

MeshLib

1.4.0.0

Generated by Doxygen 1.8.1.1

Sun Jun 14 2015 11:18:42

Contents

1	Meshlib	1
1.1	Introduction	1
1.2	Build	1
1.3	Contents	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	mesh Struct Reference	7
4.1.1	Field Documentation	8
4.1.1.1	dummy	8
4.1.1.2	faces	8
4.1.1.3	fareas	8
4.1.1.4	fcolors	8
4.1.1.5	fnormals	8
4.1.1.6	is_faces	8
4.1.1.7	is_fareas	8
4.1.1.8	is_fcolors	8
4.1.1.9	is_fnormals	8
4.1.1.10	is_loaded	8
4.1.1.11	is_trimesh	8
4.1.1.12	is_vcolors	9
4.1.1.13	is_vertices	9
4.1.1.14	is_vfaces	9
4.1.1.15	is_vnormals	9
4.1.1.16	num_faces	9
4.1.1.17	num_vertices	9
4.1.1.18	origin_type	9

4.1.1.19	vcolors	9
4.1.1.20	vertices	9
4.1.1.21	vfaces	9
4.1.1.22	vnormals	9
4.2	mesh_color Struct Reference	10
4.2.1	Field Documentation	10
4.2.1.1	a	10
4.2.1.2	b	10
4.2.1.3	g	10
4.2.1.4	r	10
4.3	mesh_face Struct Reference	10
4.3.1	Field Documentation	10
4.3.1.1	num_vertices	10
4.3.1.2	vertices	11
4.4	mesh_rotation Struct Reference	11
4.4.1	Field Documentation	11
4.4.1.1	data	11
4.5	mesh_struct Struct Reference	11
4.5.1	Field Documentation	11
4.5.1.1	items	11
4.5.1.2	num_items	11
4.6	mesh_struct2 Struct Reference	12
4.6.1	Field Documentation	12
4.6.1.1	items	12
4.6.1.2	num_items	12
4.7	mesh_transform Struct Reference	12
4.7.1	Field Documentation	12
4.7.1.1	data	12
4.8	mesh_vector3 Struct Reference	12
4.8.1	Field Documentation	13
4.8.1.1	x	13
4.8.1.2	y	13
4.8.1.3	z	13
4.9	mesh_vface Struct Reference	13
4.9.1	Field Documentation	13
4.9.1.1	faces	13
4.9.1.2	num_faces	13
5	File Documentation	15
5.1	meshcalc.c File Reference	15

5.1.1	Detailed Description	16
5.1.2	Function Documentation	16
5.1.2.1	mesh_calc_face_normal	16
5.1.2.2	mesh_calc_face_normals	17
5.1.2.3	mesh_calc_triangle_area	17
5.1.2.4	mesh_calc_vertex_adjacency	18
5.1.2.5	mesh_calc_vertex_normals	19
5.1.2.6	mesh_cross_normal	20
5.1.2.7	mesh_cross_vector3	20
5.1.2.8	mesh_find	21
5.1.2.9	mesh_find2	21
5.1.2.10	mesh_upsample	21
5.2	meshclean.c File Reference	21
5.2.1	Detailed Description	22
5.2.2	Function Documentation	23
5.2.2.1	mesh_remove_boundary_faces	23
5.2.2.2	mesh_remove_boundary_vertices	23
5.2.2.3	mesh_remove_close_vertices	23
5.2.2.4	mesh_remove_ear_faces	24
5.2.2.5	mesh_remove_triangles_with_small_area	24
5.2.2.6	mesh_remove_unreferenced_vertices	25
5.2.2.7	mesh_remove_zero_area_faces	25
5.3	meshcreate.c File Reference	26
5.3.1	Detailed Description	27
5.3.2	Function Documentation	27
5.3.2.1	mesh_create_mesh_new	27
5.3.2.2	mesh_create_mesh_new_cone	28
5.3.2.3	mesh_create_mesh_new_cuboid	28
5.3.2.4	mesh_create_mesh_new_cylinder	29
5.3.2.5	mesh_create_mesh_new_ellipsoid	29
5.3.2.6	mesh_free_mesh	30
5.4	meshdraw.c File Reference	30
5.4.1	Detailed Description	31
5.4.2	Function Documentation	31
5.4.2.1	mesh_draw_mesh	31
5.5	mesherror.c File Reference	32
5.5.1	Detailed Description	32
5.5.2	Function Documentation	33
5.5.2.1	mesh_error	33
5.6	meshfilter.c File Reference	34

5.6.1	Detailed Description	35
5.6.2	Function Documentation	35
5.6.2.1	mesh_bilateral_filter	35
5.6.2.2	mesh_laplacian_filter	36
5.6.2.3	mesh_restricted_laplacian_filter	36
5.7	meshlib.h File Reference	37
5.7.1	Detailed Description	41
5.7.2	Macro Definition Documentation	41
5.7.2.1	_CRT_SECURE_NO_DEPRECATED	41
5.7.2.2	FLOATDATA	41
5.7.2.3	INTDATA	41
5.7.2.4	MESH_CLONE_ALL_PROPS	41
5.7.2.5	MESH_CLONE_F_ALL_PROPS	42
5.7.2.6	MESH_CLONE_FACES	42
5.7.2.7	MESH_CLONE_FAREAS	42
5.7.2.8	MESH_CLONE_FCOLORS	42
5.7.2.9	MESH_CLONE_FNORMALS	42
5.7.2.10	MESH_CLONE_V_ALL_PROPS	42
5.7.2.11	MESH_CLONE_VCOLORS	42
5.7.2.12	MESH_CLONE_VERTICES	42
5.7.2.13	MESH_CLONE_VFACES	42
5.7.2.14	MESH_CLONE_VNORMALS	42
5.7.2.15	MESH_ERR_FNOTOPEN	42
5.7.2.16	MESH_ERR_INCOMPATIBLE	42
5.7.2.17	MESH_ERR_MALLOC	43
5.7.2.18	MESH_ERR_SIZE_MISMATCH	43
5.7.2.19	MESH_ERR_UNKNOWN	43
5.7.2.20	MESH_FLOATDATA_TYPE	43
5.7.2.21	MESH_INTDATA_TYPE	43
5.7.2.22	MESH_ORIGIN_TYPE_BUILD	43
5.7.2.23	MESH_ORIGIN_TYPE_COFF	43
5.7.2.24	MESH_ORIGIN_TYPE_NCOFF	43
5.7.2.25	MESH_ORIGIN_TYPE_NOFF	43
5.7.2.26	MESH_ORIGIN_TYPE_OFF	43
5.7.2.27	MESH_ORIGIN_TYPE_PLY_ASCII	43
5.7.2.28	MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN	43
5.7.2.29	MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN	44
5.7.2.30	MESH_ORIGIN_TYPE_XYZ	44
5.7.2.31	MESH_PI	44
5.7.2.32	MESH_TWOPI	44

5.7.3	Typedef Documentation	44
5.7.3.1	FILEPOINTER	44
5.7.3.2	INTDATA2	44
5.7.3.3	mesh	44
5.7.3.4	MESH	44
5.7.3.5	mesh_color	44
5.7.3.6	MESH_COLOR	44
5.7.3.7	mesh_face	44
5.7.3.8	MESH_FACE	44
5.7.3.9	mesh_normal	45
5.7.3.10	MESH_NORMAL	45
5.7.3.11	mesh_rotation	45
5.7.3.12	MESH_ROTATION	45
5.7.3.13	mesh_struct	45
5.7.3.14	MESH_STRUCT	45
5.7.3.15	mesh_struct2	45
5.7.3.16	MESH_STRUCT2	45
5.7.3.17	mesh_transform	45
5.7.3.18	MESH_TRANSFORM	45
5.7.3.19	mesh_vector3	45
5.7.3.20	MESH_VECTOR3	45
5.7.3.21	mesh_vertex	46
5.7.3.22	MESH_VERTEX	46
5.7.3.23	mesh_vface	46
5.7.3.24	MESH_VFACE	46
5.7.4	Function Documentation	46
5.7.4.1	mesh_bilateral_filter	46
5.7.4.2	mesh_calc_face_normal	46
5.7.4.3	mesh_calc_face_normals	47
5.7.4.4	mesh_calc_triangle_area	48
5.7.4.5	mesh_calc_vertex_adjacency	49
5.7.4.6	mesh_calc_vertex_normals	49
5.7.4.7	mesh_clone_mesh	50
5.7.4.8	mesh_combine_mesh	51
5.7.4.9	mesh_count_words_in_line	52
5.7.4.10	mesh_create_mesh_new	52
5.7.4.11	mesh_create_mesh_new_cone	53
5.7.4.12	mesh_create_mesh_new_cuboid	53
5.7.4.13	mesh_create_mesh_new_cylinder	54
5.7.4.14	mesh_create_mesh_new_ellipsoid	54

5.7.4.15	mesh_cross_normal	55
5.7.4.16	mesh_cross_vector3	55
5.7.4.17	mesh_draw_mesh	55
5.7.4.18	mesh_error	56
5.7.4.19	mesh_find	57
5.7.4.20	mesh_find2	58
5.7.4.21	mesh_free_mesh	58
5.7.4.22	mesh_go_next_word	58
5.7.4.23	mesh_isnumeric	58
5.7.4.24	mesh_laplacian_filter	59
5.7.4.25	mesh_load_file	59
5.7.4.26	mesh_load_off	60
5.7.4.27	mesh_load_ply	61
5.7.4.28	mesh_load_xyz	61
5.7.4.29	mesh_read_word	62
5.7.4.30	mesh_remove_boundary_faces	62
5.7.4.31	mesh_remove_boundary_vertices	63
5.7.4.32	mesh_remove_close_vertices	63
5.7.4.33	mesh_remove_ear_faces	63
5.7.4.34	mesh_remove_triangles_with_small_area	64
5.7.4.35	mesh_remove_unreferenced_vertices	64
5.7.4.36	mesh_remove_zero_area_faces	65
5.7.4.37	mesh_restricted_laplacian_filter	66
5.7.4.38	mesh_rotate	66
5.7.4.39	mesh_rotation_create	67
5.7.4.40	mesh_rotation_free	67
5.7.4.41	mesh_rotation_set_angleaxis	68
5.7.4.42	mesh_rotation_set_matrix	68
5.7.4.43	mesh_scale	69
5.7.4.44	mesh_skip_line	69
5.7.4.45	mesh_translate	69
5.7.4.46	mesh_translate_vector	69
5.7.4.47	mesh_upsample	70
5.7.4.48	mesh_vertex_rotate	70
5.7.4.49	mesh_write_file	71
5.7.4.50	mesh_write_off	71
5.7.4.51	mesh_write_ply	72
5.7.4.52	mesh_write_xyz	73
5.8	meshload.c File Reference	73
5.8.1	Detailed Description	74

5.8.2	Function Documentation	74
5.8.2.1	mesh_load_file	74
5.8.2.2	mesh_load_off	75
5.8.2.3	mesh_load_ply	76
5.8.2.4	mesh_load_xyz	76
5.9	meshops.c File Reference	77
5.9.1	Detailed Description	78
5.9.2	Function Documentation	78
5.9.2.1	mesh_clone_mesh	78
5.9.2.2	mesh_combine_mesh	79
5.10	meshtext.c File Reference	79
5.10.1	Detailed Description	80
5.10.2	Function Documentation	80
5.10.2.1	mesh_count_words_in_line	80
5.10.2.2	mesh_go_next_word	81
5.10.2.3	mesh_isnumeric	81
5.10.2.4	mesh_read_word	81
5.10.2.5	mesh_skip_line	81
5.11	meshtransform.c File Reference	82
5.11.1	Detailed Description	82
5.11.2	Function Documentation	83
5.11.2.1	mesh_rotate	83
5.11.2.2	mesh_rotation_create	83
5.11.2.3	mesh_rotation_free	84
5.11.2.4	mesh_rotation_set_angleaxis	84
5.11.2.5	mesh_rotation_set_matrix	85
5.11.2.6	mesh_scale	85
5.11.2.7	mesh_translate	86
5.11.2.8	mesh_translate_vector	86
5.11.2.9	mesh_vertex_rotate	86
5.12	meshwrite.c File Reference	87
5.12.1	Detailed Description	87
5.12.2	Function Documentation	88
5.12.2.1	mesh_write_file	88
5.12.2.2	mesh_write_off	88
5.12.2.3	mesh_write_ply	89
5.12.2.4	mesh_write_xyz	90

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:

mesh	7
mesh_color	10
mesh_face	10
mesh_rotation	11
mesh_struct	11
mesh_struct2	12
mesh_transform	12
mesh_vector3	12
mesh_vface	13

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

meshcalc.c	This file contains functions pertaining to different mesh computations	15
meshclean.c	This file contains functions pertaining to different mesh cleaning algorithms	21
meshcreate.c	This file contains functions pertaining to mesh creation and freeing	26
meshdraw.c	This file contains functions pertaining to mesh drawing in OpenGL	30
mesherror.c	This file contains functions pertaining to handling errors	32
meshfilter.c	This file contains functions pertaining to different mesh filtering algorithms	34
meshlib.h	This header file contains declarations of all functions of meshlib	37
meshload.c	This file contains functions pertaining to loading different mesh file types	73
meshops.c	This file contains functions pertaining to mesh combinatorial operations	77
meshtext.c	This file contains functions pertaining to different text routines	79
meshtransform.c	This file contains functions pertaining to different mesh transformations	82
meshwrite.c	This file contains functions pertaining to writing different mesh file types	87

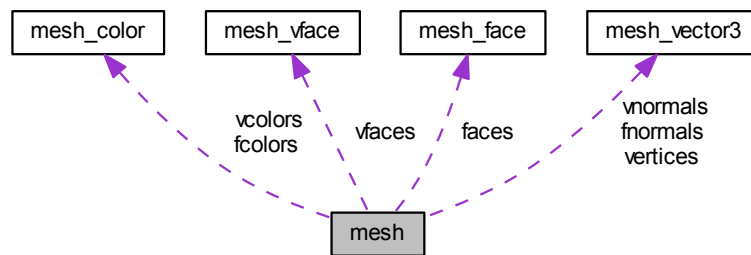
Chapter 4

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_vnormals`
- `uint8_t is_fnormals`
- `uint8_t is_vcolors`
- `uint8_t is_fcolors`
- `uint8_t is_vfaces`
- `uint8_t is_fareas`
- `INTDATA num_vertices`
- `INTDATA num_faces`
- `MESH_VERTEX vertices`
- `MESH_FACE faces`
- `MESH_NORMAL vnormals`
- `MESH_NORMAL fnormals`

- [MESH_COLOR](#) vcolors
- [MESH_COLOR](#) fcolors
- [MESH_VFACE](#) vfaces
- [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_FACE](#) faces

Pointer to faces

4.1.1.3 [FLOATDATA](#)* fareas

Pointer to face areas

4.1.1.4 [MESH_COLOR](#) fcolors

Pointer to face colors

4.1.1.5 [MESH_NORMAL](#) fnormals

Pointer to face normals

4.1.1.6 [uint8_t](#) is_faces

Has faces?

