

**MeshLib**

**1.4.0.0**

Generated by Doxygen 1.8.9.1

Sun Dec 6 2015 00:57:49



# Contents

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

4.1.1.19	is_vnormals	9
4.1.1.20	num_edges	9
4.1.1.21	num_faces	9
4.1.1.22	num_vertices	9
4.1.1.23	origin_type	10
4.1.1.24	vcolors	10
4.1.1.25	vertices	10
4.1.1.26	vfaces	10
4.1.1.27	vnormals	10
4.2	mesh_adjface Struct Reference	10
4.2.1	Field Documentation	10
4.2.1.1	faces	10
4.2.1.2	num_faces	10
4.3	mesh_color Struct Reference	10
4.3.1	Field Documentation	11
4.3.1.1	a	11
4.3.1.2	b	11
4.3.1.3	g	11
4.3.1.4	r	11
4.4	mesh_edge Struct Reference	11
4.4.1	Field Documentation	11
4.4.1.1	faces	11
4.4.1.2	vertices	11
4.5	mesh_face Struct Reference	12
4.5.1	Field Documentation	12
4.5.1.1	num_vertices	12
4.5.1.2	vertices	12
4.6	mesh_rotation Struct Reference	12
4.6.1	Field Documentation	12
4.6.1.1	data	12
4.7	mesh_struct Struct Reference	12
4.7.1	Field Documentation	13
4.7.1.1	items	13
4.7.1.2	num_items	13
4.8	mesh_struct2 Struct Reference	13
4.8.1	Field Documentation	13
4.8.1.1	items	13
4.8.1.2	num_items	13
4.9	mesh_struct3 Struct Reference	13
4.9.1	Field Documentation	13

4.9.1.1	items	13
4.9.1.2	num_items	14
4.10	mesh_transform Struct Reference	14
4.10.1	Field Documentation	14
4.10.1.1	data	14
4.11	mesh_vector3 Struct Reference	14
4.11.1	Field Documentation	14
4.11.1.1	x	14
4.11.1.2	y	14
4.11.1.3	z	14
<b>5</b>	<b>File Documentation</b>	<b>17</b>
5.1	meshcalc.c File Reference	17
5.1.1	Detailed Description	18
5.1.2	Function Documentation	18
5.1.2.1	mesh_calc_edges	18
5.1.2.2	mesh_calc_face_adjacency	19
5.1.2.3	mesh_calc_face_normal	19
5.1.2.4	mesh_calc_face_normals	20
5.1.2.5	mesh_calc_triangle_area	21
5.1.2.6	mesh_calc_vertex_adjacency	22
5.1.2.7	mesh_calc_vertex_normals	23
5.1.2.8	mesh_cross_normal	24
5.1.2.9	mesh_cross_vector3	24
5.1.2.10	mesh_find	25
5.1.2.11	mesh_find2	25
5.1.2.12	mesh_find3	25
5.1.2.13	mesh_upsample	26
5.2	meshclean.c File Reference	27
5.2.1	Detailed Description	28
5.2.2	Function Documentation	28
5.2.2.1	mesh_remove_boundary_faces	28
5.2.2.2	mesh_remove_boundary_vertices	28
5.2.2.3	mesh_remove_close_vertices	28
5.2.2.4	mesh_remove_ear_faces	29
5.2.2.5	mesh_remove_triangles_with_small_area	29
5.2.2.6	mesh_remove_unreferenced_vertices	30
5.2.2.7	mesh_remove_zero_area_faces	31
5.3	meshcreate.c File Reference	31
5.3.1	Detailed Description	32

5.3.2	Function Documentation	32
5.3.2.1	mesh_create_mesh_new	32
5.3.2.2	mesh_create_mesh_new_cone	33
5.3.2.3	mesh_create_mesh_new_cuboid	34
5.3.2.4	mesh_create_mesh_new_cylinder	34
5.3.2.5	mesh_create_mesh_new_ellipsoid	35
5.3.2.6	mesh_free_mesh	35
5.4	meshdraw.c File Reference	36
5.4.1	Detailed Description	36
5.4.2	Function Documentation	37
5.4.2.1	mesh_draw_mesh	37
5.4.2.2	mesh_draw_mesh_smooth	37
5.5	mesherror.c File Reference	38
5.5.1	Detailed Description	38
5.5.2	Function Documentation	38
5.5.2.1	mesh_error	38
5.6	meshfilter.c File Reference	39
5.6.1	Detailed Description	40
5.6.2	Function Documentation	40
5.6.2.1	mesh_bilateral_filter	40
5.6.2.2	mesh_laplacian_filter	41
5.6.2.3	mesh_restricted_laplacian_filter	41
5.7	meshlib.h File Reference	42
5.7.1	Detailed Description	47
5.7.2	Macro Definition Documentation	47
5.7.2.1	_CRT_SECURE_NO_DEPRECATED	47
5.7.2.2	FLOATDATA	47
5.7.2.3	INTDATA	47
5.7.2.4	MESH_CLONE_ALL_PROPS	47
5.7.2.5	MESH_CLONE_EDGES	47
5.7.2.6	MESH_CLONE_F_ALL_PROPS	47
5.7.2.7	MESH_CLONE_FACES	47
5.7.2.8	MESH_CLONE_FAREAS	47
5.7.2.9	MESH_CLONE_FCOLORS	47
5.7.2.10	MESH_CLONE_FFACES	48
5.7.2.11	MESH_CLONE_FNORMALS	48
5.7.2.12	MESH_CLONE_V_ALL_PROPS	48
5.7.2.13	MESH_CLONE_VCOLORS	48
5.7.2.14	MESH_CLONE_VERTICES	48
5.7.2.15	MESH_CLONE_VFACES	48

5.7.2.16	MESH_CLONE_VNORMALS	48
5.7.2.17	MESH_ERR_FNOTOPEN	48
5.7.2.18	MESH_ERR_INCOMPATIBLE	48
5.7.2.19	MESH_ERR_MALLOC	48
5.7.2.20	MESH_ERR_SIZE_MISMATCH	48
5.7.2.21	MESH_ERR_UNKNOWN	48
5.7.2.22	MESH_FLOATDATA_TYPE	49
5.7.2.23	MESH_INTDATA_TYPE	49
5.7.2.24	MESH_ORIGIN_TYPE_BUILD	49
5.7.2.25	MESH_ORIGIN_TYPE_COFF	49
5.7.2.26	MESH_ORIGIN_TYPE_NCOFF	49
5.7.2.27	MESH_ORIGIN_TYPE_NOFF	49
5.7.2.28	MESH_ORIGIN_TYPE_OFF	49
5.7.2.29	MESH_ORIGIN_TYPE_PLY_ASCII	49
5.7.2.30	MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN	49
5.7.2.31	MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN	49
5.7.2.32	MESH_ORIGIN_TYPE_XYZ	49
5.7.2.33	MESH_PI	49
5.7.2.34	MESH_TWOPI	50
5.7.3	Typedef Documentation	50
5.7.3.1	FILEPOINTER	50
5.7.3.2	INTDATA2	50
5.7.3.3	INTDATA3	50
5.7.3.4	mesh	50
5.7.3.5	MESH	50
5.7.3.6	mesh_adjface	50
5.7.3.7	mesh_color	50
5.7.3.8	MESH_COLOR	50
5.7.3.9	mesh_edge	50
5.7.3.10	MESH_EDGE	50
5.7.3.11	mesh_face	50
5.7.3.12	MESH_FACE	51
5.7.3.13	mesh_fface	51
5.7.3.14	MESH_FFACE	51
5.7.3.15	mesh_normal	51
5.7.3.16	MESH_NORMAL	51
5.7.3.17	mesh_rotation	51
5.7.3.18	MESH_ROTATION	51
5.7.3.19	mesh_struct	51
5.7.3.20	MESH_STRUCT	51

5.7.3.21	<a href="#">mesh_struct2</a>	51
5.7.3.22	<a href="#">MESH_STRUCT2</a>	51
5.7.3.23	<a href="#">mesh_struct3</a>	51
5.7.3.24	<a href="#">MESH_STRUCT3</a>	52
5.7.3.25	<a href="#">mesh_transform</a>	52
5.7.3.26	<a href="#">MESH_TRANSFORM</a>	52
5.7.3.27	<a href="#">mesh_vector3</a>	52
5.7.3.28	<a href="#">MESH_VECTOR3</a>	52
5.7.3.29	<a href="#">mesh_vertex</a>	52
5.7.3.30	<a href="#">MESH_VERTEX</a>	52
5.7.3.31	<a href="#">mesh_vface</a>	52
5.7.3.32	<a href="#">MESH_VFACE</a>	52
5.7.4	<a href="#">Function Documentation</a>	52
5.7.4.1	<a href="#">mesh_bilateral_filter</a>	52
5.7.4.2	<a href="#">mesh_calc_edges</a>	53
5.7.4.3	<a href="#">mesh_calc_face_adjacency</a>	54
5.7.4.4	<a href="#">mesh_calc_face_normal</a>	55
5.7.4.5	<a href="#">mesh_calc_face_normals</a>	56
5.7.4.6	<a href="#">mesh_calc_triangle_area</a>	56
5.7.4.7	<a href="#">mesh_calc_vertex_adjacency</a>	57
5.7.4.8	<a href="#">mesh_calc_vertex_normals</a>	58
5.7.4.9	<a href="#">mesh_clone_mesh</a>	59
5.7.4.10	<a href="#">mesh_combine_mesh</a>	60
5.7.4.11	<a href="#">mesh_count_words_in_line</a>	60
5.7.4.12	<a href="#">mesh_create_mesh_new</a>	60
5.7.4.13	<a href="#">mesh_create_mesh_new_cone</a>	61
5.7.4.14	<a href="#">mesh_create_mesh_new_cuboid</a>	62
5.7.4.15	<a href="#">mesh_create_mesh_new_cylinder</a>	62
5.7.4.16	<a href="#">mesh_create_mesh_new_ellipsoid</a>	63
5.7.4.17	<a href="#">mesh_cross_normal</a>	63
5.7.4.18	<a href="#">mesh_cross_vector3</a>	64
5.7.4.19	<a href="#">mesh_draw_mesh</a>	64
5.7.4.20	<a href="#">mesh_draw_mesh_smooth</a>	65
5.7.4.21	<a href="#">mesh_error</a>	66
5.7.4.22	<a href="#">mesh_find</a>	67
5.7.4.23	<a href="#">mesh_find2</a>	67
5.7.4.24	<a href="#">mesh_find3</a>	68
5.7.4.25	<a href="#">mesh_free_mesh</a>	68
5.7.4.26	<a href="#">mesh_go_next_word</a>	68
5.7.4.27	<a href="#">mesh_isnumeric</a>	69



5.7.4.28	<a href="#">mesh_laplacian_filter</a>	69
5.7.4.29	<a href="#">mesh_load_file</a>	69
5.7.4.30	<a href="#">mesh_load_off</a>	70
5.7.4.31	<a href="#">mesh_load_ply</a>	71
5.7.4.32	<a href="#">mesh_load_xyz</a>	72
5.7.4.33	<a href="#">mesh_read_word</a>	73
5.7.4.34	<a href="#">mesh_read_word_only</a>	73
5.7.4.35	<a href="#">mesh_remove_boundary_faces</a>	73
5.7.4.36	<a href="#">mesh_remove_boundary_vertices</a>	73
5.7.4.37	<a href="#">mesh_remove_close_vertices</a>	74
5.7.4.38	<a href="#">mesh_remove_ear_faces</a>	74
5.7.4.39	<a href="#">mesh_remove_triangles_with_small_area</a>	75
5.7.4.40	<a href="#">mesh_remove_unreferenced_vertices</a>	75
5.7.4.41	<a href="#">mesh_remove_zero_area_faces</a>	76
5.7.4.42	<a href="#">mesh_restricted_laplacian_filter</a>	77
5.7.4.43	<a href="#">mesh_rotate</a>	77
5.7.4.44	<a href="#">mesh_rotation_create</a>	78
5.7.4.45	<a href="#">mesh_rotation_free</a>	78
5.7.4.46	<a href="#">mesh_rotation_set_angleaxis</a>	79
5.7.4.47	<a href="#">mesh_rotation_set_matrix</a>	79
5.7.4.48	<a href="#">mesh_scale</a>	80
5.7.4.49	<a href="#">mesh_skip_line</a>	81
5.7.4.50	<a href="#">mesh_translate</a>	81
5.7.4.51	<a href="#">mesh_translate_vector</a>	81
5.7.4.52	<a href="#">mesh_upsample</a>	82
5.7.4.53	<a href="#">mesh_vertex_rotate</a>	82
5.7.4.54	<a href="#">mesh_write_file</a>	83
5.7.4.55	<a href="#">mesh_write_off</a>	83
5.7.4.56	<a href="#">mesh_write_ply</a>	84
5.7.4.57	<a href="#">mesh_write_xyz</a>	85
5.8	<a href="#">meshload.c File Reference</a>	85
5.8.1	<a href="#">Detailed Description</a>	86
5.8.2	<a href="#">Function Documentation</a>	86
5.8.2.1	<a href="#">mesh_load_file</a>	86
5.8.2.2	<a href="#">mesh_load_off</a>	87
5.8.2.3	<a href="#">mesh_load_ply</a>	88
5.8.2.4	<a href="#">mesh_load_xyz</a>	88
5.9	<a href="#">meshops.c File Reference</a>	89
5.9.1	<a href="#">Detailed Description</a>	90
5.9.2	<a href="#">Function Documentation</a>	90

