.11.4.29 mesh_calc_face_normals()

```
MESHLIBAPI int mesh_calc_face_normals (
            MESH m )
```

Computes face normals of a given mesh.

Parameters

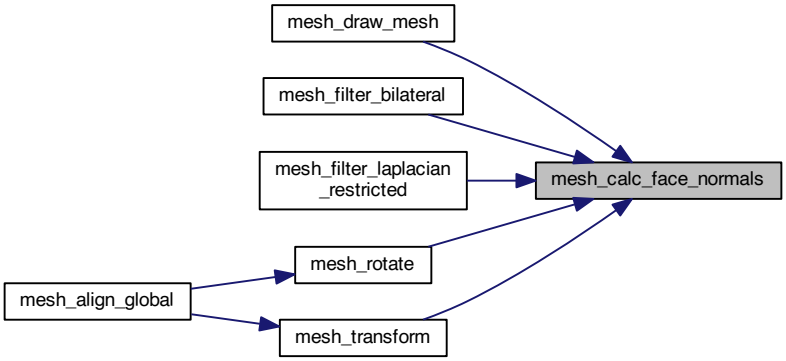
in	*m*	Input mesh

Returns

Error code

Here is the call graph for this function:

Here is the caller graph for this function:



6.11.4.30 mesh_calc_signed_area()

```
MESHLIBAPI int mesh_calc_signed_area (
            MESH m,
            MESH_VECTOR3 area )
```

Computes signed area of a triangle mesh.

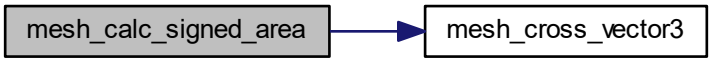
Parameters

in	*m*	Given mesh
in	*area*	Output area

Returns

Error code

Here is the call graph for this function:



6.11.4.31 mesh_calc_triangle_area()

```
MESHLIBAPI FLOATDATA mesh_calc_triangle_area (
             MESH_VERTEX a,
             MESH_VERTEX b,
             MESH_VERTEX c )
```

Computes area of a triangle.

Parameters

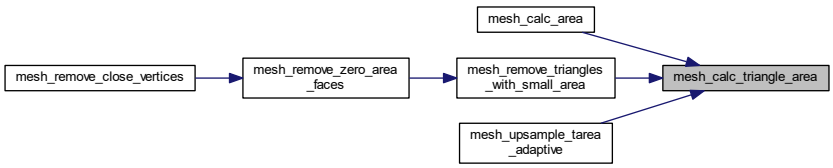
in	*a*	First vertex
in	*b*	Second vertex
in	*c*	Third vertex

Returns

Area

Here is the call graph for this function:

Here is the caller graph for this function:



6.11.4.32 mesh_calc_vertex_adjacency()

```
MESHLIBAPI int mesh_calc_vertex_adjacency (
            MESH m )
```

Computes vertex adjacent faces of a given mesh.

Parameters

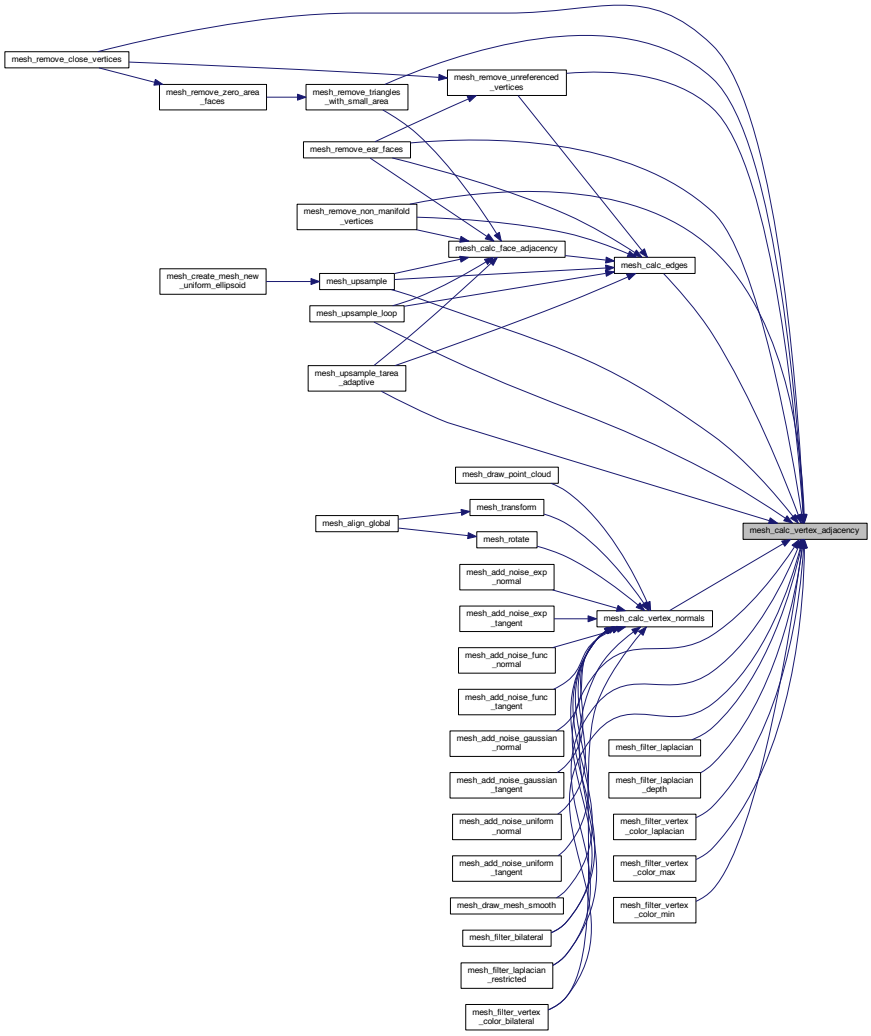
in	*m*	Input mesh

Returns

Error code

Here is the call graph for this function:

Here is the caller graph for this function:



6.11.4.33 mesh_calc_vertex_normals()

```
MESHLIBAPI int mesh_calc_vertex_normals (
            MESH m )
```

Computes vertex normals of a given mesh.

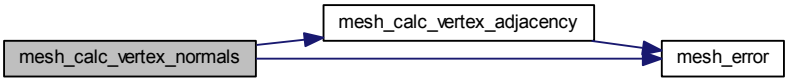
Parameters

in	*m*	Input mesh

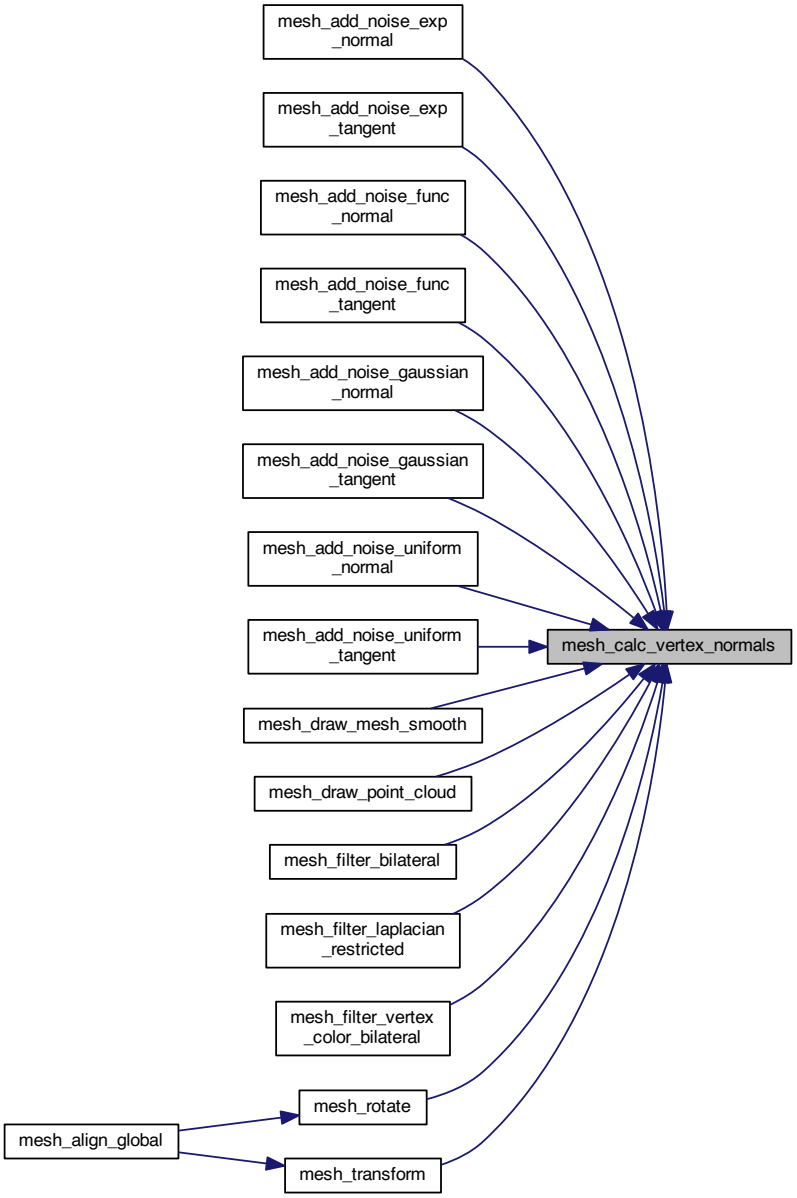
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.34 mesh_calc_volume()

```
MESHLIBAPI FLOATDATA mesh_calc_volume (
            MESH m )
```

Computes volume of a triangle mesh.

Parameters

in	*m*	Given mesh

Returns

Volume (INF, if not trimesh)

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.4.35 mesh_clone_mesh()

```
MESHLIBAPI MESH mesh_clone_mesh (
            MESH m,
            uint16_t flags )
```

Clones a given mesh into another mesh.

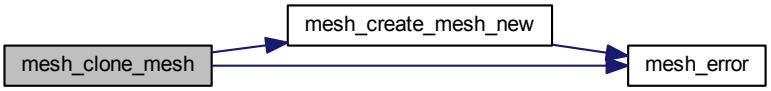
Parameters

in	*m*	Input mesh to clone
in	*flags*	Flags to copy which properties (MESH_PROPS_VERTICES/MESH_PROPS_VNORMALS/↩ MESH_PROPS_VCOLORS/MESH_PROPS_VFACES/MESH_PROPS_VSCALARS/MESH_↩ PROPS_V_ALL_PROPS/MESH_PROPS_FACES/MESH_PROPS_FNORMALS/MESH_↩ PROPS_FCOLORS/MESH_PROPS_FAREAS/MESH_PROPS_FSCALARS/MESH_PROPS↩ _F_ALL_PROPS/MESH_PROPS_ALL_PROPS)

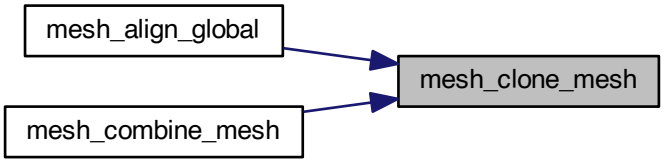
Returns

Output cloned mesh

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.36 mesh_combine_mesh()

```
MESHLIBAPI MESH mesh_combine_mesh (
            MESH m1,
            MESH m2 )
```

Combines a given mesh with another given mesh.

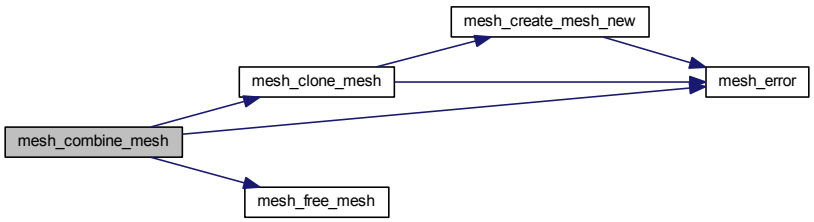
Parameters

in	*m1*	Input mesh to combine with
in	*m2*	Input mesh to combine

Returns

Output combined mesh

Here is the call graph for this function:



6.11.4.37 mesh_count_words_in_line()

```
MESHLIBAPI int mesh_count_words_in_line (
            FILEPOINTER fp,
            int * count )
```

Counts number of words in the current line.

Parameters

in	*fp*	Pointer to input file
out	*count*	Count

Returns

Status 0 - Normal/ 1- EOF

6.11.4.38 mesh_create_mesh_new()

```
MESHLIBAPI MESH mesh_create_mesh_new ( )
```

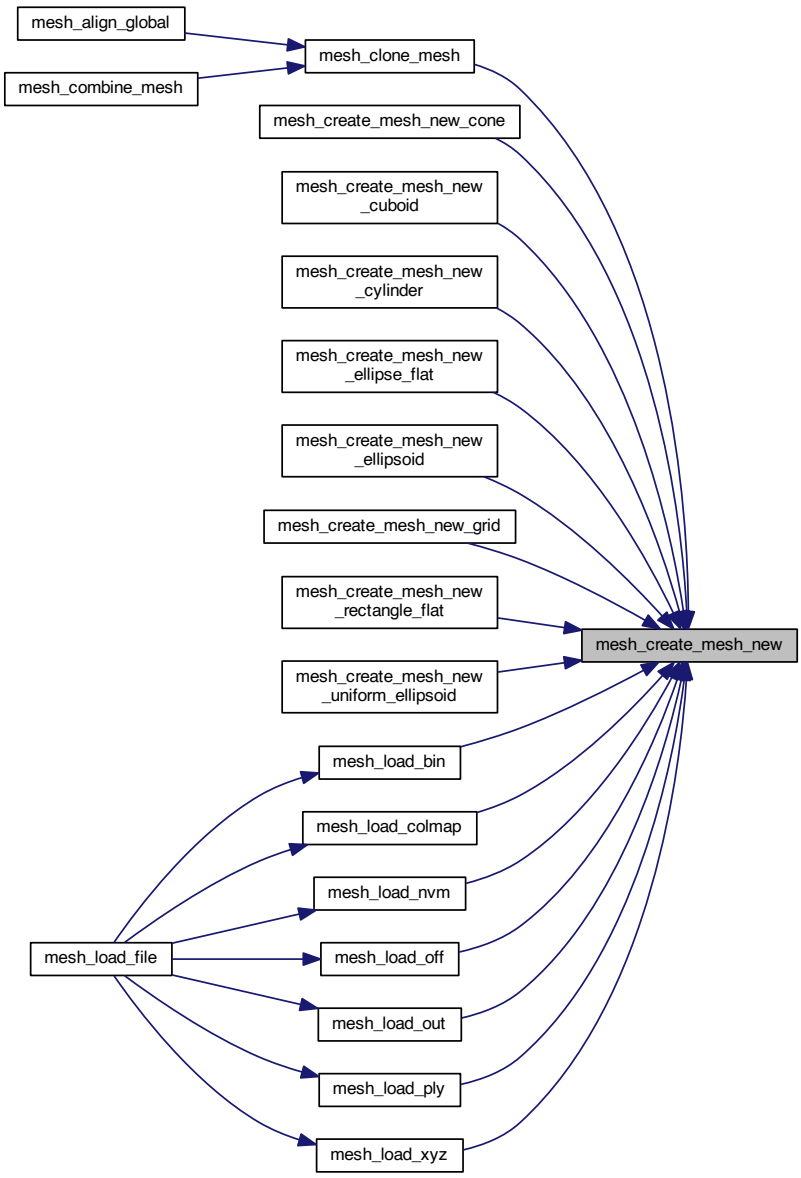
Creates a new mesh.