4.1.1.7 [uint8_t](#) is_fareas

Has face areas?

4.1.1.8 [uint8_t](#) is_fcolors

Has face colors?

4.1.1.9 [uint8_t](#) is_fnormals

Has face normals?

4.1.1.10 [uint8_t](#) is_loaded

Is loaded?

4.1.1.11 [uint8_t](#) is_trimesh

Is trimesh?

4.1.1.12 `uint8_t is_vcolors`

Has vertex colors?

4.1.1.13 `uint8_t is_vertices`

Has vertices?

4.1.1.14 `uint8_t is_vfaces`

Has vertex adjacent faces?

4.1.1.15 `uint8_t is_vnormals`

Has vertex normals?

4.1.1.16 `INTDATA num_faces`

Number of faces

4.1.1.17 `INTDATA num_vertices`

Number of vertices

4.1.1.18 `uint8_t origin_type`

Origin type

4.1.1.19 `MESH_COLOR vcolors`

Pointer to vertex colors

4.1.1.20 `MESH_VERTEX vertices`

Pointer to vertices

4.1.1.21 `MESH_VFACE vfaces`

Pointer to vertex adjacency faces

4.1.1.22 `MESH_NORMAL vnormals`

Pointer to vertex normals

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

- [meshlib.h](#)

4.2 mesh_color Struct Reference

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

Data Fields

- [FLOATDATA r](#)
- [FLOATDATA g](#)
- [FLOATDATA b](#)
- [FLOATDATA a](#)

4.2.1 Field Documentation

4.2.1.1 FLOATDATA a

Alpha channel

4.2.1.2 FLOATDATA b

Green channel

4.2.1.3 FLOATDATA g

Blue channel

4.2.1.4 FLOATDATA r

Red channel

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

- [meshlib.h](#)

4.3 mesh_face Struct Reference

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

Data Fields

- [INTDATA num_vertices](#)
- [INTDATA * vertices](#)

4.3.1 Field Documentation

4.3.1.1 INTDATA num_vertices

Number of vertices

4.3.1.2 INTDATA* vertices

Pointer to vertex indices

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

- [meshlib.h](#)

4.4 mesh_rotation Struct Reference

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

Data Fields

- [FLOATDATA data](#) [9]

4.4.1 Field Documentation

4.4.1.1 FLOATDATA data[9]

Matrix data

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

- [meshlib.h](#)

4.5 mesh_struct Struct Reference

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

Data Fields

- [INTDATA num_items](#)
- [INTDATA * items](#)

4.5.1 Field Documentation

4.5.1.1 INTDATA* items

Pointer to INTDATA items

4.5.1.2 INTDATA num_items

Number of items

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

- [meshlib.h](#)

4.6 mesh_struct2 Struct Reference

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

Data Fields

- [INTDATA num_items](#)
- [INTDATA2 * items](#)

4.6.1 Field Documentation

4.6.1.1 INTDATA2* items

Pointer to INTDATA2 items

4.6.1.2 INTDATA num_items

Number of items

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

- [meshlib.h](#)

4.7 mesh_transform Struct Reference

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

Data Fields

- [FLOATDATA * data](#)

4.7.1 Field Documentation

4.7.1.1 FLOATDATA* data

Matrix data

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

- [meshlib.h](#)

4.8 mesh_vector3 Struct Reference

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

Data Fields

- [FLOATDATA x](#)
- [FLOATDATA y](#)
- [FLOATDATA z](#)

4.8.1 Field Documentation

4.8.1.1 FLOATDATA x

x co-ordinate

4.8.1.2 FLOATDATA y

y co-ordinate

4.8.1.3 FLOATDATA z

z co-ordinate

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

- [meshlib.h](#)

4.9 mesh_vface Struct Reference

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

Data Fields

- [INTDATA num_faces](#)
- [INTDATA * faces](#)

4.9.1 Field Documentation

4.9.1.1 INTDATA* faces

Pointer to adjacent face indices

4.9.1.2 INTDATA num_faces

Number of adjacent faces

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

- [meshlib.h](#)

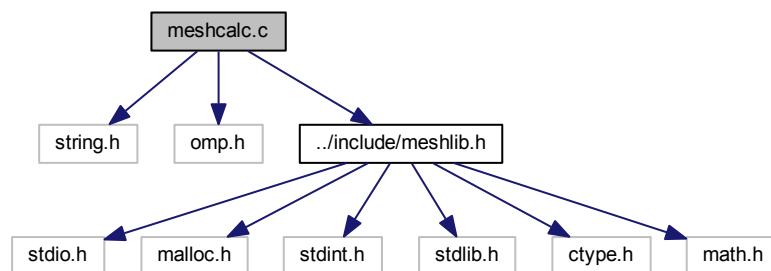
Chapter 5

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_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_vertex_adjacency` (`MESH` m)
Computes vertex 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.
- [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

1.4.0.0

Copyright

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

5.1.2 Function Documentation

5.1.2.1 `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

Here is the caller graph for this function:



5.1.2.2 int mesh_calc_face_normals (MESH *m*)

Computes face normals of a given mesh.

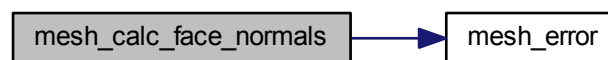
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

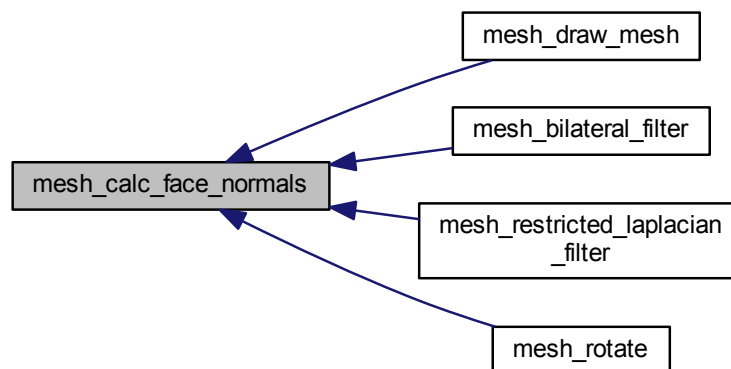
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

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

Computes area of a triangle.

Parameters

in	<i>a</i>	First vertex
in	<i>b</i>	Second vertex
in	<i>c</i>	Third vertex

Returns

Area

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.4 int mesh_calc_vertex_adjacency (MESH *m*)

Computes vertex adjacent faces of a given mesh.

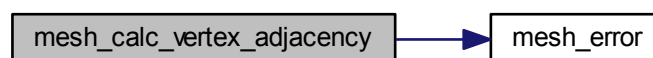
Parameters

<i>in</i>	<i>m</i>	Input mesh
-----------	----------	------------

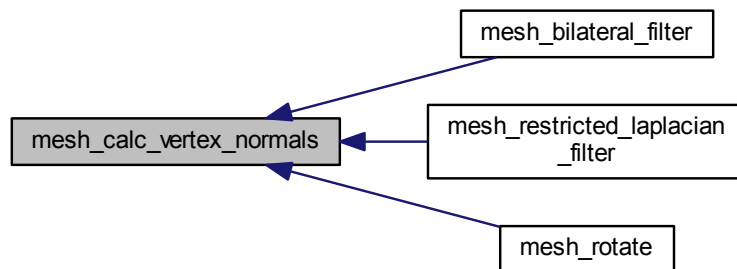
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.2.6 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

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

Computes the cross product of two 3-d vectors.

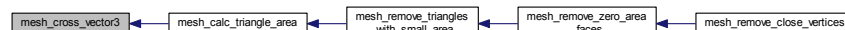
Parameters

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:



5.1.2.8 INTDATA mesh_find (MESH_STRUCT *s*, INTDATA *q*)

Finds an item in an INTDATA structure.

Parameters

in	<i>s</i>	Input INTDATA structure
in	<i>q</i>	Query INTDATA

Returns

Index or -1

5.1.2.9 INTDATA mesh_find2 (MESH_STRUCT2 *s*, INTDATA *q*)

Finds an item in an INTDATA2 structure.

Parameters

in	<i>s</i>	Input INTDATA2 structure
in	<i>q</i>	Query INTDATA2

Returns

Index or -1

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

Upsamples a given mesh.

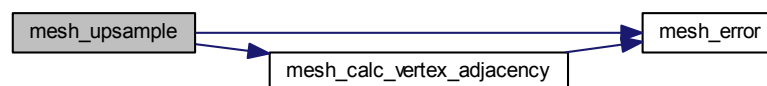
Parameters

in	<i>m</i>	Input mesh
in	<i>iters</i>	Number of iterations

Returns

Error code

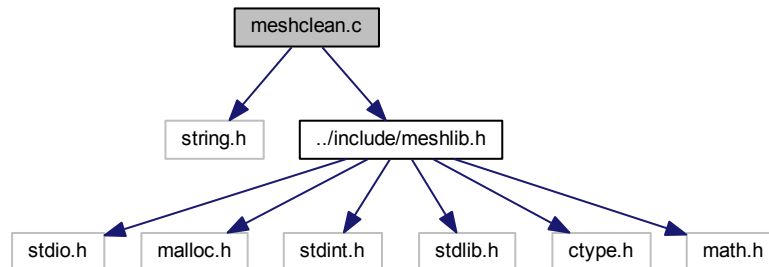
Here is the call graph for this function:



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.

5.2.1 Detailed Description

This file contains functions pertaining to different mesh cleaning algorithms.

Author

Sk. Mohammadul Haque

Version

1.4.0.0

Copyright

Copyright (c) 2013, 2014, 2015 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	<i>m</i>	Input mesh
in	<i>iters</i>	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	<i>m</i>	Input mesh
in	<i>iters</i>	Number of iterations

Returns

Error code

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

Removes close vertices.

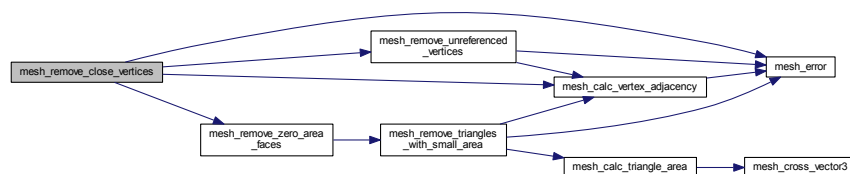
Parameters

in	<i>m</i>	Input mesh
in	<i>r</i>	Maximum distance between two vertices

Returns

Error code

Here is the call graph for this function:



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

Removes ear faces and connecting vertices.

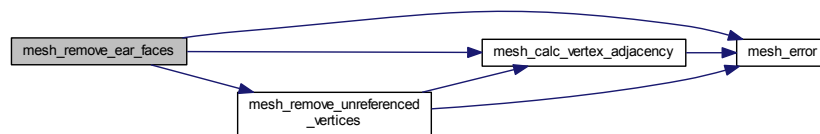
Parameters

in	<i>m</i>	Input mesh
in	<i>niters</i>	Number of iterations

Returns

Error code

Here is the call graph for this function:



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

Removes triangles with area smaller than a given value.

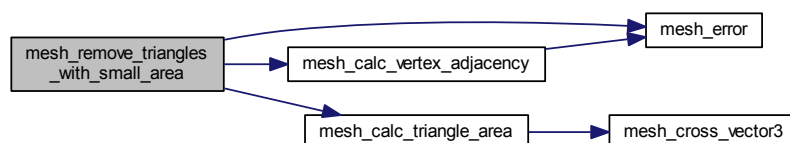
Parameters

in	<i>m</i>	Input mesh
in	<i>area</i>	Given area

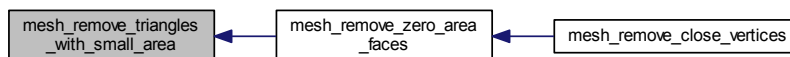
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.2.6 int mesh_remove_unreferenced_vertices (MESH *m*)

Removes unreferenced vertices.

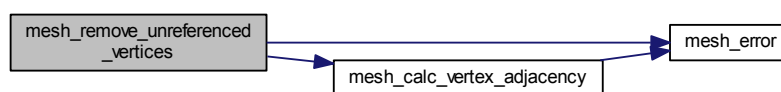
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

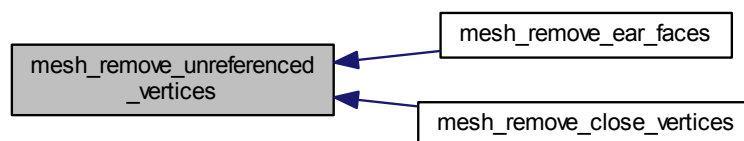
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.2.7 int mesh_remove_zero_area_faces (MESH *m*)

Removes triangles with zero area.