5.9.2.1	<code>mesh_clone_mesh</code>	90
5.9.2.2	<code>mesh_combine_mesh</code>	91
5.10	<code>meshtext.c</code> File Reference	91
5.10.1	Detailed Description	92
5.10.2	Function Documentation	92
5.10.2.1	<code>mesh_count_words_in_line</code>	92
5.10.2.2	<code>mesh_go_next_word</code>	93
5.10.2.3	<code>mesh_isnumeric</code>	93
5.10.2.4	<code>mesh_read_word</code>	93
5.10.2.5	<code>mesh_read_word_only</code>	93
5.10.2.6	<code>mesh_skip_line</code>	94
5.11	<code>meshtransform.c</code> File Reference	94
5.11.1	Detailed Description	95
5.11.2	Function Documentation	95
5.11.2.1	<code>mesh_rotate</code>	95
5.11.2.2	<code>mesh_rotation_create</code>	96
5.11.2.3	<code>mesh_rotation_free</code>	96
5.11.2.4	<code>mesh_rotation_set_angleaxis</code>	96
5.11.2.5	<code>mesh_rotation_set_matrix</code>	97
5.11.2.6	<code>mesh_scale</code>	97
5.11.2.7	<code>mesh_translate</code>	98
5.11.2.8	<code>mesh_translate_vector</code>	98
5.11.2.9	<code>mesh_vertex_rotate</code>	99
5.12	<code>meshwrite.c</code> File Reference	100
5.12.1	Detailed Description	100
5.12.2	Function Documentation	101
5.12.2.1	<code>mesh_write_file</code>	101
5.12.2.2	<code>mesh_write_off</code>	101
5.12.2.3	<code>mesh_write_ply</code>	102
5.12.2.4	<code>mesh_write_xyz</code>	103
<b>Index</b>		<b>105</b>

# 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:

<a href="#">mesh</a>	7
<a href="#">mesh_adjface</a>	10
<a href="#">mesh_color</a>	10
<a href="#">mesh_edge</a>	11
<a href="#">mesh_face</a>	12
<a href="#">mesh_rotation</a>	12
<a href="#">mesh_struct</a>	12
<a href="#">mesh_struct2</a>	13
<a href="#">mesh_struct3</a>	13
<a href="#">mesh_transform</a>	14
<a href="#">mesh_vector3</a>	14



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">meshcalc.c</a>	This file contains functions pertaining to different mesh computations . . . . .	17
<a href="#">meshclean.c</a>	This file contains functions pertaining to different mesh cleaning algorithms . . . . .	27
<a href="#">meshcreate.c</a>	This file contains functions pertaining to mesh creation and freeing . . . . .	31
<a href="#">meshdraw.c</a>	This file contains functions pertaining to mesh drawing in OpenGL . . . . .	36
<a href="#">mesherror.c</a>	This file contains functions pertaining to handling errors . . . . .	38
<a href="#">meshfilter.c</a>	This file contains functions pertaining to different mesh filtering algorithms . . . . .	39
<a href="#">meshlib.h</a>	This header file contains declarations of all functions of meshlib . . . . .	42
<a href="#">meshload.c</a>	This file contains functions pertaining to loading different mesh file types . . . . .	85
<a href="#">meshops.c</a>	This file contains functions pertaining to mesh combinatorial operations . . . . .	89
<a href="#">meshtext.c</a>	This file contains functions pertaining to different text routines . . . . .	91
<a href="#">meshtransform.c</a>	This file contains functions pertaining to different mesh transformations . . . . .	94
<a href="#">meshwrite.c</a>	This file contains functions pertaining to writing different mesh file types . . . . .	100





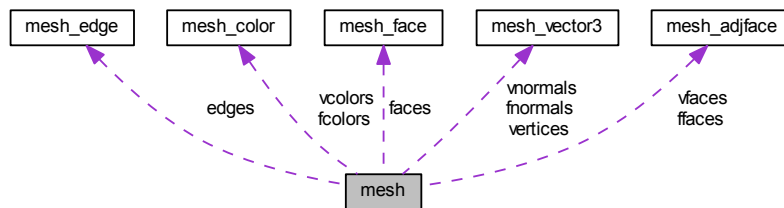
## Chapter 4

# 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](#)



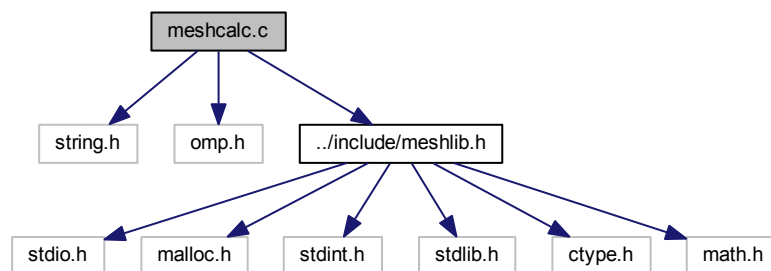
## Chapter 5

# 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_NORMAL` n)  
*Computes the face normal given 3 vertices.*
- int `mesh_calc_vertex_normals` (`MESH` m)  
*Computes vertex normals of a given mesh.*
- int `mesh_calc_face_normals` (`MESH` m)  
*Computes face normals of a given mesh.*
- int `mesh_calc_edges` (`MESH` m)  
*Computes edges of a given mesh.*

- int [mesh\\_calc\\_vertex\\_adjacency](#) (MESH m)  
*Computes vertex adjacent faces of a given mesh.*
- int [mesh\\_calc\\_face\\_adjacency](#) (MESH m)  
*Computes face adjacent faces of a given mesh.*
- INTDATA [mesh\\_find](#) (MESH\_STRUCT s, INTDATA q)  
*Finds an item in an INTDATA structure.*
- INTDATA [mesh\\_find2](#) (MESH\_STRUCT2 s, INTDATA q)  
*Finds an item in an INTDATA2 structure.*
- INTDATA [mesh\\_find3](#) (MESH\_STRUCT3 s, INTDATA q)  
*Finds an item in an INTDATA3 structure.*
- int [mesh\\_upsample](#) (MESH m, int iters)  
*Upsamples a given mesh.*
- 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 int mesh\_calc\_edges ( MESH m )

Computes edges of a given mesh.

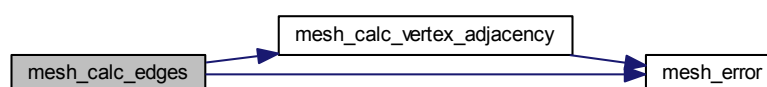
##### Parameters

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

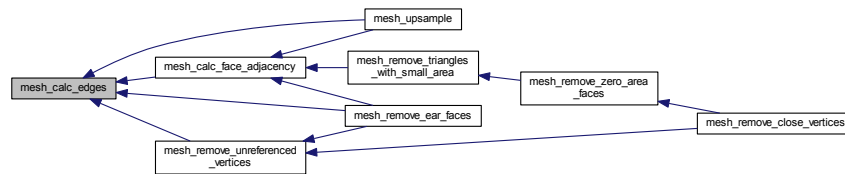
##### Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.1.2.2 int mesh\_calc\_face\_adjacency ( MESH *m* )

Computes face adjacent faces of a given mesh.

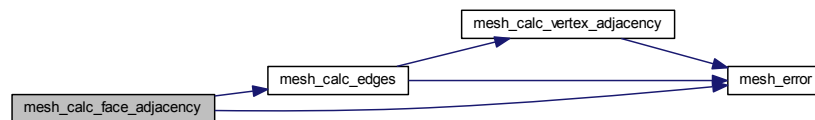
Parameters

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

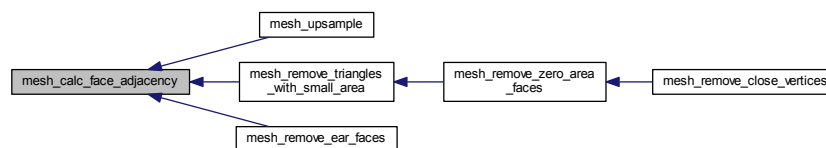
Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.1.2.3 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

Here is the caller graph for this function:

**5.1.2.4 int mesh\_calc\_face\_normals ( MESH  $m$  )**

Computes face normals of a given mesh.

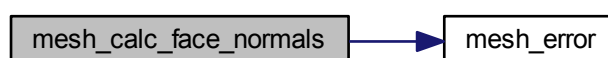
**Parameters**

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

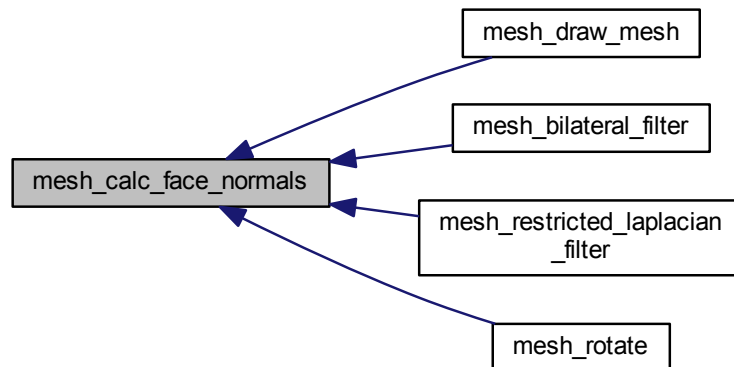
**Returns**

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	<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:



5.1.2.6 `int mesh_calc_vertex_adjacency ( MESH m )`

Computes vertex adjacent faces of a given mesh.



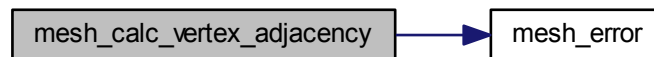
### Parameters

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

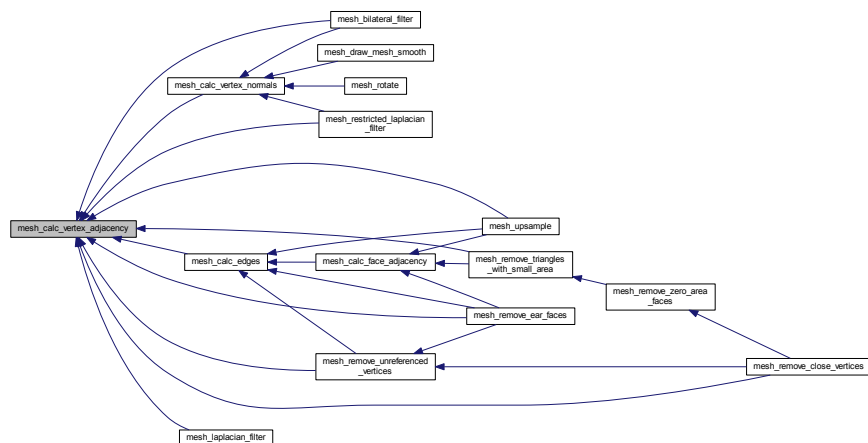
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.1.2.7 int mesh\_calc\_vertex\_normals ( MESH *m* )

Computes vertex normals of a given mesh.