Returns

Output mesh

Here is the call graph for this function:

Here is the caller graph for this function:



6.11.4.39 mesh_create_mesh_new_cone()

```
MESHLIBAPI MESH mesh_create_mesh_new_cone (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a cone mesh.

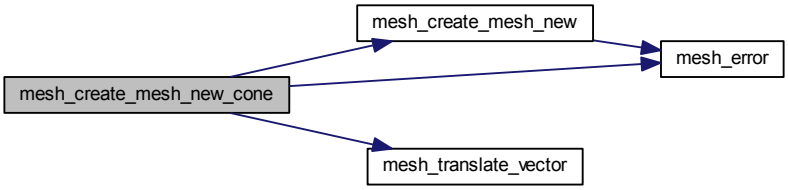
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.11.4.40 mesh_create_mesh_new_cuboid()

```
MESHLIBAPI MESH mesh_create_mesh_new_cuboid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a cuboid mesh.

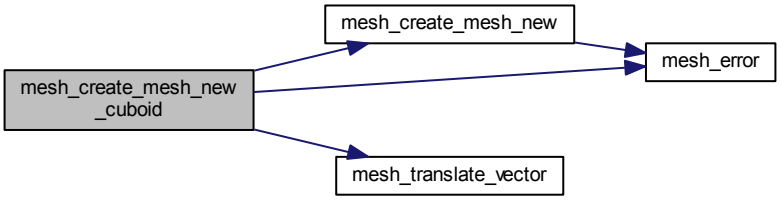
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.11.4.41 mesh_create_mesh_new_cylinder()

```
MESHLIBAPI MESH mesh_create_mesh_new_cylinder (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a cylinder mesh.

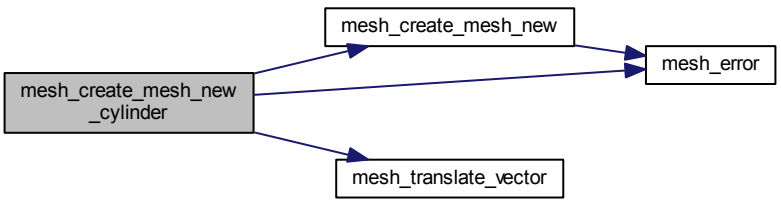
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.11.4.42 mesh_create_mesh_new_ellipse_flat()

```
MESHLIBAPI MESH mesh_create_mesh_new_ellipse_flat (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a flat ellipse mesh.

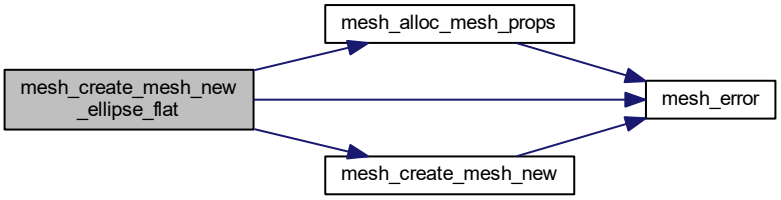
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.11.4.43 mesh_create_mesh_new_ellipsoid()

```
MESHLIBAPI MESH mesh_create_mesh_new_ellipsoid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates an ellipsoid mesh.

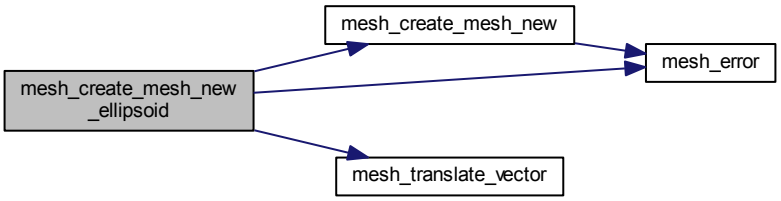
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.11.4.44 mesh_create_mesh_new_grid()

```
MESHLIBAPI MESH mesh_create_mesh_new_grid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos,
            INTDATA m,
            INTDATA n )
```

Creates a grid mesh.

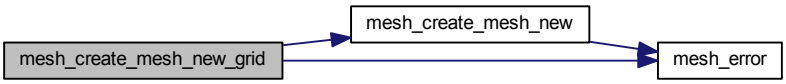
Parameters

in	*sz*	Size vector
in	*pos*	Position vector
in	*m*	Number of x-samples
in	*n*	Number of y-samples

Returns

Output mesh

Here is the call graph for this function:



6.11.4.45 mesh_create_mesh_new_rectangle_flat()

```
MESHLIBAPI MESH mesh_create_mesh_new_rectangle_flat (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos )
```

Creates a flat rectangle mesh.

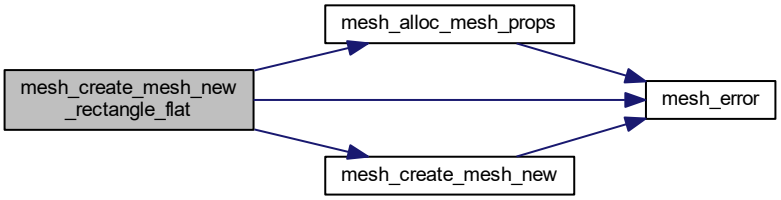
Parameters

in	*sz*	Size vector
in	*pos*	Position vector

Returns

Output mesh

Here is the call graph for this function:



6.11.4.46 mesh_create_mesh_new_uniform_ellipsoid()

```
MESHLIBAPI MESH mesh_create_mesh_new_uniform_ellipsoid (
            MESH_VECTOR3 sz,
            MESH_VECTOR3 pos,
            int n )
```

Creates a uniformly sampled ellipsoid mesh.

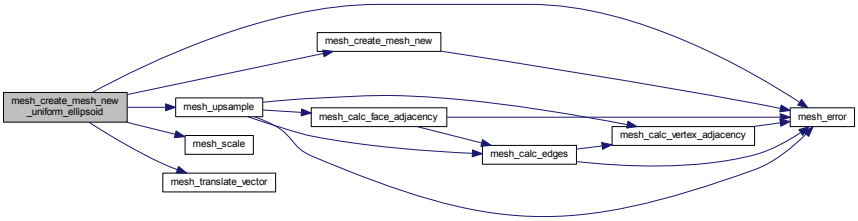
Parameters

in	*sz*	Size vector
in	*pos*	Position vector
in	*n*	Number of upsample ~3

Returns

Output mesh

Here is the call graph for this function:



6.11.4.47 mesh_cross_normal()

```
MESHLIBAPI void mesh_cross_normal (
            MESH_NORMAL x,
            MESH_NORMAL y,
            MESH_NORMAL z )
```

Computes the normalized cross product of two normals.

Parameters

in	*x*	First normal
in	*y*	Second normal
out	*z*	Output cross product $\frac{\mathbf{x}\times\mathbf{y}}{\|\mathbf{x}\times\mathbf{y}\|_2}$

Returns

NULL

6.11.4.48 mesh_cross_vector3()

```
MESHLIBAPI void mesh_cross_vector3 (
            MESH_VECTOR3 x,
            MESH_VECTOR3 y,
            MESH_VECTOR3 z )
```

Computes the cross product of two 3-d vectors.

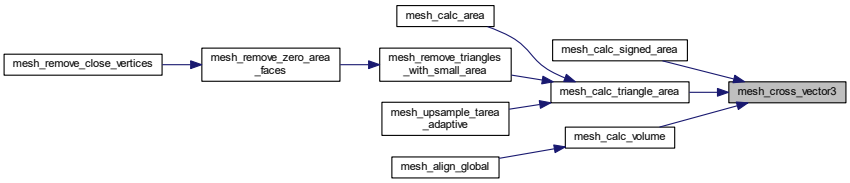
Parameters

in	*x*	First vector
in	*y*	Second vector
out	*z*	Output cross product $\mathbf{x} \times \mathbf{y}$

Returns

NULL

Here is the caller graph for this function:



6.11.4.49 mesh_draw_mesh()

```
MESHLIBAPI void mesh_draw_mesh (
            MESH m )
```

Draws a given mesh in OpenGL context in flat shading.

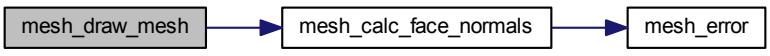
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:



6.11.4.50 mesh_draw_mesh_smooth()

```
MESHLIBAPI void mesh_draw_mesh_smooth (
            MESH m )
```

Draws a given mesh in OpenGL context in smoothing shading.

Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:

6.11.4.51 mesh_draw_point_cloud()

```
MESHLIBAPI void mesh_draw_point_cloud (
            MESH m )
```

Draws a given mesh in OpenGL context as pointcloud.

Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:

6.11.4.52 mesh_error()

```
MESHLIBAPI void mesh_error (
            int type )
```

Displays error message and exits.

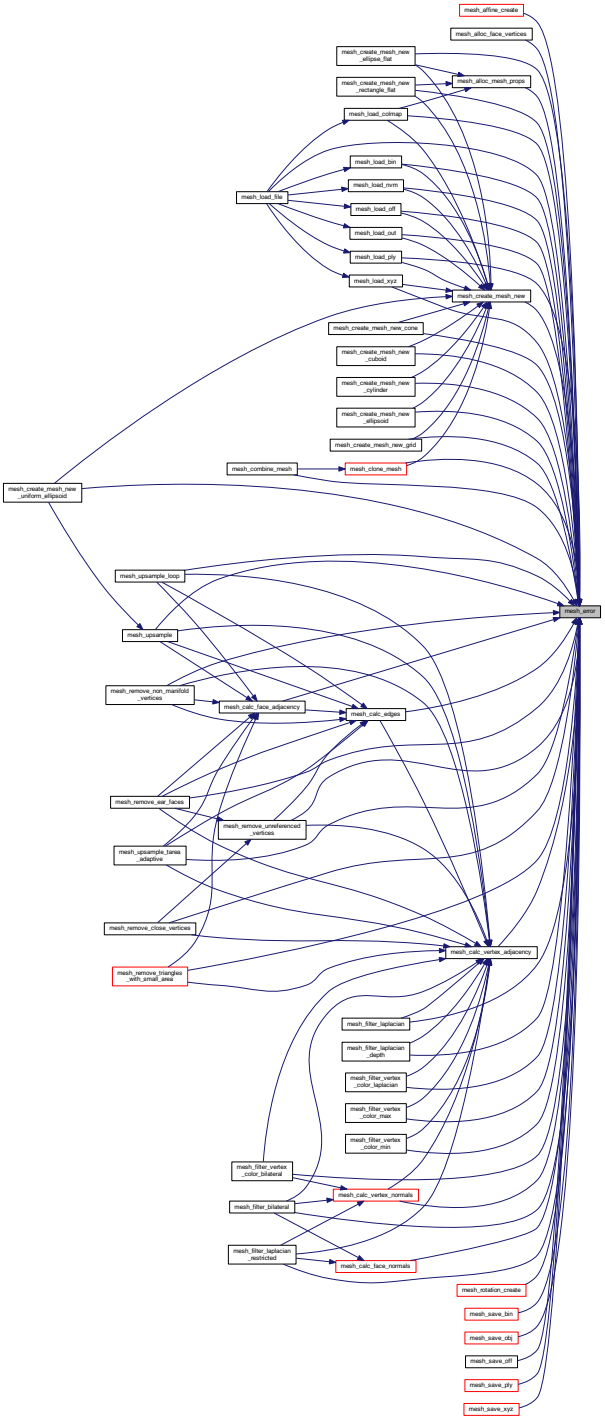
Parameters

in	*type*	Error type (MESH_ERR_MALLOC/MESH_ERR_SIZE_MISMATCH/MESH_ERR_FNOTOPEN)

Returns

NULL

Here is the caller graph for this function:



6.11.4.53 mesh_filter_bilateral()

```
MESHLIBAPI int mesh_filter_bilateral (
            MESH m,
```

```
            FLOATDATA sigma_c,
            FLOATDATA sigma_s,
            int niters )
```

Mesh bilateral filter.

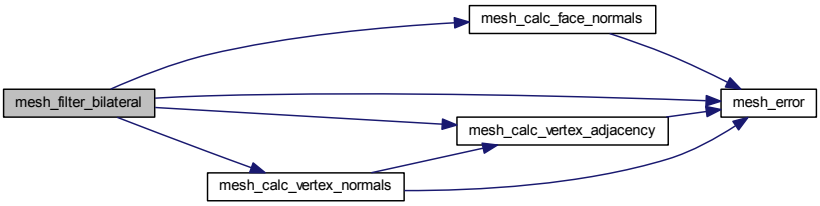
Parameters

in	*m*	Input mesh
in	*sigma_c*	Range standard deviation
in	*sigma_s*	Spatial standard deviation
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.11.4.54 mesh_filter_laplacian()

```
MESHLIBAPI int mesh_filter_laplacian (
            MESH m,
            FLOATDATA r )
```

Mesh Laplacian filter.

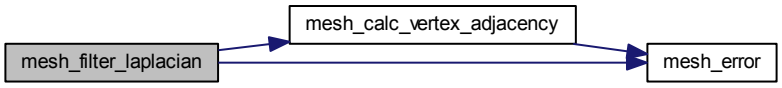
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion

Returns

Error code

Here is the call graph for this function:



6.11.4.55 mesh_filter_laplacian_depth()

```
MESHLIBAPI int mesh_filter_laplacian_depth (
            MESH m,
            FLOATDATA r,
            MESH_VECTOR3 vp )
```

Mesh Depth Laplacian filter.

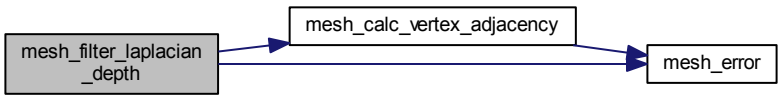
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion
in	*vp*	View-point

Returns

Error code

Here is the call graph for this function:



6.11.4.56 mesh_filter_laplacian_restricted()

```
MESHLIBAPI int mesh_filter_laplacian_restricted (
            MESH m,
            FLOATDATA r,
            FLOATDATA ang )
```

Restricted Mesh Laplacian filter.

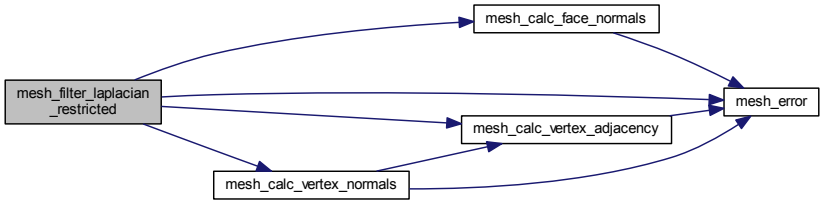
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion
in	*ang*	Minimum angle in degrees to suppress filtering

Returns

Error code

Here is the call graph for this function:



6.11.4.57 mesh_filter_taubin()

```
MESHLIBAPI int mesh_filter_taubin (
            MESH m,
            FLOATDATA lambd,
            FLOATDATA mu,
            int niters )
```

Mesh $\lambda - \mu$ Taubin filter.

Parameters

in	*m*	Input mesh
in	*lambd*	λ value
in	*mu*	μ value
in	*niters*	Number of iterations

Returns

Error code

6.11.4.58 mesh_filter_vertex_color_bilateral()

