MeshLib

1.4.2.0

Generated by Doxygen 1.8.9.1

Fri Sep 9 2016 01:16:08

Contents

Chapter 1

Meshlib

1.1 Introduction

Meshlib is a simple mesh library written in C.

1.2 Build

To build the whole project, Code::blocks is required.

1.3 Contents

Load/Write PLY, OFF, ASC files.

Basic Vertex Manipulations.

Basic Vertex Transformations.

Basic Face Manipulations.

Bilateral Filtering.

Laplacian Filtering.

Mesh Cleaning Algorithms.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

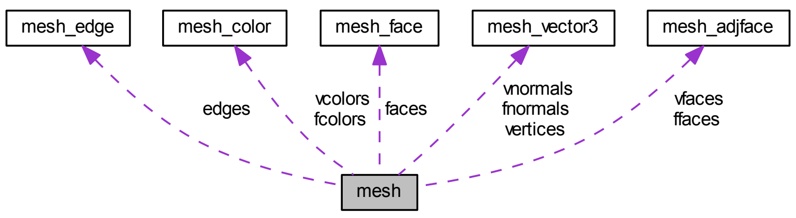
Chapter 4

Data Structure Documentation

4.1 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
- 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
- FLOATDATA ∗ fareas
- uint8_t is_trimesh
- uint8_t dummy

4.1.1 Field Documentation

4.1.1.1 uint8_t dummy

4.1.1.2 MESH_EDGE edges

Pointer to edges

4.1.1.3 MESH_FACE faces

Pointer to faces

4.1.1.4 FLOATDATA∗ fareas

Pointer to face areas

4.1.1.5 MESH_COLOR fcolors

Pointer to face colors

4.1.1.6 MESH_FFACE ffaces

Pointer to face adjacent faces

4.1.1.7 MESH_NORMAL fnormals

Pointer to face normals

4.1.1.8 uint8_t is_edges

Has edges?

4.1.1.9 uint8_t is_faces

Has faces?

4.1.1.10 uint8_t is_fareas

Has face areas?

4.1.1.11 uint8_t is_fcolors

Has face colors?

4.1.1.12 uint8_t is_ffaces

Has face adjacent faces?

4.1.1.13 uint8_t is_fnormals

Has face normals?

4.1.1.14 uint8_t is_loaded

Is loaded?

4.1.1.15 uint8_t is_trimesh

Is trimesh?

4.1.1.16 uint8_t is_vcolors

Has vertex colors?

4.1.1.17 uint8_t is_vertices

Has vertices?

4.1.1.18 uint8_t is_vfaces

Has vertex adjacent faces?

4.1.1.19 uint8_t is_vnormals

Has vertex normals?

4.1.1.20 INTDATA num_edges

Number of edges

4.1.1.21 INTDATA num_faces

Number of faces

4.1.1.22 INTDATA num_vertices

Number of vertices

4.1.1.23 uint8_t origin_type

Origin type

4.1.1.24 MESH_COLOR vcolors

Pointer to vertex colors

4.1.1.25 MESH_VERTEX vertices

Pointer to vertices

4.1.1.26 MESH_VFACE vfaces

Pointer to vertex adjacent faces

4.1.1.27 MESH_NORMAL vnormals

Pointer to vertex normals

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

- meshlib.h

4.2 mesh_adjface Struct Reference

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

Data Fields

- INTDATA num_faces
- INTDATA ∗ faces

4.2.1 Field Documentation

4.2.1.1 INTDATA∗ faces

Pointer to adjacent face indices

4.2.1.2 INTDATA num_faces

Number of adjacent faces

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

- meshlib.h

4.3 mesh_color Struct Reference

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

Data Fields

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

4.3.1 Field Documentation

4.3.1.1 FLOATDATA a

Alpha channel

4.3.1.2 FLOATDATA b

Green channel

4.3.1.3 FLOATDATA g

Blue channel

4.3.1.4 FLOATDATA r

Red channel

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

- meshlib.h

4.4 mesh_edge Struct Reference

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

Data Fields

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

4.4.1 Field Documentation

4.4.1.1 INTDATA faces[2]

Edge faces

4.4.1.2 INTDATA vertices[2]

Edge vertices

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

- meshlib.h

4.5 mesh_face Struct Reference

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

Data Fields

- INTDATA num_vertices
- INTDATA ∗ vertices

4.5.1 Field Documentation

4.5.1.1 INTDATA num_vertices

Number of vertices

4.5.1.2 INTDATA∗ vertices

Pointer to vertex indices

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

- meshlib.h

4.6 mesh_rotation Struct Reference

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

Data Fields

- FLOATDATA data [9]