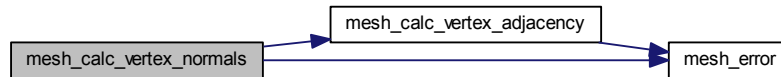
## Parameters

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

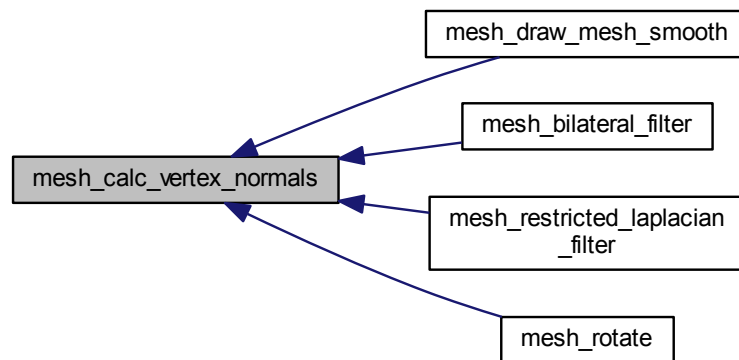
**Returns**

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.

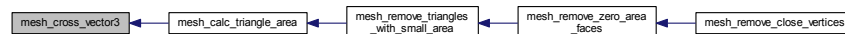
## Parameters

in	x	First vector
in	y	Second vector
out	z	Output cross product $x \times y$

## Returns

NULL

Here is the caller graph for this function:



## 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.

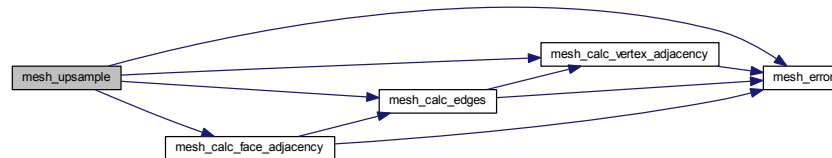
## Parameters

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

## Returns

Error code

Here is the call graph for this function:

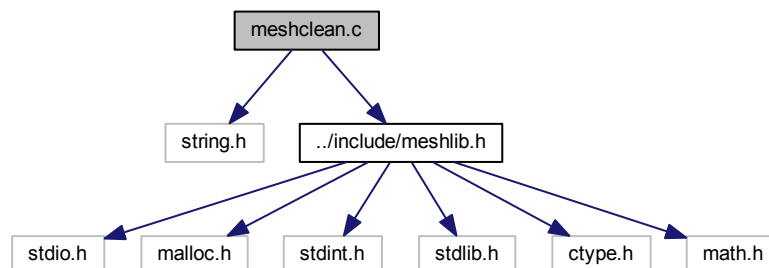


## 5.2 meshclean.c File Reference

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.

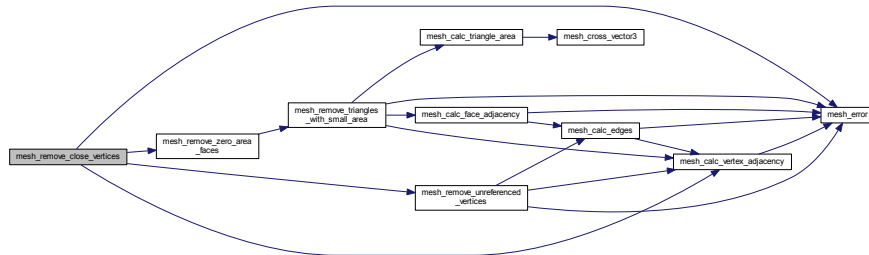
## Parameters

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.

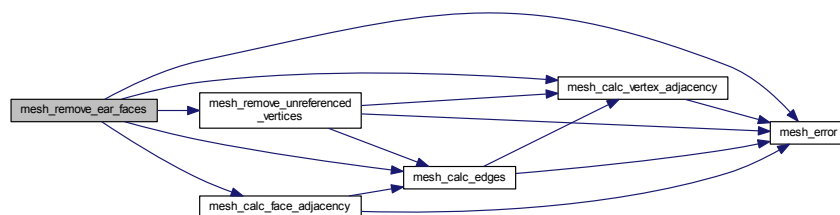
## Parameters

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.

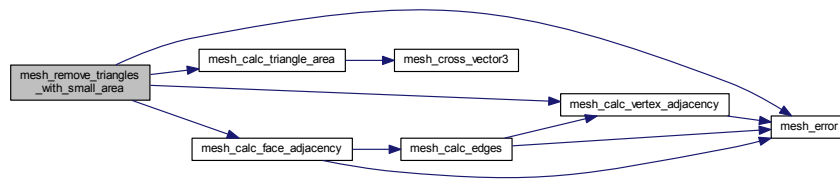
## Parameters

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

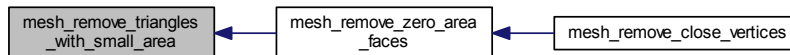
## Returns

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.

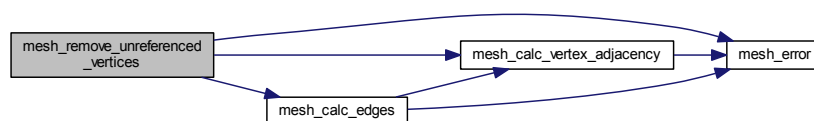
## Parameters

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

## Returns

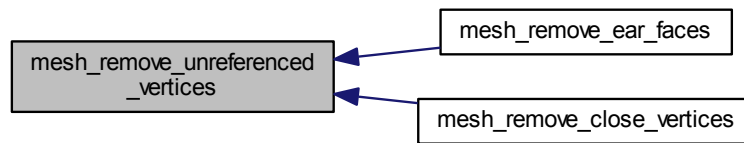
Error code

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.

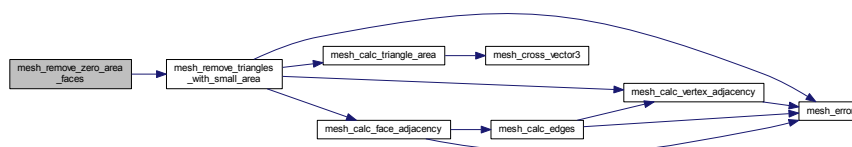
##### Parameters

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

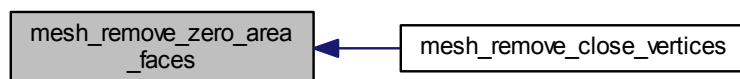
##### Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

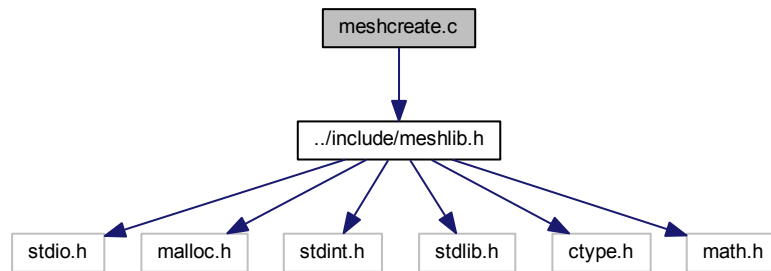


## 5.3 meshcreate.c File Reference

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.

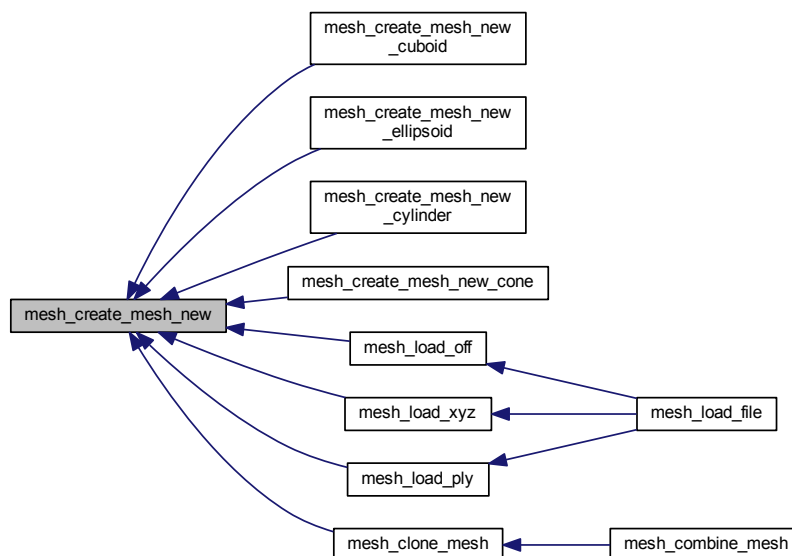
## Returns

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.

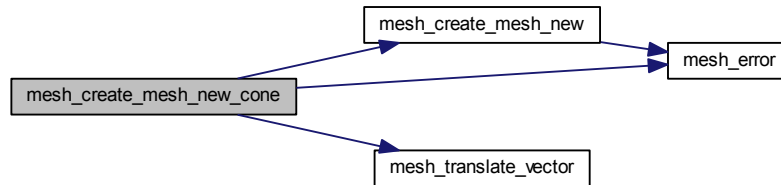
## Parameters

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

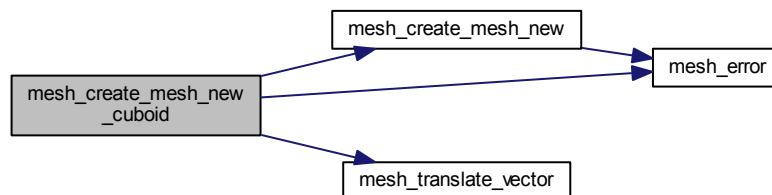
**Parameters**

in	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**

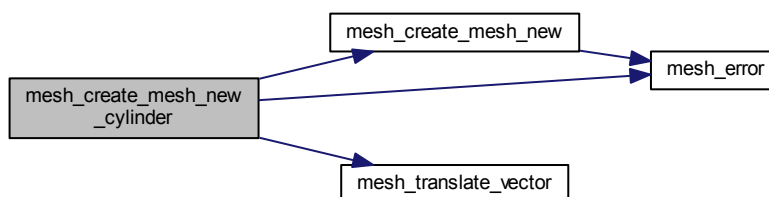

---

in	<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.

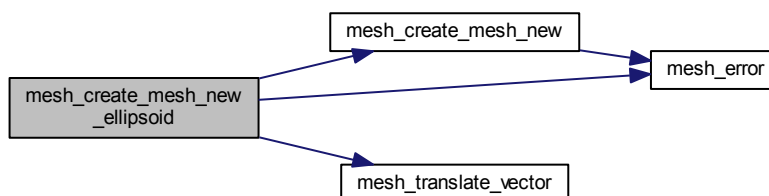
**Parameters**

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.