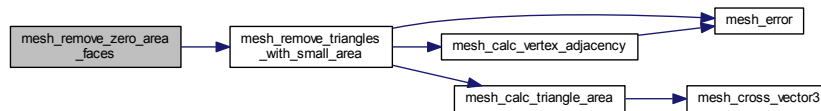
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

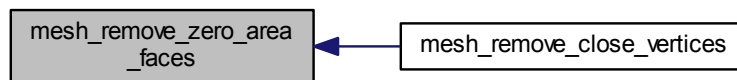
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

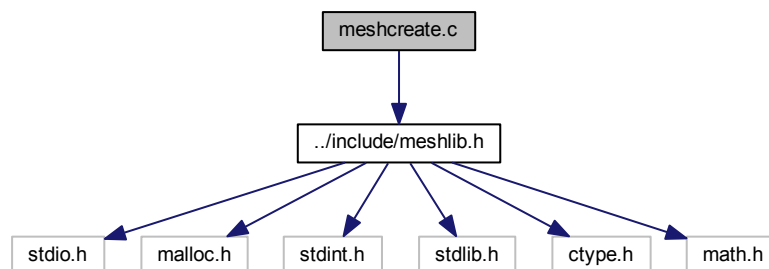


5.3 meshcreate.c File Reference

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_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

1.4.0.0

Copyright

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

5.3.2 Function Documentation

5.3.2.1 `MESH mesh_create_mesh_new ()`

Creates a new mesh.

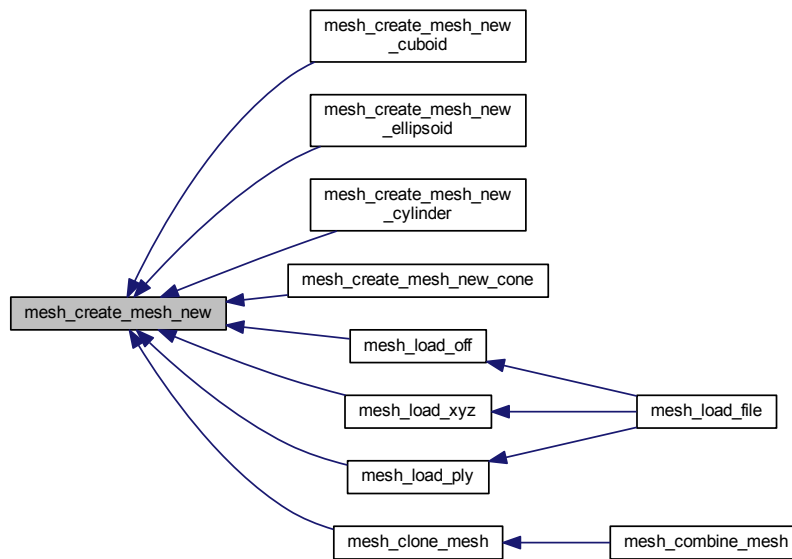
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.2 MESH mesh_create_mesh_new.cone (MESH_VECTOR3 sz, MESH_VECTOR3 pos)

Creates a cone mesh.

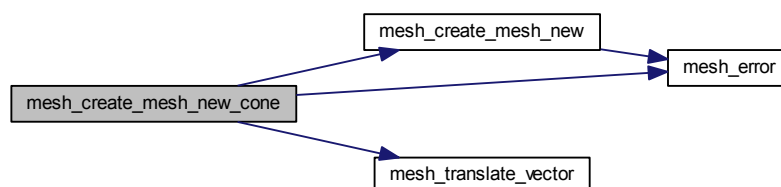
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	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.

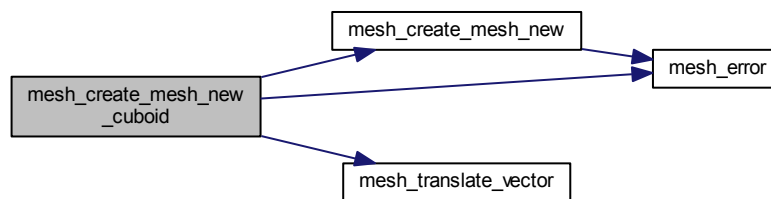
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	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.

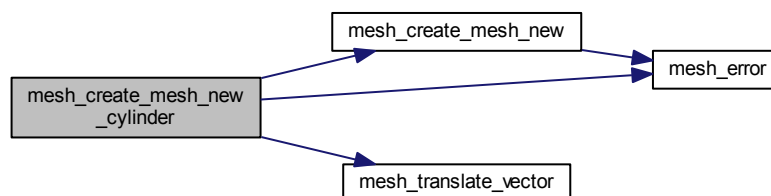
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	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.

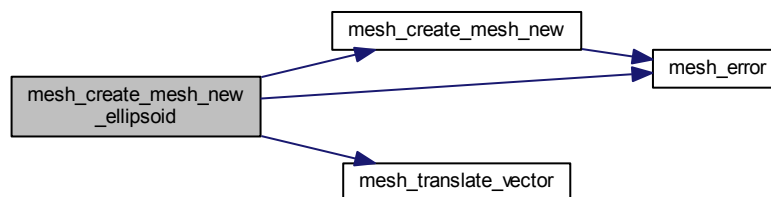
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	Position vector

Returns

Output mesh

Here is the call graph for this function:



5.3.2.6 void mesh_free_mesh (MESH m)

Frees a mesh.

Parameters

in	<i>m</i>	Input mesh
----	----------	------------

Returns

NULL

Here is the caller graph for this function:



5.4 meshdraw.c File Reference

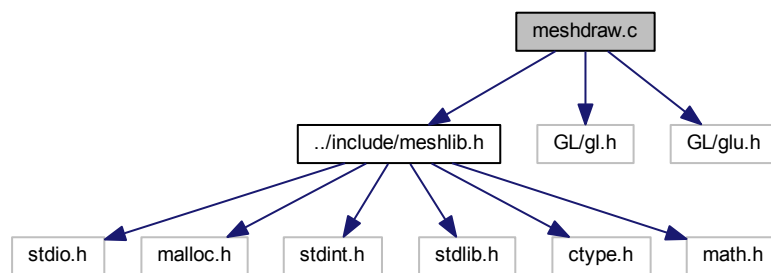
This file contains functions pertaining to mesh drawing in OpenGL.

```

#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.

5.4.1 Detailed Description

This file contains functions pertaining to mesh drawing in OpenGL.

Author

Sk. Mohammadul Haque

Version

1.4.0.0

Copyright

Copyright (c) 2013, 2014, 2015 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.

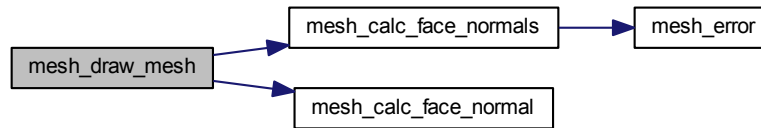
Parameters

<i>in</i>	<i>m</i>	Input mesh
-----------	----------	------------

Returns

NULL

Here is the call graph for this function:

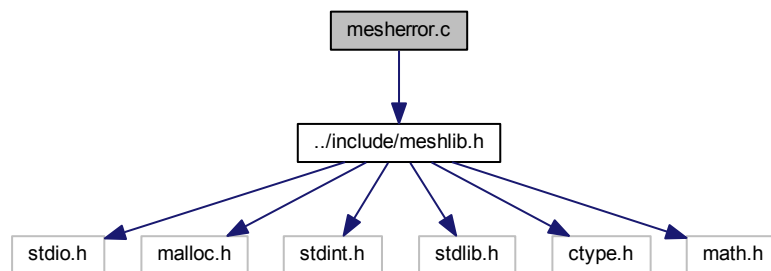


5.5 mesherror.c File Reference

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

1.4.0.0

Copyright

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

5.5.2 Function Documentation

5.5.2.1 void mesh_error (int *type*)

Displays error message and exits.

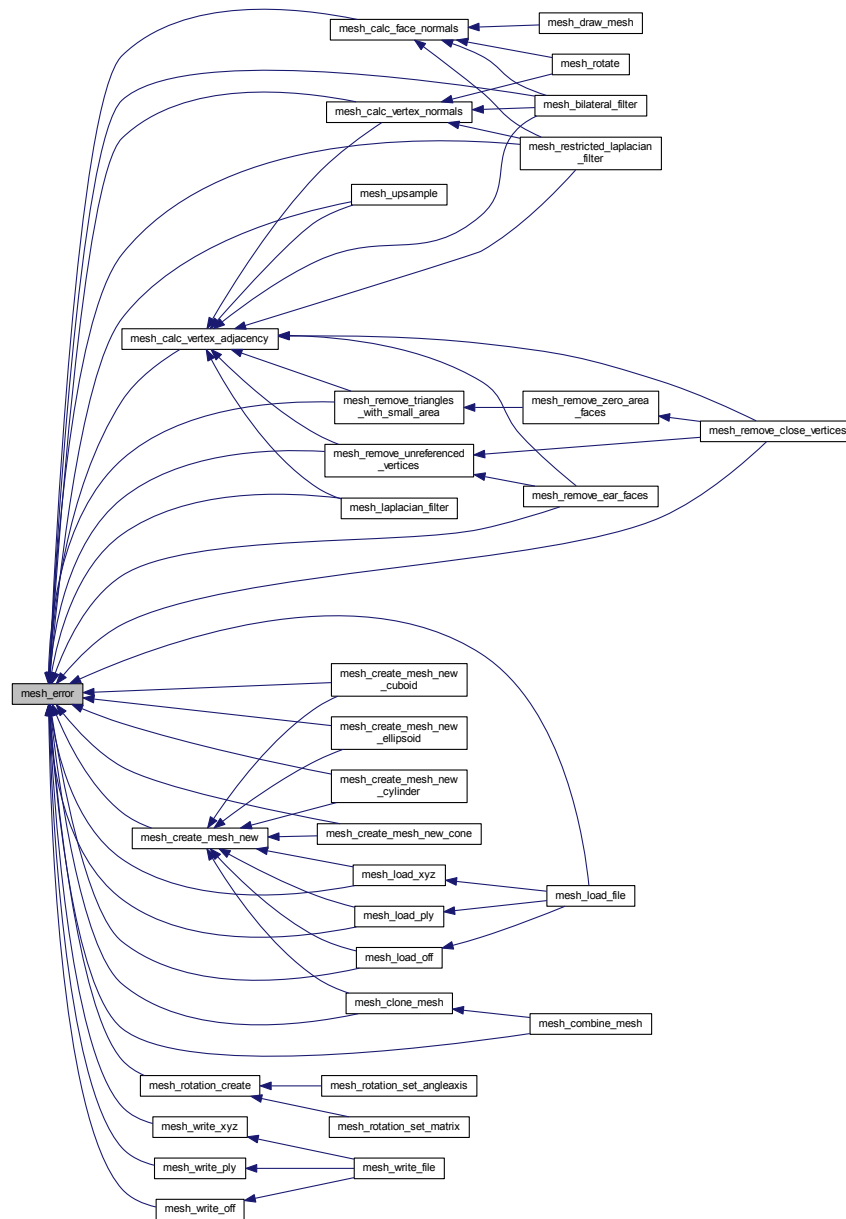
Parameters

<i>in</i>	<i>type</i>	Error type (MESH_ERR_MALLOC/MESH_ERR_SIZE_MISMATCH/MESH_ERR_FNOTOPEN)
-----------	-------------	---

Returns

NULL

Here is the caller graph for this function:

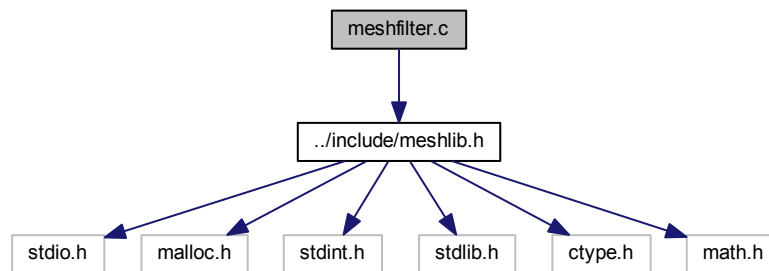


5.6 meshfilter.c File Reference

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

1.4.0.0

Copyright

Copyright (c) 2013, 2014, 2015 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.

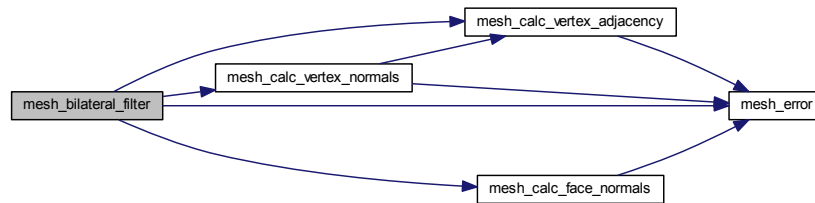
Parameters

in	<i>m</i>	Input mesh
in	<i>sigma_c</i>	Range standard deviation
in	<i>sigma_s</i>	Spatial standard deviation
in	<i>niters</i>	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.

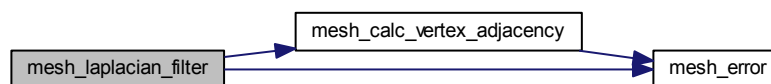
Parameters

in	<i>m</i>	Input mesh
in	<i>r</i>	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.

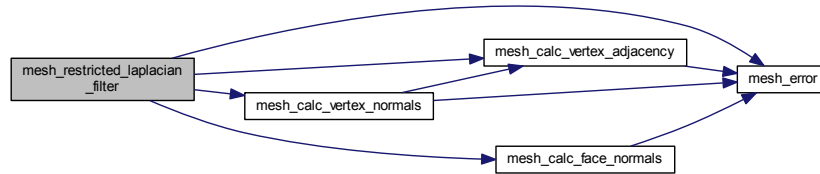
Parameters

in	<i>m</i>	Input mesh
in	<i>r</i>	Amount of diffusion
in	<i>ang</i>	Minimum angle in degrees to suppress filtering

Returns

Error code

Here is the call graph for this function:

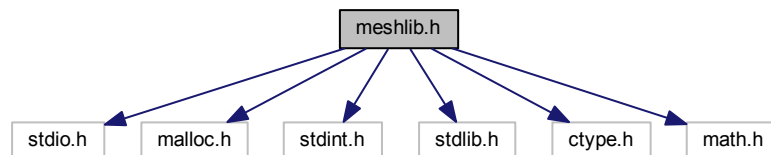


5.7 meshlib.h File Reference

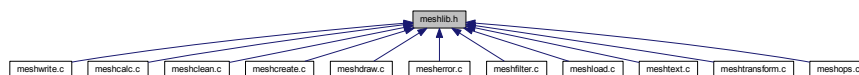
This header file contains declarations of all functions of meshlib.