4.6.1 Field Documentation

4.6.1.1 FLOATDATA data[9]

Matrix data

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

- meshlib.h

4.7 mesh_struct Struct Reference

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

Data Fields

- INTDATA num_items
- INTDATA ∗ items

4.7.1 Field Documentation

4.7.1.1 INTDATA∗ items

Pointer to INTDATA items

4.7.1.2 INTDATA num_items

Number of items

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

- meshlib.h

4.8 mesh_struct2 Struct Reference

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

Data Fields

- INTDATA num_items
- INTDATA2 ∗ items

4.8.1 Field Documentation

4.8.1.1 INTDATA2∗ items

Pointer to INTDATA2 items

4.8.1.2 INTDATA num_items

Number of items

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

- meshlib.h

4.9 mesh_struct3 Struct Reference

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

Data Fields

- INTDATA num_items
- INTDATA3 ∗ items

4.9.1 Field Documentation

4.9.1.1 INTDATA3∗ items

Pointer to INTDATA3 items

4.9.1.2 INTDATA num_items

Number of items

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

- meshlib.h

4.10 mesh_transform Struct Reference

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

Data Fields

- FLOATDATA * data

4.10.1 Field Documentation

4.10.1.1 FLOATDATA* data

Matrix data

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

- meshlib.h

4.11 mesh_vector3 Struct Reference

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

Data Fields

- FLOATDATA x
- FLOATDATA y
- FLOATDATA z

4.11.1 Field Documentation

4.11.1.1 FLOATDATA x

x co-ordinate

4.11.1.2 FLOATDATA y

y co-ordinate

4.11.1.3 FLOATDATA z

z co-ordinate

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

- meshlib.h

Chapter 5

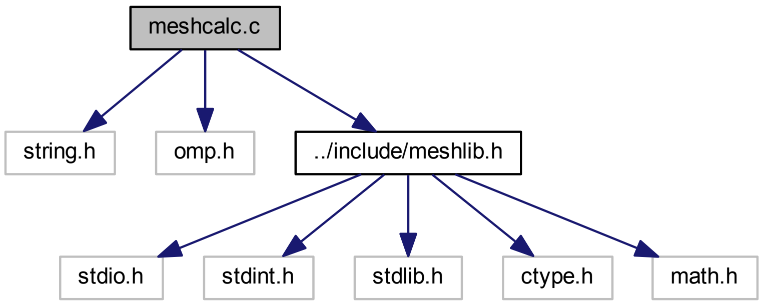
File Documentation

5.1 meshcalc.c File Reference

This file contains functions pertaining to different mesh computations.

```
#include <string.h>
#include <omp.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_NO←RMAL 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.
- FLOATDATA mesh_calc_triangle_area (MESH_VERTEX a, MESH_VERTEX b, MESH_VERTEX c)

  Computes area of a triangle.

5.1.1 Detailed Description

This file contains functions pertaining to different mesh computations.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.1.2 Function Documentation

5.1.2.1 int mesh_calc_edges (MESH *m*)

Computes edges of a given mesh.

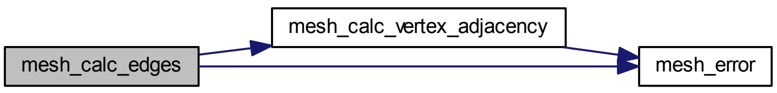
Parameters

in	*m*	Input mesh

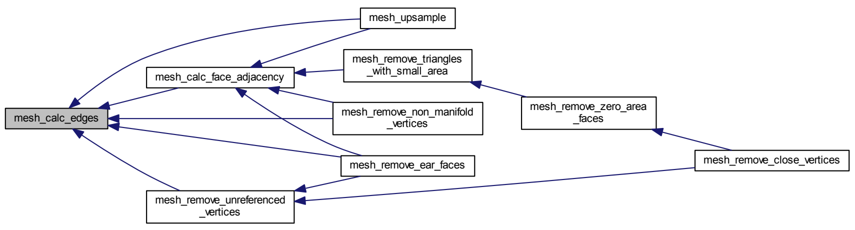
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_upsample
mesh_remove_triangles_with_small_area
mesh_calc_face_adjacency
mesh_remove_zero_area_faces
mesh_calc_edges
mesh_remove_non_manifold_vertices
mesh_remove_ear_faces
mesh_remove_close_vertices
mesh_remove_unreferenced_vertices

5.1.2.2 int mesh_calc_face_adjacency (**MESH** *m*)

Computes face adjacent faces of a given mesh.

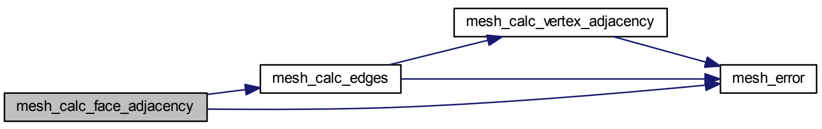
Parameters

in	*m*	Input mesh

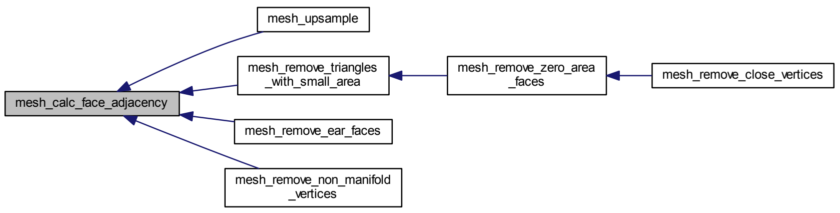
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_upsample
mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices
mesh_calc_face_adjacency
mesh_remove_ear_faces
mesh_remove_non_manifold_vertices

5.1.2.3 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

5.1.2.4 int mesh_calc_face_normals (MESH *m*)

Computes face normals of a given mesh.

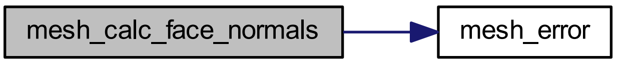
Parameters

in	*m*	Input mesh

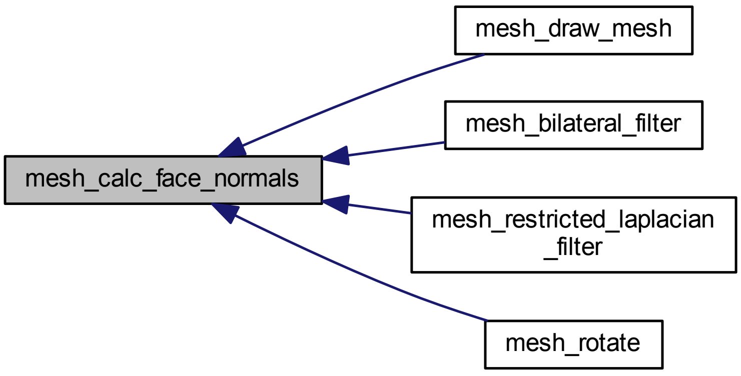
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.5 FLOATDATA mesh_calc_triangle_area (MESH_VERTEX *a*, MESH_VERTEX *b*, MESH_VERTEX *c*)

Computes area of a triangle.

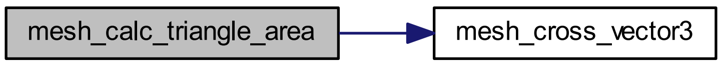
Parameters

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

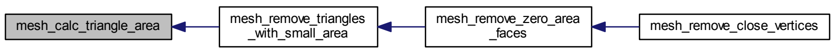
Returns

Area

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_calc_triangle_area
mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices

5.1.2.6 int mesh_calc_vertex_adjacency (MESH *m*)

Computes vertex adjacent faces of a given mesh.

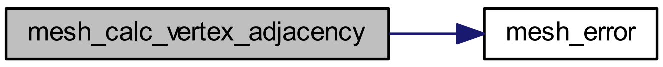
Parameters

`in`	*m*	Input mesh

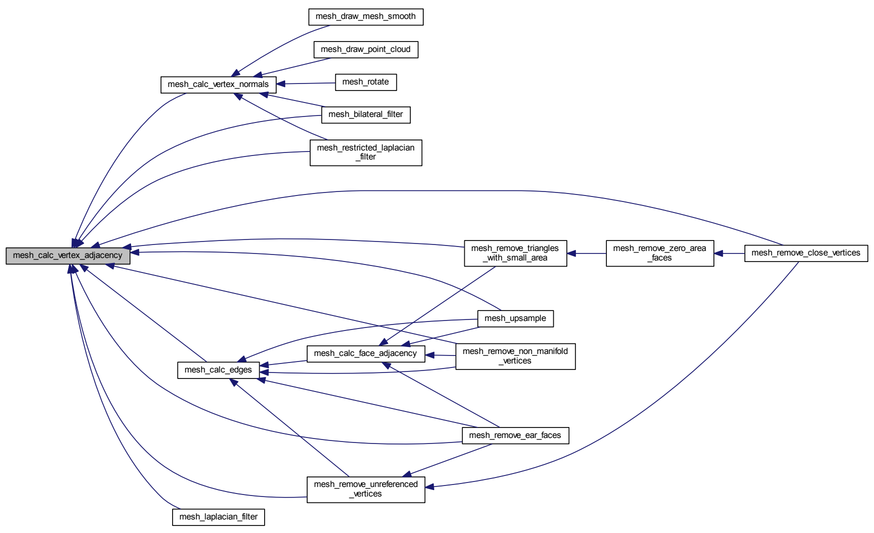
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_draw_mesh_smooth
mesh_draw_point_cloud
mesh_calc_vertex_normals
mesh_rotate
mesh_bilateral_filter
mesh_restricted_laplacian_filter
mesh_calc_vertex_adjacency
mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices
mesh_upsample
mesh_calc_face_adjacency
mesh_remove_non_manifold_vertices
mesh_calc_edges
mesh_remove_ear_faces
mesh_remove_unreferenced_vertices
mesh_laplacian_filter

5.1.2.7 int mesh_calc_vertex_normals (**MESH** *m*)

Computes vertex normals of a given mesh.

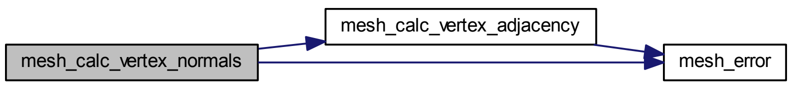
Parameters

in	*m*	Input mesh

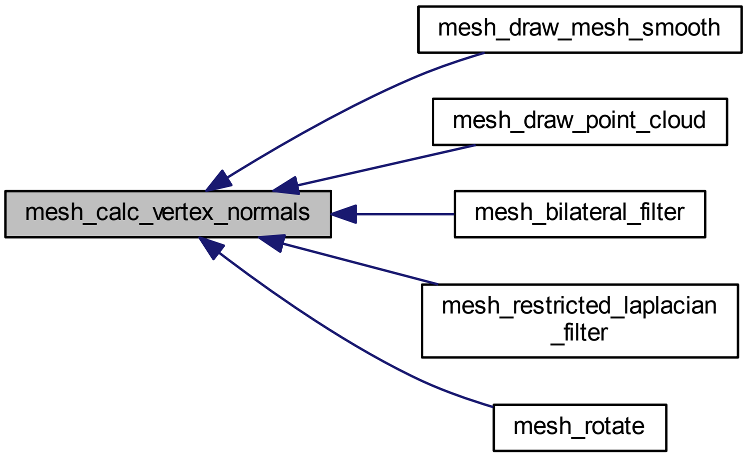
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.8 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

5.1.2.9 void mesh_cross_vector3 (**MESH_VECTOR3** *x*, **MESH_VECTOR3** *y*, **MESH_VECTOR3** *z*)

Computes the cross product of two 3-d vectors.

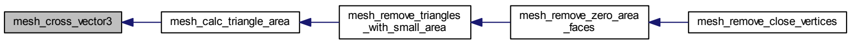
Parameters

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:


mesh_cross_vector3
mesh_calc_triangle_area
mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices

5.1.2.10 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

5.1.2.11 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

5.1.2.12 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

5.1.2.13 int mesh_upsample (MESH *m*, int *iters*)

Upsamples a given mesh.

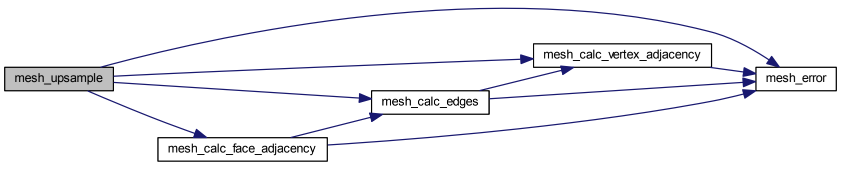
Parameters

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

Returns

Error code

Here is the call graph for this function:

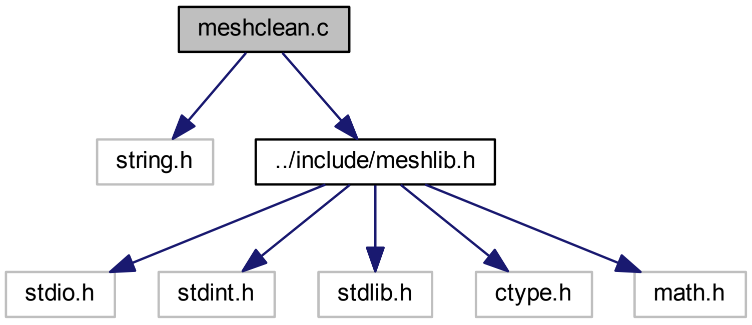


5.2 meshclean.c File Reference

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)

5.2.1 Detailed Description

This file contains functions pertaining to different mesh cleaning algorithms.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.2.2 Function Documentation

5.2.2.1 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

5.2.2.2 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

5.2.2.3 int mesh_remove_close_vertices (MESH *m*, FLOATDATA *r*)

Removes close vertices.

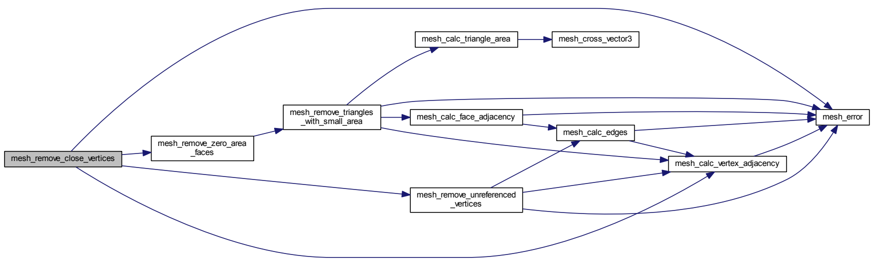
Parameters

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

Returns

Error code

Here is the call graph for this function:


mesh_remove_close_vertices
mesh_remove_zero_area_faces
mesh_remove_triangles_with_small_area
mesh_calc_triangle_area
mesh_cross_vector3
mesh_calc_face_adjacency
mesh_calc_edges
mesh_error
mesh_calc_vertex_adjacency
mesh_remove_unreferenced_vertices

5.2.2.4 int mesh_remove_ear_faces (MESH *m*, int *niters*)

Removes ear faces and connecting vertices.

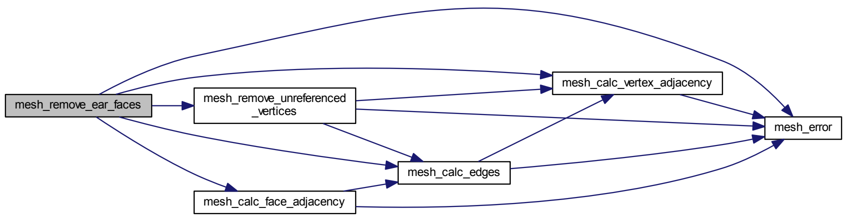
Parameters

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

Returns

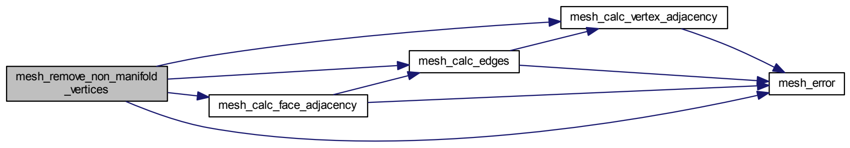
Error code

Here is the call graph for this function:



5.2.2.5 int mesh_remove_non_manifold_vertices (MESH *m*)

Here is the call graph for this function:



5.2.2.6 int mesh_remove_triangles_with_small_area (MESH *m*, FLOATDATA *area*)

Removes triangles with area smaller than a given value.

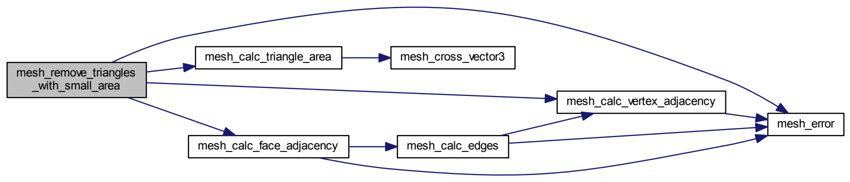
Parameters

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

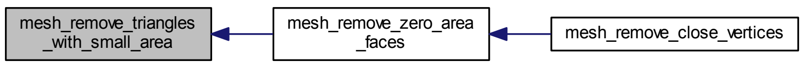
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices

5.2.2.7 int mesh_remove_unreferenced_vertices (MESH *m*)

Removes unreferenced vertices.

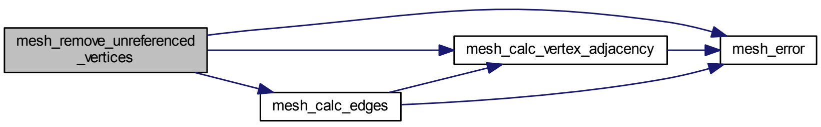
Parameters

in	*m*	Input mesh

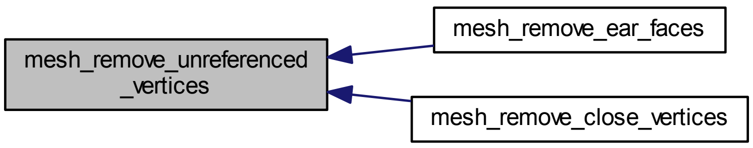
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.2.8 int mesh_remove_zero_area_faces (**MESH** *m*)

Removes triangles with zero area.

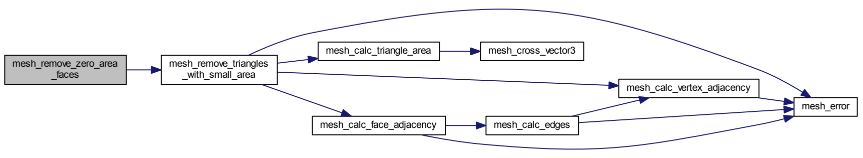
Parameters

in	*m*	Input mesh

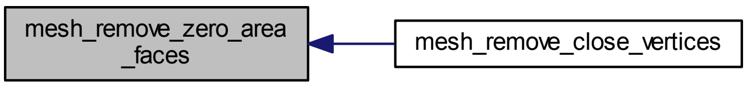
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

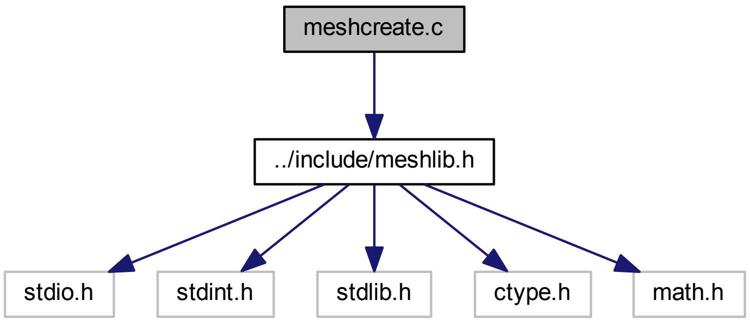


5.3 meshcreate.c File Reference

This file contains functions pertaining to mesh creation and freeing.

```
#include "../include/meshlib.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, INTDA↩ TA 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_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.

5.3.1 Detailed Description

This file contains functions pertaining to mesh creation and freeing.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.3.2 Function Documentation

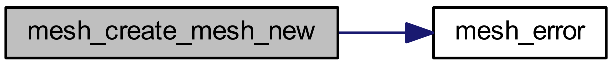
5.3.2.1 MESH mesh_create_mesh_new ( )

Creates a new mesh.

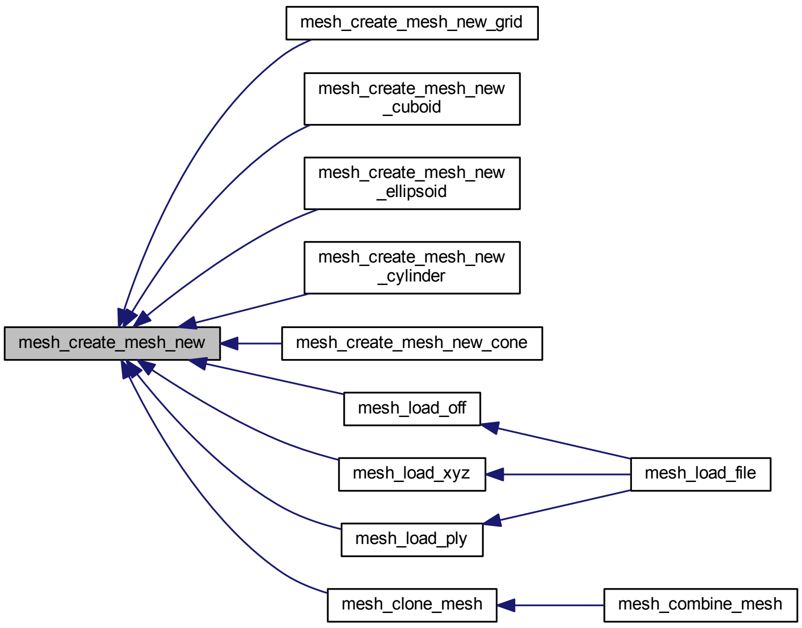
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_create_mesh_new_grid
mesh_create_mesh_new _cuboid
mesh_create_mesh_new _ellipsoid
mesh_create_mesh_new _cylinder
mesh_create_mesh_new
mesh_create_mesh_new_cone
mesh_load_off
mesh_load_xyz
mesh_load_file
mesh_load_ply
mesh_clone_mesh
mesh_combine_mesh

5.3.2.2 MESH mesh_create_mesh_new_cone (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates a cone mesh.

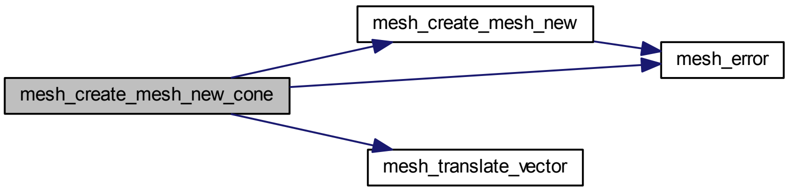
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.3.2.3 MESH mesh_create_mesh_new_cuboid (MESH_VECTOR3 *sz,* MESH_VECTOR3 *pos*)

Creates a cuboid mesh.