```
MESHLIBAPI int mesh_filter_vertex_color_bilateral (
            MESH m,
            FLOATDATA sigma_k,
            FLOATDATA sigma_c,
            FLOATDATA sigma_s,
            int niters )
```

Mesh bilateral vertex color filter.

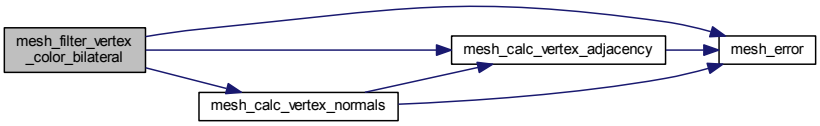
Parameters

in	*m*	Input mesh
in	*sigma_k*	Color standard deviation
in	*sigma_c*	Range standard deviation
in	*sigma_s*	Spatial standard deviation
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.11.4.59 mesh_filter_vertex_color_laplacian()

```
MESHLIBAPI int mesh_filter_vertex_color_laplacian (
            MESH m,
            FLOATDATA r )
```

Mesh Laplacian vertex color filter.

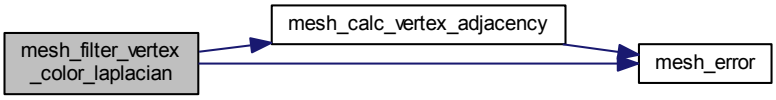
Parameters

in	*m*	Input mesh
in	*r*	Amount of diffusion

Returns

Error code

Here is the call graph for this function:



6.11.4.60 mesh_filter_vertex_color_max()

```
MESHLIBAPI int mesh_filter_vertex_color_max (
            MESH m,
            INTDATA niters )
```

Mesh maximum intensity vertex color filter.

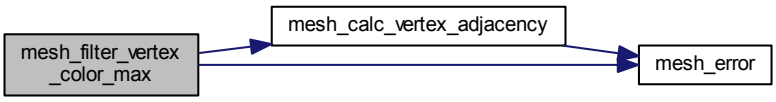
Parameters

in	*m*	Input mesh
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.11.4.61 mesh_filter_vertex_color_min()

```
MESHLIBAPI int mesh_filter_vertex_color_min (
            MESH m,
            INTDATA niters )
```

Mesh minimum intensity vertex color filter.

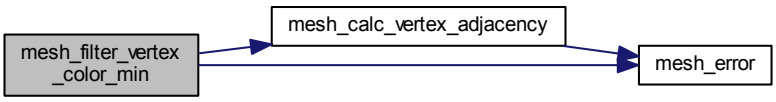
Parameters

in	*m*	Input mesh
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.11.4.62 mesh_find()

```
MESHLIBAPI INTDATA mesh_find (
            MESH_STRUCT s,
            INTDATA q )
```

Finds an item in an INTDATA structure.

Parameters

in	*s*	Input INTDATA structure
in	*q*	Query INTDATA

Returns

Index or -1

6.11.4.63 mesh_find2()

```
MESHLIBAPI INTDATA mesh_find2 (
            MESH_STRUCT2 s,
            INTDATA q )
```

Finds an item in an INTDATA2 structure.

Parameters

in	*s*	Input INTDATA2 structure
in	*q*	Query INTDATA2

Returns

Index or -1

6.11.4.64 mesh_find3()

```
MESHLIBAPI INTDATA mesh_find3 (
            MESH_STRUCT3 s,
            INTDATA q )
```

Finds an item in an INTDATA3 structure.

Parameters

in	*s*	Input INTDATA3 structure
in	*q*	Query INTDATA3

Returns

Index or -1

6.11.4.65 mesh_free_face_vertices()

```
MESHLIBAPI int mesh_free_face_vertices (
            MESH_FACE mf )
```

Frees memory for vertices of a given mesh face.

Parameters

in	*mf*	Input mesh face

Returns

Error code

6.11.4.66 mesh_free_mesh()

```
MESHLIBAPI void mesh_free_mesh (
            MESH m )
```

Frees a mesh.

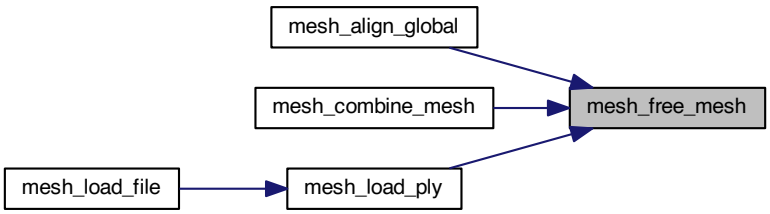
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the caller graph for this function:



6.11.4.67 mesh_free_mesh_props()

```
MESHLIBAPI int mesh_free_mesh_props (
            MESH m,
            uint16_t flags )
```

Frees memory for various properties of a given mesh.

Parameters

in	*m*	Input mesh
in	*flags*	Flags to free which properties (MESH_PROPS_VERTICES/MESH_PROPS_VNORMALS/↩ MESH_PROPS_VCOLORS/MESH_PROPS_VFACES/MESH_PROPS_VSCALARS/MESH_↩ PROPS_V_ALL_PROPS/MESH_PROPS_FACES/MESH_PROPS_FNORMALS/MESH_↩ PROPS_FCOLORS/MESH_PROPS_FAREAS/MESH_PROPS_FSCALARS/MESH_PROPS↩ _F_ALL_PROPS/MESH_PROPS_ALL_PROPS)

Returns

Error code

6.11.4.68 mesh_go_next_word()

```
MESHLIBAPI int mesh_go_next_word (
            FILEPOINTER fp )
```

Points to the next word.

Parameters

in	*fp*	Pointer to input file

Returns

Status 0 - Normal/ 1- EOF

6.11.4.69 mesh_isnumeric()

```
MESHLIBAPI int mesh_isnumeric (
            FILEPOINTER fp )
```

Checks if numeric or not.

Parameters

in	*fp*	Pointer to input file

Returns

1 for numeric/ else - for non-numeric

6.11.4.70 mesh_load_bin()

```
MESHLIBAPI MESH mesh_load_bin (
            const char * fname )
```

Reads a mesh from a CloudCompare BINv1 file.

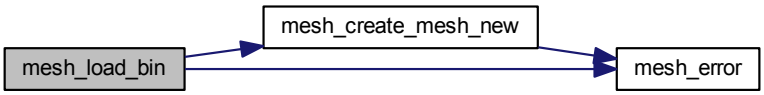
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.71 mesh_load_colmap()

```
MESHLIBAPI MESH mesh_load_colmap (
            const char * fname )
```

Reads a mesh from a COLMAP BIN file.

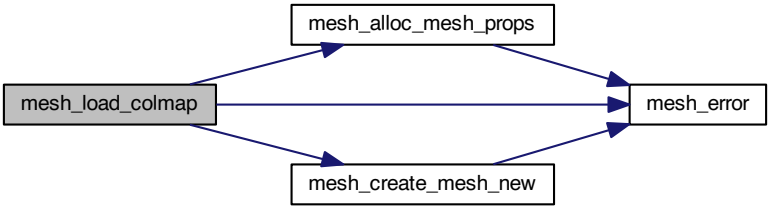
Parameters

in	*fname*	Input foldername (where points3D.bin resides)

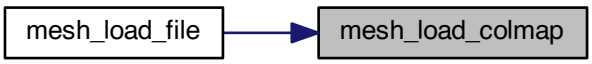
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.72 mesh_load_file()

```
MESHLIBAPI MESH mesh_load_file (
            const char * fname )
```

Reads a mesh from an OFF/PLY/ASC/XYZ/BINv1/BundlerOUT/NVM file.

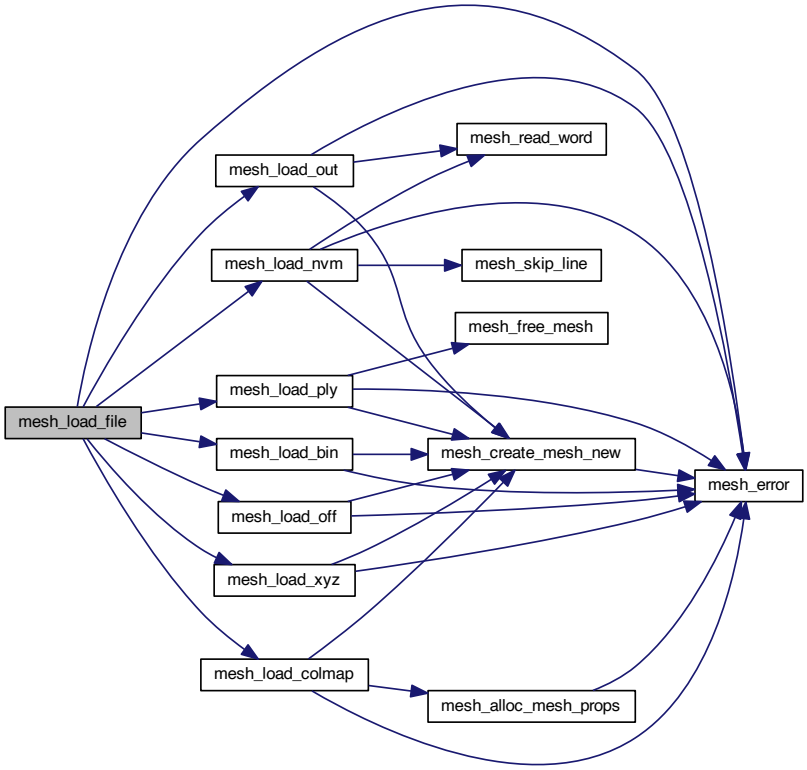
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



6.11.4.73 mesh_load_nvm()

```
MESHLIBAPI MESH mesh_load_nvm (
            const char * fname )
```

Reads a mesh from an NVM file.

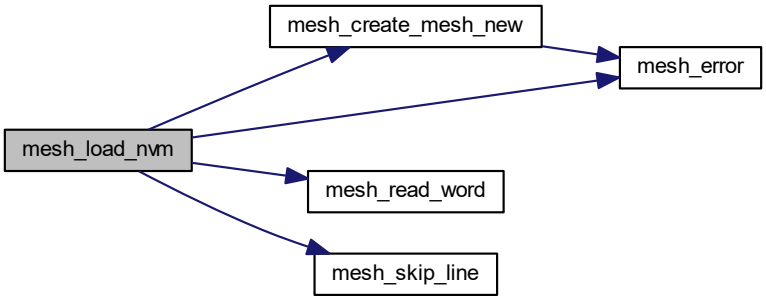
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.74 mesh_load_off()

```
MESHLIBAPI MESH mesh_load_off (
            const char * fname )
```

Reads a mesh from an OFF file.

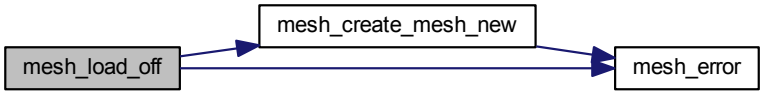
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.75 mesh_load_out()

```
MESHLIBAPI MESH mesh_load_out (
            const char * fname )
```

Reads a mesh from a Bundler OUT file.

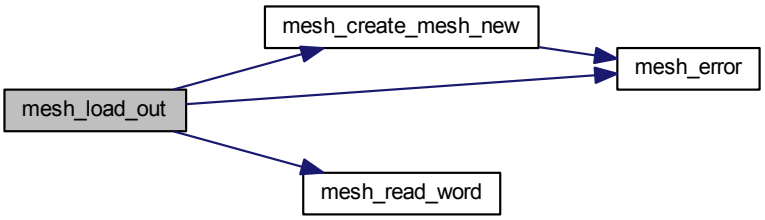
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.76 mesh_load_ply()

```
MESHLIBAPI MESH mesh_load_ply (
            const char * fname )
```

Reads a mesh from a PLY file.

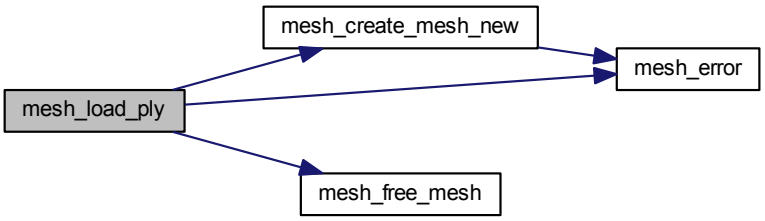
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.77 mesh_load_xyz()

```
MESHLIBAPI MESH mesh_load_xyz (
            const char * fname )
```

Read a mesh from an ASC/XYZ file.

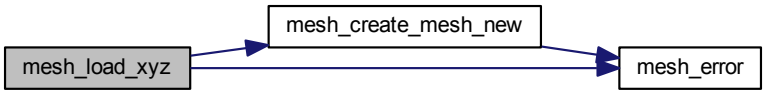
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.78 mesh_read_word()

```
MESHLIBAPI int mesh_read_word (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word and moves to the next word.

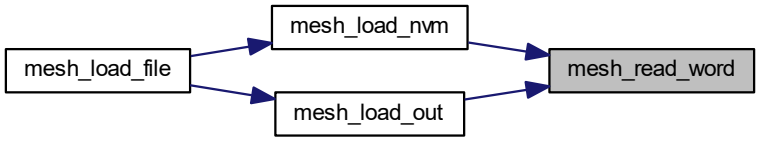
Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1 - EOF / 2 - Overflow

Here is the caller graph for this function:



6.11.4.79 mesh_read_word_only()

```
MESHLIBAPI int mesh_read_word_only (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word without moving to the next word.

Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF / 2 - Overflow

6.11.4.80 mesh_read_word_only_skip_comment()

```
MESHLIBAPI int mesh_read_word_only_skip_comment (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word skipping comment with # without moving to the next word.

Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF / 2 - Overflow / 3 - Commented

6.11.4.81 mesh_read_word_skip_comment()

```
MESHLIBAPI int mesh_read_word_skip_comment (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word skipping comment with # and moves to the next word.

Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF / 2 - Overflow / 3 - Commented

6.11.4.82 mesh_remove_boundary_faces()

```
MESHLIBAPI int mesh_remove_boundary_faces (
            MESH m,
            int iters )
```

Removes boundary faces and connecting elements.

Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

Returns

Error code

6.11.4.83 mesh_remove_boundary_vertices()

```
MESHLIBAPI int mesh_remove_boundary_vertices (
            MESH m,
            int iters )
```

Removes boundary vertices and connecting elements.

Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

Returns

Error code

6.11.4.84 mesh_remove_close_vertices()

```
MESHLIBAPI int mesh_remove_close_vertices (
            MESH m,
            FLOATDATA r )
```

Removes close vertices.

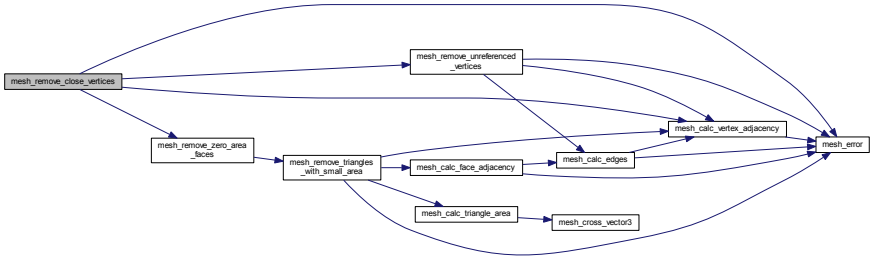
Parameters

in	*m*	Input mesh
in	*r*	Maximum distance between two vertices

Returns

Error code

Here is the call graph for this function:



6.11.4.85 mesh_remove_ear_faces()

```
MESHLIBAPI int mesh_remove_ear_faces (
            MESH m,
            int niters )
```

Removes ear faces and connecting vertices.

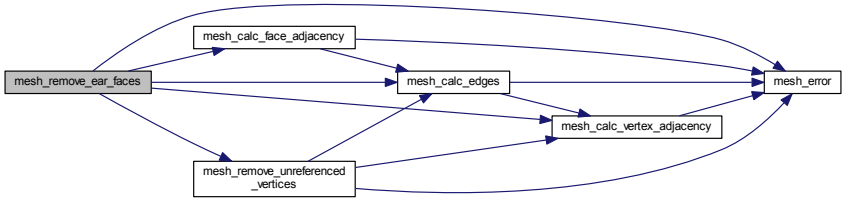
Parameters

in	*m*	Input mesh
in	*niters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.11.4.86 mesh_remove_non_manifold_vertices()

```
MESHLIBAPI int mesh_remove_non_manifold_vertices (
            MESH m )
```

Removes non-manifold vertices.

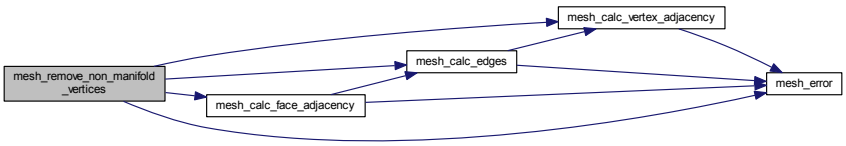
Parameters

in	*m*	Input mesh

Returns

Error code

Here is the call graph for this function:



6.11.4.87 mesh_remove_triangles_with_small_area()

```
MESHLIBAPI int mesh_remove_triangles_with_small_area (
            MESH m,
            FLOATDATA area )
```

Removes triangles with area smaller than a given value.

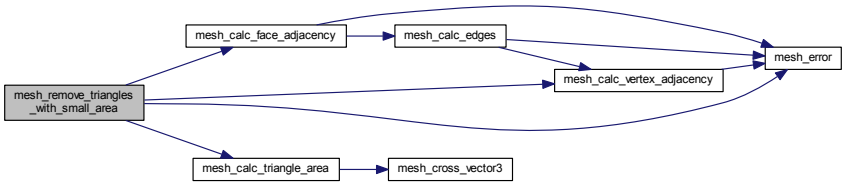
Parameters

in	*m*	Input mesh
in	*area*	Given area

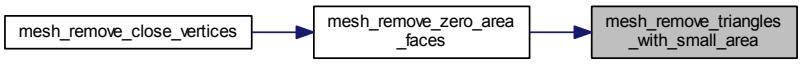
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.88 mesh_remove_unreferenced_vertices()

```
MESHLIBAPI int mesh_remove_unreferenced_vertices (
            MESH m )
```

Removes unreferenced vertices.

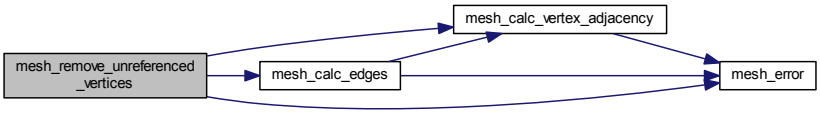
Parameters

in	*m*	Input mesh

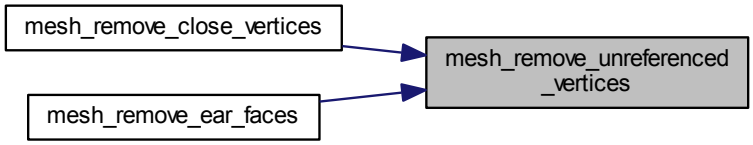
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.89 mesh_remove_zero_area_faces()

```
MESHLIBAPI int mesh_remove_zero_area_faces (
            MESH m )
```

Removes triangles with zero area.

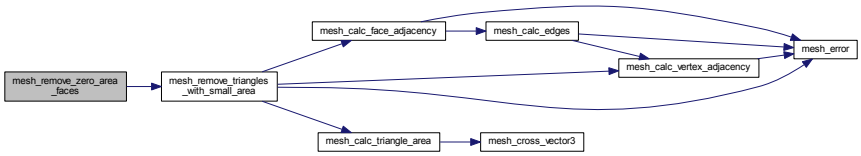
Parameters

in	*m*	Input mesh

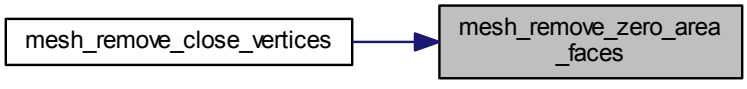
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.90 mesh_rotate()

```
MESHLIBAPI int mesh_rotate (
            MESH m,
            MESH_ROTATION r )
```

Rotates a mesh by a given rotation.

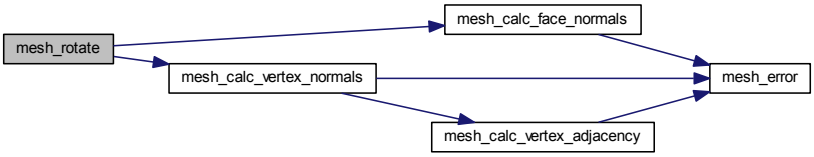
Parameters

in	*m*	Input vertex
in	*r*	Input rotation

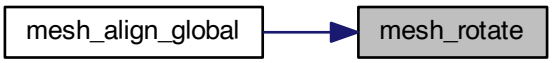
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.91 mesh_rotation_create()

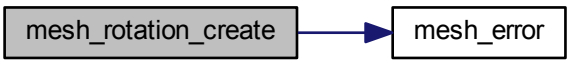
```
MESHLIBAPI MESH_ROTATION mesh_rotation_create ( )
```

Creates a new rotation.

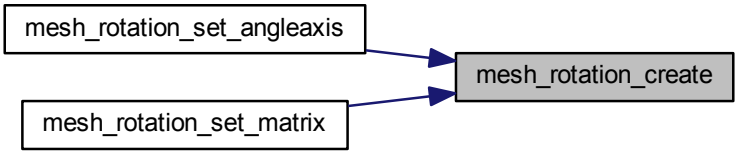
Returns

Output rotation

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.92 mesh_rotation_free()

```
MESHLIBAPI void mesh_rotation_free (
            MESH_ROTATION r )
```

Frees a given rotation.

Parameters

r	Input rotation

Returns

NULL

6.11.4.93 mesh_rotation_set_angleaxis()

```
MESHLIBAPI MESH_ROTATION mesh_rotation_set_angleaxis (
            FLOATDATA ang,
            MESH_NORMAL axis,
            MESH_ROTATION r )
```

Sets rotation from angle axis.

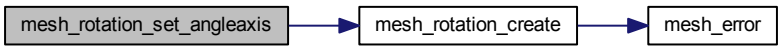
Parameters

in	*ang*	Input angle of rotation
out	*axis*	Input axis of rotation
out	*r*	Input rotation

Returns

Output rotation

Here is the call graph for this function:



6.11.4.94 mesh_rotation_set_matrix()

```
MESHLIBAPI MESH_ROTATION mesh_rotation_set_matrix (
            FLOATDATA * mat,
            MESH_ROTATION r )
```

Sets rotation from a matrix.

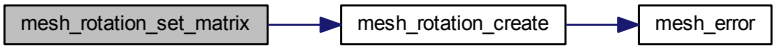
Parameters

in	*mat*	Input matrix
out	*r*	Input rotation

Returns

Output rotation

Here is the call graph for this function:



6.11.4.95 mesh_save_bin()

```
MESHLIBAPI int mesh_save_bin (
            MESH m,
            const char * fname )
```

Saves a mesh to a BINv1 file.

Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.4.96 mesh_save_file()

```
MESHLIBAPI int mesh_save_file (
            MESH m,
            const char * fname )
```

Saves a mesh to an OFF/PLY/ASC/XYZ/BIN/OBJ file.

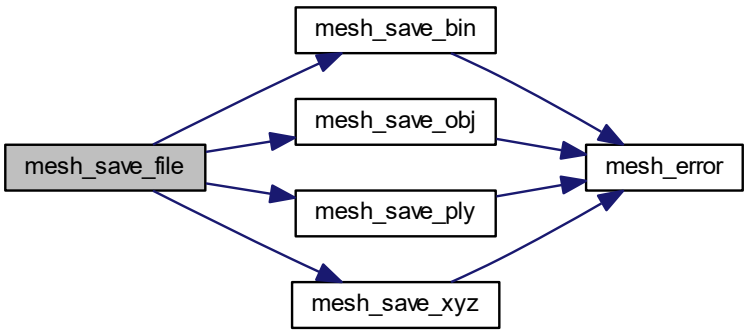
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



6.11.4.97 mesh_save_obj()

```
MESHLIBAPI int mesh_save_obj (
            MESH m,
            const char * fname )
```

Saves a mesh to an OBJ file.

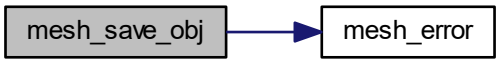
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.98 mesh_save_off()

```
MESHLIBAPI int mesh_save_off (
            MESH m,
            const char * fname )
```

Saves a mesh to an OFF file.

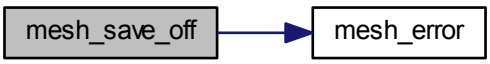
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



6.11.4.99 mesh_save_ply()

```
MESHLIBAPI int mesh_save_ply (
            MESH m,
            const char * fname )
```

Saves a mesh to a PLY file.

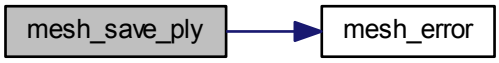
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.100 mesh_save_xyz()

```
MESHLIBAPI int mesh_save_xyz (
            MESH m,
            const char * fname )
```

Saves a mesh to an XYZ file.

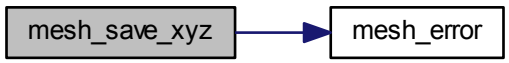
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.101 mesh_scale()

```
MESHLIBAPI int mesh_scale (
            MESH m,
            FLOATDATA sx,
            FLOATDATA sy,
            FLOATDATA sz )
```

Scales a mesh by x, y and z amounts.

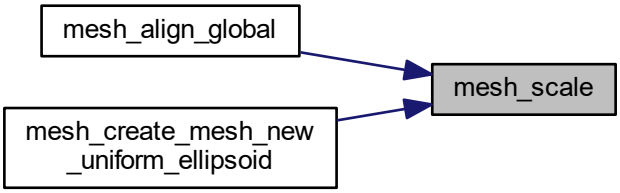
Parameters

in	*m*	Input mesh
in	*sx*	X component
in	*sy*	Y component
in	*sz*	Z component

Returns

Error code

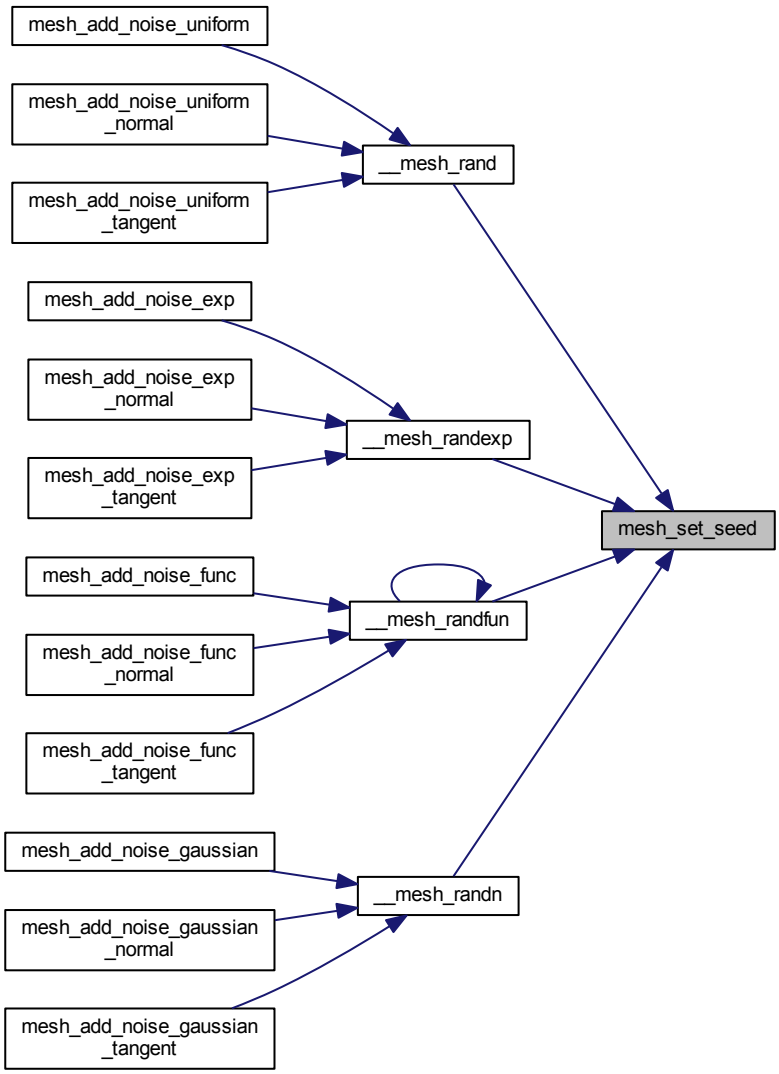
Here is the caller graph for this function:



6.11.4.102 mesh_set_seed()

```
MESHLIBAPI void mesh_set_seed (
            int seed )
```

Here is the caller graph for this function:



6.11.4.103 mesh_skip_line()

```
MESHLIBAPI int mesh_skip_line (
            FILEPOINTER fp )
```

Skips to next line.

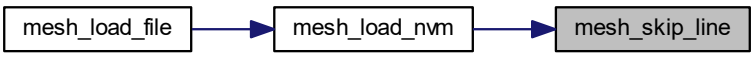
Parameters

in	*fp*	Pointer to input file

Returns

Status 0 - Normal/ 1- EOF

Here is the caller graph for this function:



6.11.4.104 mesh_transform()

```
MESHLIBAPI int mesh_transform (
            MESH m,
            MESH_AFFINE tx )
```

Transforms a mesh by a given affine transformation.

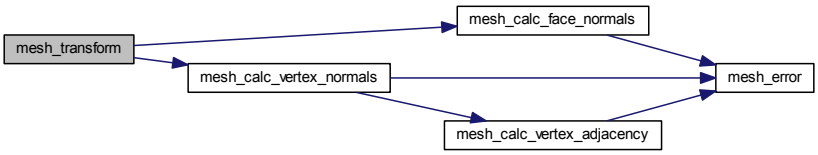
Parameters

in	*m*	Input vertex
in	*tx*	Input affine transformation

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.11.4.105 mesh_translate()

```
MESHLIBAPI int mesh_translate (
            MESH m,
            FLOATDATA x,
            FLOATDATA y,
            FLOATDATA z )
```

Translates a mesh by x, y and z amounts.

Parameters

in	*m*	Input mesh
in	*x*	X component
in	*y*	Y component
in	*z*	Z component

Returns

Error code

6.11.4.106 mesh_translate_vector()