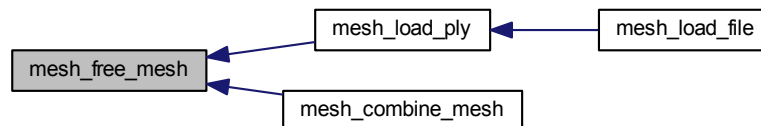
**Parameters**

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

**Returns**

NULL

Here is the caller graph for this function:



## 5.4 meshdraw.c File Reference

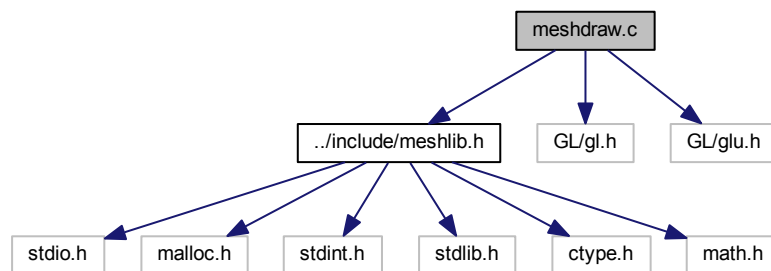
This file contains functions pertaining to mesh drawing in OpenGL.

```
#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.*

### 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 in flat shading.

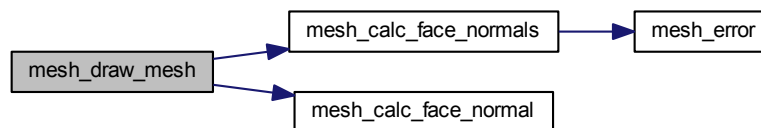
**Parameters**

<i>in</i>	<i>m</i>	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.

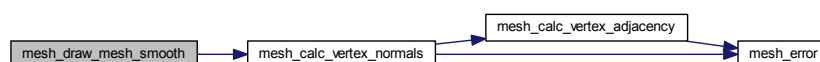
**Parameters**

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

**Returns**

NULL

Here is the call graph for this function:

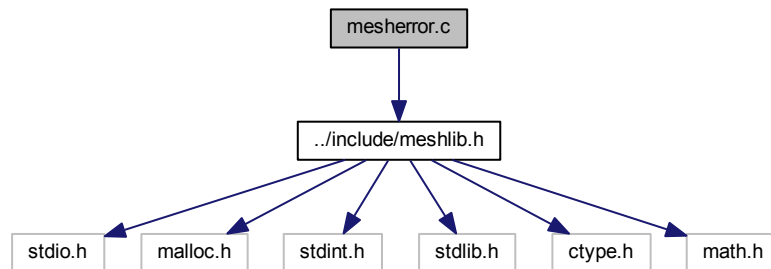


## 5.5 mesherror.c File Reference

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

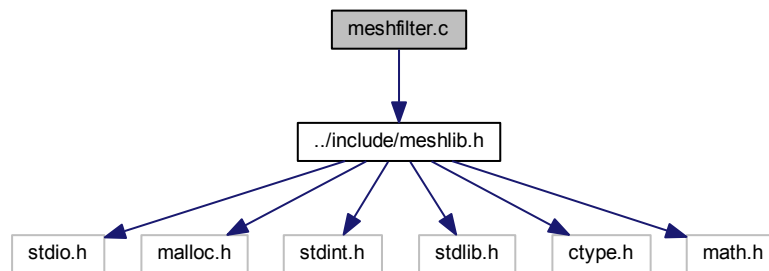
---





```
#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

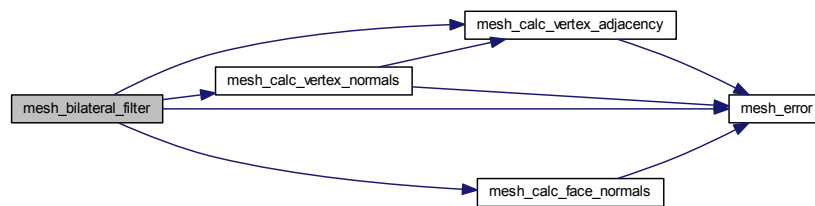
---

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.

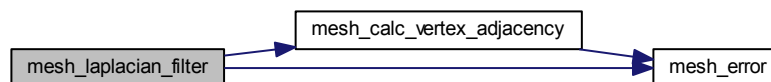
**Parameters**

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.

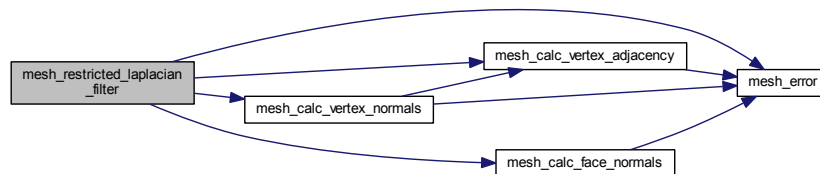
## Parameters

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:

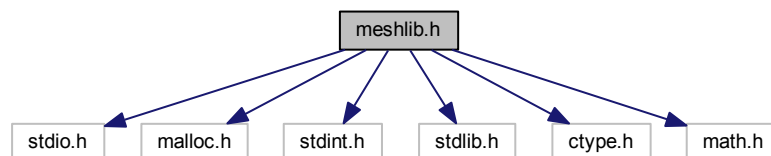


## 5.7 meshlib.h File Reference

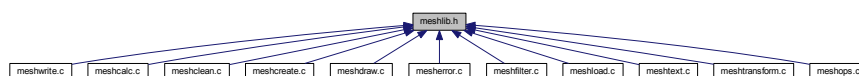
This header file contains declarations of all functions of meshlib.

```
#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\\_struct3](#)
- struct [mesh\\_face](#)
- struct [mesh\\_edge](#)
- struct [mesh\\_adjface](#)
- struct [mesh\\_rotation](#)
- struct [mesh\\_transform](#)
- struct [mesh](#)

## Macros

- `#define _CRT_SECURE_NO_DEPRECATED`
- `#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_TWOP (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_CLONE_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

- 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\\_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.*
- 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.*
- [INTDATA mesh\\_find3](#) ([MESH\\_STRUCT3](#) s, [INTDATA](#) q)  
*Finds an item in an INTDATA3 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 and moves to the next word.*
- int [mesh\\_read\\_word\\_only](#) (FILEPOINTER fp, char \*c\_word, int sz)  
*Reads current word without moving to the next word.*
- int [mesh\\_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 in flat shading.*
- void [mesh\\_draw\\_mesh\\_smooth](#) (MESH m)  
*Draws a given mesh in OpenGL context in smoothing shading.*



### 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 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)**

$\pi$

5.7.2.34 `#define MESH_TWOPI (6.28318530718)`

$2\pi$

### 5.7.3 Typedef Documentation

5.7.3.1 `typedef struct _jobuf* 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 int mesh\_bilateral\_filter ( MESH *m*, FLOATDATA *sigma\_c*, FLOATDATA *sigma\_s*, int *niters* )**

Mesh bilateral filter.

Parameters

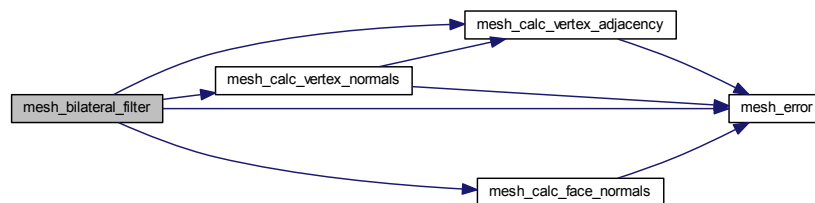
in	<i>m</i>	Input mesh
in	<i>sigma_c</i>	Range standard deviation
in	<i>sigma_s</i>	Spatial standard deviation

<code>in</code>	<code>niters</code>	Number of iterations
-----------------	---------------------	----------------------

**Returns**

Error code

Here is the call graph for this function:

**5.7.4.2 int mesh\_calc\_edges ( MESH m )**

Computes edges of a given mesh.

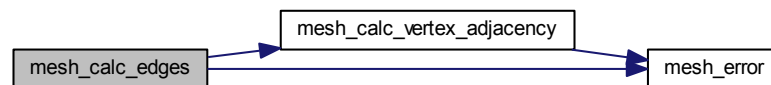
**Parameters**

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

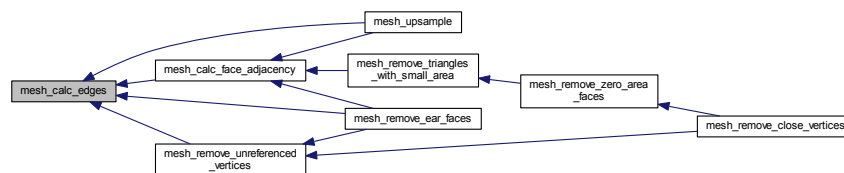
**Returns**

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.4.3 `int mesh_calc_face_adjacency ( MESH m )`

Computes face adjacent faces of a given mesh.



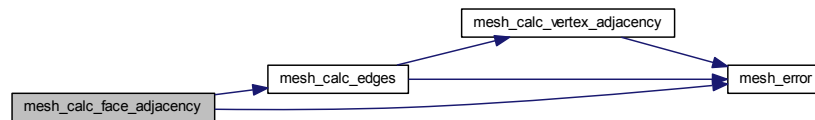
## Parameters

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

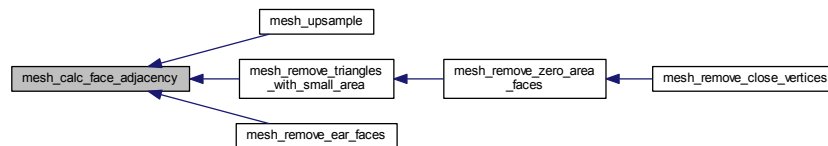
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.4 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	<i>v1</i>	First vertex
in	<i>v2</i>	Second vertex
in	<i>v3</i>	Third vertex
out	<i>n</i>	Output face normal $\mathbf{n}_f$

## Returns

NULL

Here is the caller graph for this function:



#### 5.7.4.5 int mesh\_calc\_face\_normals ( MESH *m* )

Computes face normals of a given mesh.

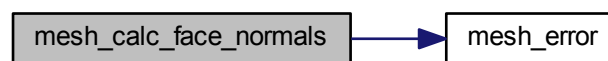
##### Parameters

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

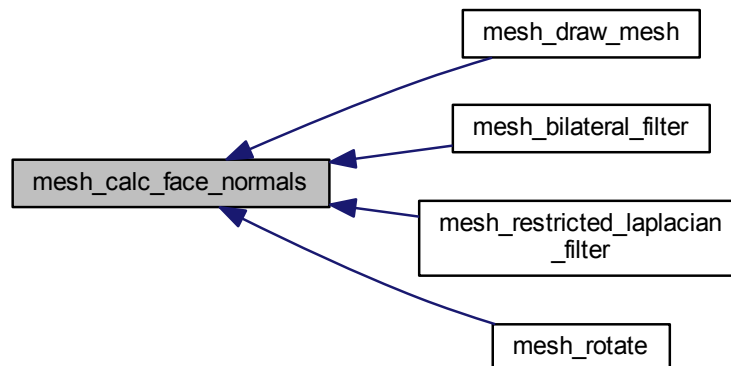
##### Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.6 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:

**5.7.4.7 int mesh\_calc\_vertex\_adjacency ( MESH *m* )**

Computes vertex adjacent faces of a given mesh.

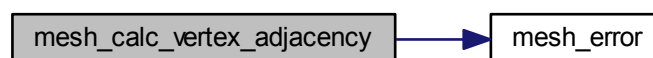
**Parameters**

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

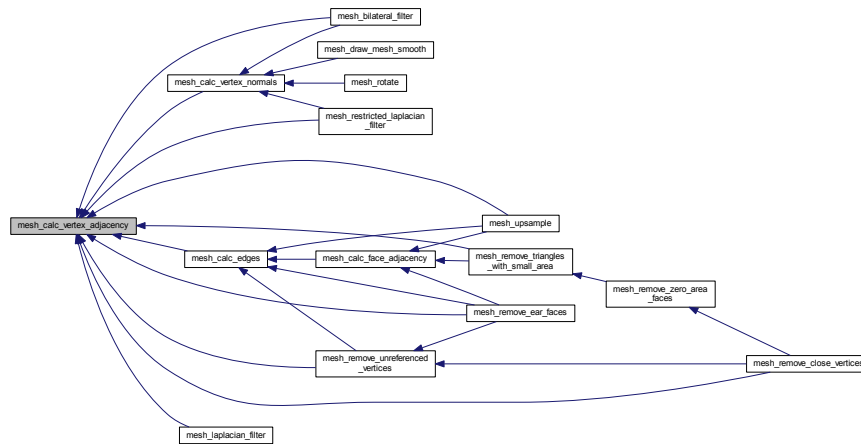
**Returns**

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.8 int mesh\_calc\_vertex\_normals ( MESH *m* )

Computes vertex normals of a given mesh.

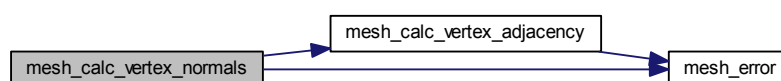
##### Parameters

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

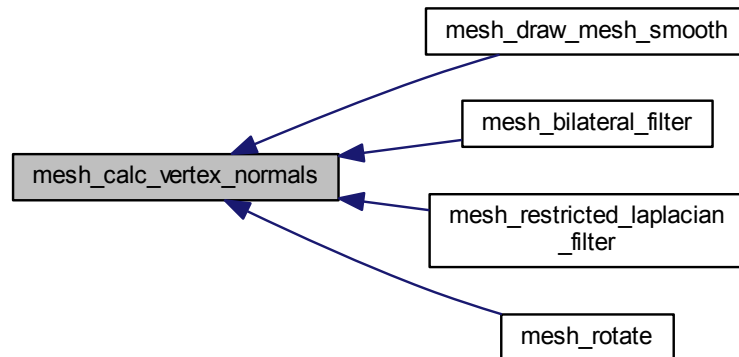
##### Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.9 MESH mesh\_clone\_mesh ( MESH *m*, uint16\_t *flags* )

Clones a given mesh into another mesh.

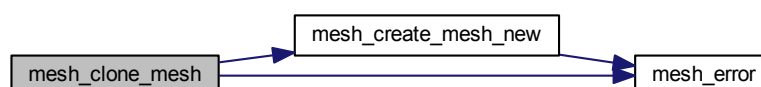
##### Parameters

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:



#### 5.7.4.10 MESH mesh\_combine\_mesh ( MESH *m1*, MESH *m2* )

Combines a given mesh with another given mesh.

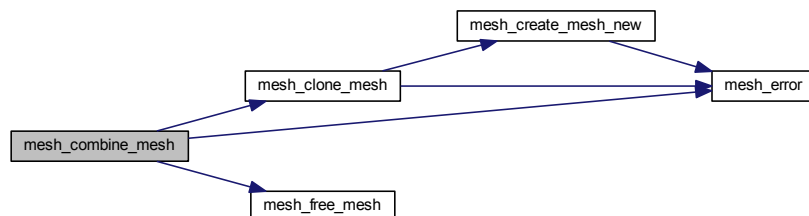
##### Parameters

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:



#### 5.7.4.11 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

#### 5.7.4.12 MESH mesh\_create\_mesh\_new ( )

Creates a new mesh.

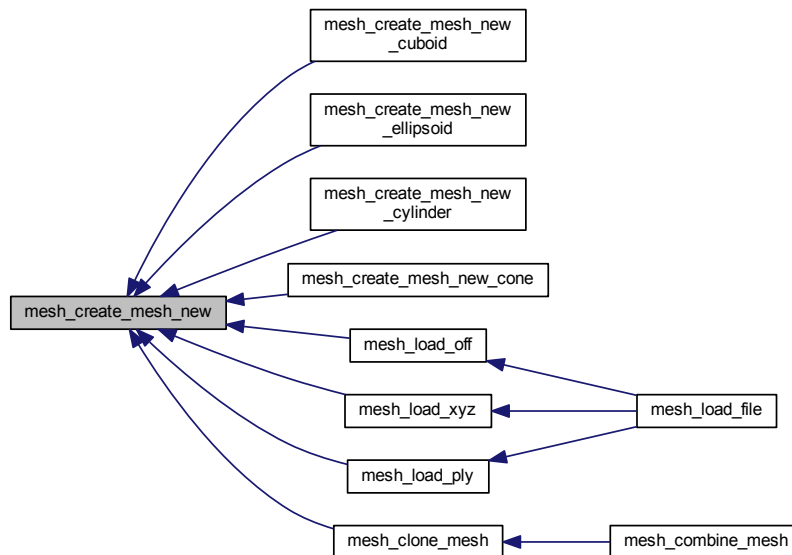
## Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.13 MESH mesh\_create\_mesh\_new\_cone ( MESH\_VECTOR3 sz, MESH\_VECTOR3 pos )

Creates a cone mesh.

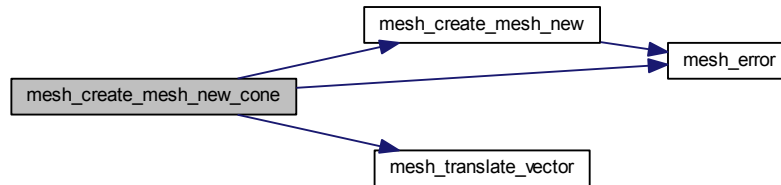
## Parameters

in	sz	Size vector
in	pos	Position vector

**Returns**

Output mesh

Here is the call graph for this function:



#### 5.7.4.14 MESH mesh\_create\_mesh\_new\_cuboid ( MESH\_VECTOR3 sz, MESH\_VECTOR3 pos )

Creates a cuboid mesh.

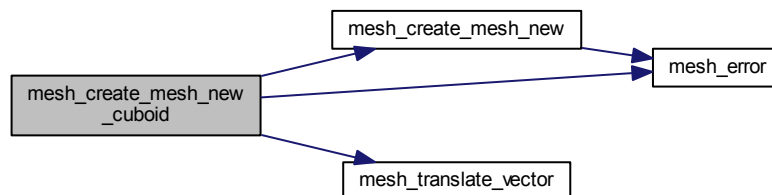
**Parameters**

in	sz	Size vector
in	pos	Position vector

**Returns**

Output mesh

Here is the call graph for this function:



#### 5.7.4.15 MESH mesh\_create\_mesh\_new\_cylinder ( MESH\_VECTOR3 sz, MESH\_VECTOR3 pos )

Creates a cylinder mesh.

**Parameters**


---

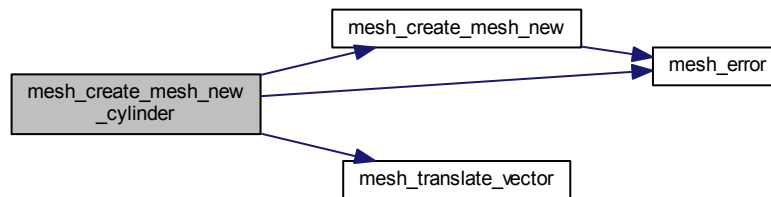


in	<i>sz</i>	Size vector
in	<i>pos</i>	Position vector

**Returns**

Output mesh

Here is the call graph for this function:



#### 5.7.4.16 MESH mesh\_create\_mesh\_new\_ellipsoid ( MESH\_VECTOR3 sz, MESH\_VECTOR3 pos )

Creates an ellipsoid mesh.

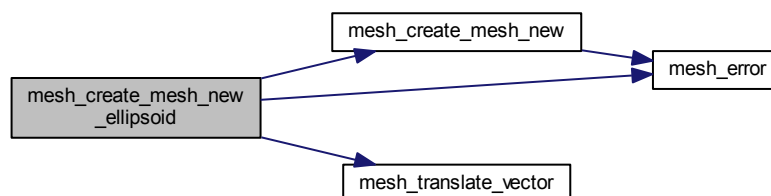
**Parameters**

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

**Returns**

Output mesh

Here is the call graph for this function:



#### 5.7.4.17 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

**5.7.4.18 void mesh\_cross\_vector3 ( MESH\_VECTOR3  $x$ , MESH\_VECTOR3  $y$ , MESH\_VECTOR3  $z$  )**

Computes the cross product of two 3-d vectors.

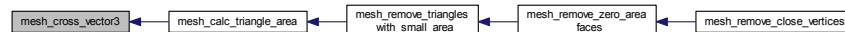
**Parameters**

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:

**5.7.4.19 void mesh\_draw\_mesh ( MESH  $m$  )**

Draws a given mesh in OpenGL context in flat shading.

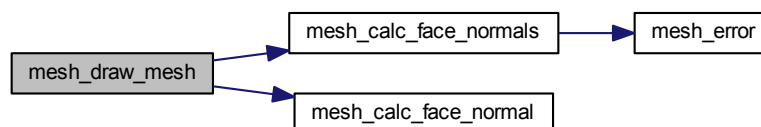
**Parameters**

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

**Returns**

NULL

Here is the call graph for this function:



#### 5.7.4.20 void mesh\_draw\_mesh\_smooth ( MESH *m* )

Draws a given mesh in OpenGL context in smoothing shading.

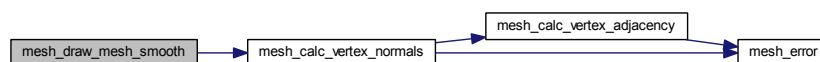
## Parameters

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

## Returns

NULL

Here is the call graph for this function:

5.7.4.21 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