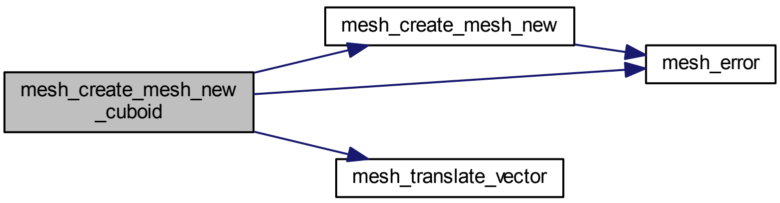
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.3.2.4 MESH mesh_create_mesh_new_cylinder (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates a cylinder mesh.

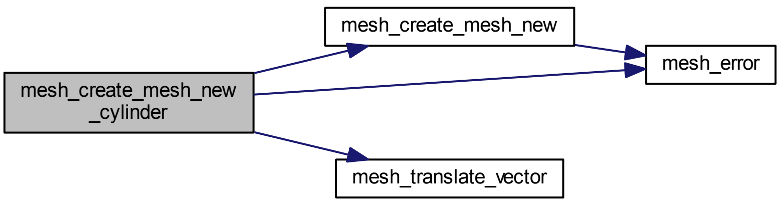
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.3.2.5 MESH mesh_create_mesh_new_ellipsoid (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates an ellipsoid mesh.

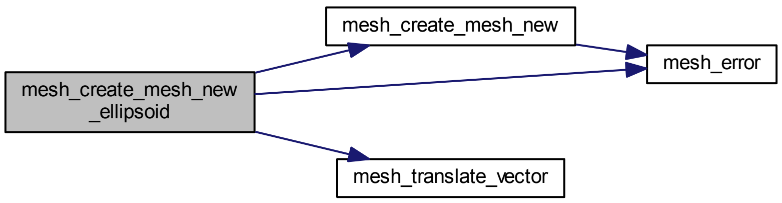
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.3.2.6 MESH mesh_create_mesh_new_grid (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*, INTDATA *m*, INTDATA *n*)

Creates a grid mesh.

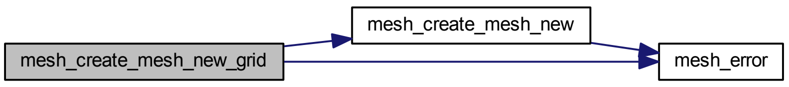
Parameters

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:



5.3.2.7 void mesh_free_mesh (MESH *m*)

Frees a mesh.

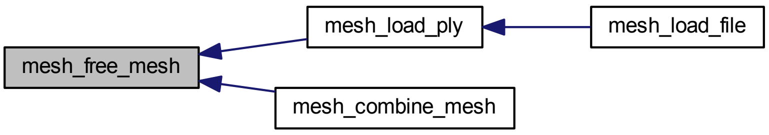
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the caller graph for this function:

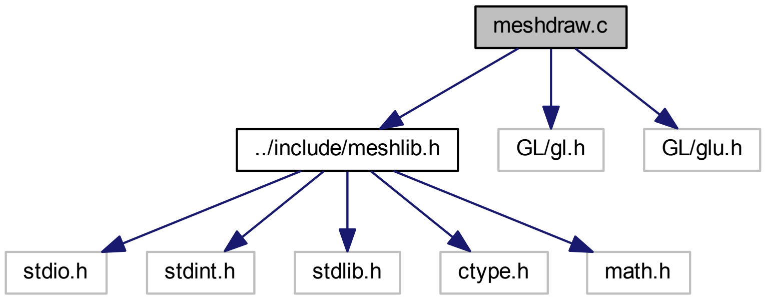


5.4 meshdraw.c File Reference

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.

5.4.1 Detailed Description

This file contains functions pertaining to mesh drawing in OpenGL.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.4.2 Function Documentation

5.4.2.1 void mesh_draw_mesh (MESH *m*)

Draws a given mesh in OpenGL context in flat shading.

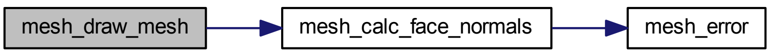
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:



5.4.2.2 void mesh_draw_mesh_smooth (MESH *m*)

Draws a given mesh in OpenGL context in smoothing shading.

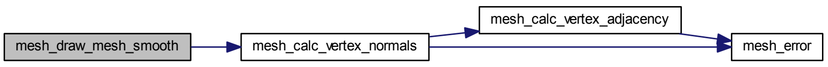
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:



5.4.2.3 void mesh_draw_point_cloud (MESH *m*)

Draws a given mesh in OpenGL context as pointcloud.

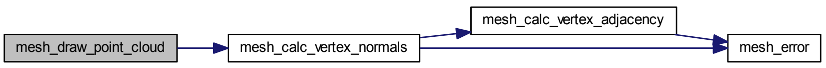
Parameters

`in`	*m*	Input mesh

Returns

NULL

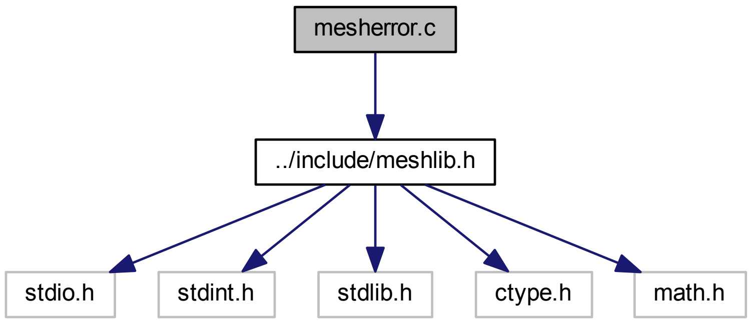
Here is the call graph for this function:



5.5 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.

5.5.1 Detailed Description

This file contains functions pertaining to handling errors.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.5.2 Function Documentation

5.5.2.1 void mesh_error (int *type*)

Displays error message and exits.

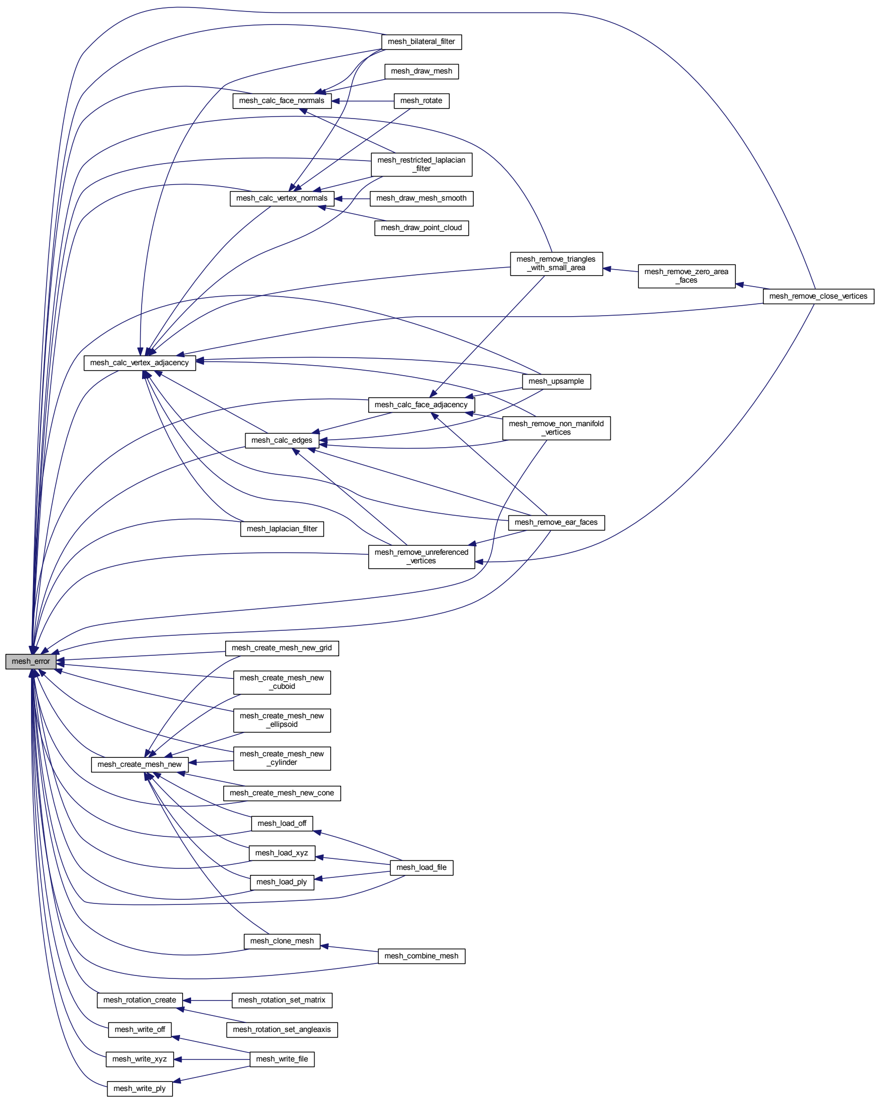
Parameters

in	*type*	Error type (MESH_ERR_MALLOC/MESH_ERR_SIZE_MISMATCH/MESH←_ERR_FNOTOPEN)

Returns

NULL

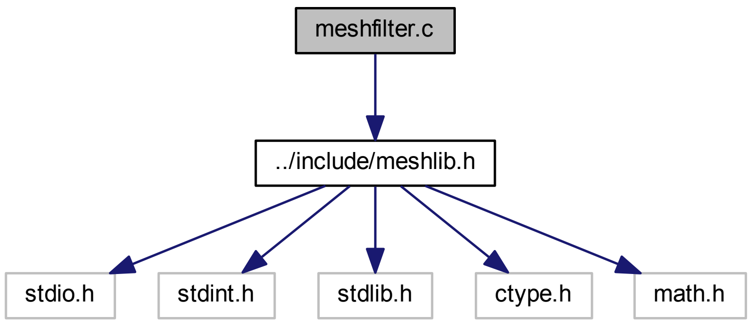
Here is the caller graph for this function:



5.6 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_bilateral_filter (MESH m, FLOATDATA sigma_c, FLOATDATA sigma_s, int niters)

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

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

  Restricted Mesh Laplacian filter.

5.6.1 Detailed Description

This file contains functions pertaining to different mesh filtering algorithms.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.6.2 Function Documentation

5.6.2.1 int mesh_bilateral_filter (**MESH** *m*, **FLOATDATA** *sigma_c*, **FLOATDATA** *sigma_s*, int *niters*)

Mesh bilateral filter.

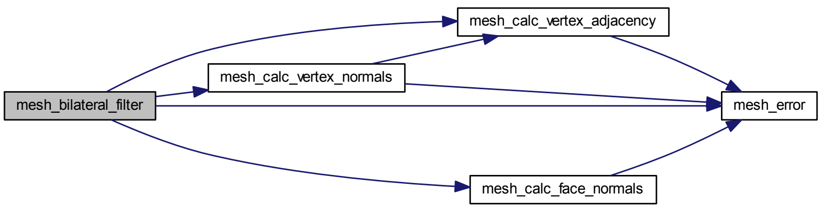
Parameters

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:



5.6.2.2 int mesh_laplacian_filter (MESH *m*, FLOATDATA *r*)

Mesh Laplacian filter.

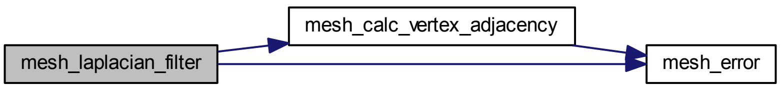
Parameters

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

Returns

Error code

Here is the call graph for this function:



5.6.2.3 int mesh_restricted_laplacian_filter (MESH *m*, FLOATDATA *r*, FLOATDATA *ang*)

Restricted Mesh Laplacian filter.

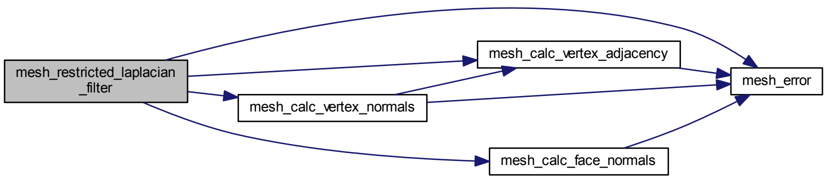
Parameters

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:

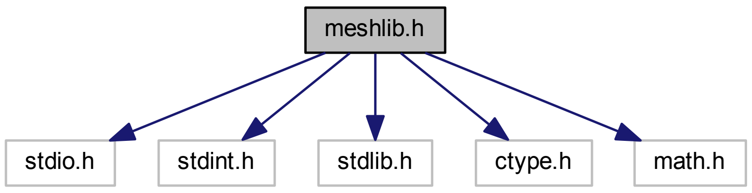


5.7 meshlib.h File Reference

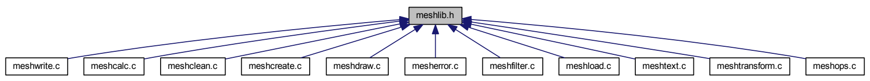
This header file contains declarations of all functions of meshlib.