```
MESHLIBAPI int mesh_translate_vector (
            MESH m,
            MESH_VECTOR3 v )
```

Translates a mesh by a given 3-d vector.

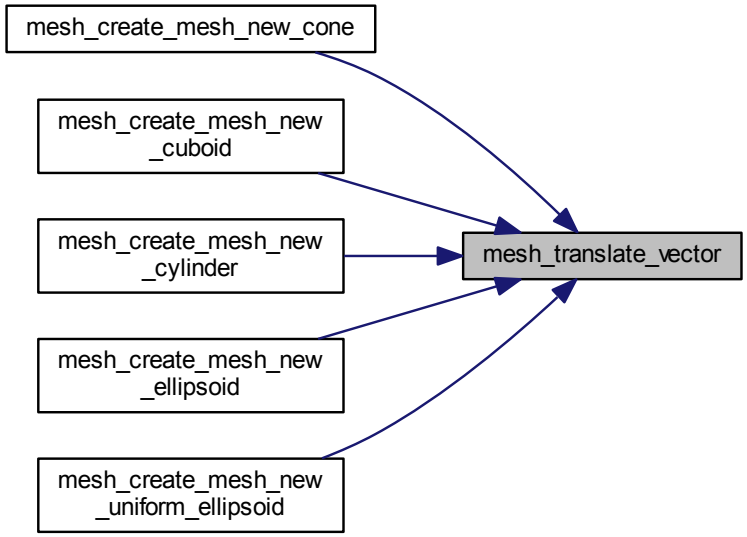
Parameters

in	*m*	Input mesh
in	*v*	Input vector

Returns

Error code

Here is the caller graph for this function:



6.11.4.107 mesh_upsample()

```
MESHLIBAPI int mesh_upsample (
            MESH m,
            int iters )
```

Upsamples a given mesh.

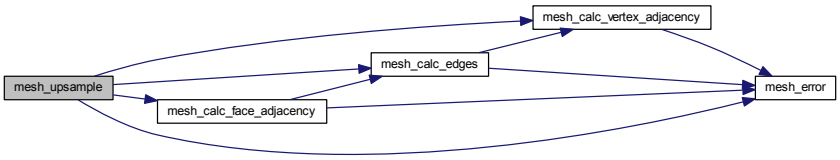
Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

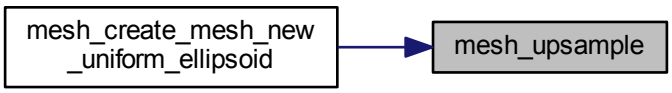
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.4.108 mesh_upsample_loop()

```
MESHLIBAPI int mesh_upsample_loop (
            MESH m,
            int iters )
```

Upsamples a given mesh using Loop's algorithm.

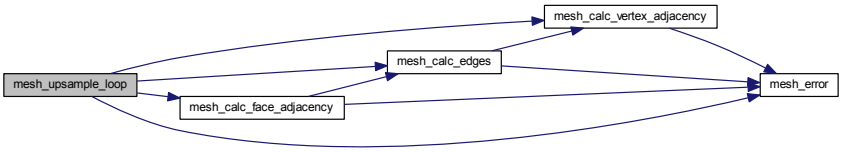
Parameters

in	*m*	Input mesh
in	*iters*	Number of iterations

Returns

Error code

Here is the call graph for this function:



6.11.4.109 mesh_upsample_tarea_adaptive()

```
MESHLIBAPI int mesh_upsample_tarea_adaptive (
            MESH m,
            int miters,
            FLOATDATA e )
```

Upsamples a given mesh upto a given triangle area threshold.

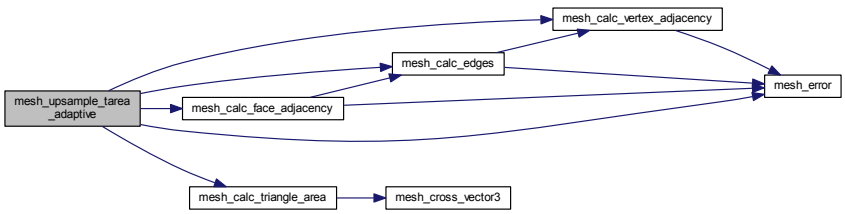
Parameters

in	*m*	Input mesh
in	*miters*	Maximum number of iterations
in	*e*	Triangle area threshold

Returns

Error code

Here is the call graph for this function:



6.11.4.110 mesh_vertex_rotate()

```
MESHLIBAPI MESH_VERTEX mesh_vertex_rotate (
            MESH_VERTEX v,
            MESH_ROTATION r )
```

Rotates a vertex by a given rotation.

Parameters

in	*v*	Input vertex
in	*r*	Input rotation

Returns

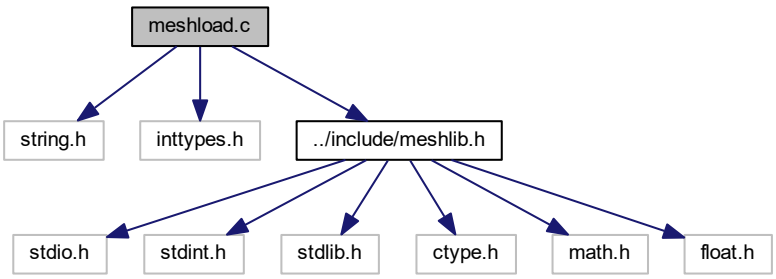
Output vertex

6.12 meshload.c File Reference

This file contains functions pertaining to loading different mesh file types.

```
#include <string.h>
#include <inttypes.h>
#include "../include/meshlib.h"
```

Include dependency graph for meshload.c:



Functions

- MESH mesh_load_file (const char *fname)

  Reads a mesh from an OFF/PLY/ASC/XYZ/BINv1/BundlerOUT/NVM file.
- MESH mesh_load_off (const char *fname)

  Reads a mesh from an OFF file.
- MESH mesh_load_xyz (const char *fname)

  Read a mesh from an ASC/XYZ file.
- MESH mesh_load_ply (const char *fname)

Reads a mesh from a PLY file.

- MESH mesh_load_bin (const char *fname)

  Reads a mesh from a CloudCompare BINv1 file.

- MESH mesh_load_out (const char *fname)

  Reads a mesh from a Bundler OUT file.

- MESH mesh_load_nvm (const char *fname)

  Reads a mesh from an NVM file.

- MESH mesh_load_colmap (const char *fname)

  Reads a mesh from a COLMAP BIN file.

6.12.1 Detailed Description

This file contains functions pertaining to loading different mesh file types.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.12.2 Function Documentation

6.12.2.1 mesh_load_bin()

```
MESH mesh_load_bin (
            const char * fname )
```

Reads a mesh from a CloudCompare BINv1 file.

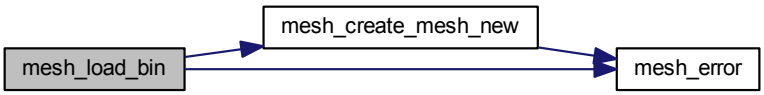
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.12.2.2 mesh_load_colmap()

```
MESH mesh_load_colmap (
            const char * fname )
```

Reads a mesh from a COLMAP BIN file.

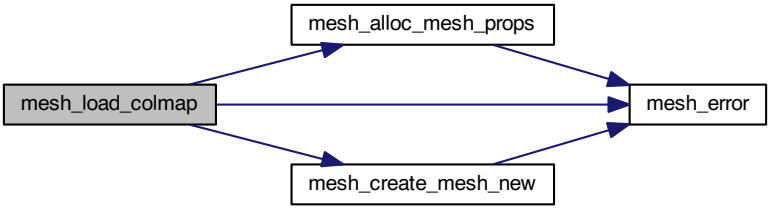
Parameters

in	*fname*	Input foldername (where points3D.bin resides)

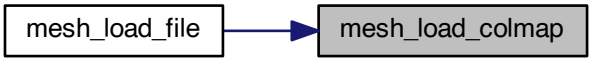
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.2.3 mesh_load_file()

```
MESH mesh_load_file (
            const char * fname )
```

Reads a mesh from an OFF/PLY/ASC/XYZ/BINv1/BundlerOUT/NVM file.

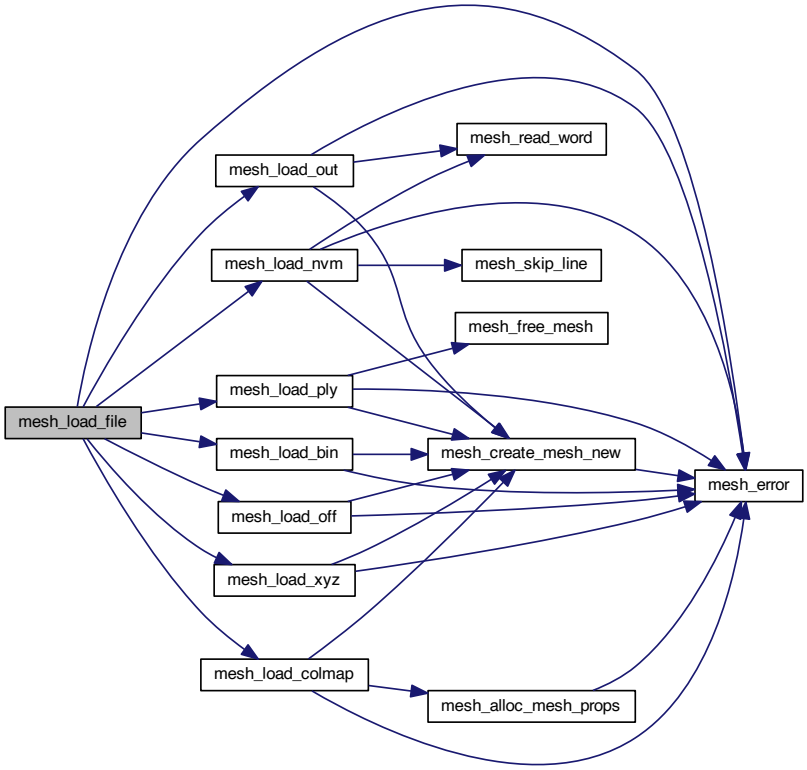
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



6.12.2.4 mesh_load_nvm()

```
MESH mesh_load_nvm (
            const char * fname )
```

Reads a mesh from an NVM file.

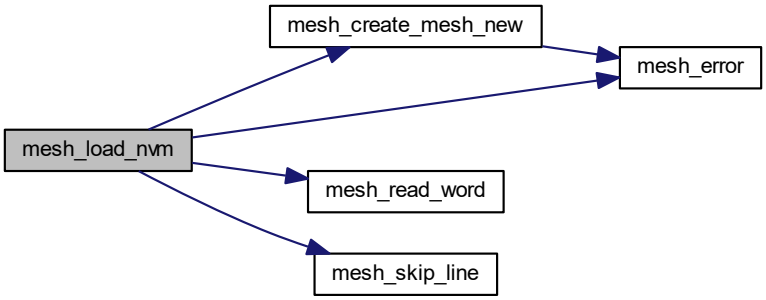
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.12.2.5 mesh_load_off()

```
MESH mesh_load_off (
            const char * fname )
```

Reads a mesh from an OFF file.

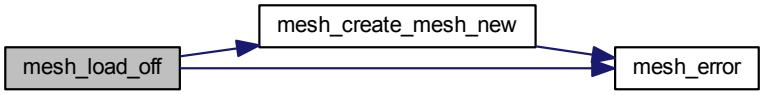
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.12.2.6 mesh_load_out()

```
MESH mesh_load_out (
            const char * fname )
```

Reads a mesh from a Bundler OUT file.

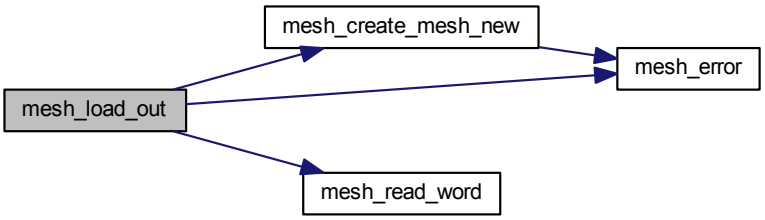
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.12.2.7 mesh_load_ply()

```
MESH mesh_load_ply (
            const char * fname )
```

Reads a mesh from a PLY file.

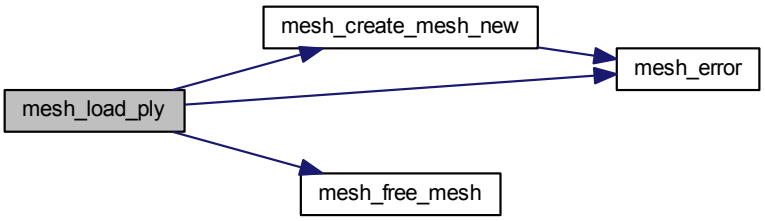
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

6.12.2.8 mesh_load_xyz()

```
MESH mesh_load_xyz (
            const char * fname )
```

Read a mesh from an ASC/XYZ file.

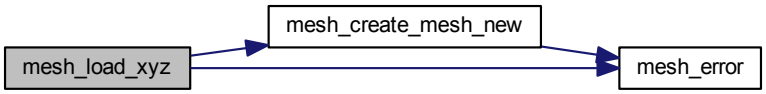
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



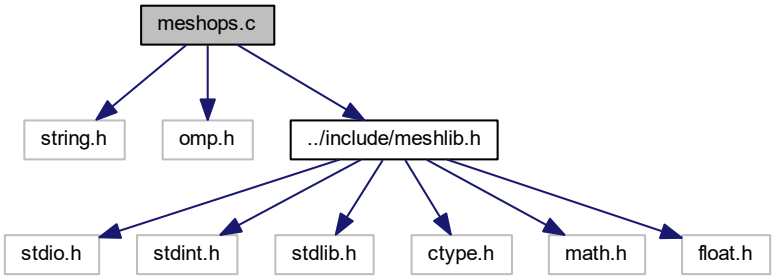
Here is the caller graph for this function:

6.13 meshops.c File Reference

This file contains functions pertaining to mesh combinatorial operations.