```
#include <stdio.h>
#include <malloc.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_face](#)

- struct [mesh_vface](#)
- struct [mesh_rotation](#)
- struct [mesh_transform](#)
- struct [mesh](#)

Macros

- `#define _CRT_SECURE_NO_DEPRECATED`
- `#define MESH_INTDATA_TYPE 0`
- `#define MESH_FLOATDATA_TYPE 0`
- `#define INTDATA int32_t /* do not change this, careful see meshload fscanf and other functions */`
- `#define FLOATDATA float /* 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_F_ALL_PROPS (MESH_CLONE_FACES | __MESH_CLONE_F_ALL_PROPS)`
- `#define MESH_CLONE_ALL_PROPS (0xFF)`

Typedefs

- `typedef struct _iobuf * FILEPOINTER`
- `typedef INTDATA INTDATA2 [2]`
- `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_face` `mesh_face`
- typedef `mesh_face` * `MESH_FACE`
- typedef struct `mesh_vface` `mesh_vface`
- typedef `mesh_vface` * `MESH_VFACE`
- 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

- void `mesh_error` (int type)
Displays error message and exits.
- `MESH` `mesh_create_mesh_new` ()
Creates a new mesh.
- void `mesh_free_mesh` (`MESH` m)
Frees a 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.
- `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.
- `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.
- 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.
- 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_vertex_adjacency](#) (MESH m)
Computes vertex adjacent faces of a given mesh.
- int [mesh_upsample](#) (MESH m, int iters)
Upsamples a given mesh.
- 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.
- FLOATDATA [mesh_calc_triangle_area](#) (MESH_VERTEX a, MESH_VERTEX b, MESH_VERTEX c)
Computes area of a triangle.
- void [mesh_calc_face_normal](#) (MESH_VERTEX v1, MESH_VERTEX v2, MESH_VERTEX v3, MESH_NORMAL n)
Computes the face normal given 3 vertices.
- 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.
- 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_unreferenced_vertices](#) (MESH m)
Removes unreferenced vertices.
- int [mesh_remove_zero_area_faces](#) (MESH m)
Removes triangles with zero area.
- int [mesh_remove_close_vertices](#) (MESH m, FLOATDATA r)
Removes close vertices.
- int [mesh_remove_ear_faces](#) (MESH m, int niters)
Removes ear faces and connecting vertices.
- int [mesh_isnumeric](#) (FILEPOINTER fp)
Checks if numeric or not.
- int [mesh_go_next_word](#) (FILEPOINTER fp)
Points to the next word.
- int [mesh_read_word](#) (FILEPOINTER fp, char *c_word, int sz)
Reads current word.
- int [mesh_count_words_in_line](#) (FILEPOINTER fp, int *count)
Counts number of words in the current line.
- int [mesh_skip_line](#) (FILEPOINTER fp)
Skips to next line.
- 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.
- 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_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_VERTEX` 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.
- void `mesh_draw_mesh` (`MESH` m)
Draws a given mesh in OpenGL context.

5.7.1 Detailed Description

This header file contains declarations of all functions of meshlib.

Author

Sk. Mohammadul Haque

Version

1.4.0.0

Copyright

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

5.7.2 Macro Definition Documentation

5.7.2.1 `#define _CRT_SECURE_NO_DEPRECATED`

5.7.2.2 `#define FloatData float /* 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 (0xFF)`

Clone mesh all properties

5.7.2.5 `#define MESH_CLONE_F_ALL_PROPS (MESH_CLONE_FACES | __MESH_CLONE_F_ALL_PROPS)`

Clone mesh all face properties

5.7.2.6 `#define MESH_CLONE_FACES (MESH_CLONE_VERTICES | __MESH_CLONE_FACES)`

Clone mesh faces

5.7.2.7 `#define MESH_CLONE_FAREAS (MESH_CLONE_FACES | __MESH_CLONE_FAREAS)`

Clone mesh faces and face areas

5.7.2.8 `#define MESH_CLONE_FCOLORS (MESH_CLONE_FACES | __MESH_CLONE_FCOLORS)`

Clone mesh faces and face colors

5.7.2.9 `#define MESH_CLONE_FNORMALS (MESH_CLONE_FACES | __MESH_CLONE_FNORMALS)`

Clone mesh faces and face normals

5.7.2.10 `#define MESH_CLONE_V_ALL_PROPS (0x0F)`

Clone mesh all vertex properties

5.7.2.11 `#define MESH_CLONE_VCOLORS (MESH_CLONE_VERTICES | __MESH_CLONE_VCOLORS)`

Clone mesh vertices and vertex colors

5.7.2.12 `#define MESH_CLONE_VERTICES (0x01)`

Clone mesh vertices

5.7.2.13 `#define MESH_CLONE_VFACES (MESH_CLONE_VERTICES | __MESH_CLONE_VFACES)`

Clone mesh vertices and vertex face adjacency

5.7.2.14 `#define MESH_CLONE_VNORMALS (MESH_CLONE_VERTICES | __MESH_CLONE_VNORMALS)`

Clone mesh vertices and vertex normals

5.7.2.15 `#define MESH_ERR_FNOTOPEN 2`

Mesh error type - file open

5.7.2.16 `#define MESH_ERR_INCOMPATIBLE 3`

Mesh error type - incompatible data

5.7.2.17 #define MESH_ERR_MALLOC 0

Mesh error type - allocation

5.7.2.18 #define MESH_ERR_SIZE_MISMATCH 1

Mesh error type - size mismatch

5.7.2.19 #define MESH_ERR_UNKNOWN 4

Mesh error type - unknown

5.7.2.20 #define MESH_FLOATDATA_TYPE 0

Float datatype selector

5.7.2.21 #define MESH_INTDATA_TYPE 0

Integer datatype selector

5.7.2.22 #define MESH_ORIGIN_TYPE_BUILD 00

Mesh origin type - create new

5.7.2.23 #define MESH_ORIGIN_TYPE_COFF 13

Mesh origin type - COFF file

5.7.2.24 #define MESH_ORIGIN_TYPE_NCOFF 14

Mesh origin type - NCOFF file

5.7.2.25 #define MESH_ORIGIN_TYPE_NOFF 12

Mesh origin type - NOFF file

5.7.2.26 #define MESH_ORIGIN_TYPE_OFF 11

Mesh origin type - OFF file

5.7.2.27 #define MESH_ORIGIN_TYPE_PLY_ASCII 30

Mesh origin type - PLY ascii file

5.7.2.28 #define MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN 32

Mesh origin type - PLY binary BE file

5.7.2.29 `#define MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN 31`

Mesh origin type - PLY binary LE file

5.7.2.30 `#define MESH_ORIGIN_TYPE_XYZ 20`

Mesh origin type - XYZ file

5.7.2.31 `#define MESH_PI (3.14159265359)`

π

5.7.2.32 `#define MESH_TWOPI (6.28318530718)`

2π

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 struct mesh mesh`

Mesh

5.7.3.4 `typedef mesh* MESH`

Pointer to mesh

5.7.3.5 `typedef struct mesh_color mesh_color`

5.7.3.6 `typedef mesh_color* MESH_COLOR`

Color

5.7.3.7 `typedef struct mesh_face mesh_face`

Face

5.7.3.8 `typedef mesh_face* MESH_FACE`

Pointer to face

5.7.3.9 `typedef mesh_vector3 mesh_normal`

Normal

5.7.3.10 `typedef mesh_normal* MESH_NORMAL`

Normal pointer

5.7.3.11 `typedef struct mesh_rotation mesh_rotation`

Rotation

5.7.3.12 `typedef mesh_rotation* MESH_ROTATION`

Pointer to rotation

5.7.3.13 `typedef struct mesh_struct mesh_struct`

INTDATA Structure

5.7.3.14 `typedef mesh_struct* MESH_STRUCT`

INTDATA Structure pointer

5.7.3.15 `typedef struct mesh_struct2 mesh_struct2`

INTDATA2 Structure

5.7.3.16 `typedef mesh_struct2* MESH_STRUCT2`

INTDATA2 Structure pointer

5.7.3.17 `typedef struct mesh_transform mesh_transform`

Transformation

5.7.3.18 `typedef mesh_transform* MESH_TRANSFORM`

Pointer to transformation

5.7.3.19 `typedef struct mesh_vector3 mesh_vector3`

Generic 3-d vector

5.7.3.20 `typedef mesh_vector3* MESH_VECTOR3`

Generic 3-d vector pointer

5.7.3.21 `typedef mesh_vector3 mesh_vertex`

Vertex

5.7.3.22 `typedef mesh_vertex* MESH_VERTEX`

Vertex pointer

5.7.3.23 `typedef struct mesh_vface mesh_vface`

Vertex adjacent faces

5.7.3.24 `typedef mesh_vface* MESH_VFACE`

Pointer to vertex adjacent faces

5.7.4 Function Documentation

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

Mesh bilateral filter.

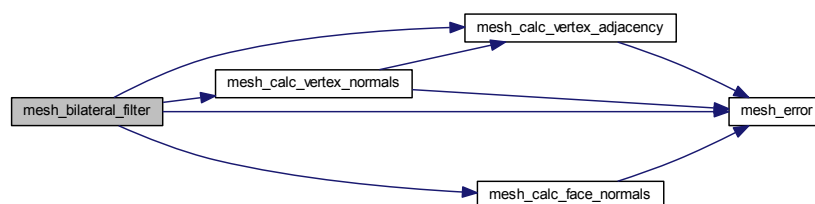
Parameters

<code>in</code>	<code>m</code>	Input mesh
<code>in</code>	<code>sigma_c</code>	Range standard deviation
<code>in</code>	<code>sigma_s</code>	Spatial standard deviation
<code>in</code>	<code>niters</code>	Number of iterations

Returns

Error code

Here is the call graph for this function:



5.7.4.2 `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

Here is the caller graph for this function:

5.7.4.3 int mesh_calc_face_normals (MESH m)

Computes face normals of a given mesh.

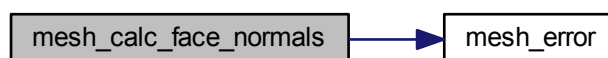
Parameters

in	m	Input mesh
----	-----	------------

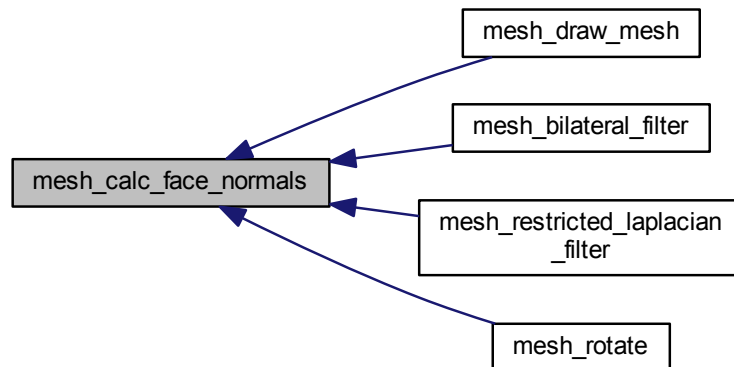
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.4 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:



5.7.4.5 int mesh_calc_vertex_adjacency (MESH *m*)

Computes vertex adjacent faces of a given mesh.

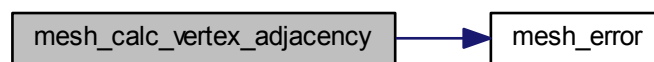
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

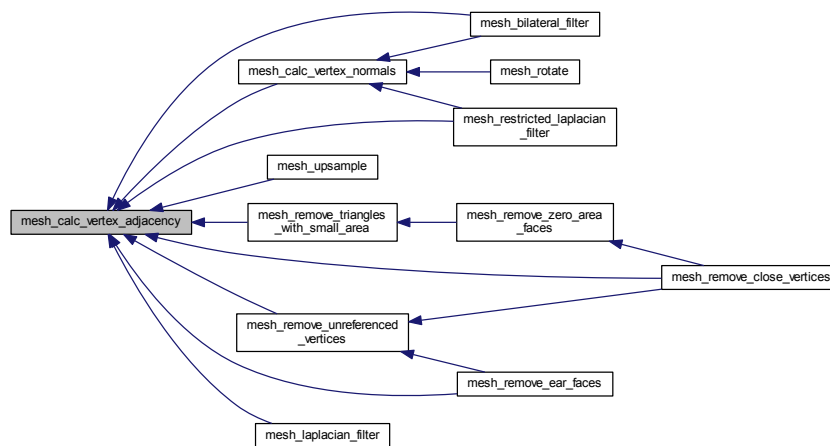
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.6 int mesh_calc_vertex_normals (MESH *m*)

Computes vertex normals of a given mesh.