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

Include dependency graph for meshlib.h:



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



Data Structures

- 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_transform
- struct mesh

Macros

- #define _CRT_SECURE_NO_DEPRECATE
- #define MESHLIBAPI extern
- #define MESH_INTDATA_TYPE 0
- #define MESH_FLOATDATA_TYPE 1
- #define INTDATA int32_t /∗ do not change this, careful see meshload fscanf and other functions ∗/
- #define FLOATDATA double /∗ do not change this, careful see meshload fscanf and other functions ∗/
- #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_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_CLONE_VERTICES (0x01)
- #define MESH_CLONE_VNORMALS (MESH_CLONE_VERTICES | __MESH_CLONE_VNORMALS)
- #define MESH_CLONE_VCOLORS (MESH_CLONE_VERTICES | __MESH_CLONE_VCOLORS)
- #define MESH_CLONE_VFACES (MESH_CLONE_VERTICES | __MESH_CLONE_VFACES)
- #define MESH_CLONE_V_ALL_PROPS (0x0F)
- #define MESH_CLONE_FACES (MESH_CLONE_VERTICES | __MESH_CLONE_FACES)
- #define MESH_CLONE_FNORMALS (MESH_CLONE_FACES | __MESH_CLONE_FNORMALS)
- #define MESH_CLONE_FCOLORS (MESH_CLONE_FACES | __MESH_CLONE_FCOLORS)
- #define MESH_CLONE_FAREAS (MESH_CLONE_FACES | __MESH_CLONE_FAREAS)
- #define MESH_CLONE_FFACES (MESH_CLONE_FACES | __MESH_CLONE_FFACES)
- #define MESH_CLONE_F_ALL_PROPS (MESH_CLONE_FACES | __MESH_CLONE_F_ALL_PROPS)
- #define MESH_CLONE_EDGES (MESH_CLONE_VERTICES | __MESH_CLONE_FACES | __MESH_C←
LONE_EDGES)
- #define MESH_CLONE_ALL_PROPS (0xFFFF)

Typedefs

- typedef struct _iobuf ∗ FILEPOINTER
- typedef INTDATA INTDATA2[2]
- typedef INTDATA INTDATA3[3]
- 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 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_transform mesh_transform
- typedef mesh_transform ∗ MESH_TRANSFORM
- 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.
- MESH mesh_create_mesh_new_grid (MESH_VECTOR3 sz, MESH_VECTOR3 pos, INTDATA m, INTDA←
  TA 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_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_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 MESH mesh_load_file (const char ∗fname)

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

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

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

  Reads a mesh from a PLY file.
- MESHLIBAPI int mesh_write_file (MESH m, const char ∗fname)

  Write a mesh to an OFF/PLY/ASC/XYZ file.
- MESHLIBAPI int mesh_write_off (MESH m, const char ∗fname)

  Write a mesh to an OFF file.
- MESHLIBAPI int mesh_write_xyz (MESH m, const char ∗fname)

  Write a mesh to an XYZ file.
- MESHLIBAPI int mesh_write_ply (MESH m, const char ∗fname)

  Write a mesh to an PLY 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 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_VE↩
  RTEX 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 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)
- 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_only (FILEPOINTER fp, char ∗c_word, int sz)

  Reads current word withot 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_bilateral_filter (MESH m, FLOATDATA sigma_c, FLOATDATA sigma_s, int niters)

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

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

  Restricted Mesh Laplacian 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_VERTEX 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 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.

5.7.1 Detailed Description

This header file contains declarations of all functions of meshlib.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.7.2 Macro Definition Documentation

5.7.2.1 #define _CRT_SECURE_NO_DEPRECATE

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

Float datatype

5.7.2.3 #define INTDATA int32_t /∗ do not change this, careful see meshload fscanf and other functions ∗/

Integer datatype

5.7.2.4 #define MESH_CLONE_ALL_PROPS (0xFFFF)

Clone mesh all properties

5.7.2.5 #define MESH_CLONE_EDGES (MESH_CLONE_VERTICES | __MESH_CLONE_FACES | __MESH_CLONE_EDGES)

Clone mesh edges

5.7.2.6 #define MESH_CLONE_F_ALL_PROPS (MESH_CLONE_FACES | __MESH_CLONE_F_ALL_PROPS)

Clone mesh all face properties

5.7.2.7 #define MESH_CLONE_FACES (MESH_CLONE_VERTICES | __MESH_CLONE_FACES)

Clone mesh faces

5.7.2.8 #define MESH_CLONE_FAREAS (MESH_CLONE_FACES | __MESH_CLONE_FAREAS)

Clone mesh faces and face areas

5.7.2.9 #define MESH_CLONE_FCOLORS (MESH_CLONE_FACES | __MESH_CLONE_FCOLORS)

Clone mesh faces and face colors

5.7.2.10 #define MESH_CLONE_FFACES (MESH_CLONE_FACES | __MESH_CLONE_FFACES)

Clone mesh faces and face face adjacency

5.7.2.11 #define MESH_CLONE_FNORMALS (MESH_CLONE_FACES | __MESH_CLONE_FNORMALS)

Clone mesh faces and face normals

5.7.2.12 #define MESH_CLONE_V_ALL_PROPS (0x0F)

Clone mesh all vertex properties

5.7.2.13 #define MESH_CLONE_VCOLORS (MESH_CLONE_VERTICES | __MESH_CLONE_VCOLORS)

Clone mesh vertices and vertex colors

5.7.2.14 #define MESH_CLONE_VERTICES (0x01)

Clone mesh vertices

5.7.2.15 #define MESH_CLONE_VFACES (MESH_CLONE_VERTICES | __MESH_CLONE_VFACES)

Clone mesh vertices and vertex face adjacency

5.7.2.16 #define MESH_CLONE_VNORMALS (MESH_CLONE_VERTICES | __MESH_CLONE_VNORMALS)

Clone mesh vertices and vertex normals

5.7.2.17 #define MESH_ERR_FNOTOPEN 2

Mesh error type - file open

5.7.2.18 #define MESH_ERR_INCOMPATIBLE 3

Mesh error type - incompatible data

5.7.2.19 #define MESH_ERR_MALLOC 0

Mesh error type - allocation

5.7.2.20 #define MESH_ERR_SIZE_MISMATCH 1

Mesh error type - size mismatch

5.7.2.21 #define MESH_ERR_UNKNOWN 4

Mesh error type - unknown

5.7.2.22 #define MESH_FLOATDATA_TYPE 1

Float datatype selector

5.7.2.23 #define MESH_INTDATA_TYPE 0

Integer datatype selector

5.7.2.24 #define MESH_ORIGIN_TYPE_BUILD 00

Mesh origin type - create new

5.7.2.25 #define MESH_ORIGIN_TYPE_COFF 13

Mesh origin type - COFF file

5.7.2.26 #define MESH_ORIGIN_TYPE_NCOFF 14

Mesh origin type - NCOFF file

5.7.2.27 #define MESH_ORIGIN_TYPE_NOFF 12

Mesh origin type - NOFF file

5.7.2.28 #define MESH_ORIGIN_TYPE_OFF 11

Mesh origin type - OFF file

5.7.2.29 #define MESH_ORIGIN_TYPE_PLY_ASCII 30

Mesh origin type - PLY ascii file

5.7.2.30 #define MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN 32

Mesh origin type - PLY binary BE file

5.7.2.31 #define MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN 31

Mesh origin type - PLY binary LE file

5.7.2.32 #define MESH_ORIGIN_TYPE_XYZ 20

Mesh origin type - XYZ file

5.7.2.33 #define MESH_PI (3.14159265359)

π

5.7.2.34 #define MESH_TWOPI (6.28318530718)

2π

5.7.2.35 #define MESHLIBAPI extern

5.7.3 Typedef Documentation

5.7.3.1 typedef struct _iobuf* FILEPOINTER

File pointer

5.7.3.2 typedef INTDATA INTDATA2[2]

2- element INTDATA

5.7.3.3 typedef INTDATA INTDATA3[3]

3- element INTDATA

5.7.3.4 typedef struct mesh mesh

Mesh

5.7.3.5 typedef mesh* MESH

Pointer to mesh

5.7.3.6 typedef struct mesh_adjface mesh_adjface

Adjacent face structure

5.7.3.7 typedef struct mesh_color mesh_color

5.7.3.8 typedef mesh_color* MESH_COLOR

Color

5.7.3.9 typedef struct mesh_edge mesh_edge

Edge

5.7.3.10 typedef struct **mesh_edge*** **MESH_EDGE**

Pointer to edge

5.7.3.11 typedef struct **mesh_face** **mesh_face**

Face

5.7.3.12 typedef **mesh_face*** **MESH_FACE**

Pointer to face

5.7.3.13 typedef struct **mesh_adjface** **mesh_fface**

Face adjacent faces

5.7.3.14 typedef **mesh_fface*** **MESH_FFACE**

Pointer to face adjacent faces

5.7.3.15 typedef **mesh_vector3** **mesh_normal**

Normal

5.7.3.16 typedef **mesh_normal*** **MESH_NORMAL**

Normal pointer

5.7.3.17 typedef struct **mesh_rotation** **mesh_rotation**

Rotation

5.7.3.18 typedef **mesh_rotation*** **MESH_ROTATION**

Pointer to rotation

5.7.3.19 typedef struct **mesh_struct** **mesh_struct**

INTDATA Structure

5.7.3.20 typedef **mesh_struct*** **MESH_STRUCT**

INTDATA Structure pointer

5.7.3.21 typedef struct **mesh_struct2** **mesh_struct2**

INTDATA2 Structure

5.7.3.22 typedef mesh_struct2∗ MESH_STRUCT2

INTDATA2 Structure pointer

5.7.3.23 typedef struct mesh_struct3 mesh_struct3

INTDATA3 Structure

5.7.3.24 typedef mesh_struct3∗ MESH_STRUCT3

INTDATA3 Structure pointer

5.7.3.25 typedef struct mesh_transform mesh_transform

Transformation

5.7.3.26 typedef mesh_transform∗ MESH_TRANSFORM

Pointer to transformation

5.7.3.27 typedef struct mesh_vector3 mesh_vector3

Generic 3-d vector

5.7.3.28 typedef mesh_vector3∗ MESH_VECTOR3

Generic 3-d vector pointer

5.7.3.29 typedef mesh_vector3 mesh_vertex

Vertex

5.7.3.30 typedef mesh_vertex∗ MESH_VERTEX

Vertex pointer

5.7.3.31 typedef struct mesh_adjface mesh_vface

Vertex adjacent faces

5.7.3.32 typedef mesh_vface∗ MESH_VFACE

Pointer to vertex adjacent faces

5.7.4 Function Documentation

5.7.4.1 MESHLIBAPI int mesh_bilateral_filter (MESH *m,* FLOATDATA *sigma_c,* FLOATDATA *sigma_s,* int *niters*)

Mesh bilateral filter.

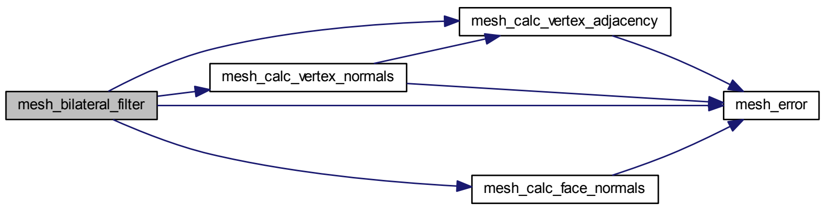
Parameters

`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:



5.7.4.2 MESHLIBAPI int mesh_calc_edges (MESH *m*)

Computes edges of a given mesh.

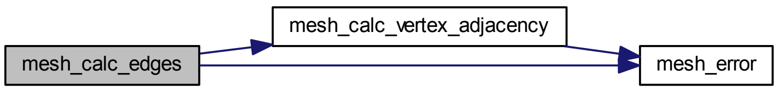
Parameters

`in`	*m*	Input mesh

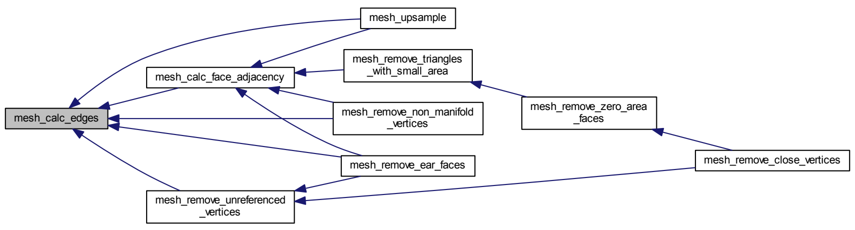
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_upsample
mesh_remove_triangles _with_small_area
mesh_calc_face_adjacency
mesh_calc_edges
mesh_remove_non_manifold _vertices
mesh_remove_zero_area _faces
mesh_remove_ear_faces
mesh_remove_close_vertices
mesh_remove_unreferenced _vertices

5.7.4.3 MESHLIBAPI int mesh_calc_face_adjacency (MESH *m*)

Computes face adjacent faces of a given mesh.