**Parameters**

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

**Returns**

Index or -1

**5.7.4.24 INTDATA mesh\_find3 ( MESH\_STRUCT3 *s*, INTDATA *q* )**

Finds an item in an INTDATA3 structure.

**Parameters**

in	<i>s</i>	Input INTDATA3 structure
in	<i>q</i>	Query INTDATA3

**Returns**

Index or -1

**5.7.4.25 void mesh\_free\_mesh ( MESH *m* )**

Frees a mesh.

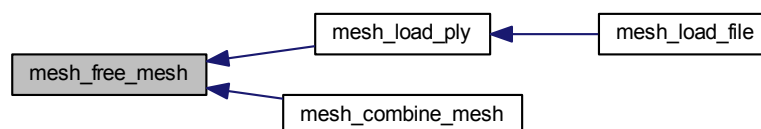
**Parameters**

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

**Returns**

NULL

Here is the caller graph for this function:

**5.7.4.26 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

5.7.4.27 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

5.7.4.28 int mesh\_laplacian\_filter ( MESH *m*, FLOATDATA *r* )

Mesh Laplacian filter.

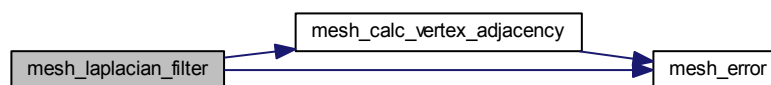
## Parameters

<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:

5.7.4.29 MESH mesh\_load\_file ( const char \* *fname* )

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

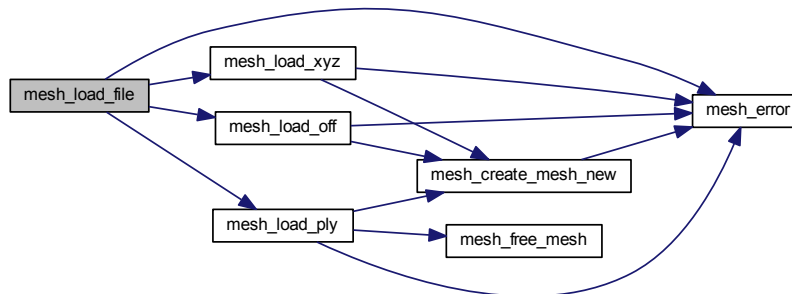
## Parameters

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

**Returns**

Output mesh

Here is the call graph for this function:



#### 5.7.4.30 MESH mesh\_load\_off ( const char \* *fname* )

Reads a mesh from an OFF file.

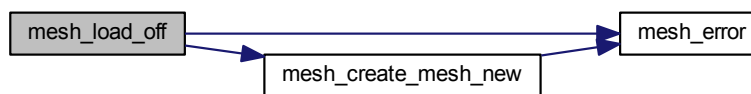
**Parameters**

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

**Returns**

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:





#### 5.7.4.31 MESH mesh\_load\_ply ( const char \* *fname* )

Reads a mesh from a PLY file.

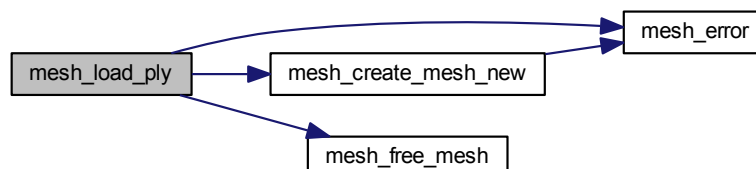
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.32 MESH mesh\_load\_xyz ( const char \* fname )

Read a mesh from an ASC/XYZ file.

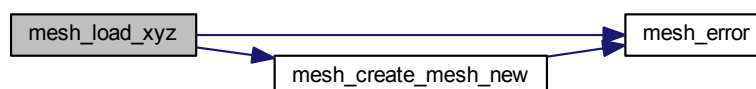
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.33 int mesh\_read\_word ( FILEPOINTER *fp*, char \* *c\_word*, int *sz* )

Reads current word and moves to the next 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

#### 5.7.4.34 int mesh\_read\_word\_only ( FILEPOINTER *fp*, char \* *c\_word*, int *sz* )

Reads current word without moving to the next 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

#### 5.7.4.35 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

#### 5.7.4.36 int mesh\_remove\_boundary\_vertices ( MESH *m*, int *iters* )

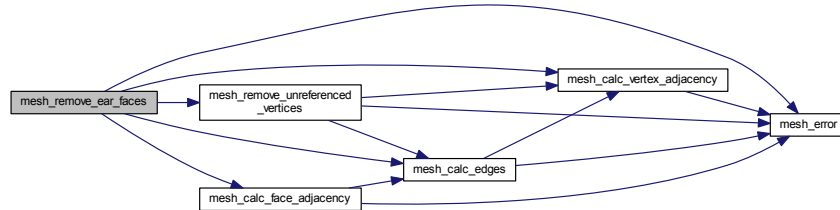
Removes boundary vertices and connecting elements.



## Returns

Error code

Here is the call graph for this function:



#### 5.7.4.39 int mesh\_remove\_triangles\_with\_small\_area ( MESH *m*, FLOATDATA *area* )

Removes triangles with area smaller than a given value.

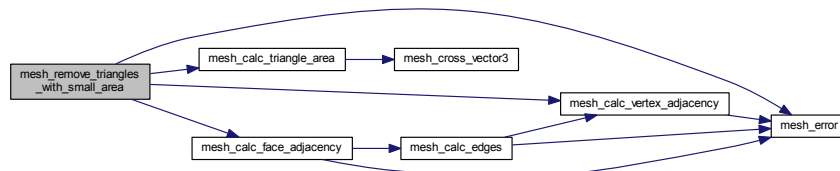
## Parameters

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

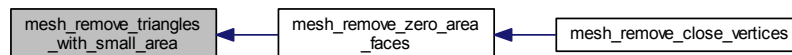
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.40 int mesh\_remove\_unreferenced\_vertices ( MESH *m* )

Removes unreferenced vertices.

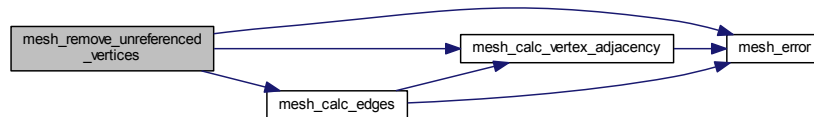
## Parameters

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

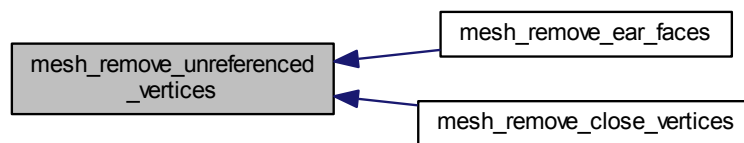
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.41 int mesh\_remove\_zero\_area\_faces ( MESH *m* )

Removes triangles with zero area.

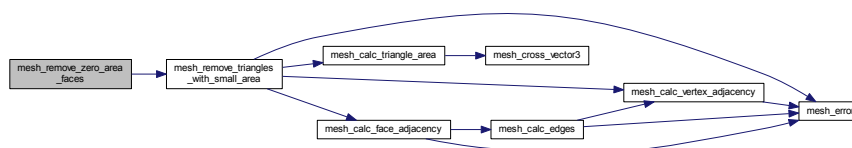
## Parameters

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

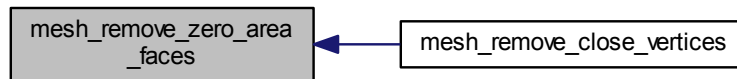
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.42 `int mesh_restricted_laplacian_filter ( MESH m, FLOATDATA r, FLOATDATA ang )`

Restricted Mesh Laplacian filter.

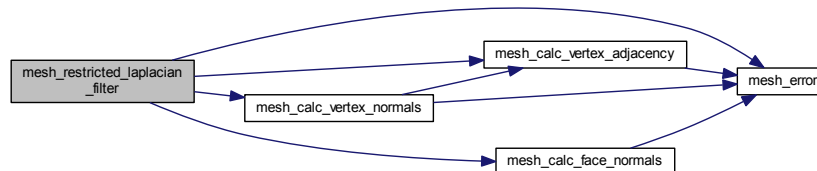
##### Parameters

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

##### Returns

Error code

Here is the call graph for this function:



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

Rotates a mesh by a given rotation.

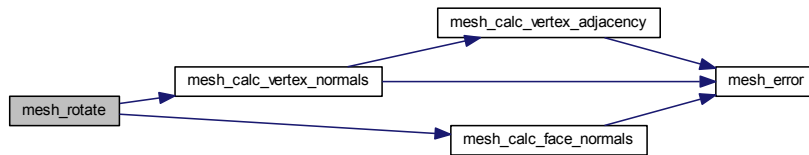
##### Parameters

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

**Returns**

Error code

Here is the call graph for this function:

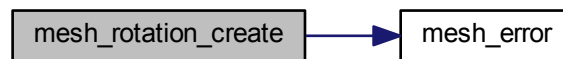
**5.7.4.44 MESH\_ROTATION mesh\_rotation\_create ( )**

Creates a new rotation.

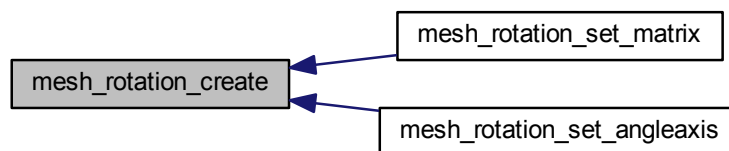
**Returns**

Output rotation

Here is the call graph for this function:



Here is the caller graph for this function:

**5.7.4.45 void mesh\_rotation\_free ( MESH\_ROTATION r )**

Frees a given rotation.



## Parameters

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

## Returns

NULL

**5.7.4.46 MESH\_ROTATION** `mesh_rotation_set_angleaxis ( FLOATDATA ang, MESH_NORMAL axis, MESH_ROTATION r )`

Sets rotation from angle axis.

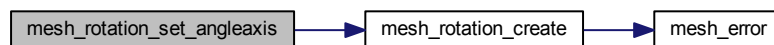
## Parameters

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:



**5.7.4.47 MESH\_ROTATION** `mesh_rotation_set_matrix ( FLOATDATA *mat, MESH_ROTATION r )`

Sets rotation from a matrix.

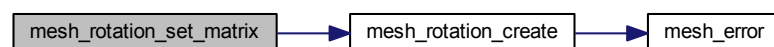
## Parameters

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

## Returns

Output rotation

Here is the call graph for this function:



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

Scales a mesh by x, y and z amounts.

## Parameters

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

## Returns

Error code

5.7.4.49 `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

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

Translates a mesh by x, y and z amounts.

## Parameters

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

## Returns

Error code

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

Translates a mesh by a given 3-d vector.

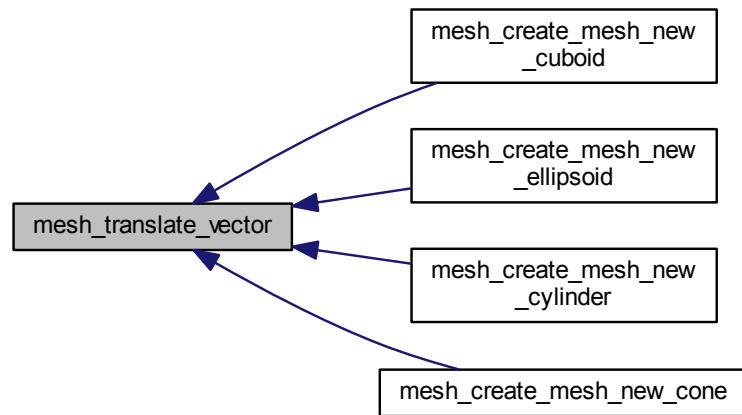
## Parameters

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

**Returns**

Error code

Here is the caller graph for this function:



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

Upsamples a given mesh.

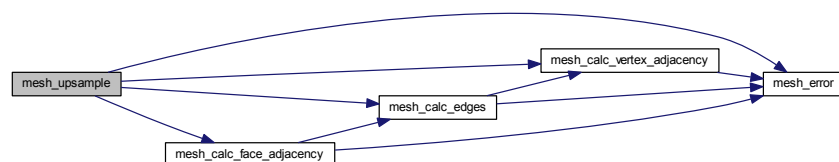
**Parameters**

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

**Returns**

Error code

Here is the call graph for this function:



#### 5.7.4.53 `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

5.7.4.54 int mesh\_write\_file ( MESH *m*, const char \* *fname* )

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

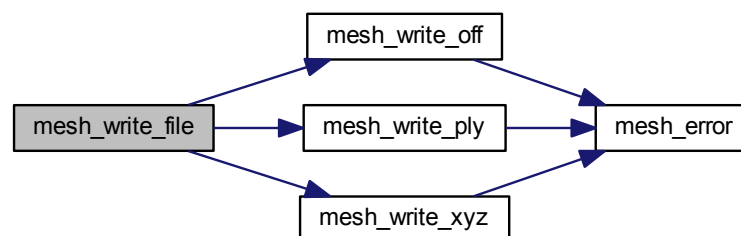
## Parameters

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

## Returns

Error code

Here is the call graph for this function:

5.7.4.55 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:



#### 5.7.4.56 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:



#### 5.7.4.57 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:



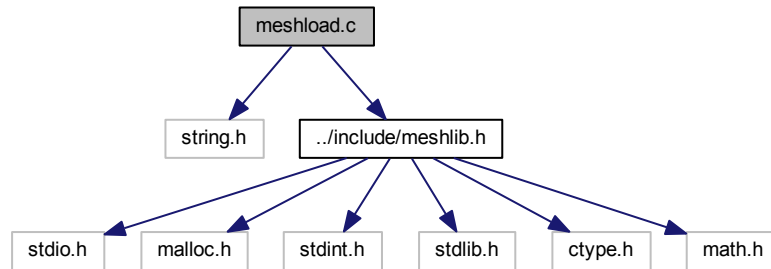
Here is the caller graph for this function:



## 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.



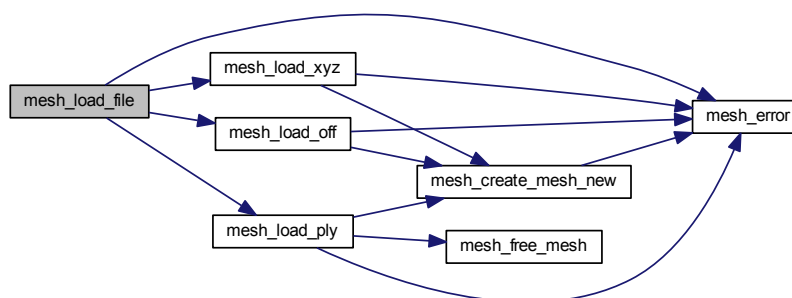