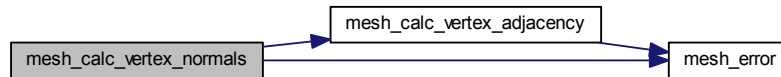
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

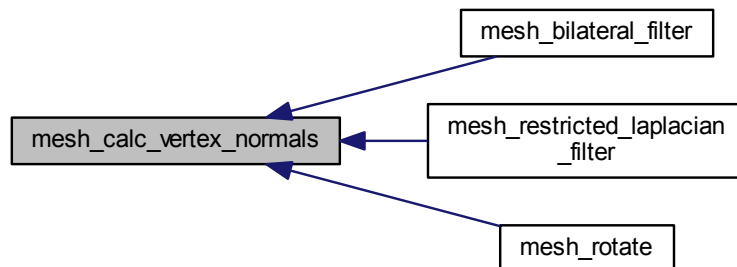
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.7 MESH mesh_clone_mesh (MESH m, uint16_t flags)

Clones a given mesh into another mesh.

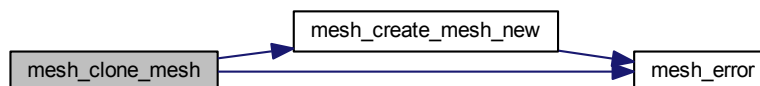
Parameters

in	<i>m</i>	Input mesh to clone
in	<i>flags</i>	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:



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

Combines a given mesh with another given mesh.

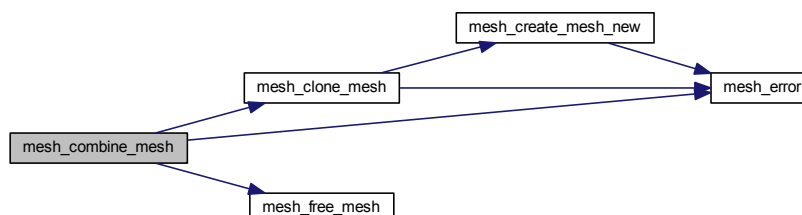
Parameters

in	<i>m1</i>	Input mesh to combine with
in	<i>m2</i>	Input mesh to combine

Returns

Output combined mesh

Here is the call graph for this function:



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

Counts number of words in the current line.

Parameters

in	<i>fp</i>	Pointer to input file
out	<i>count</i>	Count

Returns

Status 0 - Normal/ 1- EOF

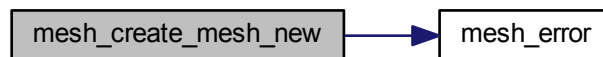
5.7.4.10 MESH mesh_create_mesh_new ()

Creates a new mesh.

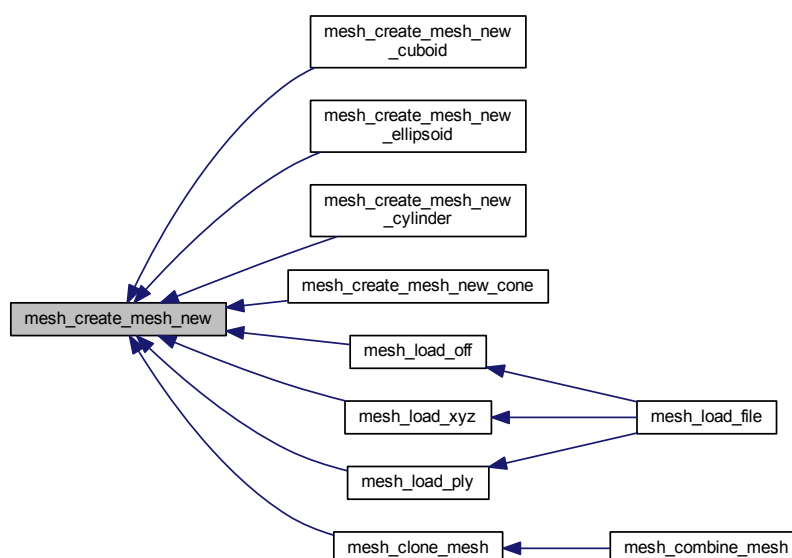
Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



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

Creates a cone mesh.

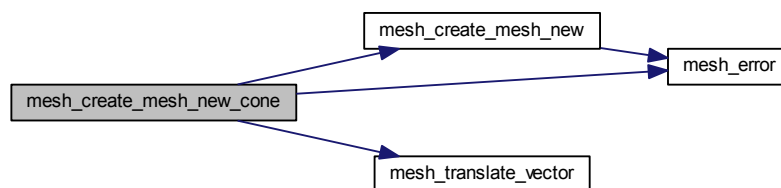
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	Position vector

Returns

Output mesh

Here is the call graph for this function:



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

Creates a cuboid mesh.

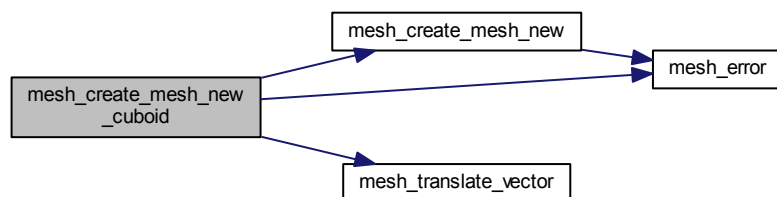
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	Position vector

Returns

Output mesh

Here is the call graph for this function:



5.7.4.13 MESH mesh.create_mesh_new.cylinder (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates a cylinder mesh.

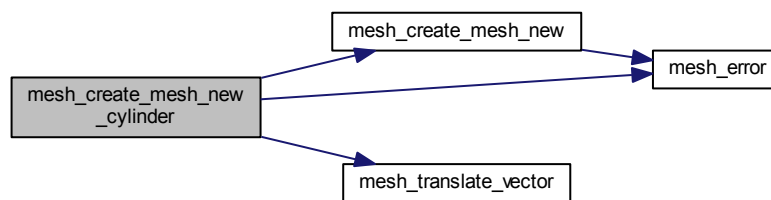
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	Position vector

Returns

Output mesh

Here is the call graph for this function:



5.7.4.14 MESH mesh.create_mesh_new.ellipsoid (MESH_VECTOR3 *sz*, MESH_VECTOR3 *pos*)

Creates an ellipsoid mesh.

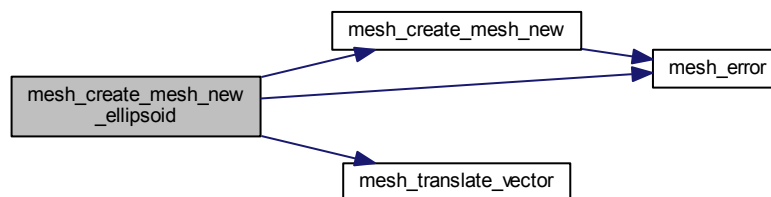
Parameters

in	<i>sz</i>	Size vector
in	<i>pos</i>	Position vector

Returns

Output mesh

Here is the call graph for this function:



5.7.4.15 void mesh_cross_normal (MESH_NORMAL *x*, MESH_NORMAL *y*, MESH_NORMAL *z*)

Computes the normalized cross product of two normals.

Parameters

in	<i>x</i>	First normal
in	<i>y</i>	Second normal
out	<i>z</i>	Output cross product $\frac{\mathbf{x} \times \mathbf{y}}{\ \mathbf{x} \times \mathbf{y}\ _2}$

Returns

NULL

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

Computes the cross product of two 3-d vectors.

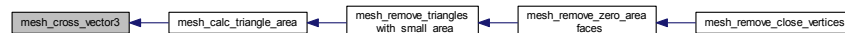
Parameters

in	<i>x</i>	First vector
in	<i>y</i>	Second vector
out	<i>z</i>	Output cross product $\mathbf{x} \times \mathbf{y}$

Returns

NULL

Here is the caller graph for this function:



5.7.4.17 void mesh_draw_mesh (MESH *m*)

Draws a given mesh in OpenGL context.

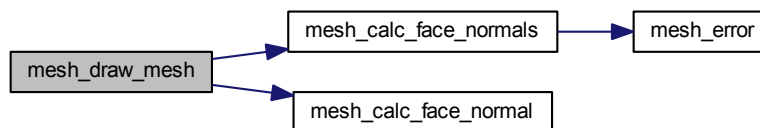
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

Returns

NULL

Here is the call graph for this function:



5.7.4.18 void mesh_error (int *type*)

Displays error message and exits.

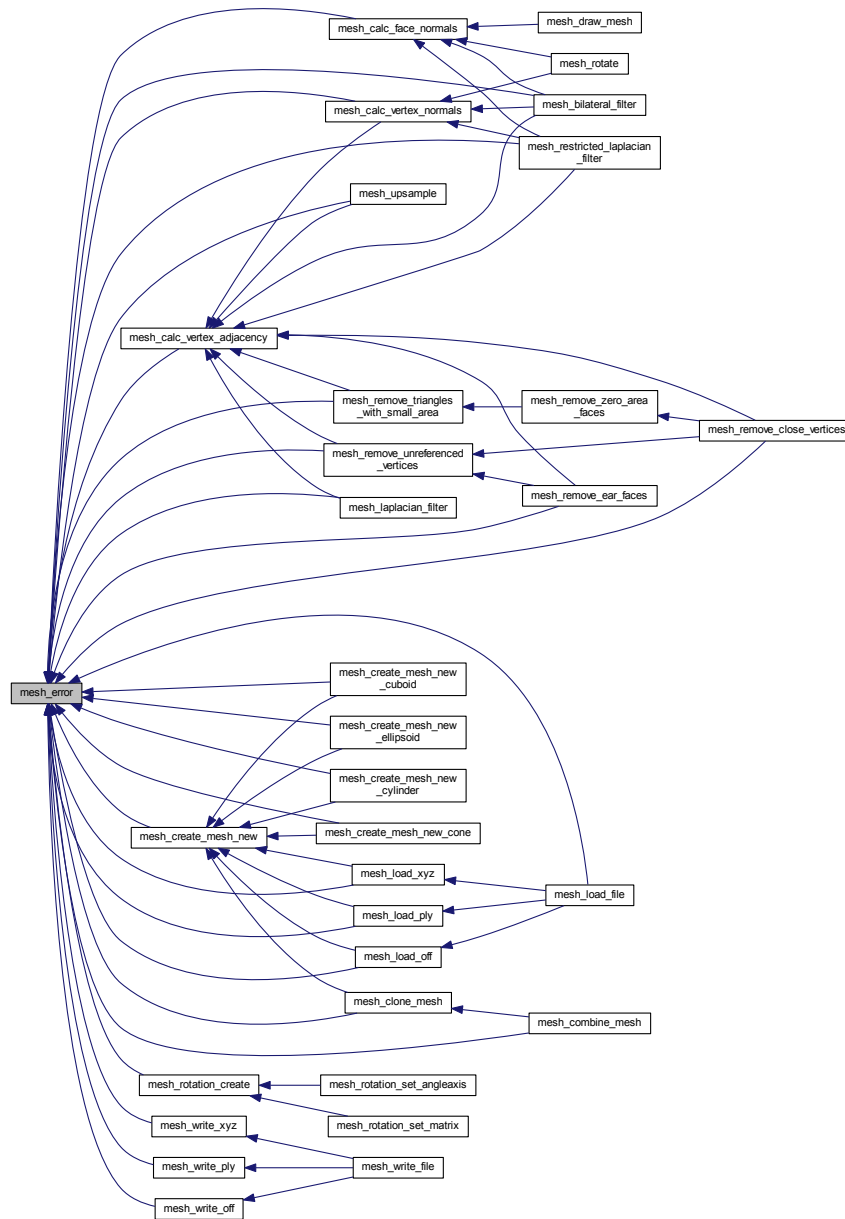
Parameters

<i>in</i>	<i>type</i>	
		Error type (MESH_ERR_MALLOC/MESH_ERR_SIZE_MISMATCH/MESH_ERR_FNOTOPEN)

Returns

NULL

Here is the caller graph for this function:



5.7.4.19 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

5.7.4.20 INTDATA mesh_find2 (MESH_STRUCT2 *s*, INTDATA *q*)

Finds an item in an INTDATA2 structure.

Parameters

<i>in</i>	<i>s</i>	Input INTDATA2 structure
<i>in</i>	<i>q</i>	Query INTDATA2

Returns

Index or -1

5.7.4.21 void mesh_free_mesh (MESH *m*)

Frees a mesh.

Parameters

<i>in</i>	<i>m</i>	Input mesh
-----------	----------	------------

Returns

NULL

Here is the caller graph for this function:



5.7.4.22 int mesh_go_next_word (FILEPOINTER *fp*)

Points to the next word.

Parameters

<i>in</i>	<i>fp</i>	Pointer to input file
-----------	-----------	-----------------------

Returns

Status 0 - Normal/ 1- EOF

5.7.4.23 int mesh_isnumeric (FILEPOINTER *fp*)

Checks if numeric or not.

Parameters

<i>in</i>	<i>fp</i>	Pointer to input file
-----------	-----------	-----------------------

Returns

1 for numeric/ else - for non-numeric

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

Mesh Laplacian filter.