```
#include <string.h>
#include <omp.h>
#include "../include/meshlib.h"
```

Include dependency graph for meshops.c:



Functions

- MESH mesh_clone_mesh (MESH m, uint16_t flags)

  Clones a given mesh into another mesh.
- MESH mesh_combine_mesh (MESH m1, MESH m2)

  Combines a given mesh with another given mesh.
- int mesh_alloc_mesh_props (MESH m, INTDATA nv, INTDATA nf, INTDATA ne, uint16_t flags)

  Allocates memory for various properties of a given mesh.
- int mesh_free_mesh_props (MESH m, uint16_t flags)

  Frees memory for various properties of a given mesh.
- int mesh_alloc_face_vertices (MESH_FACE mf, INTDATA nv)

  Allocates memory for vertices of a given mesh face.
- int mesh_free_face_vertices (MESH_FACE mf)

  Frees memory for vertices of a given mesh face.

6.13.1 Detailed Description

This file contains functions pertaining to mesh combinatorial operations.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.13.2 Function Documentation

6.13.2.1 mesh_alloc_face_vertices()

```
int mesh_alloc_face_vertices (
            MESH_FACE mf,
            INTDATA nv )
```

Allocates memory for vertices of a given mesh face.

Parameters

in	*mf*	Input mesh face
in	*nv*	New number of vertices

Returns

Error code

Here is the call graph for this function:

6.13.2.2 mesh_alloc_mesh_props()

```
int mesh_alloc_mesh_props (
            MESH m,
            INTDATA nv,
            INTDATA nf,
            INTDATA ne,
            uint16_t flags )
```

Allocates memory for various properties of a given mesh.

Parameters

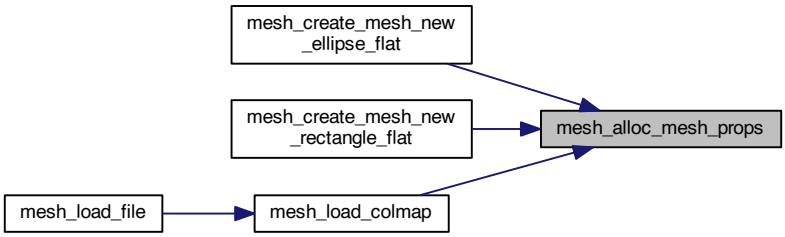
in	*m*	Input mesh
in	*nv*	New number of vertices
in	*nf*	New number of faces
in	*ne*	New number of edges
in	*flags*	Flags to allocate which properties (MESH_PROPS_VERTICES/MESH_PROPS_↩ VNORMALS/MESH_PROPS_VCOLORS/MESH_PROPS_VFACES/MESH_PROPS_↩ VSCALARS/MESH_PROPS_V_ALL_PROPS/MESH_PROPS_FACES/MESH_PROPS_↩ FNORMALS/MESH_PROPS_FCOLORS/MESH_PROPS_FAREAS/MESH_PROPS_↩ FSCALARS/MESH_PROPS_F_ALL_PROPS/MESH_PROPS_ALL_PROPS)

Returns

Error code

Here is the call graph for this function:

Here is the caller graph for this function:



6.13.2.3 mesh_clone_mesh()

```
MESH mesh_clone_mesh (
            MESH m,
            uint16_t flags )
```

Clones a given mesh into another mesh.

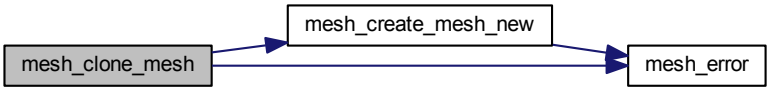
Parameters

in	*m*	Input mesh to clone
in	*flags*	Flags to copy which properties (MESH_PROPS_VERTICES/MESH_PROPS_VNORMALS/↩ MESH_PROPS_VCOLORS/MESH_PROPS_VFACES/MESH_PROPS_VSCALARS/MESH_↩ PROPS_V_ALL_PROPS/MESH_PROPS_FACES/MESH_PROPS_FNORMALS/MESH_↩ PROPS_FCOLORS/MESH_PROPS_FAREAS/MESH_PROPS_FSCALARS/MESH_PROPS↩ _F_ALL_PROPS/MESH_PROPS_ALL_PROPS)

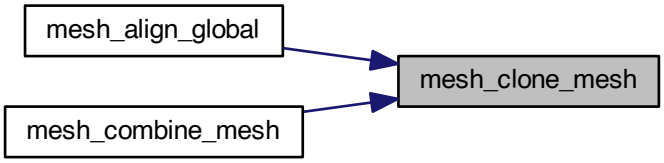
Returns

Output cloned mesh

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.2.4 mesh_combine_mesh()

```
MESH mesh_combine_mesh (
            MESH m1,
            MESH m2 )
```

Combines a given mesh with another given mesh.

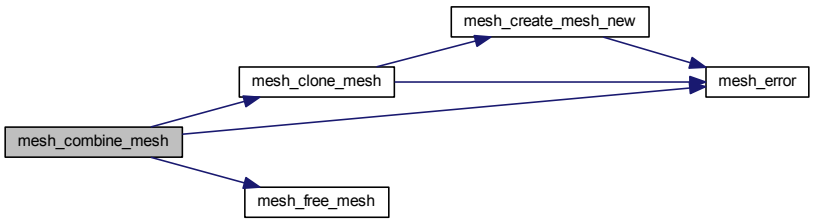
Parameters

in	*m1*	Input mesh to combine with
in	*m2*	Input mesh to combine

Returns

Output combined mesh

Here is the call graph for this function:



6.13.2.5 mesh_free_face_vertices()

```
int mesh_free_face_vertices (
            MESH_FACE mf )
```

Frees memory for vertices of a given mesh face.

Parameters

in	*mf*	Input mesh face

Returns

Error code

6.13.2.6 mesh_free_mesh_props()

```
int mesh_free_mesh_props (
            MESH m,
            uint16_t flags )
```

Frees memory for various properties of a given mesh.

Parameters

in	*m*	Input mesh
in	*flags*	Flags to free which properties (MESH_PROPS_VERTICES/MESH_PROPS_VNORMALS/↩ MESH_PROPS_VCOLORS/MESH_PROPS_VFACES/MESH_PROPS_VSCALARS/MESH_↩ PROPS_V_ALL_PROPS/MESH_PROPS_FACES/MESH_PROPS_FNORMALS/MESH_↩ PROPS_FCOLORS/MESH_PROPS_FAREAS/MESH_PROPS_FSCALARS/MESH_PROPS↩ _F_ALL_PROPS/MESH_PROPS_ALL_PROPS)

Returns

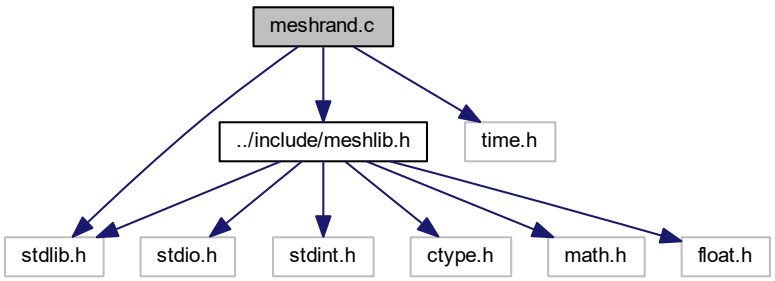
Error code

6.14 meshrand.c File Reference

This file contains functions pertaining to different mesh random perturbations.

```
#include "../include/meshlib.h"
#include <stdlib.h>
#include <time.h>
```

Include dependency graph for meshrand.c:



Functions

- void mesh_set_seed (int seed)
- FLOATDATA __mesh_rand (void)
- FLOATDATA __mesh_randn (void)
- FLOATDATA __mesh_randexp (void)
- FLOATDATA __mesh_randfun (FLOATDATA(∗fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)
- int mesh_add_noise_uniform (MESH m, FLOATDATA sigma)

  Adds uniform random noise to a mesh.
- int mesh_add_noise_gaussian (MESH m, FLOATDATA sigma)

  Adds Gaussian random noise to a mesh.
- int mesh_add_noise_exp (MESH m, FLOATDATA sigma)

  Adds exponential random noise to a mesh.
- int mesh_add_noise_func (MESH m, FLOATDATA sigma, FLOATDATA(∗fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)

  Adds random noise given by density function to a mesh.
- int mesh_add_noise_uniform_normal (MESH m, FLOATDATA sigma)

  Adds uniform random noise along normals to a mesh.
- int mesh_add_noise_gaussian_normal (MESH m, FLOATDATA sigma)

  Adds Gaussian random noise along normals to a mesh.
- int mesh_add_noise_exp_normal (MESH m, FLOATDATA sigma)

  Adds exponential random noise along normals to a mesh.
- int mesh_add_noise_func_normal (MESH m, FLOATDATA sigma, FLOATDATA(∗fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)

Adds random noise given by density function along normals to a mesh.

- int mesh_add_noise_uniform_tangent (MESH m, FLOATDATA sigma)

  Adds uniform random noise along tangent planes to a mesh.
- int mesh_add_noise_gaussian_tangent (MESH m, FLOATDATA sigma)

  Adds Gaussian random noise along tangent planes to a mesh.
- int mesh_add_noise_exp_tangent (MESH m, FLOATDATA sigma)

  Adds exponential random noise along tangent planes to a mesh.
- int mesh_add_noise_func_tangent (MESH m, FLOATDATA sigma, FLOATDATA(*fun)(FLOATDATA), FLOATDATA xmin, FLOATDATA xmax)

  Adds random noise given by density function along tangent planes to a mesh.

Variables

- unsigned int MESH_RAND_SEED = 0
- int MESH_SET_RAND_SEED = 0

6.14.1 Detailed Description

This file contains functions pertaining to different mesh random perturbations.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

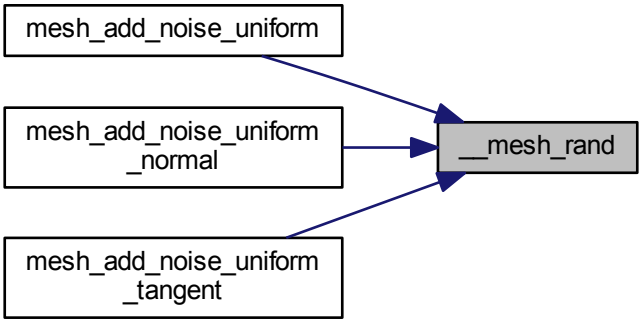
6.14.2 Function Documentation

6.14.2.1 __mesh_rand()

```
FLOATDATA __mesh_rand (
            void  )
```

Here is the call graph for this function:

Here is the caller graph for this function:

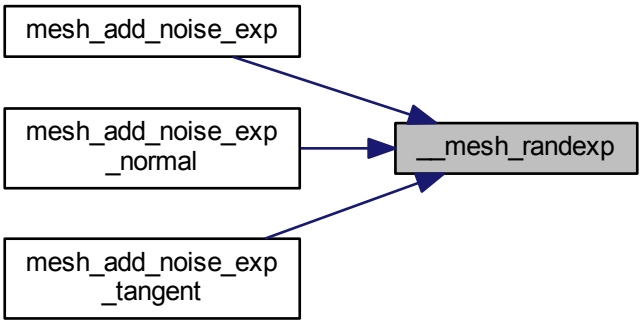


6.14.2.2 __mesh_randexp()

```
FLOATDATA __mesh_randexp (
            void  )
```

Here is the call graph for this function:

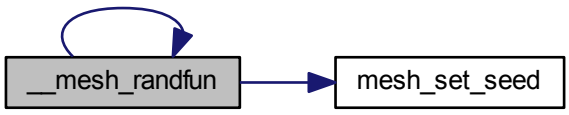
Here is the caller graph for this function:



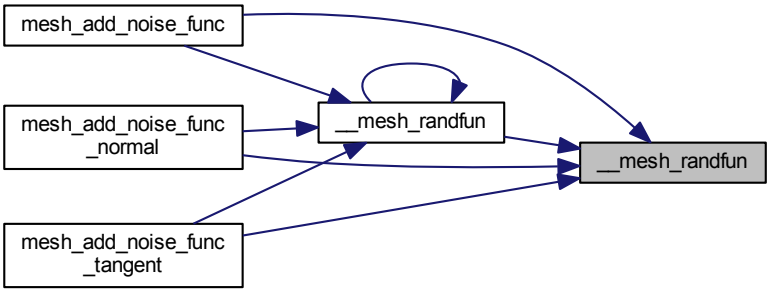
6.14.2.3 __mesh_randfun()

```
FLOATDATA __mesh_randfun (
              FLOATDATA(*)(FLOATDATA) fun,
              FLOATDATA xmin,
              FLOATDATA xmax )
```

Here is the call graph for this function:



Here is the caller graph for this function:

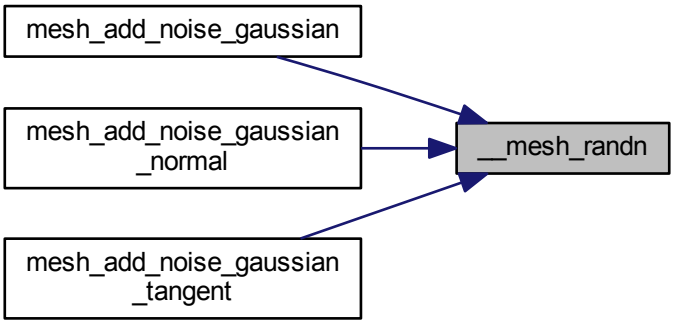


6.14.2.4 __mesh_randn()

```
FLOATDATA __mesh_randn (
              void  )
```

Here is the call graph for this function:

Here is the caller graph for this function:



6.14.2.5 mesh_add_noise_exp()

```
int mesh_add_noise_exp (
            MESH m,
            FLOATDATA sigma )
```

Adds exponential random noise to a mesh.

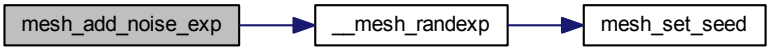
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.6 mesh_add_noise_exp_normal()

```
int mesh_add_noise_exp_normal (
            MESH m,
            FLOATDATA sigma )
```

Adds exponential random noise along normals to a mesh.

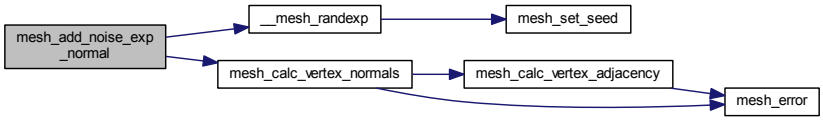
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.7 mesh_add_noise_exp_tangent()

```
int mesh_add_noise_exp_tangent (
            MESH m,
            FLOATDATA sigma )
```

Adds exponential random noise along tangent planes to a mesh.

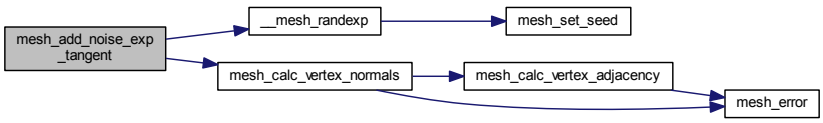
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.8 mesh_add_noise_func()

```
int mesh_add_noise_func (
            MESH m,
```

```
            FLOATDATA sigma,
            FLOATDATA(*)(FLOATDATA) fun,
            FLOATDATA xmin,
            FLOATDATA xmax )
```

Adds random noise given by density function to a mesh.

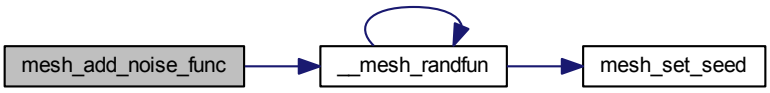
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation
in	*fun*	Density function
in	*xmin*	Lower limit
in	*xmax*	Upper limit

Returns

Error code

Here is the call graph for this function:



6.14.2.9 mesh_add_noise_func_normal()

```
int mesh_add_noise_func_normal (
            MESH m,
            FLOATDATA sigma,
            FLOATDATA(*)(FLOATDATA) fun,
            FLOATDATA xmin,
            FLOATDATA xmax )
```

Adds random noise given by density function along normals to a mesh.

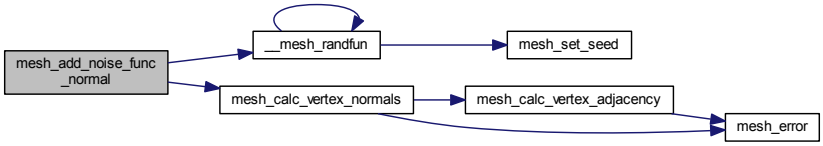
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation
in	*fun*	Density function
in	*xmin*	Lower limit
in	*xmax*	Upper limit

Returns

Error code

Here is the call graph for this function:



6.14.2.10 mesh_add_noise_func_tangent()