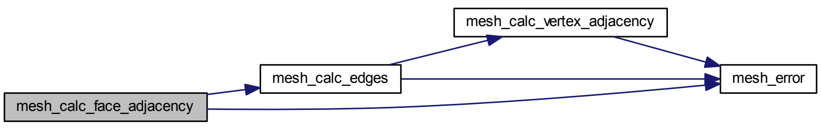
Parameters

in	*m*	Input mesh

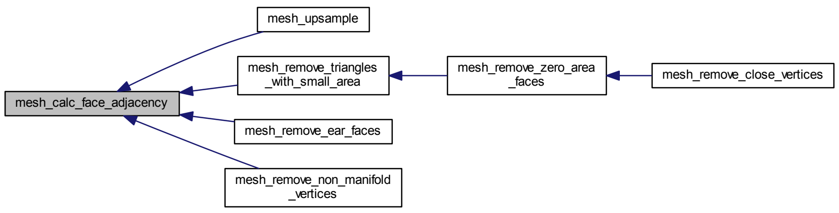
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_upsample
mesh_remove_triangles _with_small_area
mesh_remove_zero_area _faces
mesh_remove_close_vertices
mesh_calc_face_adjacency
mesh_remove_ear_faces
mesh_remove_non_manifold _vertices

5.7.4.4 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

5.7.4.5 MESHLIBAPI int mesh_calc_face_normals (MESH *m*)

Computes face normals of a given mesh.

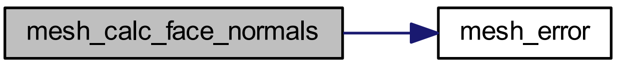
Parameters

in	*m*	Input mesh

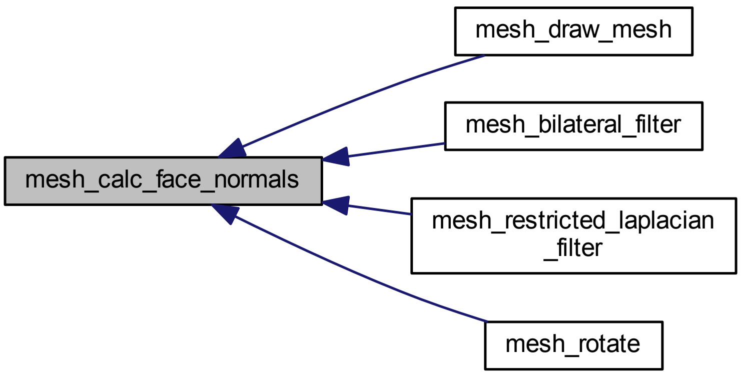
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.6 MESHLIBAPI FLOATDATA mesh_calc_triangle_area (MESH_VERTEX *a*, MESH_VERTEX *b*, MESH_VERTEX *c*)

Computes area of a triangle.

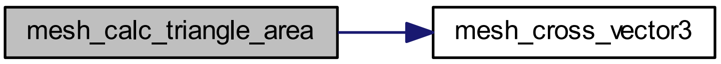
Parameters

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

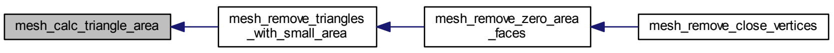
Returns

Area

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_calc_triangle_area
mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices

5.7.4.7 MESHLIBAPI int mesh_calc_vertex_adjacency (MESH *m*)

Computes vertex adjacent faces of a given mesh.

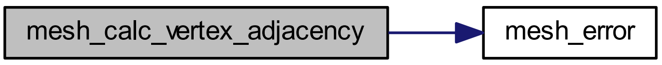
Parameters

in	*m*	Input mesh

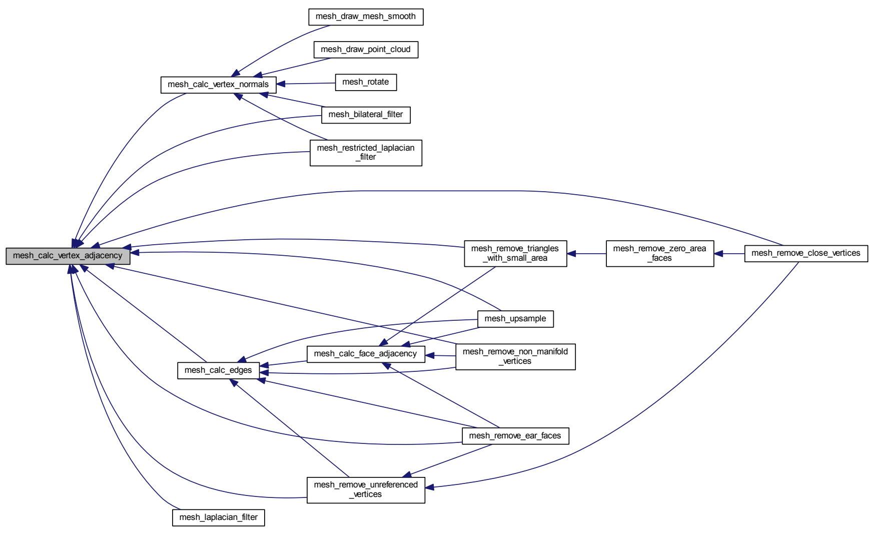
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_draw_mesh_smooth
mesh_draw_point_cloud
mesh_calc_vertex_normals
mesh_rotate
mesh_bilateral_filter
mesh_restricted_laplacian _filter
mesh_calc_vertex_adjacency
mesh_remove_triangles _with_small_area
mesh_remove_zero_area _faces
mesh_remove_close_vertices
mesh_upsample
mesh_calc_face_adjacency
mesh_remove_non_manifold _vertices
mesh_calc_edges
mesh_remove_ear_faces
mesh_remove_unreferenced _vertices
mesh_laplacian_filter

5.7.4.8 MESHLIBAPI int mesh_calc_vertex_normals (MESH *m*)

Computes vertex normals of a given mesh.

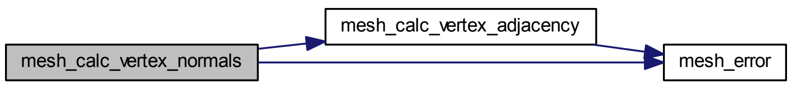
Parameters

in	*m*	Input mesh

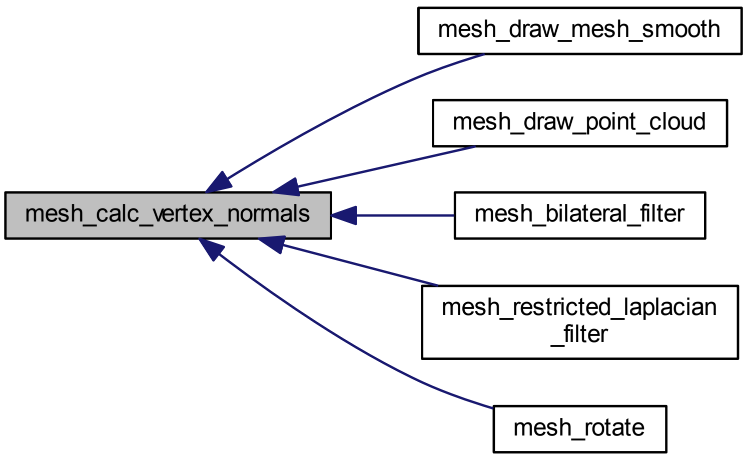
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.9 MESHLIBAPI MESH mesh_clone_mesh (MESH *m*, uint16_t *flags*)

Clones a given mesh into another mesh.

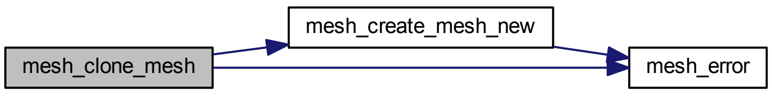
Parameters

in	*m*	Input mesh to clone
in	*flags*	Flags to copy which properties (MESH_CLONE_VERTICES/MESH_CLONE_VNORMALS/MESH_CLONE_VCOLORS/MESH_CLONE_VFACES/MESH_CLONE_V_ALL_PROPS/MESH_CLONE_FACES/MESH_CLONE_FNORMALS/MESH_CLONE_FCOLORS/MESH_CLONE_FAREAS/MESH_CLONE_F_ALL_PROPS/MESH_CLONE_ALL_PROPS)

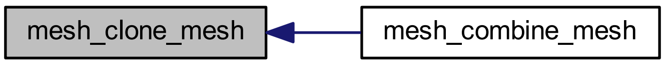
Returns

Output cloned mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.10 MESHLIBAPI MESH mesh_combine_mesh (MESH *m1*, MESH *m2*)

Combines a given mesh with another given mesh.

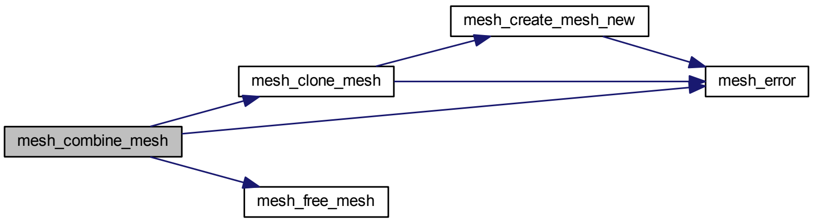
Parameters

`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:



5.7.4.11 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

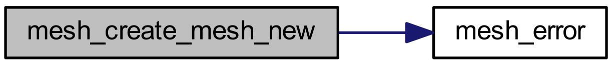
5.7.4.12 MESHLIBAPI MESH mesh_create_mesh_new ( )

Creates a new mesh.

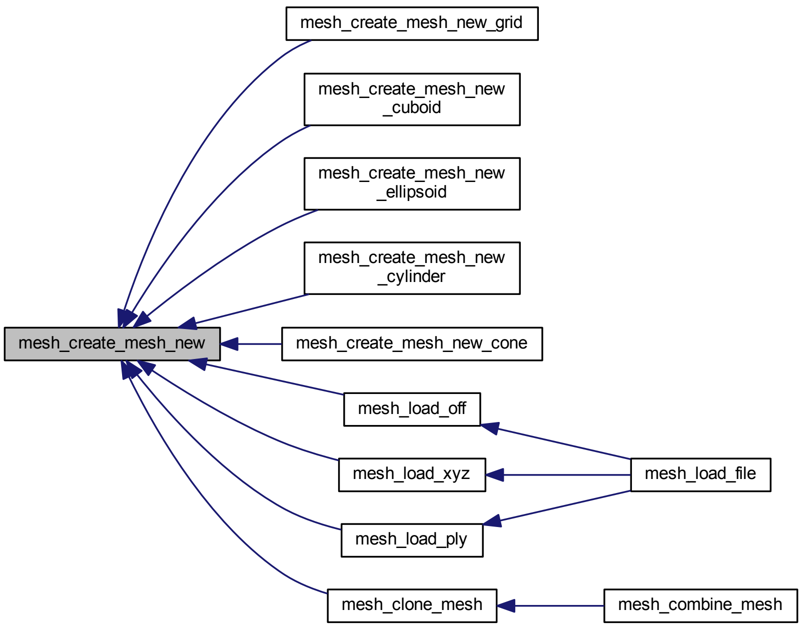
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_create_mesh_new_grid
mesh_create_mesh_new_cuboid
mesh_create_mesh_new_ellipsoid
mesh_create_mesh_new_cylinder
mesh_create_mesh_new
mesh_create_mesh_new_cone
mesh_load_off
mesh_load_xyz
mesh_load_file
mesh_load_ply
mesh_clone_mesh
mesh_combine_mesh

5.7.4.13 MESHLIBAPI MESH mesh_create_mesh_new_cone (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates a cone mesh.

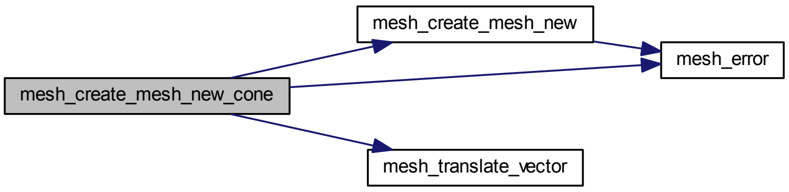
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.7.4.14 MESHLIBAPI MESH mesh_create_mesh_new_cuboid (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates a cuboid mesh.

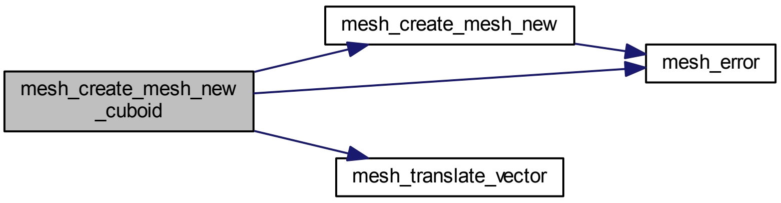
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.7.4.15 MESHLIBAPI MESH mesh_create_mesh_new_cylinder (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates a cylinder mesh.

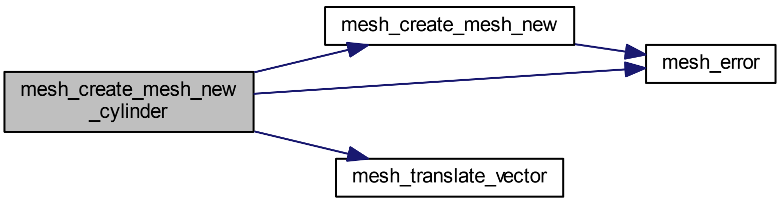
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.7.4.16 MESHLIBAPI MESH mesh_create_mesh_new_ellipsoid (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates an ellipsoid mesh.

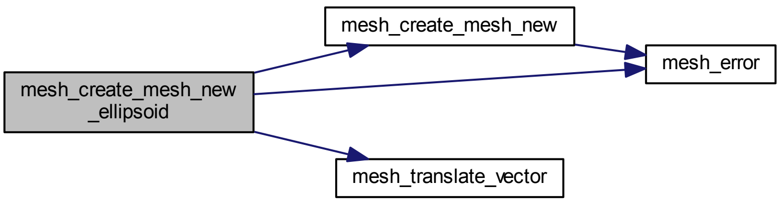
Parameters

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

Returns

Output mesh

Here is the call graph for this function:



5.7.4.17 MESH mesh_create_mesh_new_grid (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*, INTDATA *m*, INTDATA *n*)

Creates a grid mesh.