## Parameters

<i>in</i>	<i>fname</i>	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.

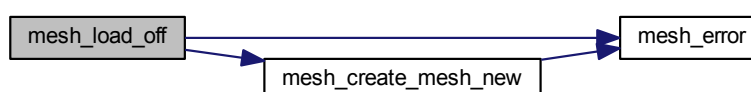
## Parameters

<i>in</i>	<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.

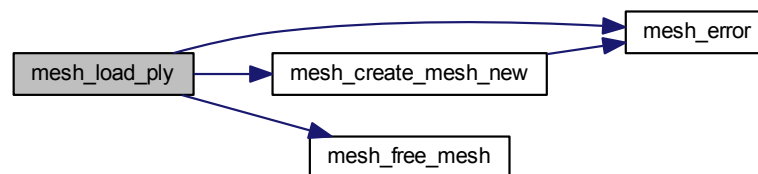
Parameters

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.

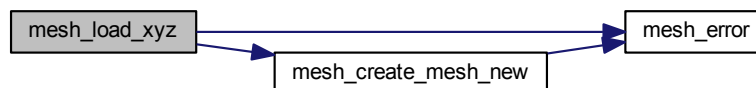
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

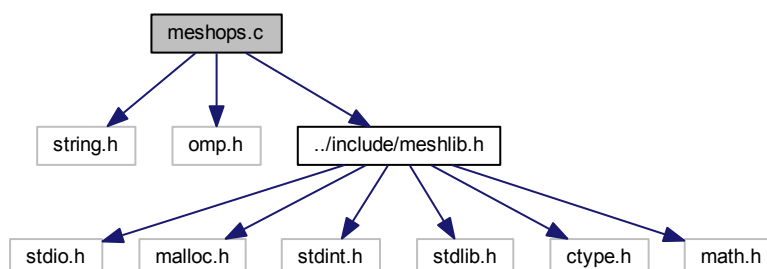


## 5.9 meshops.c File Reference

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.

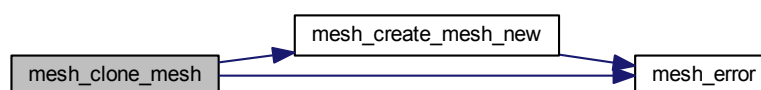
##### Parameters

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.

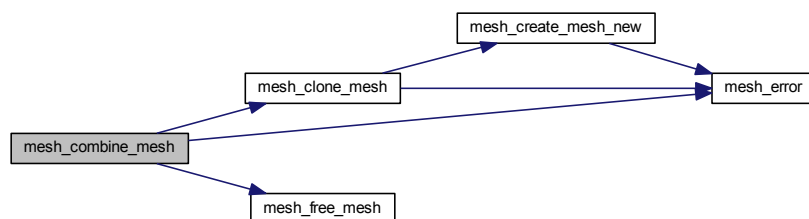
##### Parameters

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:

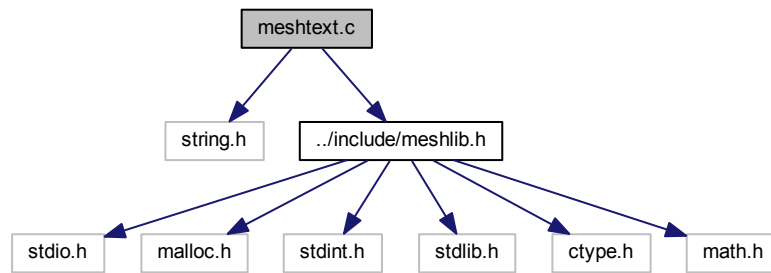


## 5.10 meshtext.c File Reference

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 and moves to the next word.*
- int [mesh\\_read\\_word\\_only](#) (FILEPOINTER fp, char \*c\_word, int sz)  
*Reads current word without moving to the next word.*
- int [mesh\\_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	<i>fp</i>	Pointer to input file
out	<i>count</i>	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	<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**

in	<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 and moves to the next 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

**5.10.2.5 int mesh\_read\_word\_only ( FILEPOINTER *fp*, char \* *c\_word*, int *sz* )**

Reads current word without moving to the next 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

**5.10.2.6 int mesh\_skip\_line ( FILEPOINTER fp )**

Skips to next line.

**Parameters**

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

**Returns**

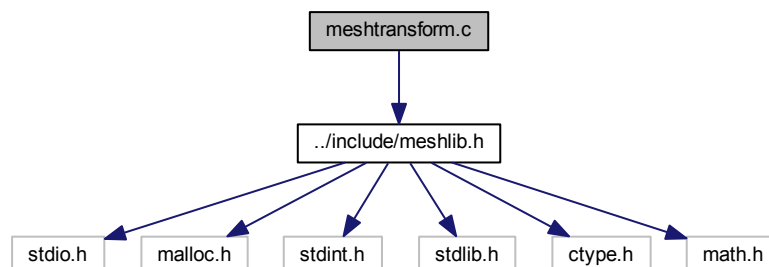
Status 0 - Normal/ 1- EOF

**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.

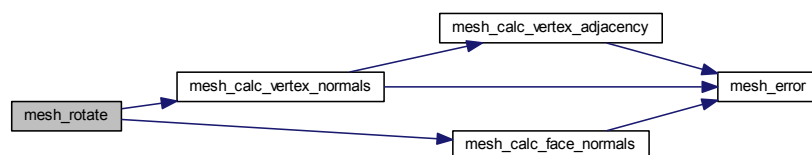
#### Parameters

in	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.

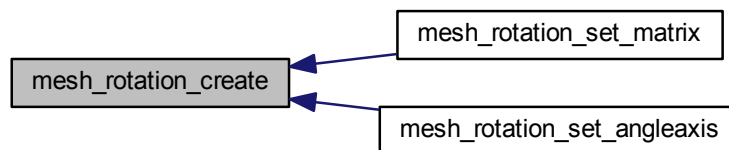
#### Returns

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

<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.

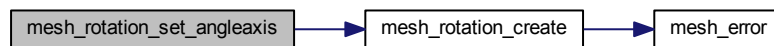
## Parameters

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.

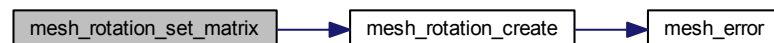
## Parameters

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	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	<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.

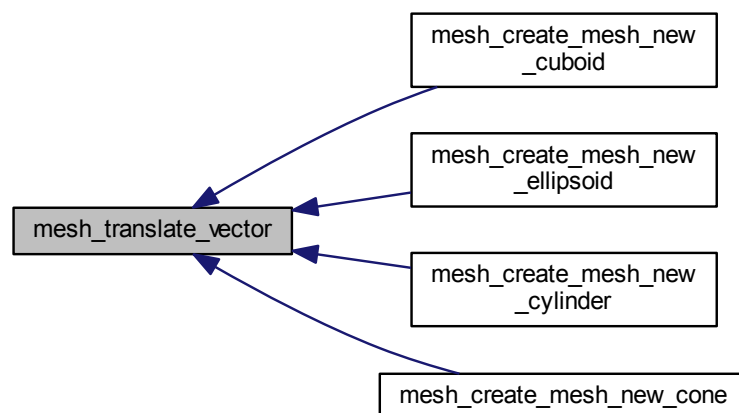
**Parameters**

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

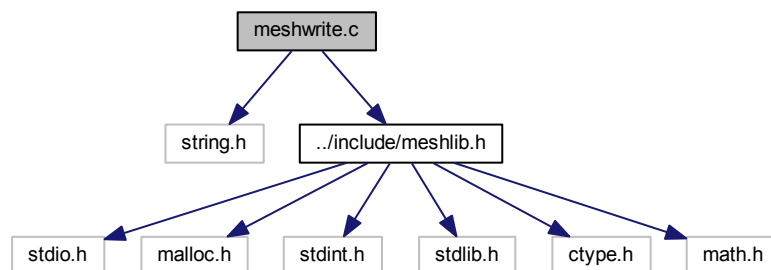
**Returns**

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.

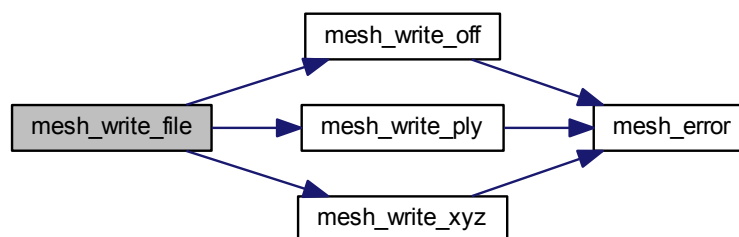
## Parameters

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



### 5.12.2.3 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

`_CRT_SECURE_NO_DEPRECATED`  
    [meshlib.h, 47](#)

a  
    [mesh\\_color, 11](#)

b  
    [mesh\\_color, 11](#)

data  
    [mesh\\_rotation, 12](#)  
    [mesh\\_transform, 14](#)

dummy  
    [mesh, 8](#)

edges  
    [mesh, 8](#)

FILEPOINTER  
    [meshlib.h, 50](#)

FLOATDATA  
    [meshlib.h, 47](#)

faces  
    [mesh, 8](#)  
    [mesh\\_adjface, 10](#)  
    [mesh\\_edge, 11](#)

fareas  
    [mesh, 8](#)

fcolors  
    [mesh, 8](#)

ffaces  
    [mesh, 8](#)

fnormals  
    [mesh, 8](#)

g  
    [mesh\\_color, 11](#)

INTDATA  
    [meshlib.h, 47](#)

INTDATA2  
    [meshlib.h, 50](#)

INTDATA3  
    [meshlib.h, 50](#)

is\_edges  
    [mesh, 8](#)

is\_faces  
    [mesh, 8](#)

is\_fareas  
    [mesh, 8](#)

is\_fcolors

[mesh, 8](#)

is\_ffaces  
    [mesh, 9](#)

is\_fnormals  
    [mesh, 9](#)

is\_loaded  
    [mesh, 9](#)

is\_trimesh  
    [mesh, 9](#)

is\_vcolors  
    [mesh, 9](#)

is\_vertices  
    [mesh, 9](#)

is\_vfaces  
    [mesh, 9](#)

is\_vnormals  
    [mesh, 9](#)

items  
    [mesh\\_struct, 13](#)  
    [mesh\\_struct2, 13](#)  
    [mesh\\_struct3, 13](#)

MESH  
    [meshlib.h, 50](#)  
MESH\_CLONE\_ALL\_PROPS  
    [meshlib.h, 47](#)  
MESH\_CLONE\_EDGES  
    [meshlib.h, 47](#)  
MESH\_CLONE\_F\_ALL\_PROPS  
    [meshlib.h, 47](#)  
MESH\_CLONE\_FACES  
    [meshlib.h, 47](#)  
MESH\_CLONE\_FAREAS  
    [meshlib.h, 47](#)  
MESH\_CLONE\_FCOLORS  
    [meshlib.h, 47](#)  
MESH\_CLONE\_FFACES  
    [meshlib.h, 47](#)  
MESH\_CLONE\_FNORMALS  
    [meshlib.h, 48](#)  
MESH\_CLONE\_V\_ALL\_PROPS  
    [meshlib.h, 48](#)  
MESH\_CLONE\_VCOLORS  
    [meshlib.h, 48](#)  
MESH\_CLONE\_VERTICES  
    [meshlib.h, 48](#)  
MESH\_CLONE\_VFACES  
    [meshlib.h, 48](#)  
MESH\_CLONE\_VNORMALS  
    [meshlib.h, 48](#)

MESH\_COLOR  
     meshlib.h, 50  
 MESH\_EDGE  
     meshlib.h, 50  
 MESH\_ERR\_FNOTOPEN  
     meshlib.h, 48  
 MESH\_ERR\_INCOMPATIBLE  
     meshlib.h, 48  
 MESH\_ERR\_MALLOC  
     meshlib.h, 48  
 MESH\_ERR\_SIZE\_MISMATCH  
     meshlib.h, 48  
 MESH\_ERR\_UNKNOWN  
     meshlib.h, 48  
 MESH\_FACE  
     meshlib.h, 50  
 MESH\_FFACE  
     meshlib.h, 51  
 MESH\_FLOATDATA\_TYPE  
     meshlib.h, 48  
 MESH\_INTDATA\_TYPE  
     meshlib.h, 49  
 MESH\_NORMAL  
     meshlib.h, 51  
 MESH\_ORIGIN\_TYPE\_BUILD  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_COFF  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_NCOFF  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_NOFF  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_OFF  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_PLY\_ASCII  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_PLY\_BINARY\_BIG\_ENDIAN  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_PLY\_BINARY\_LITTLE\_ENDIAN  
     meshlib.h, 49  
 MESH\_ORIGIN\_TYPE\_XYZ  
     meshlib.h, 49  
 MESH\_PI  
     meshlib.h, 49  
 MESH\_ROTATION  
     meshlib.h, 51  
 MESH\_STRUCT  
     meshlib.h, 51  
 MESH\_STRUCT2  
     meshlib.h, 51  
 MESH\_STRUCT3  
     meshlib.h, 51  
 MESH\_TRANSFORM  
     meshlib.h, 52  
 MESH\_TWOP  
     meshlib.h, 49  
 MESH\_VECTOR3  
     meshlib.h, 52  
 MESH\_VERTEX  
     meshlib.h, 52  
 MESH\_VFACE  
     meshlib.h, 52  
 mesh, 7  
     dummy, 8  
     edges, 8  
     faces, 8  
     fareas, 8  
     fcolors, 8  
     ffaces, 8  
     fnormals, 8  
     is\_edges, 8  
     is\_faces, 8  
     is\_fareas, 8  
     is\_fcolors, 8  
     is\_ffaces, 9  
     is\_fnormals, 9  
     is\_loaded, 9  
     is\_trimesh, 9  
     is\_vcolors, 9  
     is\_vertices, 9  
     is\_vfaces, 9  
     is\_vnormals, 9  
     meshlib.h, 50  
     num\_edges, 9  
     num\_faces, 9  
     num\_vertices, 9  
     origin\_type, 9  
     vcolors, 10  
     vertices, 10  
     vfaces, 10  
     vnormals, 10  
 mesh\_adjface, 10  
     faces, 10  
     meshlib.h, 50  
     num\_faces, 10  
 mesh\_bilateral\_filter  