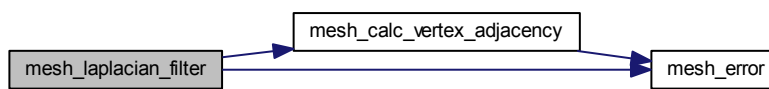
Parameters

<i>in</i>	<i>m</i>	Input mesh
<i>in</i>	<i>r</i>	Amount of diffusion

Returns

Error code

Here is the call graph for this function:

5.7.4.25 MESH mesh_load_file (const char * *fname*)

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

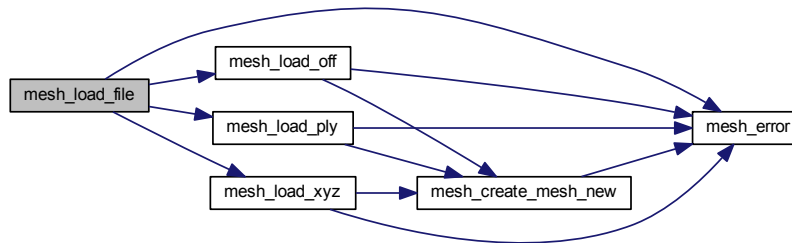
Parameters

<i>in</i>	<i>fname</i>	Input filename
-----------	--------------	----------------

Returns

Output mesh

Here is the call graph for this function:



5.7.4.26 MESH mesh.load_off (const char * fname)

Reads a mesh from an OFF file.

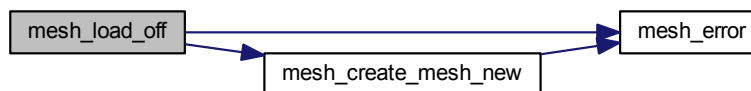
Parameters

in	fname	Input filename
----	-------	----------------

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.27 MESH mesh.load_ply (const char * *fname*)

Reads a mesh from a PLY file.

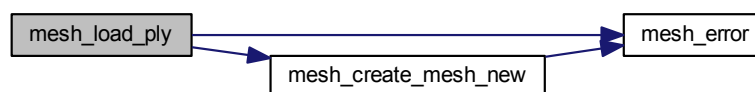
Parameters

in	<i>fname</i>	Input filename
----	--------------	----------------

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.28 MESH mesh.load_xyz (const char * *fname*)

Read a mesh from an ASC/XYZ file.

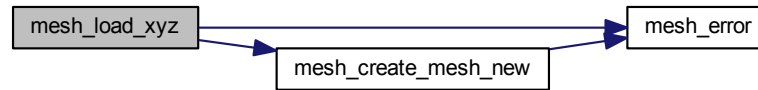
Parameters

in	<i>fname</i>	Input filename
----	--------------	----------------

Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



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

Reads current word.

Parameters

in	<i>fp</i>	Pointer to input file
out	<i>c_word</i>	Variable to store the word
in	<i>sz</i>	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF

5.7.4.30 int mesh_remove_boundary_faces (MESH *m*, int *iters*)

Removes boundary faces and connecting elements.

Parameters

in	<i>m</i>	Input mesh
in	<i>iters</i>	Number of iterations

Returns

Error code

5.7.4.31 int mesh_remove_boundary_vertices (MESH *m*, int *iters*)

Removes boundary vertices and connecting elements.

Parameters

in	<i>m</i>	Input mesh
in	<i>iters</i>	Number of iterations

Returns

Error code

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

Removes close vertices.

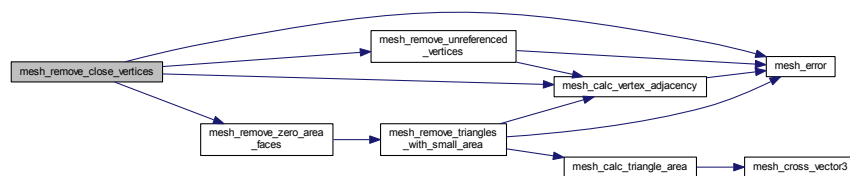
Parameters

in	<i>m</i>	Input mesh
in	<i>r</i>	Maximum distance between two vertices

Returns

Error code

Here is the call graph for this function:

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

Removes ear faces and connecting vertices.

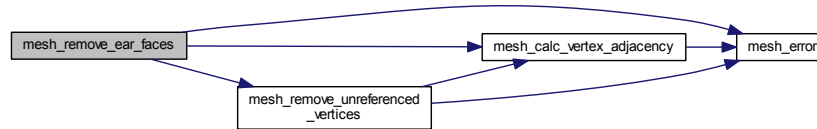
Parameters

in	<i>m</i>	Input mesh
in	<i>niters</i>	Number of iterations

Returns

Error code

Here is the call graph for this function:



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

Removes triangles with area smaller than a given value.

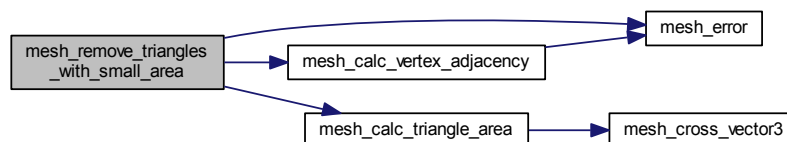
Parameters

in	<i>m</i>	Input mesh
in	<i>area</i>	Given area

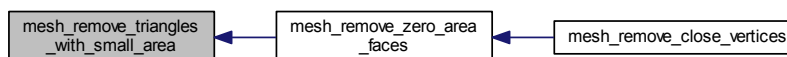
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.35 int mesh_remove_unreferenced_vertices (MESH *m*)

Removes unreferenced vertices.

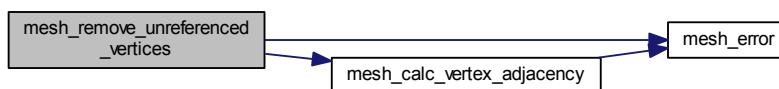
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

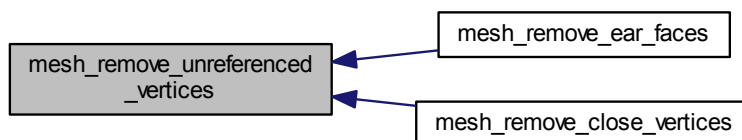
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

5.7.4.36 int mesh_remove_zero_area_faces (MESH *m*)

Removes triangles with zero area.

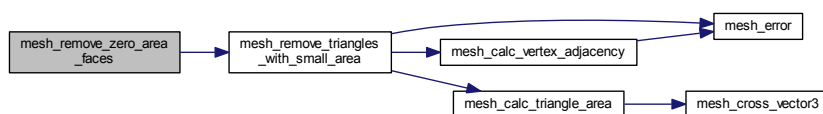
Parameters

in	<i>m</i>	Input mesh
----	----------	------------

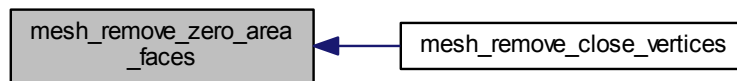
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



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

Restricted Mesh Laplacian filter.

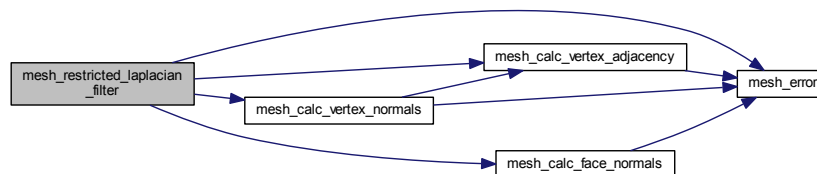
Parameters

in	<i>m</i>	Input mesh
in	<i>r</i>	Amount of diffusion
in	<i>ang</i>	Minimum angle in degrees to suppress filtering

Returns

Error code

Here is the call graph for this function:



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

Rotates a mesh by a given rotation.

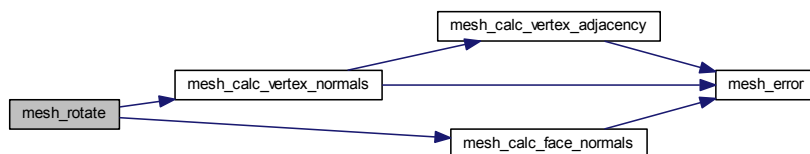
Parameters

in	<i>m</i>	Input vertex
in	<i>r</i>	Input rotation

Returns

Error code

Here is the call graph for this function:



5.7.4.39 MESH_ROTATION mesh_rotation_create ()

Creates a new rotation.

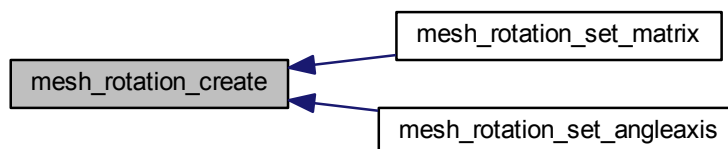
Returns

Output rotation

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.40 void mesh_rotation_free (MESH_ROTATION r)

Frees a given rotation.

Parameters

	<i>r</i>	Input rotation
--	----------	----------------

Returns

NULL

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

Sets rotation from angle axis.

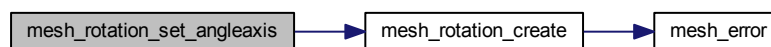
Parameters

in	<i>ang</i>	Input angle of rotation
out	<i>axis</i>	Input axis of rotation
out	<i>r</i>	Input rotation

Returns

Output rotation

Here is the call graph for this function:



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

Sets rotation from a matrix.

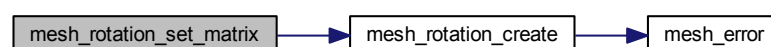
Parameters

in	<i>mat</i>	Input matrix
out	<i>r</i>	Input rotation

Returns

Output rotation

Here is the call graph for this function:



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

Scales a mesh by x, y and z amounts.

Parameters

<code>in</code>	<code><i>m</i></code>	Input mesh
<code>in</code>	<code><i>sx</i></code>	X component
<code>in</code>	<code><i>sy</i></code>	Y component
<code>in</code>	<code><i>sz</i></code>	Z component

Returns

Error code

5.7.4.44 `int mesh_skip_line (FILEPOINTER fp)`

Skips to next line.

Parameters

<code>in</code>	<code><i>fp</i></code>	Pointer to input file
-----------------	------------------------	-----------------------

Returns

Status 0 - Normal/ 1- EOF

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

Translates a mesh by x, y and z amounts.

Parameters

<code>in</code>	<code><i>m</i></code>	Input mesh
<code>in</code>	<code><i>x</i></code>	X component
<code>in</code>	<code><i>y</i></code>	Y component
<code>in</code>	<code><i>z</i></code>	Z component

Returns

Error code

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

Translates a mesh by a given 3-d vector.

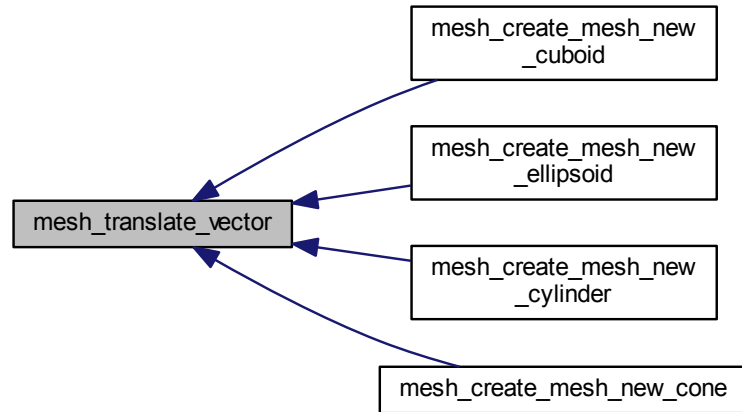
Parameters

<code>in</code>	<code><i>m</i></code>	Input mesh
<code>in</code>	<code><i>v</i></code>	Input vector

Returns

Error code

Here is the caller graph for this function:



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

Upsamples a given mesh.

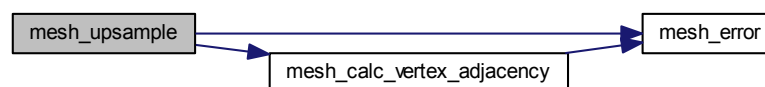
Parameters

<code>in</code>	<code><i>m</i></code>	Input mesh
<code>in</code>	<code><i>iters</i></code>	Number of iterations

Returns

Error code

Here is the call graph for this function:



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

Rotates a vertex by a given rotation.

Parameters

in	<i>v</i>	Input vertex
in	<i>r</i>	Input rotation

Returns

Output vertex

5.7.4.49 int mesh_write_file (MESH *m*, const char * *fname*)

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

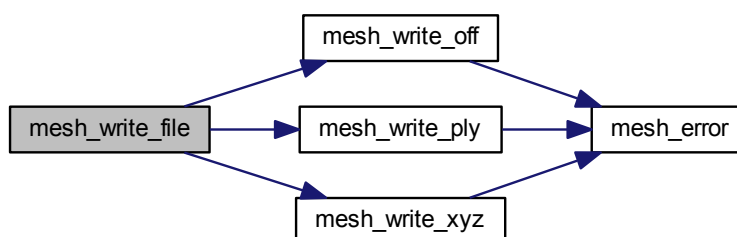
Parameters

in	<i>m</i>	Input mesh
in	<i>fname</i>	Output filename

Returns

Error code

Here is the call graph for this function:

5.7.4.50 int mesh_write_off (MESH *m*, const char * *fname*)

Write a mesh to an OFF file.

Parameters

in	<i>m</i>	Input mesh
in	<i>fname</i>	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.51 int mesh_write_ply (MESH *m*, const char * *fname*)

Write a mesh to an PLY file.

Parameters

in	<i>m</i>	Input mesh
in	<i>fname</i>	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



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

Write a mesh to an XYZ file.