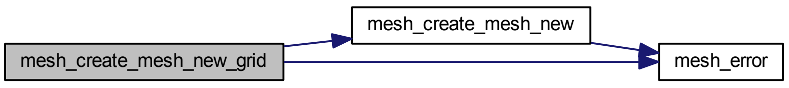
Parameters

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:



5.7.4.18 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

5.7.4.19 MESHLIBAPI void mesh_cross_vector3 (MESH_VECTOR3 *x*, MESH_VECTOR3 *y*, MESH_VECTOR3 *z*)

Computes the cross product of two 3-d vectors.

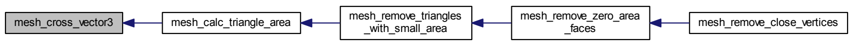
Parameters

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:


mesh_cross_vector3
mesh_calc_triangle_area
mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices

5.7.4.20 MESHLIBAPI void mesh_draw_mesh (MESH *m*)

Draws a given mesh in OpenGL context in flat shading.

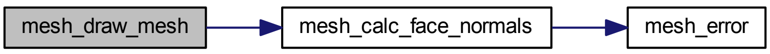
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:



5.7.4.21 MESHLIBAPI void mesh_draw_mesh_smooth (MESH *m*)

Draws a given mesh in OpenGL context in smoothing shading.

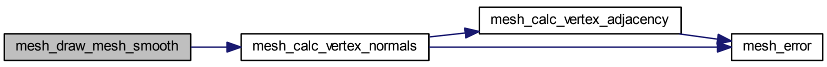
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:



5.7.4.22 MESHLIBAPI void mesh_draw_point_cloud (MESH *m*)

Draws a given mesh in OpenGL context as pointcloud.

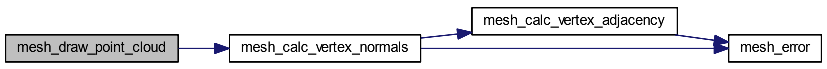
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the call graph for this function:



5.7.4.23 MESHLIBAPI void mesh_error (int *type*)

Displays error message and exits.

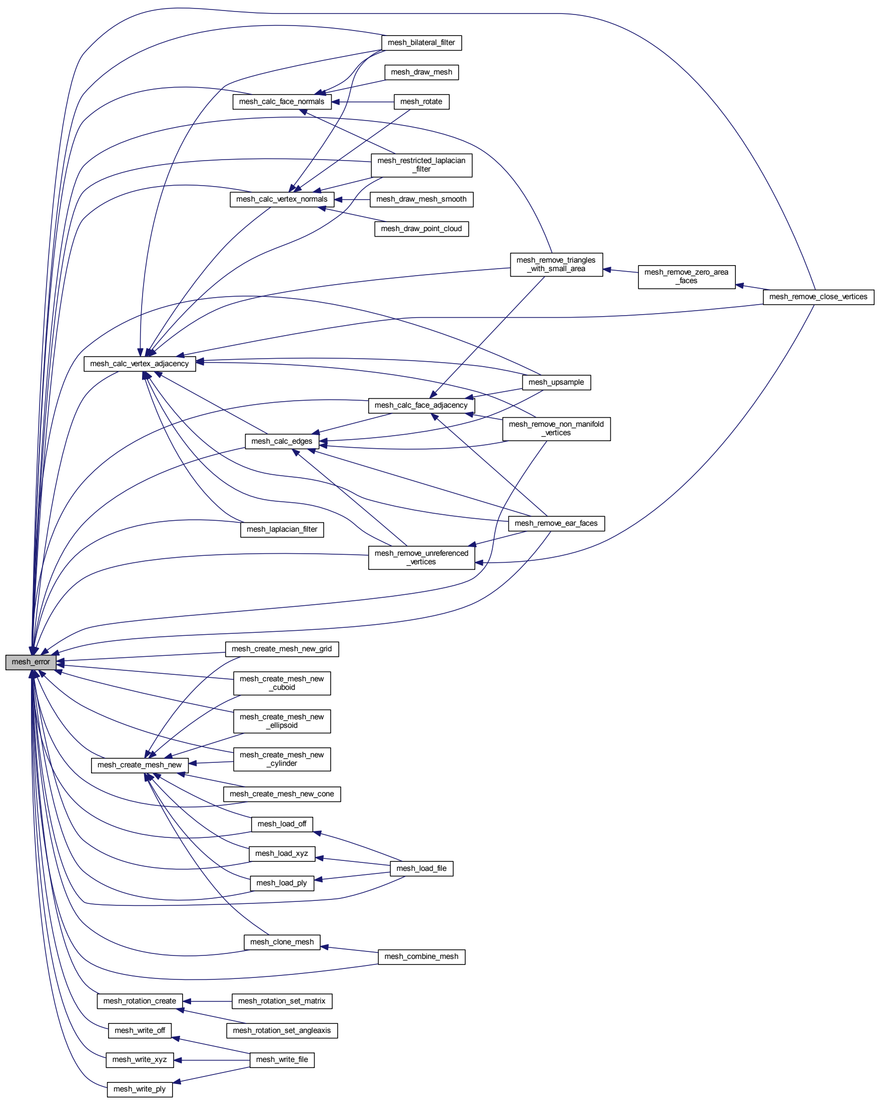
Parameters

in	*type*	Error type (MESH_ERR_MALLOC/MESH_ERR_SIZE_MISMATCH/MESH←_ERR_FNOTOPEN)

Returns

NULL

Here is the caller graph for this function:



5.7.4.24 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

5.7.4.25 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

5.7.4.26 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

5.7.4.27 MESHLIBAPI void mesh_free_mesh (MESH *m*)

Frees a mesh.

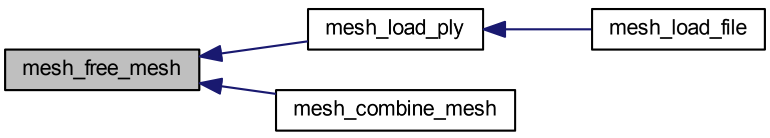
Parameters

in	*m*	Input mesh

Returns

NULL

Here is the caller graph for this function:



5.7.4.28 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

5.7.4.29 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

5.7.4.30 MESHLIBAPI int mesh_laplacian_filter (MESH *m*, FLOATDATA *r*)

Mesh Laplacian filter.

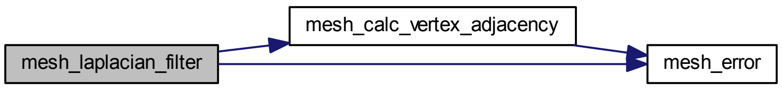
Parameters

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

Returns

Error code

Here is the call graph for this function:



5.7.4.31 MESHLIBAPI MESH mesh_load_file (const char ∗ *fname*)

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

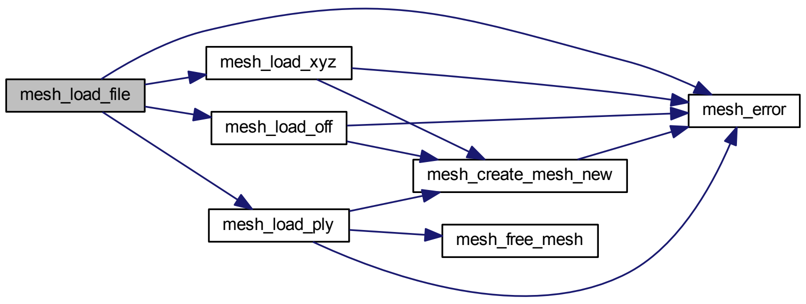
Parameters

in	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



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

Reads a mesh from an OFF file.

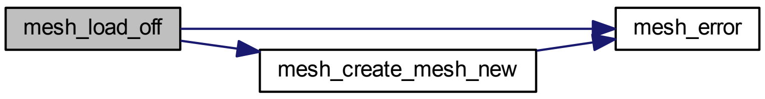
Parameters

in	*fname*	Input filename

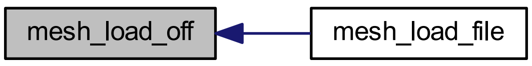
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



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

Reads a mesh from a PLY file.

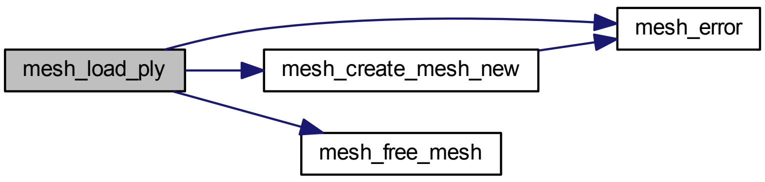
Parameters

`in`	*fname*	Input filename

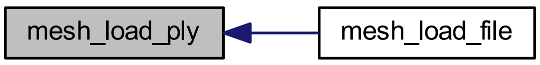
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.34 MESHLIBAPI MESH mesh_load_xyz (const char ∗ *fname*)

Read a mesh from an ASC/XYZ file.

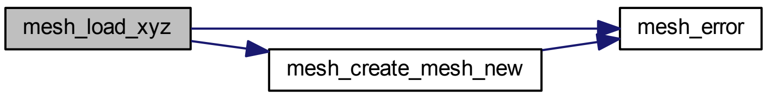
Parameters

`in`	*fname*	Input filename

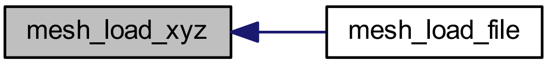
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.35 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

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

Reads current word withot 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

5.7.4.37 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

5.7.4.38 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

5.7.4.39 MESHLIBAPI int mesh_remove_close_vertices (MESH *m*, FLOATDATA *r*)

Removes close vertices.

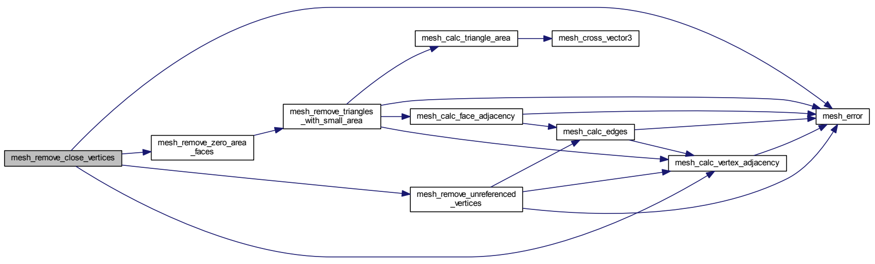
Parameters

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

Returns

Error code

Here is the call graph for this function:


mesh_calc_triangle_area
mesh_cross_vector3
mesh_remove_triangles_with_small_area
mesh_calc_face_adjacency
mesh_error
mesh_calc_edges
mesh_remove_zero_area_faces
mesh_remove_close_vertices
mesh_calc_vertex_adjacency
mesh_remove_unreferenced_vertices

5.7.4.40 MESHLIBAPI int mesh_remove_ear_faces (MESH *m*, int *niters*)

Removes ear faces and connecting vertices.

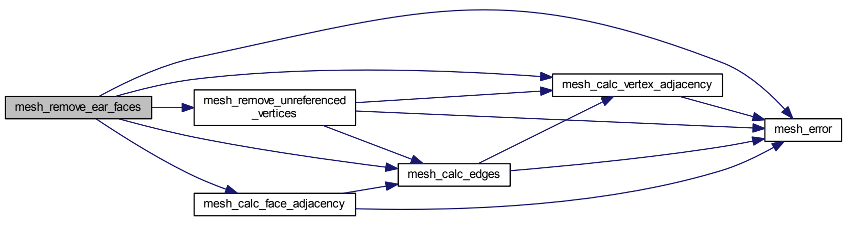
Parameters

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

Returns

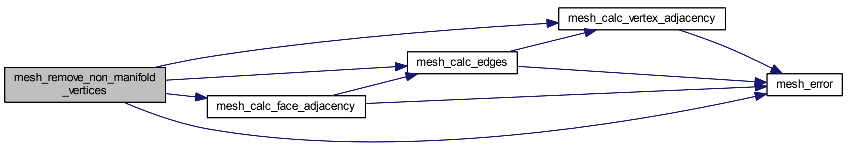
Error code

Here is the call graph for this function:



5.7.4.41 MESHLIBAPI int mesh_remove_non_manifold_vertices (MESH *m*)

Here is the call graph for this function:



5.7.4.42 MESHLIBAPI int mesh_remove_triangles_with_small_area (MESH *m*, FLOATDATA *area*)

Removes triangles with area smaller than a given value.

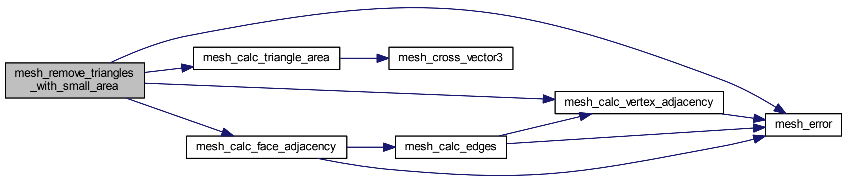
Parameters

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

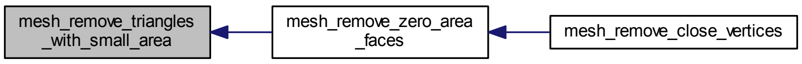
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:


mesh_remove_triangles_with_small_area
mesh_remove_zero_area_faces
mesh_remove_close_vertices

5.7.4.43 MESHLIBAPI int mesh_remove_unreferenced_vertices (MESH *m*)

Removes unreferenced vertices.