```
int mesh_add_noise_func_tangent (
            MESH m,
            FLOATDATA sigma,
            FLOATDATA(*)(FLOATDATA) fun,
            FLOATDATA xmin,
            FLOATDATA xmax )
```

Adds random noise given by density function along tangent planes to a mesh.

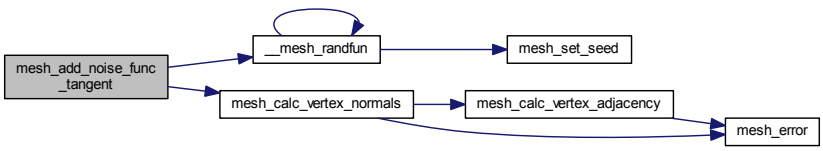
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation
in	*fun*	Density function
in	*xmin*	Lower limit
in	*xmax*	Upper limit

Returns

Error code

Here is the call graph for this function:



6.14.2.11 mesh_add_noise_gaussian()

```
int mesh_add_noise_gaussian (
            MESH m,
            FLOATDATA sigma )
```

Adds Gaussian random noise to a mesh.

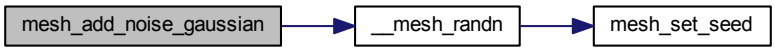
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.12 mesh_add_noise_gaussian_normal()

```
int mesh_add_noise_gaussian_normal (
            MESH m,
            FLOATDATA sigma )
```

Adds Gaussian random noise along normals to a mesh.

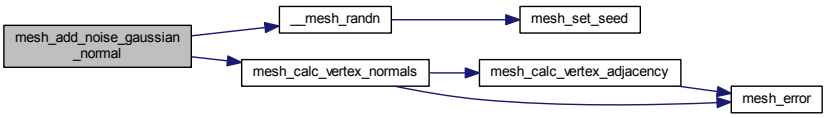
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.13 mesh_add_noise_gaussian_tangent()

```
int mesh_add_noise_gaussian_tangent (
            MESH m,
            FLOATDATA sigma )
```

Adds Gaussian random noise along tangent planes to a mesh.

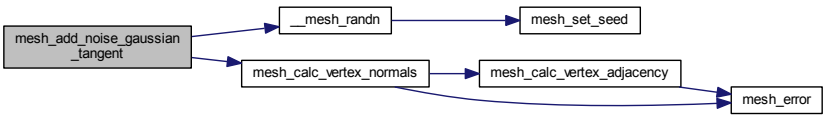
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.14 mesh_add_noise_uniform()

```
int mesh_add_noise_uniform (
            MESH m,
            FLOATDATA sigma )
```

Adds uniform random noise to a mesh.

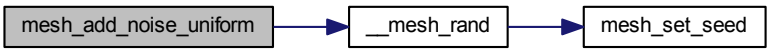
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.15 mesh_add_noise_uniform_normal()

```
int mesh_add_noise_uniform_normal (
            MESH m,
            FLOATDATA sigma )
```

Adds uniform random noise along normals to a mesh.

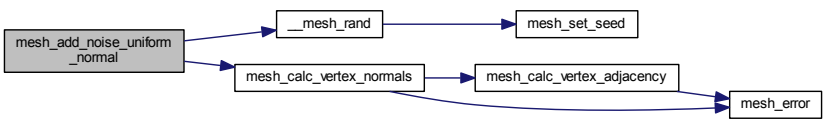
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

Here is the call graph for this function:



6.14.2.16 mesh_add_noise_uniform_tangent()

```
int mesh_add_noise_uniform_tangent (
            MESH m,
            FLOATDATA sigma )
```

Adds uniform random noise along tangent planes to a mesh.

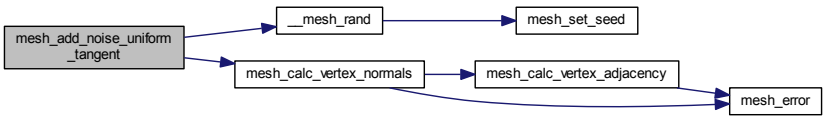
Parameters

in	*m*	Input mesh
in	*sigma*	Standard deviation

Returns

Error code

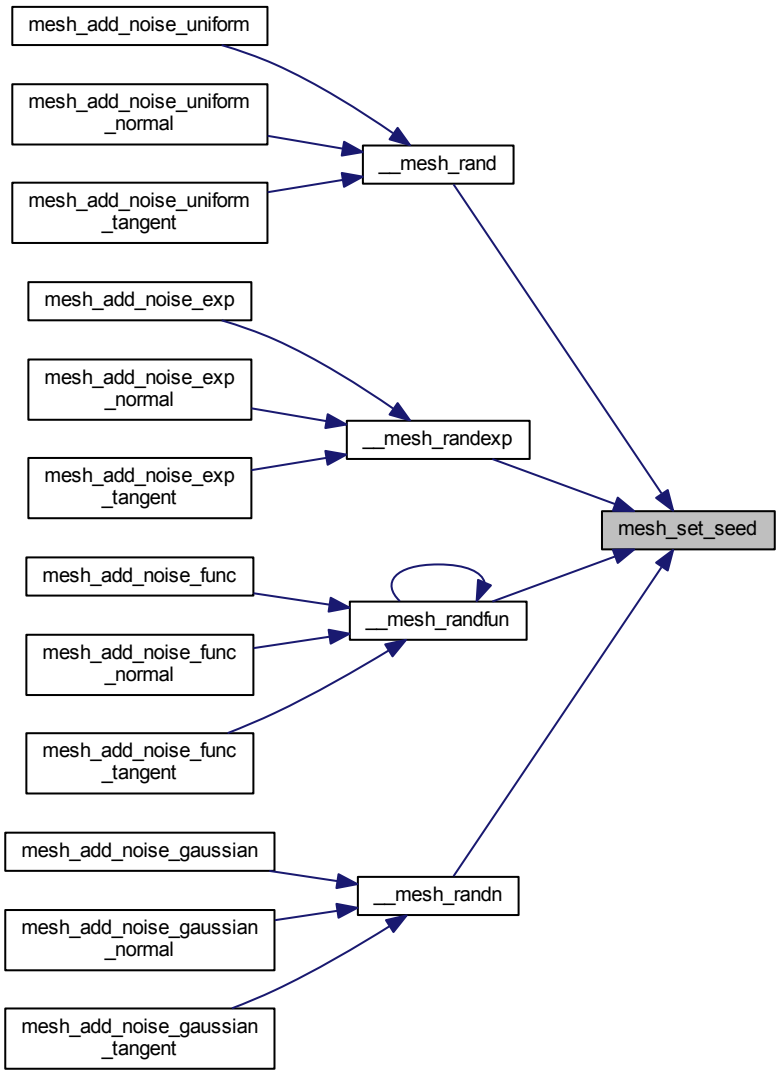
Here is the call graph for this function:



6.14.2.17 mesh_set_seed()

```
void mesh_set_seed (
            int seed )
```

Here is the caller graph for this function:



6.14.3 Variable Documentation

6.14.3.1 MESH_RAND_SEED

```
unsigned int MESH_RAND_SEED = 0
```

6.14.3.2 MESH_SET_RAND_SEED

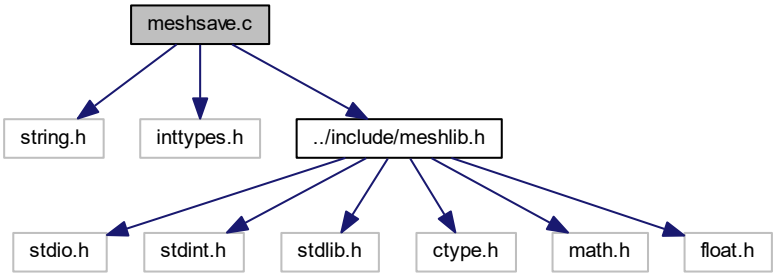
```
int MESH_SET_RAND_SEED = 0
```

6.15 meshsave.c File Reference

This file contains functions pertaining to saving different mesh file types.

```
#include <string.h>
#include <inttypes.h>
#include "../include/meshlib.h"
```

Include dependency graph for meshsave.c:



Macros

- #define _CRT_SECURE_NO_WARNINGS

Functions

- int mesh_save_file (MESH m, const char *fname)
  Saves a mesh to an OFF/PLY/ASC/XYZ/BIN/OBJ file.
- int mesh_save_off (MESH m, const char *fname)
  Saves a mesh to an OFF file.
- int mesh_save_xyz (MESH m, const char *fname)
  Saves a mesh to an XYZ file.
- int mesh_save_ply (MESH m, const char *fname)
  Saves a mesh to a PLY file.
- int mesh_save_bin (MESH m, const char *fname)
  Saves a mesh to a BINv1 file.
- int mesh_save_obj (MESH m, const char *fname)
  Saves a mesh to an OBJ file.

6.15.1 Detailed Description

This file contains functions pertaining to saving different mesh file types.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.15.2 Macro Definition Documentation

6.15.2.1 _CRT_SECURE_NO_WARNINGS

```
#define _CRT_SECURE_NO_WARNINGS
```

6.15.3 Function Documentation

6.15.3.1 mesh_save_bin()

```
int mesh_save_bin (
            MESH m,
            const char * fname )
```

Saves a mesh to a BINv1 file.

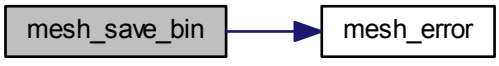
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.15.3.2 mesh_save_file()

```
int mesh_save_file (
            MESH m,
            const char * fname )
```

Saves a mesh to an OFF/PLY/ASC/XYZ/BIN/OBJ file.

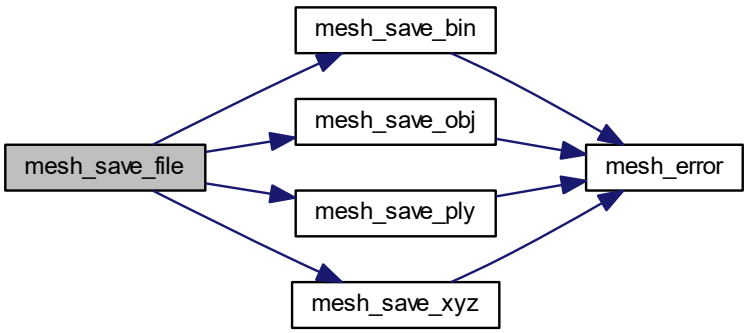
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



6.15.3.3 mesh_save_obj()

```
int mesh_save_obj (
            MESH m,
            const char * fname )
```

Saves a mesh to an OBJ file.

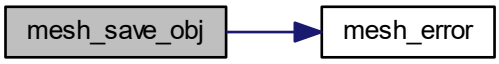
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.15.3.4 mesh_save_off()

```
int mesh_save_off (
            MESH m,
            const char * fname )
```

Saves a mesh to an OFF file.

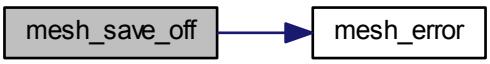
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



6.15.3.5 mesh_save_ply()

```
int mesh_save_ply (
            MESH m,
            const char * fname )
```

Saves a mesh to a PLY file.

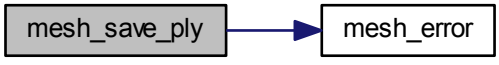
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.15.3.6 mesh_save_xyz()

```
int mesh_save_xyz (
            MESH m,
            const char * fname )
```

Saves a mesh to an XYZ file.

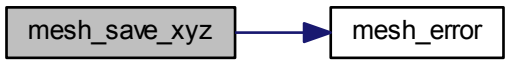
Parameters

in	*m*	Input mesh
in	*fname*	Output filename

Returns

Error code

Here is the call graph for this function:



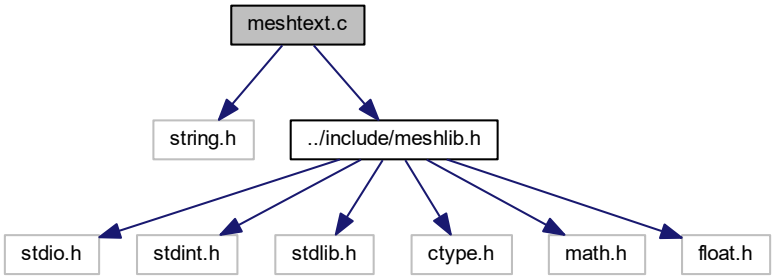
Here is the caller graph for this function:

6.16 meshtext.c File Reference

This file contains functions pertaining to different text routines.

```
#include <string.h>
#include "../include/meshlib.h"
```

Include dependency graph for meshtext.c:



Functions

- int mesh_isnumeric (FILEPOINTER fp)
  Checks if numeric or not.
- int mesh_go_next_word (FILEPOINTER fp)
  Points to the next word.
- int mesh_count_words_in_line (FILEPOINTER fp, int *count)
  Counts number of words in the current line.
- int mesh_read_word (FILEPOINTER fp, char *c_word, int sz)
  Reads current word and moves to the next word.
- int mesh_read_word_skip_comment (FILEPOINTER fp, char *c_word, int sz)
  Reads current word skipping comment with # and moves to the next word.
- int mesh_read_word_only (FILEPOINTER fp, char *c_word, int sz)
  Reads current word without moving to the next word.
- int mesh_read_word_only_skip_comment (FILEPOINTER fp, char *c_word, int sz)
  Reads current word skipping comment with # without moving to the next word.
- int mesh_skip_line (FILEPOINTER fp)
  Skips to next line.

6.16.1 Detailed Description

This file contains functions pertaining to different text routines.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.16.2 Function Documentation

6.16.2.1 mesh_count_words_in_line()

```
int mesh_count_words_in_line (
            FILEPOINTER fp,
            int * count )
```

Counts number of words in the current line.

Parameters

in	*fp*	Pointer to input file
out	*count*	Count

Returns

Status 0 - Normal/ 1- EOF

6.16.2.2 mesh_go_next_word()

```
int mesh_go_next_word (
            FILEPOINTER fp )
```

Points to the next word.

Parameters

in	*fp*	Pointer to input file

Returns

Status 0 - Normal/ 1- EOF

6.16.2.3 mesh_isnumeric()

```
int mesh_isnumeric (
            FILEPOINTER fp )
```

Checks if numeric or not.

Parameters

in	*fp*	Pointer to input file

Returns

1 for numeric/ else - for non-numeric

6.16.2.4 mesh_read_word()

```
int mesh_read_word (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word and moves to the next word.

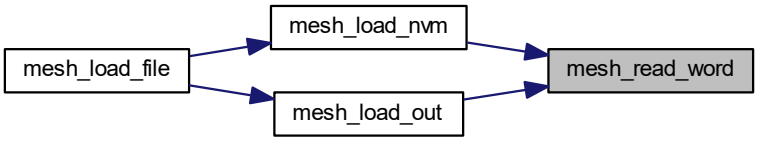
Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1 - EOF / 2 - Overflow

Here is the caller graph for this function:



6.16.2.5 mesh_read_word_only()

```
int mesh_read_word_only (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word without moving to the next word.

Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF / 2 - Overflow

6.16.2.6 mesh_read_word_only_skip_comment()

```
int mesh_read_word_only_skip_comment (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word skipping comment with # without moving to the next word.

Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF / 2 - Overflow / 3 - Commented

6.16.2.7 mesh_read_word_skip_comment()

```
int mesh_read_word_skip_comment (
            FILEPOINTER fp,
            char * c_word,
            int sz )
```

Reads current word skipping comment with # and moves to the next word.

Parameters

in	*fp*	Pointer to input file
out	*c_word*	Variable to store the word
in	*sz*	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF / 2 - Overflow / 3 - Commented

6.16.2.8 mesh_skip_line()

```
int mesh_skip_line (
            FILEPOINTER fp )
```

Skips to next line.

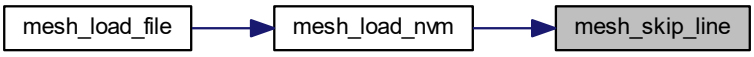
Parameters

in	*fp*	Pointer to input file

Returns

Status 0 - Normal/ 1- EOF

Here is the caller graph for this function:

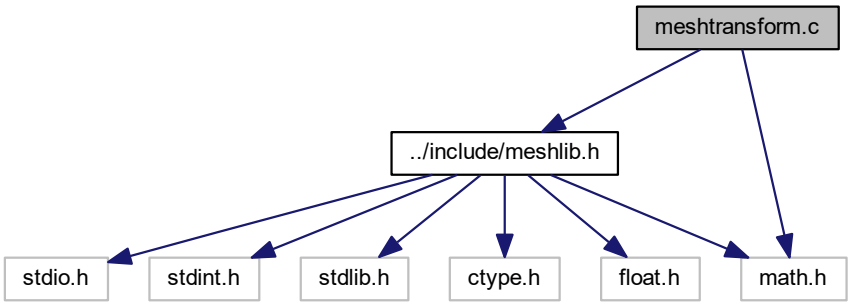


6.17 meshtransform.c File Reference

This file contains functions pertaining to different mesh transformations.

```
#include "../include/meshlib.h"
#include <math.h>
```

Include dependency graph for meshtransform.c:



Functions

- MESH_ROTATION mesh_rotation_create ()

  Creates a new rotation.
- void mesh_rotation_free (MESH_ROTATION r)

  Frees a given rotation.
- MESH_AFFINE mesh_affine_create ()

  Creates a new affine transformation.
- void mesh_affine_free (MESH_AFFINE tx)

  Frees a given affine transformation.
- MESH_ROTATION mesh_rotation_set_matrix (FLOATDATA *mat, MESH_ROTATION r)

  Sets rotation from a matrix.
- MESH_AFFINE mesh_affine_set_matrix (FLOATDATA *mat, MESH_AFFINE r)

  Sets affine transformation from a matrix.
- MESH_ROTATION mesh_rotation_set_angleaxis (FLOATDATA ang, MESH_NORMAL axis, MESH_ROTATION r)

  Sets rotation from angle axis.
- int mesh_translate (MESH m, FLOATDATA x, FLOATDATA y, FLOATDATA z)

  Translates a mesh by x, y and z amounts.
- int mesh_translate_vector (MESH m, MESH_VECTOR3 v)

  Translates a mesh by a given 3-d vector.
- int mesh_scale (MESH m, FLOATDATA sx, FLOATDATA sy, FLOATDATA sz)

  Scales a mesh by x, y and z amounts.
- MESH_VERTEX mesh_vertex_rotate (MESH_VERTEX v, MESH_ROTATION r)

  Rotates a vertex by a given rotation.
- int mesh_rotate (MESH m, MESH_ROTATION r)

  Rotates a mesh by a given rotation.
- int mesh_transform (MESH m, MESH_AFFINE tx)

  Transforms a mesh by a given affine transformation.
- MESH_AFFINE mesh_transform_set_rotation_translation (MESH_ROTATION r, MESH_VECTOR3 t, MESH_AFFINE tx)

  Sets an affine transformation with rotation and translation.
- int mesh_align_global (MESH m1, MESH m2, int flags, MESH_AFFINE tx)

  Sets an affine transformation with rotation and translation.

6.17.1 Detailed Description

This file contains functions pertaining to different mesh transformations.

Author

Sk. Mohammadul Haque

Version

1.8.0.0

Copyright

Copyright (c) 2013-2021 Sk. Mohammadul Haque.

6.17.2 Function Documentation

6.17.2.1 mesh_affine_create()

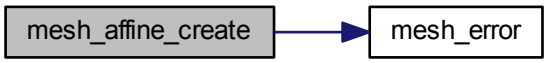
```
MESH_AFFINE mesh_affine_create ( )
```

Creates a new affine transformation.

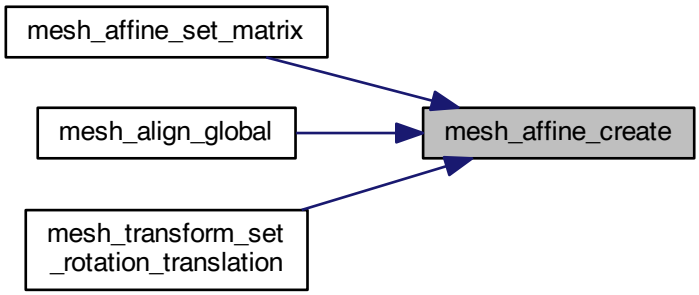
Returns

Output affine transformation

Here is the call graph for this function:



Here is the caller graph for this function:



6.17.2.2 mesh_affine_free()

```
void mesh_affine_free (
            MESH_AFFINE tx )
```

Frees a given affine transformation.

Parameters

tx	Input affine transformation

Returns

NULL

6.17.2.3 mesh_affine_set_matrix()

```
MESH_AFFINE mesh_affine_set_matrix (
            FLOATDATA * mat,
            MESH_AFFINE r )
```

Sets affine transformation from a matrix.

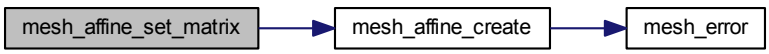
Parameters

in	*mat*	Input matrix
out	*r*	Input affine transformation

Returns

Output affine transformation

Here is the call graph for this function:



6.17.2.4 mesh_align_global()

```
int mesh_align_global (
            MESH m1,
            MESH m2,
            int flags,
            MESH_AFFINE tx )
```

Sets an affine transformation with rotation and translation.

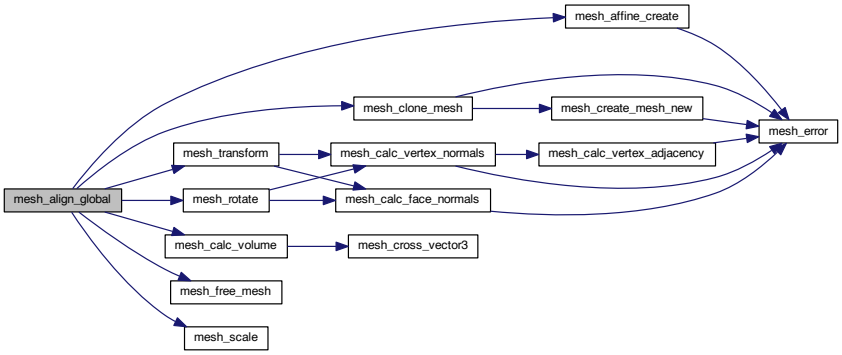
Parameters

in	*m1*	Input mesh
out	*m2*	Input mesh to align
in	*flags*	(MESH_ALIGN_GLOBAL_POSITION/MESH_ALIGN_GLOBAL_ORIENTATION/MESH_↩ ALIGN_GLOBAL_SCALE/MESH_ALIGN_GLOBAL_ALL/MESH_ALIGN_GLOBAL_DO_↩ TRANSFORM)
out	*tx*	Output affine transformation, if not null

Returns

Error code

Here is the call graph for this function:



6.17.2.5 mesh_rotate()

```
int mesh_rotate (
            MESH m,
            MESH_ROTATION r )
```

Rotates a mesh by a given rotation.

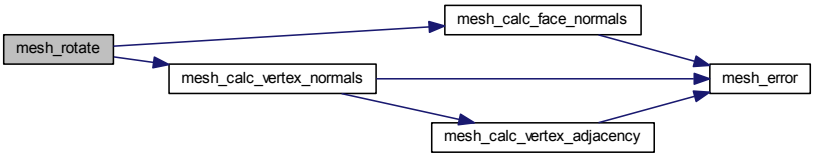
Parameters

in	*m*	Input vertex
in	*r*	Input rotation

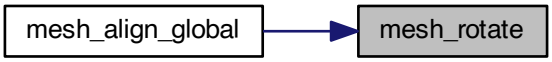
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



6.17.2.6 mesh_rotation_create()

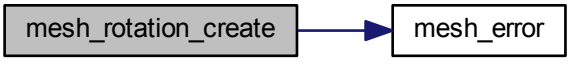
```
MESH_ROTATION mesh_rotation_create ( )
```

Creates a new rotation.

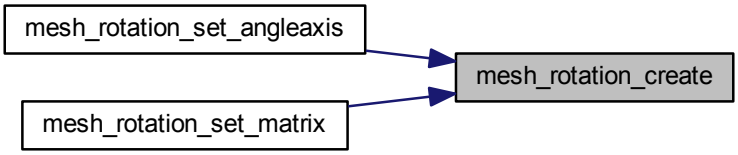
Returns

Output rotation

Here is the call graph for this function:



Here is the caller graph for this function:



6.17.2.7 mesh_rotation_free()

```
void mesh_rotation_free (
            MESH_ROTATION r )
```

Frees a given rotation.

Parameters

r	Input rotation

Returns

NULL

6.17.2.8 mesh_rotation_set_angleaxis()

```
MESH_ROTATION mesh_rotation_set_angleaxis (
            FLOATDATA ang,
            MESH_NORMAL axis,
            MESH_ROTATION r )
```

Sets rotation from angle axis.

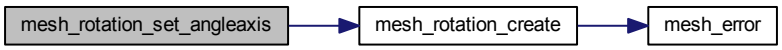
Parameters

in	*ang*	Input angle of rotation
out	*axis*	Input axis of rotation
out	*r*	Input rotation

Returns

Output rotation

Here is the call graph for this function:



6.17.2.9 mesh_rotation_set_matrix()

```
MESH_ROTATION mesh_rotation_set_matrix (
            FLOATDATA * mat,
            MESH_ROTATION r )
```

Sets rotation from a matrix.

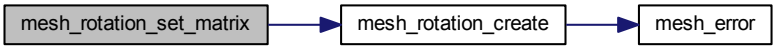
Parameters

in	*mat*	Input matrix
out	*r*	Input rotation

Returns

Output rotation

Here is the call graph for this function:



6.17.2.10 mesh_scale()

```
int mesh_scale (
            MESH m,
            FLOATDATA sx,
            FLOATDATA sy,
            FLOATDATA sz )
```

Scales a mesh by x, y and z amounts.

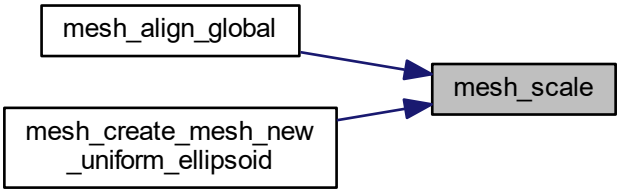
Parameters

in	*m*	Input mesh
in	*sx*	X component
in	*sy*	Y component
in	*sz*	Z component

Returns

Error code

Here is the caller graph for this function:



6.17.2.11 mesh_transform()

```
int mesh_transform (
            MESH m,
            MESH_AFFINE tx )
```

Transforms a mesh by a given affine transformation.

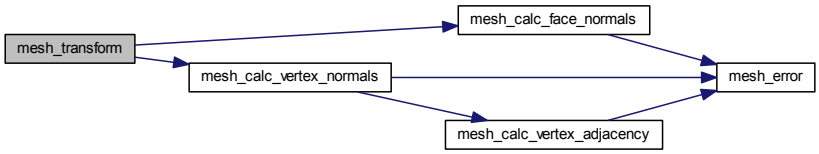
Parameters

in	*m*	Input vertex
in	*tx*	Input affine transformation

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

6.17.2.12 mesh_transform_set_rotation_translation()

```
MESH_AFFINE mesh_transform_set_rotation_translation (
             MESH_ROTATION r,
             MESH_VECTOR3 t,
             MESH_AFFINE tx )
```

Sets an affine transformation with rotation and translation.

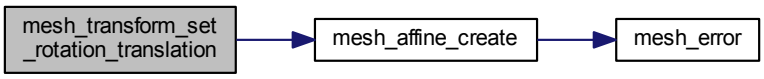
Parameters

in	*r*	Input rotation
out	*t*	Input translation
out	*tx*	Input affine transformation

Returns

Output affine transformation

Here is the call graph for this function:



6.17.2.13 mesh_translate()

```
int mesh_translate (
              MESH m,
              FLOATDATA x,
              FLOATDATA y,
              FLOATDATA z )
```

Translates a mesh by x, y and z amounts.

Parameters

in	*m*	Input mesh
in	*x*	X component
in	*y*	Y component
in	*z*	Z component

Returns

Error code

6.17.2.14 mesh_translate_vector()

```
int mesh_translate_vector (
              MESH m,
              MESH_VECTOR3 v )
```

Translates a mesh by a given 3-d vector.

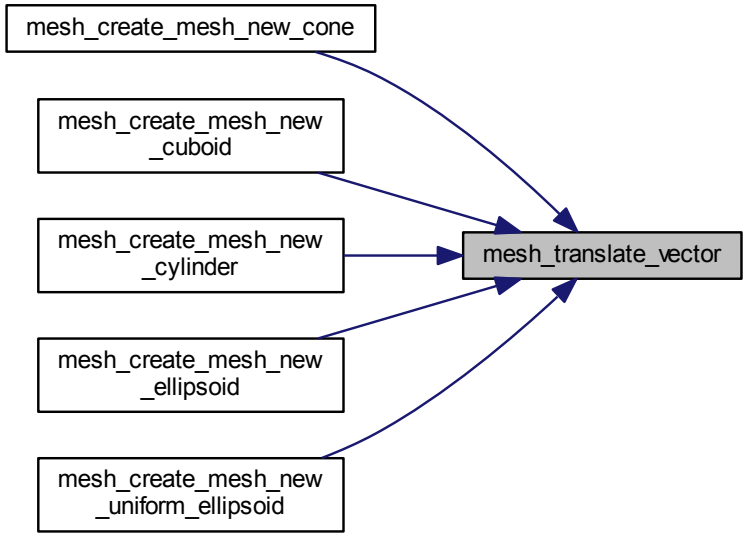
Parameters

in	*m*	Input mesh
in	*v*	Input vector

Returns

Error code

Here is the caller graph for this function:



6.17.2.15 mesh_vertex_rotate()

```
MESH_VERTEX mesh_vertex_rotate (
              MESH_VERTEX v,
              MESH_ROTATION r )
```

Rotates a vertex by a given rotation.

Parameters

in	*v*	Input vertex
in	*r*	Input rotation

Returns

Output vertex

6.18 openmesh.md File Reference

6.19 smoothmesh.md File Reference

Index