Parameters

in	<i>m</i>	Input mesh
in	<i>fname</i>	Output filename

Returns

Error code

Here is the call graph for this function:



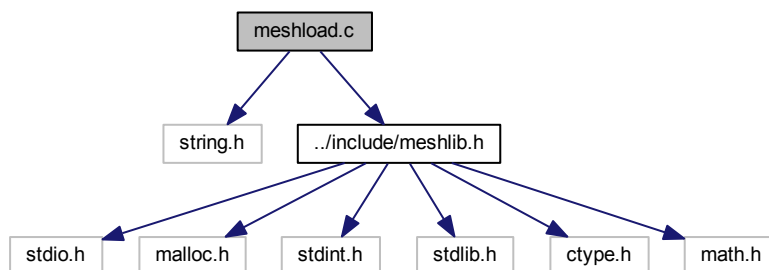
Here is the caller graph for this function:



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

1.4.0.0

Copyright

Copyright (c) 2013, 2014, 2015 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.

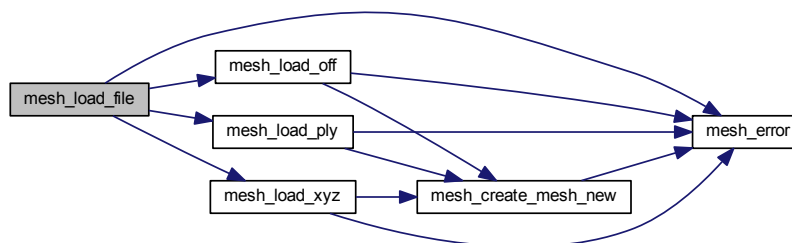
Parameters

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.

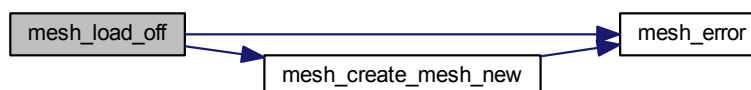
Parameters

in	<i>fname</i>	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.

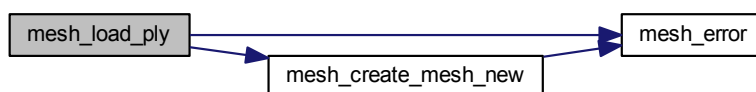
Parameters

in	<i>fname</i>	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.

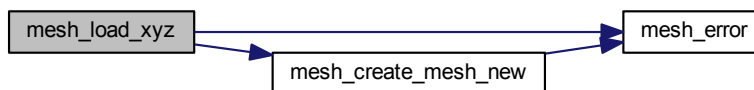
Parameters

in	<i>fname</i>	Input filename
----	--------------	----------------

Returns

Output mesh

Here is the call graph for this function:



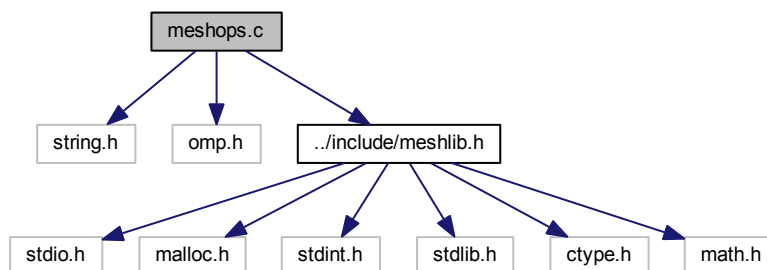
Here is the caller graph for this function:



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

1.4.0.0

Copyright

Copyright (c) 2013, 2014, 2015 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.

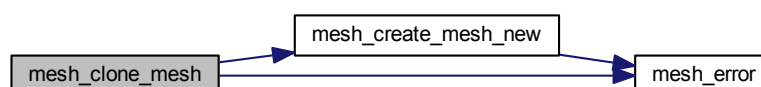
Parameters

in	<i>m</i>	Input mesh to clone
in	<i>flags</i>	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:



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

Combines a given mesh with another given mesh.

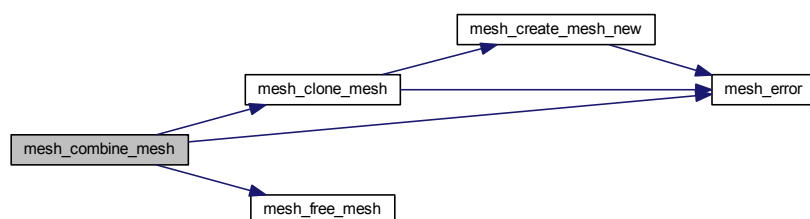
Parameters

in	<i>m1</i>	Input mesh to combine with
in	<i>m2</i>	Input mesh to combine

Returns

Output combined mesh

Here is the call graph for this function:

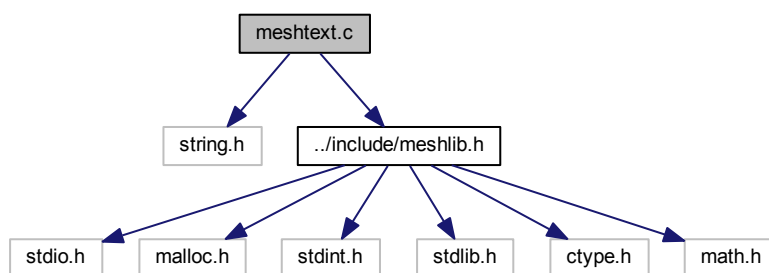


5.10 meshtext.c File Reference

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.
- 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

1.4.0.0

Copyright

Copyright (c) 2013, 2014, 2015 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

<i>in</i>	<i>fp</i>	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

<i>in</i>	<i>fp</i>	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.

Parameters

<i>in</i>	<i>fp</i>	Pointer to input file
<i>out</i>	<i>c_word</i>	Variable to store the word
<i>in</i>	<i>sz</i>	Maximum size to read

Returns

Status 0 - Normal/ 1- EOF

5.10.2.5 int mesh_skip_line (FILEPOINTER *fp*)

Skips to next line.

Parameters

<i>in</i>	<i>fp</i>	Pointer to input file
-----------	-----------	-----------------------

Returns

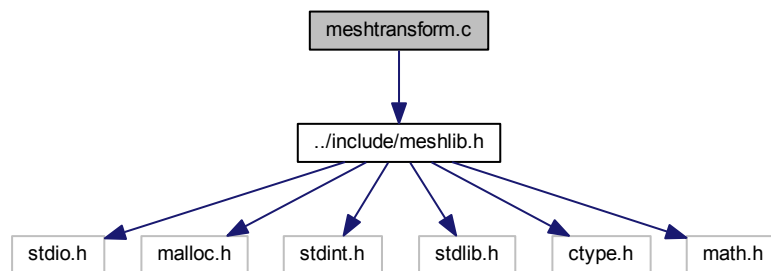
Status 0 - Normal/ 1- EOF

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 \(FLOATDATA ang, MESH_NORMAL axis, MESH_ROTATION r\)](#)
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.

5.11.1 Detailed Description

This file contains functions pertaining to different mesh transformations.

Author

Sk. Mohammadul Haque

Version

1.4.0.0

Copyright

Copyright (c) 2013, 2014, 2015 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.

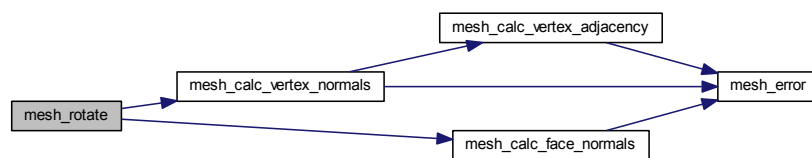
Parameters

in	<i>m</i>	Input vertex
in	<i>r</i>	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.

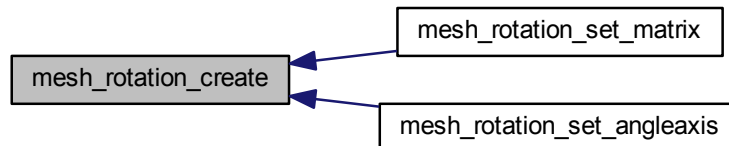
Returns

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

<i>r</i>	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.

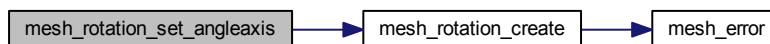
Parameters

in	<i>ang</i>	Input angle of rotation
out	<i>axis</i>	Input axis of rotation
out	<i>r</i>	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.

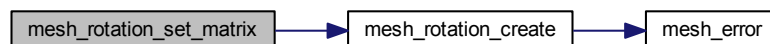
Parameters

in	<i>mat</i>	Input matrix
out	<i>r</i>	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	<i>m</i>	Input mesh
in	<i>sx</i>	X component
in	<i>sy</i>	Y component
in	<i>sz</i>	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	<i>m</i>	Input mesh
in	<i>x</i>	X component
in	<i>y</i>	Y component
in	<i>z</i>	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.

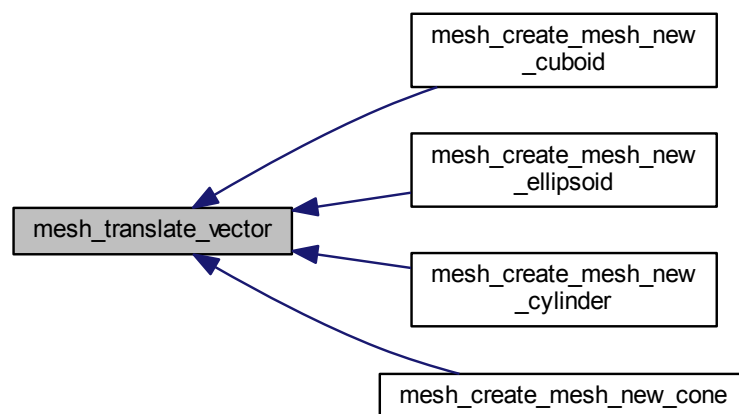
Parameters

in	<i>m</i>	Input mesh
in	<i>v</i>	Input vector

Returns

Error code

Here is the caller graph for this function:



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

Rotates a vertex by a given rotation.

Parameters

in	<i>v</i>	Input vertex
in	<i>r</i>	Input rotation

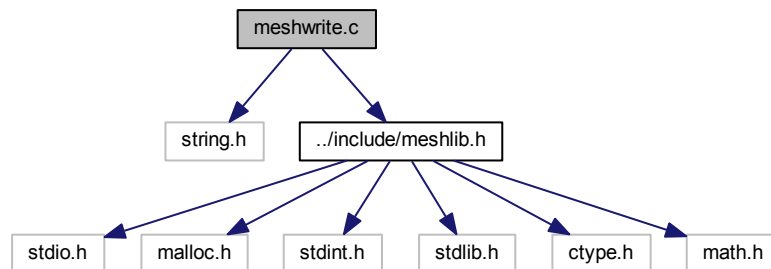
Returns

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

1.4.0.0

Copyright

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

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.

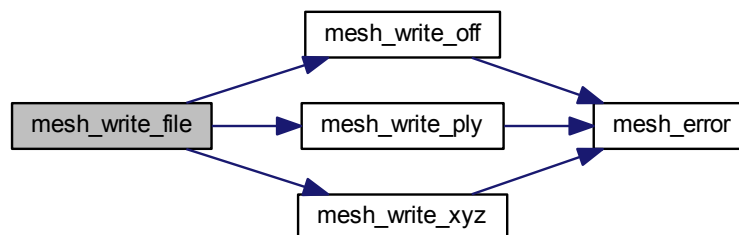
Parameters

<code>in</code>	<code><i>m</i></code>	Input mesh
<code>in</code>	<code><i>fname</i></code>	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

<code>in</code>	<code><i>m</i></code>	Input mesh
<code>in</code>	<code><i>fname</i></code>	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	<i>m</i>	Input mesh
in	<i>fname</i>	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	<i>m</i>	Input mesh
in	<i>fname</i>	Output filename

Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



Index

a

mesh_color, [10](#)

b

mesh_color, [10](#)

data

mesh_rotation, [11](#)

mesh_transform, [12](#)

dummy

mesh, [8](#)

FILEPOINTER

meshlib.h, [44](#)

FLOATDATA

meshlib.h, [41](#)

faces

mesh, [8](#)

mesh_vface, [13](#)

fareas

mesh, [8](#)

fcolors

mesh, [8](#)

fnormals

mesh, [8](#)

g

mesh_color, [10](#)

INTDATA

meshlib.h, [41](#)

INTDATA2

meshlib.h, [44](#)

is_faces

mesh, [8](#)

is_fareas

mesh, [8](#)

is_fcolors

mesh, [8](#)

is_fnormals

mesh, [8](#)

is_loaded

mesh, [8](#)

is_trimesh

mesh, [8](#)

is_vcolors

mesh, [8](#)

is_vertices

mesh, [9](#)

is_vfaces

mesh, [9](#)

is_vnormals

mesh, [9](#)

items

mesh_struct, [11](#)

mesh_struct2, [12](#)

MESH

meshlib.h, [44](#)

MESH_CLONE_FACES

meshlib.h, [42](#)

MESH_CLONE_FAREAS

meshlib.h, [42](#)

MESH_CLONE_FCOLORS

meshlib.h, [42](#)

MESH_CLONE_FNORMALS

meshlib.h, [42](#)

MESH_CLONE_VCOLORS

meshlib.h, [42](#)

MESH_CLONE_VERTICES

meshlib.h, [42](#)

MESH_CLONE_VFACES

meshlib.h, [42](#)

MESH_CLONE_VNORMALS

meshlib.h, [42](#)

MESH_COLOR

meshlib.h, [44](#)

MESH_ERR_FNOTOPEN

meshlib.h, [42](#)

MESH_ERR_MALLOC

meshlib.h, [42](#)

MESH_ERR_UNKNOWN

meshlib.h, [43](#)

MESH_FACE

meshlib.h, [44](#)

MESH_FLOATDATA_TYPE

meshlib.h, [43](#)

MESH_INTDATA_TYPE

meshlib.h, [43](#)