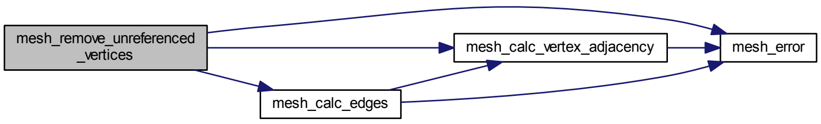
Parameters

`in`	*m*	Input mesh

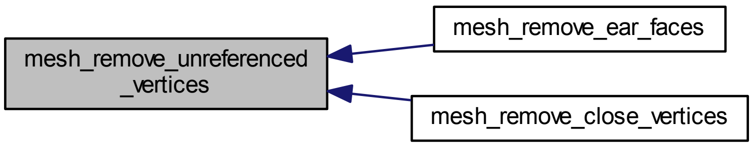
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.44 MESHLIBAPI int mesh_remove_zero_area_faces (MESH *m*)

Removes triangles with zero area.

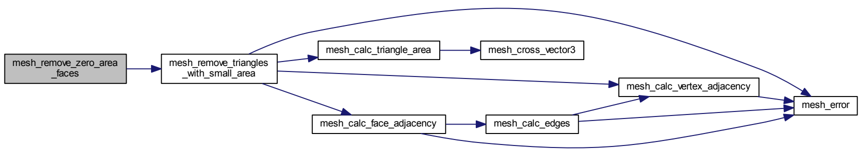
Parameters

in	*m*	Input mesh

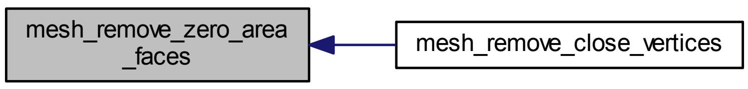
Returns

Error code

Here is the call graph for this function:


mesh_remove_zero_area _faces
mesh_remove_triangles _with_small_area
mesh_calc_triangle_area
mesh_cross_vector3
mesh_calc_vertex_adjacency
mesh_error
mesh_calc_face_adjacency
mesh_calc_edges

Here is the caller graph for this function:



5.7.4.45 MESHLIBAPI int mesh_restricted_laplacian_filter (MESH *m*, FLOATDATA *r*, FLOATDATA *ang*)

Restricted Mesh Laplacian filter.

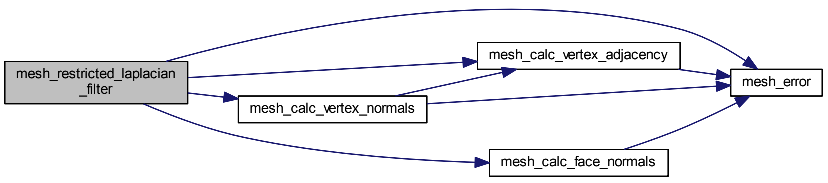
Parameters

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:



5.7.4.46 MESHLIBAPI int mesh_rotate (MESH *m*, MESH_ROTATION *r*)

Rotates a mesh by a given rotation.

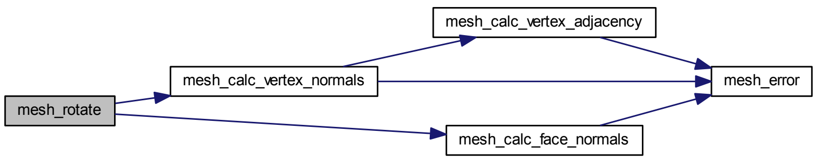
Parameters

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

Returns

Error code

Here is the call graph for this function:



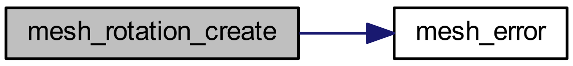
5.7.4.47 MESHLIBAPI MESH_ROTATION mesh_rotation_create ( )

Creates a new rotation.

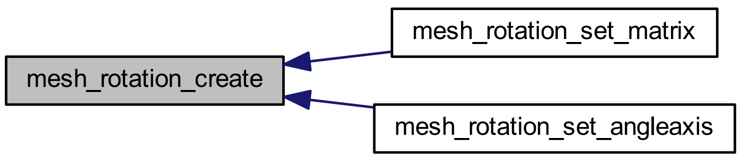
Returns

Output rotation

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.48 MESHLIBAPI void mesh_rotation_free (MESH_ROTATION *r*)

Frees a given rotation.

Parameters

r	Input rotation

Returns

NULL

5.7.4.49 MESHLIBAPI MESH_ROTATION mesh_rotation_set_angleaxis (FLOATDATA *ang*, MESH_NORMAL *axis*, MESH_ROTATION *r*)

Sets rotation from angle axis.

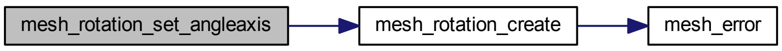
Parameters

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:



5.7.4.50 MESHLIBAPI MESH_ROTATION mesh_rotation_set_matrix (FLOATDATA ∗ *mat*, MESH_ROTATION *r*)

Sets rotation from a matrix.

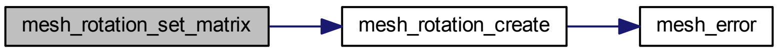
Parameters

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

Returns

Output rotation

Here is the call graph for this function:



5.7.4.51 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

5.7.4.52 MESHLIBAPI int mesh_skip_line (FILEPOINTER *fp*)

Skips to next line.

Parameters

`in`	*fp*	Pointer to input file

Returns

Status 0 - Normal/ 1- EOF

5.7.4.53 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

5.7.4.54 MESHLIBAPI int mesh_translate_vector (MESH *m*, MESH_VECTOR3 *v*)

Translates a mesh by a given 3-d vector.

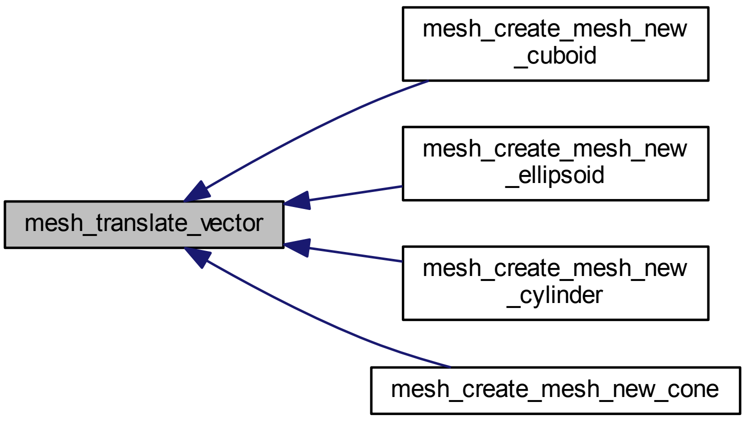
Parameters

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

Returns

Error code

Here is the caller graph for this function:


mesh_create_mesh_new _cuboid
mesh_create_mesh_new _ellipsoid
mesh_translate_vector
mesh_create_mesh_new _cylinder
mesh_create_mesh_new_cone

5.7.4.55 MESHLIBAPI int mesh_upsample (MESH *m*, int *iters*)

Upsamples a given mesh.

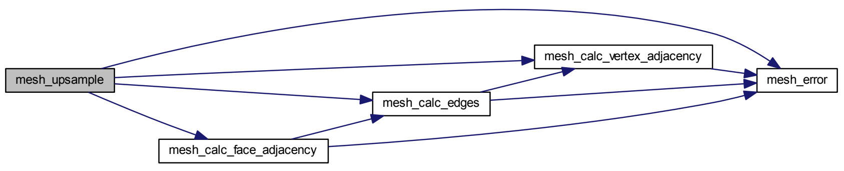
Parameters

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

Returns

Error code

Here is the call graph for this function:



5.7.4.56 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

5.7.4.57 MESHLIBAPI int mesh_write_file (MESH *m*, const char ∗ *fname*)

Write a mesh to an OFF/PLY/ASC/XYZ file.

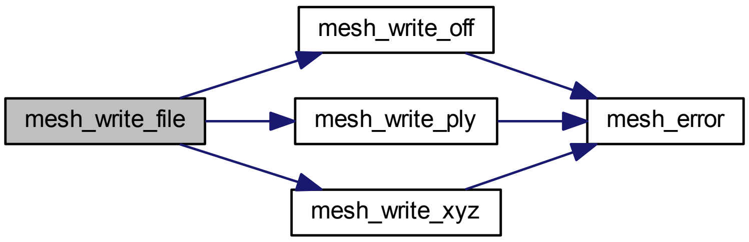
Parameters

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

Returns

Error code

Here is the call graph for this function:



5.7.4.58 MESHLIBAPI int mesh_write_off (MESH *m*, const char ∗ *fname*)

Write a mesh to an OFF file.

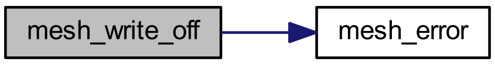
Parameters

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

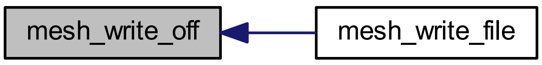
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.59 MESHLIBAPI int mesh_write_ply (MESH *m*, const char ∗ *fname*)

Write a mesh to an PLY file.

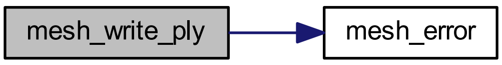
Parameters

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

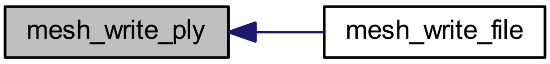
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.60 MESHLIBAPI int mesh_write_xyz (MESH *m*, const char ∗ *fname*)

Write a mesh to an XYZ file.

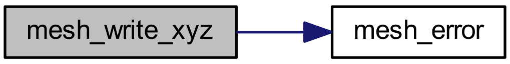
Parameters

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

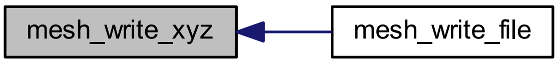
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