MESH_NORMAL

meshlib.h, [45](#)

MESH_PI

meshlib.h, [44](#)

MESH_ROTATION

meshlib.h, [45](#)

MESH_STRUCT

meshlib.h, [45](#)

MESH_STRUCT2

meshlib.h, [45](#)

MESH_TRANSFORM

meshlib.h, [45](#)

- MESH_TWOPI
 - meshlib.h, 44
- MESH_VECTOR3
 - meshlib.h, 45
- MESH_VERTEX
 - meshlib.h, 46
- MESH_VFACE
 - meshlib.h, 46
- mesh, 7
 - dummy, 8
 - faces, 8
 - fareas, 8
 - fcolors, 8
 - fnormals, 8
 - is_faces, 8
 - is_fareas, 8
 - is_fcolors, 8
 - is_fnormals, 8
 - is_loaded, 8
 - is_trimesh, 8
 - is_vcolors, 8
 - is_vertices, 9
 - is_vfaces, 9
 - is_vnormals, 9
 - meshlib.h, 44
 - num_faces, 9
 - num_vertices, 9
 - origin_type, 9
 - vcolors, 9
 - vertices, 9
 - vfaces, 9
 - vnormals, 9
- mesh_bilateral_filter
 - meshfilter.c, 35
 - meshlib.h, 46
- mesh_calc_face_normal
 - meshcalc.c, 16
 - meshlib.h, 46
- mesh_calc_face_normals
 - meshcalc.c, 16
 - meshlib.h, 47
- mesh_calc_triangle_area
 - meshcalc.c, 17
 - meshlib.h, 48
- mesh_calc_vertex_adjacency
 - meshcalc.c, 18
 - meshlib.h, 48
- mesh_calc_vertex_normals
 - meshcalc.c, 19
 - meshlib.h, 49
- mesh_clone_mesh
 - meshlib.h, 50
 - meshops.c, 78
- mesh_color, 10
 - a, 10
 - b, 10
 - g, 10
 - meshlib.h, 44
 - r, 10
- mesh_combine_mesh
 - meshlib.h, 51
 - meshops.c, 79
- mesh_count_words_in_line
 - meshlib.h, 51
 - meshtext.c, 80
- mesh_create_mesh_new
 - meshcreate.c, 27
 - meshlib.h, 52
- mesh_create_mesh_new_cone
 - meshcreate.c, 28
 - meshlib.h, 52
- mesh_create_mesh_new_cuboid
 - meshcreate.c, 28
 - meshlib.h, 53
- mesh_create_mesh_new_cylinder
 - meshcreate.c, 29
 - meshlib.h, 53
- mesh_create_mesh_new_ellipsoid
 - meshcreate.c, 29
 - meshlib.h, 54
- mesh_cross_normal
 - meshcalc.c, 20
 - meshlib.h, 54
- mesh_cross_vector3
 - meshcalc.c, 20
 - meshlib.h, 55
- mesh_draw_mesh
 - meshdraw.c, 31
 - meshlib.h, 55
- mesh_error
 - mesherror.c, 33
 - meshlib.h, 56
- mesh_face, 10
 - meshlib.h, 44
 - num_vertices, 10
 - vertices, 10
- mesh_find
 - meshcalc.c, 20
 - meshlib.h, 57
- mesh_find2
 - meshcalc.c, 21
 - meshlib.h, 57
- mesh_free_mesh
 - meshcreate.c, 30
 - meshlib.h, 58
- mesh_go_next_word
 - meshlib.h, 58
 - meshtext.c, 81
- mesh_isnumeric
 - meshlib.h, 58
 - meshtext.c, 81
- mesh_laplacian_filter
 - meshfilter.c, 36
 - meshlib.h, 59
- mesh_load_file
 - meshlib.h, 59

- meshload.c, 74
- mesh_load_off
 - meshlib.h, 60
 - meshload.c, 75
- mesh_load_ply
 - meshlib.h, 60
 - meshload.c, 75
- mesh_load_xyz
 - meshlib.h, 61
 - meshload.c, 76
- mesh_normal
 - meshlib.h, 44
- mesh_read_word
 - meshlib.h, 62
 - meshtext.c, 81
- mesh_remove_boundary_faces
 - meshclean.c, 23
 - meshlib.h, 62
- mesh_remove_boundary_vertices
 - meshclean.c, 23
 - meshlib.h, 62
- mesh_remove_close_vertices
 - meshclean.c, 23
 - meshlib.h, 63
- mesh_remove_ear_faces
 - meshclean.c, 23
 - meshlib.h, 63
- mesh_remove_triangles_with_small_area
 - meshclean.c, 24
 - meshlib.h, 64
- mesh_remove_unreferenced_vertices
 - meshclean.c, 25
 - meshlib.h, 64
- mesh_remove_zero_area_faces
 - meshclean.c, 25
 - meshlib.h, 65
- mesh_restricted_laplacian_filter
 - meshfilter.c, 36
 - meshlib.h, 66
- mesh_rotate
 - meshlib.h, 66
 - meshtransform.c, 83
- mesh_rotation, 11
 - data, 11
 - meshlib.h, 45
- mesh_rotation_create
 - meshlib.h, 67
 - meshtransform.c, 83
- mesh_rotation_free
 - meshlib.h, 67
 - meshtransform.c, 84
- mesh_rotation_set_angleaxis
 - meshlib.h, 68
 - meshtransform.c, 84
- mesh_rotation_set_matrix
 - meshlib.h, 68
 - meshtransform.c, 85
- mesh_scale
 - meshlib.h, 68
- meshtransform.c, 85
- mesh_skip_line
 - meshlib.h, 69
 - meshtext.c, 81
- mesh_struct, 11
 - items, 11
 - meshlib.h, 45
 - num_items, 11
- mesh_struct2, 12
 - items, 12
 - meshlib.h, 45
 - num_items, 12
- mesh_transform, 12
 - data, 12
 - meshlib.h, 45
- mesh_translate
 - meshlib.h, 69
 - meshtransform.c, 85
- mesh_translate_vector
 - meshlib.h, 69
 - meshtransform.c, 86
- mesh_upsample
 - meshcalc.c, 21
 - meshlib.h, 70
- mesh_vector3, 12
 - meshlib.h, 45
 - x, 13
 - y, 13
 - z, 13
- mesh_vertex
 - meshlib.h, 45
- mesh_vertex_rotate
 - meshlib.h, 70
 - meshtransform.c, 86
- mesh_vface, 13
 - faces, 13
 - meshlib.h, 46
 - num_faces, 13
- mesh_write_file
 - meshlib.h, 71
 - meshwrite.c, 88
- mesh_write_off
 - meshlib.h, 71
 - meshwrite.c, 88
- mesh_write_ply
 - meshlib.h, 72
 - meshwrite.c, 89
- mesh_write_xyz
 - meshlib.h, 73
 - meshwrite.c, 90
- meshcalc.c, 15
 - mesh_calc_face_normal, 16
 - mesh_calc_face_normals, 16
 - mesh_calc_triangle_area, 17
 - mesh_calc_vertex_adjacency, 18
 - mesh_calc_vertex_normals, 19
 - mesh_cross_normal, 20

- mesh_cross_vector3, 20
- mesh_find, 20
- mesh_find2, 21
- mesh_upsample, 21
- meshclean.c, 21
 - mesh_remove_boundary_faces, 23
 - mesh_remove_boundary_vertices, 23
 - mesh_remove_close_vertices, 23
 - mesh_remove_ear_faces, 23
 - mesh_remove_triangles_with_small_area, 24
 - mesh_remove_unreferenced_vertices, 25
 - mesh_remove_zero_area_faces, 25
- meshcreate.c, 26
 - mesh_create_mesh_new, 27
 - mesh_create_mesh_new_cone, 28
 - mesh_create_mesh_new_cuboid, 28
 - mesh_create_mesh_new_cylinder, 29
 - mesh_create_mesh_new_ellipsoid, 29
 - mesh_free_mesh, 30
- meshdraw.c, 30
 - mesh_draw_mesh, 31
- mesherror.c, 32
 - mesh_error, 33
- meshfilter.c, 34
 - mesh_bilateral_filter, 35
 - mesh_laplacian_filter, 36
 - mesh_restricted_laplacian_filter, 36
- meshlib.h, 37
 - FILEPOINTER, 44
 - FLOATDATA, 41
 - INTDATA, 41
 - INTDATA2, 44
 - MESH, 44
 - MESH_CLONE_FACES, 42
 - MESH_CLONE_FAREAS, 42
 - MESH_CLONE_FCOLORS, 42
 - MESH_CLONE_FNORMALS, 42
 - MESH_CLONE_VCOLORS, 42
 - MESH_CLONE_VERTICES, 42
 - MESH_CLONE_VFACES, 42
 - MESH_CLONE_VNORMALS, 42
 - MESH_COLOR, 44
 - MESH_ERR_FNOTOPEN, 42
 - MESH_ERR_MALLOC, 42
 - MESH_ERR_UNKNOWN, 43
 - MESH_FACE, 44
 - MESH_FLOATDATA_TYPE, 43
 - MESH_INTDATA_TYPE, 43
 - MESH_NORMAL, 45
 - MESH_PI, 44
 - MESH_ROTATION, 45
 - MESH_STRUCT, 45
 - MESH_STRUCT2, 45
 - MESH_TRANSFORM, 45
 - MESH_TWOPI, 44
 - MESH_VECTOR3, 45
 - MESH_VERTEX, 46
 - MESH_VFACE, 46
 - mesh, 44
 - mesh_bilateral_filter, 46
 - mesh_calc_face_normal, 46
 - mesh_calc_face_normals, 47
 - mesh_calc_triangle_area, 48
 - mesh_calc_vertex_adjacency, 48
 - mesh_calc_vertex_normals, 49
 - mesh_clone_mesh, 50
 - mesh_color, 44
 - mesh_combine_mesh, 51
 - mesh_count_words_in_line, 51
 - mesh_create_mesh_new, 52
 - mesh_create_mesh_new_cone, 52
 - mesh_create_mesh_new_cuboid, 53
 - mesh_create_mesh_new_cylinder, 53
 - mesh_create_mesh_new_ellipsoid, 54
 - mesh_cross_normal, 54
 - mesh_cross_vector3, 55
 - mesh_draw_mesh, 55
 - mesh_error, 56
 - mesh_face, 44
 - mesh_find, 57
 - mesh_find2, 57
 - mesh_free_mesh, 58
 - mesh_go_next_word, 58
 - mesh_isnumeric, 58
 - mesh_laplacian_filter, 59
 - mesh_load_file, 59
 - mesh_load_off, 60
 - mesh_load_ply, 60
 - mesh_load_xyz, 61
 - mesh_normal, 44
 - mesh_read_word, 62
 - mesh_remove_boundary_faces, 62
 - mesh_remove_boundary_vertices, 62
 - mesh_remove_close_vertices, 63
 - mesh_remove_ear_faces, 63
 - mesh_remove_triangles_with_small_area, 64
 - mesh_remove_unreferenced_vertices, 64
 - mesh_remove_zero_area_faces, 65
 - mesh_restricted_laplacian_filter, 66
 - mesh_rotate, 66
 - mesh_rotation, 45
 - mesh_rotation_create, 67
 - mesh_rotation_free, 67
 - mesh_rotation_set_angleaxis, 68
 - mesh_rotation_set_matrix, 68
 - mesh_scale, 68
 - mesh_skip_line, 69
 - mesh_struct, 45
 - mesh_struct2, 45
 - mesh_transform, 45
 - mesh_translate, 69
 - mesh_translate_vector, 69
 - mesh_upsample, 70
 - mesh_vector3, 45
 - mesh_vertex, 45
 - mesh_vertex_rotate, 70

- mesh_vface, [46](#)
- mesh_write_file, [71](#)
- mesh_write_off, [71](#)
- mesh_write_ply, [72](#)
- mesh_write_xyz, [73](#)
- meshload.c, [73](#)
 - mesh_load_file, [74](#)
 - mesh_load_off, [75](#)
 - mesh_load_ply, [75](#)
 - mesh_load_xyz, [76](#)
- meshops.c, [77](#)
 - mesh_clone_mesh, [78](#)
 - mesh_combine_mesh, [79](#)
- meshtext.c, [79](#)
 - mesh_count_words_in_line, [80](#)
 - mesh_go_next_word, [81](#)
 - mesh_isnumeric, [81](#)
 - mesh_read_word, [81](#)
 - mesh_skip_line, [81](#)
- meshtransform.c, [82](#)
 - mesh_rotate, [83](#)
 - mesh_rotation_create, [83](#)
 - mesh_rotation_free, [84](#)
 - mesh_rotation_set_angleaxis, [84](#)
 - mesh_rotation_set_matrix, [85](#)
 - mesh_scale, [85](#)
 - mesh_translate, [85](#)
 - mesh_translate_vector, [86](#)
 - mesh_vertex_rotate, [86](#)
- meshwrite.c, [87](#)
 - mesh_write_file, [88](#)
 - mesh_write_off, [88](#)
 - mesh_write_ply, [89](#)
 - mesh_write_xyz, [90](#)
- num_faces
 - mesh, [9](#)
 - mesh_vface, [13](#)
- num_items
 - mesh_struct, [11](#)
 - mesh_struct2, [12](#)
- num_vertices
 - mesh, [9](#)
 - mesh_face, [10](#)
- origin_type
 - mesh, [9](#)
- r
 - mesh_color, [10](#)
- vcolors
 - mesh, [9](#)
- vertices
 - mesh, [9](#)
 - mesh_face, [10](#)
- vfaces
 - mesh, [9](#)
- vnormals
 - mesh, [9](#)
- x
 - mesh_vector3, [13](#)
- y
 - mesh_vector3, [13](#)
- z
 - mesh_vector3, [13](#)