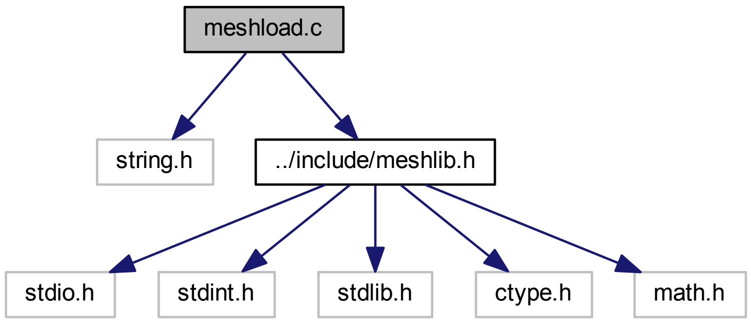


5.8 meshload.c File Reference

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

```
#include <string.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 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.

5.8.1 Detailed Description

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

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.8.2 Function Documentation

5.8.2.1 MESH mesh_load_file (const char ∗ *fname*)

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

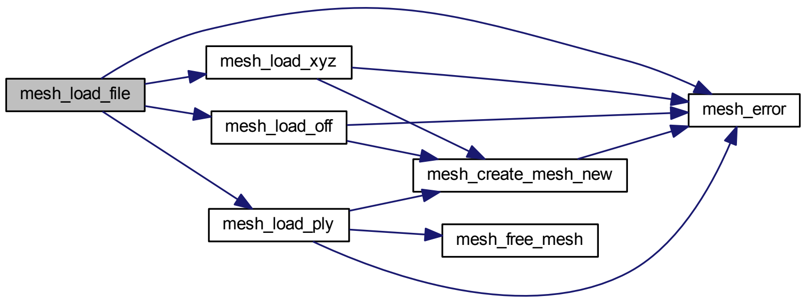
Parameters

`in`	*fname*	Input filename

Returns

Output mesh

Here is the call graph for this function:



5.8.2.2 MESH mesh_load_off (const char ∗ *fname*)

Reads a mesh from an OFF file.

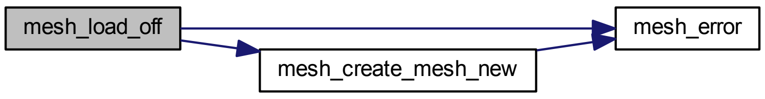
Parameters

`in`	*fname*	Input filename

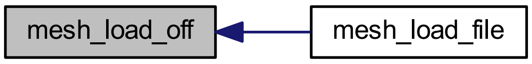
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.3 MESH mesh_load_ply (const char * *fname*)

Reads a mesh from a PLY file.

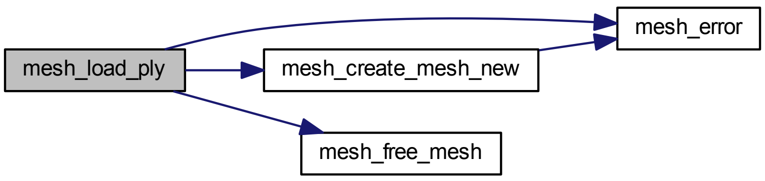
Parameters

`in`	*fname*	Input filename

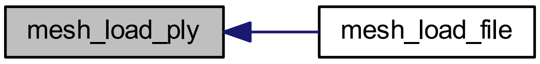
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.8.2.4 MESH mesh_load_xyz (const char * *fname*)

Read a mesh from an ASC/XYZ file.

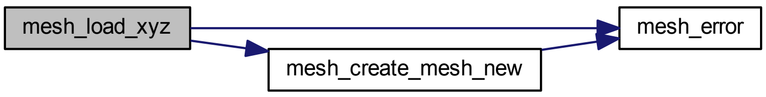
Parameters

`in`	*fname*	Input filename

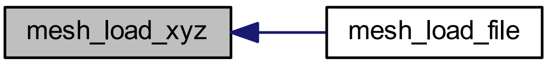
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

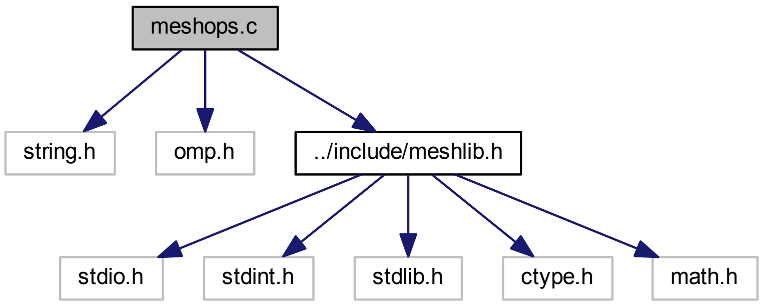


5.9 meshops.c File Reference

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.

5.9.1 Detailed Description

This file contains functions pertaining to mesh combinatorial operations.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.9.2 Function Documentation

5.9.2.1 MESH mesh_clone_mesh (MESH *m*, uint16_t *flags*)

Clones a given mesh into another mesh.

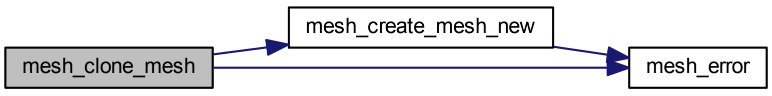
Parameters

in	*m*	Input mesh to clone
in	*flags*	Flags to copy which properties (MESH_CLONE_VERTICES/MESH_CLON↩ E_VNORMALS/MESH_CLONE_VCOLORS/MESH_CLONE_VFACES/ME↩ SH_CLONE_V_ALL_PROPS/MESH_CLONE_FACES/MESH_CLONE_FN↩ ORMALS/MESH_CLONE_FCOLORS/MESH_CLONE_FAREAS/MESH_C↩ LONE_F_ALL_PROPS/MESH_CLONE_ALL_PROPS)

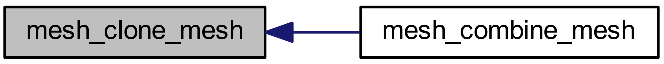
Returns

Output cloned mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.2.2 MESH mesh_combine_mesh (MESH *m1*, MESH *m2*)

Combines a given mesh with another given mesh.

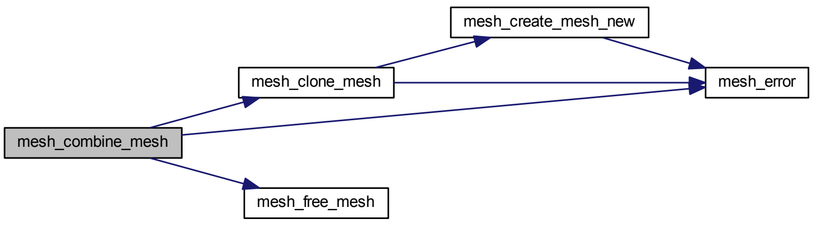
Parameters

`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:

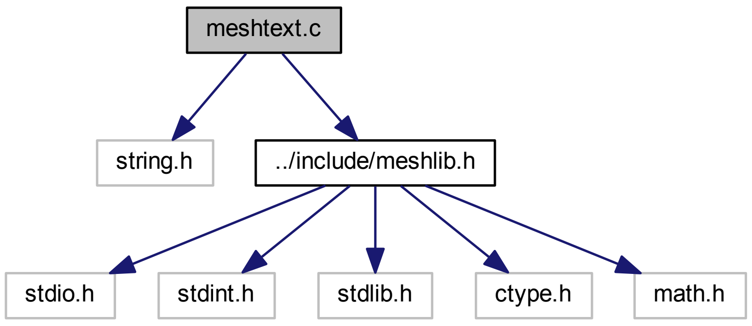


5.10 meshtext.c File Reference

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_only (FILEPOINTER fp, char ∗c_word, int sz)

  Reads current word withot moving to the next word.
- int mesh_skip_line (FILEPOINTER fp)

  Skips to next line.

5.10.1 Detailed Description

This file contains functions pertaining to different text routines.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.10.2 Function Documentation

5.10.2.1 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

5.10.2.2 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

5.10.2.3 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

5.10.2.4 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

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

Reads current word withot 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

5.10.2.6 int mesh_skip_line (FILEPOINTER *fp*)

Skips to next line.

Parameters

in	*fp*	Pointer to input file

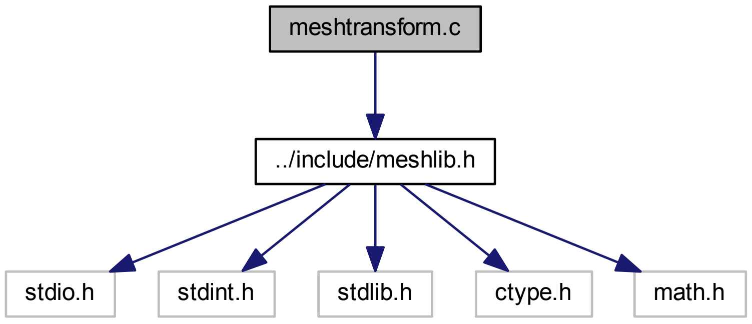
Returns

Status 0 - Normal/ 1- EOF

5.11 meshtransform.c File Reference

This file contains functions pertaining to different mesh transformations.

```
#include "../include/meshlib.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_ROTATION mesh_rotation_set_matrix (FLOATDATA ∗mat, MESH_ROTATION r)

  Sets rotation from a matrix.

- MESH_ROTATION mesh_rotation_set_angleaxis (FLOATDATA ang, MESH_NORMAL axis, MESH_ROT↩ ATION 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.

5.11.1 Detailed Description

This file contains functions pertaining to different mesh transformations.

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

Copyright (c) 2013, 2014, 2015, 2016 Sk. Mohammadul Haque.

5.11.2 Function Documentation

5.11.2.1 int mesh_rotate (MESH *m*, MESH_ROTATION *r*)

Rotates a mesh by a given rotation.

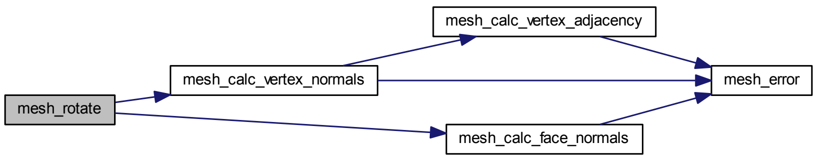
Parameters

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

Returns

Error code

Here is the call graph for this function:



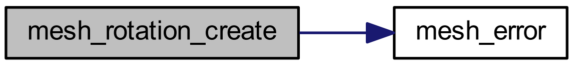
5.11.2.2 MESH_ROTATION mesh_rotation_create ( )

Creates a new rotation.

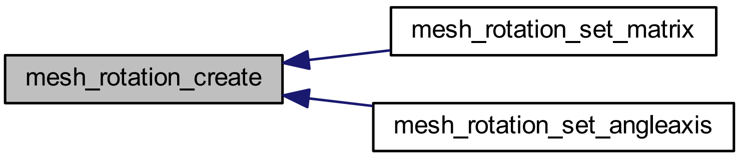
Returns

Output rotation

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.2.3 void mesh_rotation_free (MESH_ROTATION *r*)

Frees a given rotation.

Parameters

r	Input rotation

Returns

NULL

5.11.2.4 MESH_ROTATION mesh_rotation_set_angleaxis (FLOATDATA *ang*, MESH_NORMAL *axis*, MESH_ROTATION *r*)

Sets rotation from angle axis.

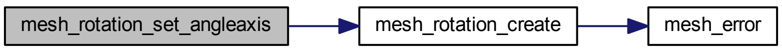
Parameters

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:



5.11.2.5 MESH_ROTATION mesh_rotation_set_matrix (FLOATDATA ∗ *mat*, MESH_ROTATION *r*)

Sets rotation from a matrix.

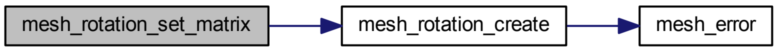
Parameters

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

Returns

Output rotation

Here is the call graph for this function:



5.11.2.6 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

5.11.2.7 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

5.11.2.8 int mesh_translate_vector (MESH *m*, MESH_VECTOR3 *v*)

Translates a mesh by a given 3-d vector.

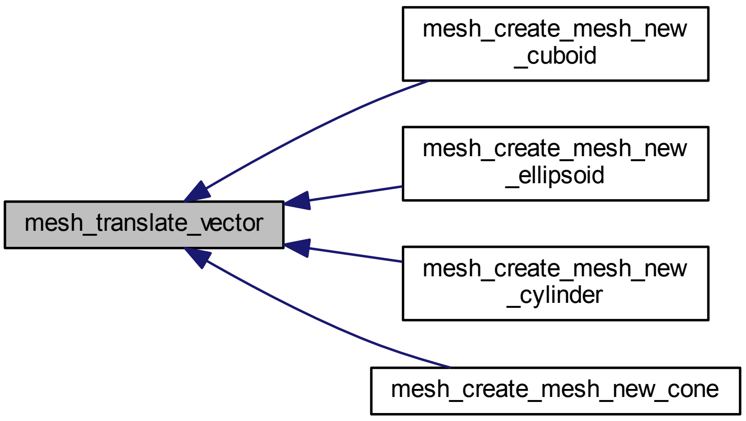
Parameters

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

Returns

Error code

Here is the caller graph for this function:


mesh_create_mesh_new _cuboid
mesh_create_mesh_new _ellipsoid
mesh_translate_vector
mesh_create_mesh_new _cylinder
mesh_create_mesh_new_cone

5.11.2.9 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

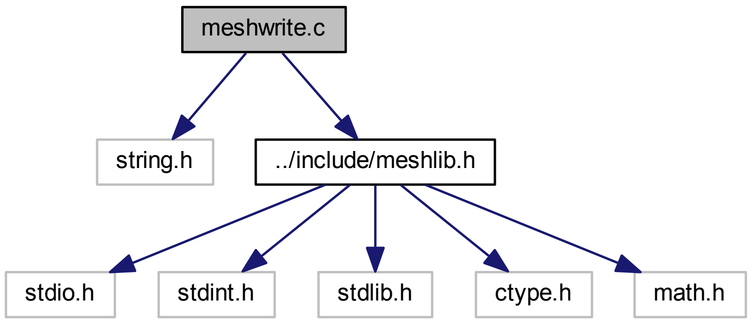
Returns

Output vertex

5.12 meshwrite.c File Reference

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

```
#include <string.h>
#include "../include/meshlib.h"
```
Include dependency graph for meshwrite.c:



Functions

- int mesh_write_file (MESH m, const char ∗fname)

  Write a mesh to an OFF/PLY/ASC/XYZ file.
- int mesh_write_off (MESH m, const char ∗fname)

  Write a mesh to an OFF file.
- int mesh_write_xyz (MESH m, const char ∗fname)

  Write a mesh to an XYZ file.
- int mesh_write_ply (MESH m, const char ∗fname)

  Write a mesh to an PLY file.

5.12.1 Detailed Description

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

Author

Sk. Mohammadul Haque

Version

1.4.2.0

Copyright

5.12.2 Function Documentation

5.12.2.1 int mesh_write_file (MESH *m*, const char ∗ *fname*)

Write a mesh to an OFF/PLY/ASC/XYZ file.

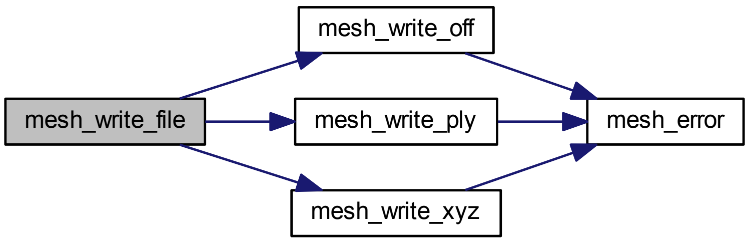
Parameters

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

Returns

Error code

Here is the call graph for this function:



5.12.2.2 int mesh_write_off (MESH *m*, const char ∗ *fname*)

Write a mesh to an OFF file.

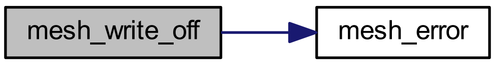
Parameters

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

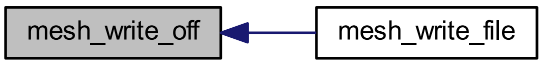
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.3 int mesh_write_ply (**MESH** *m,* const char ∗ *fname*)

Write a mesh to an PLY file.

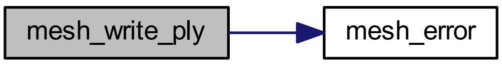
Parameters

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

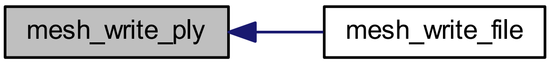
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.12.2.4 int mesh_write_xyz (MESH *m*, const char * *fname*)

Write a mesh to an XYZ file.

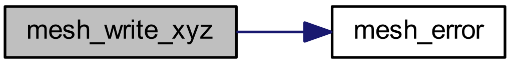
Parameters

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

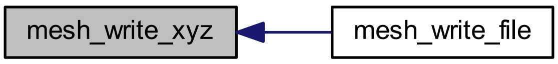
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



Index