     meshfilter.c, 40  
     meshlib.h, 52  
 mesh\_calc\_edges  
     meshcalc.c, 18  
     meshlib.h, 53  
 mesh\_calc\_face\_adjacency  
     meshcalc.c, 19  
     meshlib.h, 53  
 mesh\_calc\_face\_normal  
     meshcalc.c, 19  
     meshlib.h, 55  
 mesh\_calc\_face\_normals  
     meshcalc.c, 20  
     meshlib.h, 55  
 mesh\_calc\_triangle\_area  
     meshcalc.c, 21  
     meshlib.h, 56  
 mesh\_calc\_vertex\_adjacency  
     meshcalc.c, 21

- meshlib.h, 57
- mesh\_calc\_vertex\_normals
  - meshcalc.c, 23
  - meshlib.h, 58
- mesh\_clone\_mesh
  - meshlib.h, 59
  - meshops.c, 90
- mesh\_color, 10
  - a, 11
  - b, 11
  - g, 11
  - meshlib.h, 50
  - r, 11
- mesh\_combine\_mesh
  - meshlib.h, 60
  - meshops.c, 91
- mesh\_count\_words\_in\_line
  - meshlib.h, 60
  - meshtext.c, 92
- mesh\_create\_mesh\_new
  - meshcreate.c, 32
  - meshlib.h, 60
- mesh\_create\_mesh\_new\_cone
  - meshcreate.c, 33
  - meshlib.h, 61
- mesh\_create\_mesh\_new\_cuboid
  - meshcreate.c, 34
  - meshlib.h, 62
- mesh\_create\_mesh\_new\_cylinder
  - meshcreate.c, 34
  - meshlib.h, 62
- mesh\_create\_mesh\_new\_ellipsoid
  - meshcreate.c, 35
  - meshlib.h, 63
- mesh\_cross\_normal
  - meshcalc.c, 24
  - meshlib.h, 63
- mesh\_cross\_vector3
  - meshcalc.c, 24
  - meshlib.h, 64
- mesh\_draw\_mesh
  - meshdraw.c, 37
  - meshlib.h, 64
- mesh\_draw\_mesh\_smooth
  - meshdraw.c, 37
  - meshlib.h, 64
- mesh\_edge, 11
  - faces, 11
  - meshlib.h, 50
  - vertices, 11
- mesh\_error
  - mesherror.c, 38
  - meshlib.h, 66
- mesh\_face, 12
  - meshlib.h, 50
  - num\_vertices, 12
  - vertices, 12
- mesh\_fface
  - meshlib.h, 51
- mesh\_find
  - meshcalc.c, 25
  - meshlib.h, 67
- mesh\_find2
  - meshcalc.c, 25
  - meshlib.h, 67
- mesh\_find3
  - meshcalc.c, 25
  - meshlib.h, 68
- mesh\_free\_mesh
  - meshcreate.c, 35
  - meshlib.h, 68
- mesh\_go\_next\_word
  - meshlib.h, 68
  - meshtext.c, 93
- mesh\_isnumeric
  - meshlib.h, 69
  - meshtext.c, 93
- mesh\_laplacian\_filter
  - meshfilter.c, 41
  - meshlib.h, 69
- mesh\_load\_file
  - meshlib.h, 69
  - meshload.c, 86
- mesh\_load\_off
  - meshlib.h, 70
  - meshload.c, 87
- mesh\_load\_ply
  - meshlib.h, 70
  - meshload.c, 88
- mesh\_load\_xyz
  - meshlib.h, 72
  - meshload.c, 88
- mesh\_normal
  - meshlib.h, 51
- mesh\_read\_word
  - meshlib.h, 73
  - meshtext.c, 93
- mesh\_read\_word\_only
  - meshlib.h, 73
  - meshtext.c, 93
- mesh\_remove\_boundary\_faces
  - meshclean.c, 28
  - meshlib.h, 73
- mesh\_remove\_boundary\_vertices
  - meshclean.c, 28
  - meshlib.h, 73
- mesh\_remove\_close\_vertices
  - meshclean.c, 28
  - meshlib.h, 74
- mesh\_remove\_ear\_faces
  - meshclean.c, 29
  - meshlib.h, 74
- mesh\_remove\_triangles\_with\_small\_area
  - meshclean.c, 29
  - meshlib.h, 75
- mesh\_remove\_unreferenced\_vertices

- meshclean.c, 30
- meshlib.h, 75
- mesh\_remove\_zero\_area\_faces
  - meshclean.c, 31
  - meshlib.h, 76
- mesh\_restricted\_laplacian\_filter
  - meshfilter.c, 41
  - meshlib.h, 77
- mesh\_rotate
  - meshlib.h, 77
  - meshtransform.c, 95
- mesh\_rotation, 12
  - data, 12
  - meshlib.h, 51
- mesh\_rotation\_create
  - meshlib.h, 78
  - meshtransform.c, 95
- mesh\_rotation\_free
  - meshlib.h, 78
  - meshtransform.c, 96
- mesh\_rotation\_set\_angleaxis
  - meshlib.h, 79
  - meshtransform.c, 96
- mesh\_rotation\_set\_matrix
  - meshlib.h, 79
  - meshtransform.c, 97
- mesh\_scale
  - meshlib.h, 79
  - meshtransform.c, 97
- mesh\_skip\_line
  - meshlib.h, 81
  - meshtext.c, 94
- mesh\_struct, 12
  - items, 13
  - meshlib.h, 51
  - num\_items, 13
- mesh\_struct2, 13
  - items, 13
  - meshlib.h, 51
  - num\_items, 13
- mesh\_struct3, 13
  - items, 13
  - meshlib.h, 51
  - num\_items, 13
- mesh\_transform, 14
  - data, 14
  - meshlib.h, 52
- mesh\_translate
  - meshlib.h, 81
  - meshtransform.c, 98
- mesh\_translate\_vector
  - meshlib.h, 81
  - meshtransform.c, 98
- mesh\_upsample
  - meshcalc.c, 25
  - meshlib.h, 82
- mesh\_vector3, 14
  - meshlib.h, 52
  - x, 14
  - y, 14
  - z, 14
- mesh\_vertex
  - meshlib.h, 52
- mesh\_vertex\_rotate
  - meshlib.h, 82
  - meshtransform.c, 98
- mesh\_vface
  - meshlib.h, 52
- mesh\_write\_file
  - meshlib.h, 83
  - meshwrite.c, 101
- mesh\_write\_off
  - meshlib.h, 83
  - meshwrite.c, 101
- mesh\_write\_ply
  - meshlib.h, 84
  - meshwrite.c, 102
- mesh\_write\_xyz
  - meshlib.h, 85
  - meshwrite.c, 103
- meshcalc.c, 17
  - mesh\_calc\_edges, 18
  - mesh\_calc\_face\_adjacency, 19
  - mesh\_calc\_face\_normal, 19
  - mesh\_calc\_face\_normals, 20
  - mesh\_calc\_triangle\_area, 21
  - mesh\_calc\_vertex\_adjacency, 21
  - mesh\_calc\_vertex\_normals, 23
  - mesh\_cross\_normal, 24
  - mesh\_cross\_vector3, 24
  - mesh\_find, 25
  - mesh\_find2, 25
  - mesh\_find3, 25
  - mesh\_upsample, 25
- meshclean.c, 27
  - mesh\_remove\_boundary\_faces, 28
  - mesh\_remove\_boundary\_vertices, 28
  - mesh\_remove\_close\_vertices, 28
  - mesh\_remove\_ear\_faces, 29
  - mesh\_remove\_triangles\_with\_small\_area, 29
  - mesh\_remove\_unreferenced\_vertices, 30
  - mesh\_remove\_zero\_area\_faces, 31
- meshcreate.c, 31
  - mesh\_create\_mesh\_new, 32
  - mesh\_create\_mesh\_new\_cone, 33
  - mesh\_create\_mesh\_new\_cuboid, 34
  - mesh\_create\_mesh\_new\_cylinder, 34
  - mesh\_create\_mesh\_new\_ellipsoid, 35
  - mesh\_free\_mesh, 35
- meshdraw.c, 36
  - mesh\_draw\_mesh, 37
  - mesh\_draw\_mesh\_smooth, 37
- mesherror.c, 38
  - mesh\_error, 38
- meshfilter.c, 39
  - mesh\_bilateral\_filter, 40

- mesh\_laplacian\_filter, 41
- mesh\_restricted\_laplacian\_filter, 41
- meshlib.h, 42
- \_CRT\_SECURE\_NO\_DEPRECATED, 47
- FILEPOINTER, 50
- FLOATDATA, 47
- INTDATA, 47
- INTDATA2, 50
- INTDATA3, 50
- MESH, 50
- MESH\_CLONE\_ALL\_PROPS, 47
- MESH\_CLONE\_EDGES, 47
- MESH\_CLONE\_F\_ALL\_PROPS, 47
- MESH\_CLONE\_FACES, 47
- MESH\_CLONE\_FAREAS, 47
- MESH\_CLONE\_FCOLORS, 47
- MESH\_CLONE\_FFACES, 47
- MESH\_CLONE\_FNORMALS, 48
- MESH\_CLONE\_V\_ALL\_PROPS, 48
- MESH\_CLONE\_VCOLORS, 48
- MESH\_CLONE\_VERTICES, 48
- MESH\_CLONE\_VFACES, 48
- MESH\_CLONE\_VNORMALS, 48
- MESH\_COLOR, 50
- MESH\_EDGE, 50
- MESH\_ERR\_FNOTOPEN, 48
- MESH\_ERR\_INCOMPATIBLE, 48
- MESH\_ERR\_MALLOC, 48
- MESH\_ERR\_SIZE\_MISMATCH, 48
- MESH\_ERR\_UNKNOWN, 48
- MESH\_FACE, 50
- MESH\_FFACE, 51
- MESH\_FLOATDATA\_TYPE, 48
- MESH\_INTDATA\_TYPE, 49
- MESH\_NORMAL, 51
- MESH\_ORIGIN\_TYPE\_BUILD, 49
- MESH\_ORIGIN\_TYPE\_COFF, 49
- MESH\_ORIGIN\_TYPE\_NCOFF, 49
- MESH\_ORIGIN\_TYPE\_NOFF, 49
- MESH\_ORIGIN\_TYPE\_OFF, 49
- MESH\_ORIGIN\_TYPE\_PLY\_ASCII, 49
- MESH\_ORIGIN\_TYPE\_PLY\_BINARY\_BIG\_ENDIAN, 49
- MESH\_ORIGIN\_TYPE\_PLY\_BINARY\_LITTLE\_ENDIAN, 49
- MESH\_ORIGIN\_TYPE\_XYZ, 49
- MESH\_PI, 49
- MESH\_ROTATION, 51
- MESH\_STRUCT, 51
- MESH\_STRUCT2, 51
- MESH\_STRUCT3, 51
- MESH\_TRANSFORM, 52
- MESH\_TWOPI, 49
- MESH\_VECTOR3, 52
- MESH\_VERTEX, 52
- MESH\_VFACE, 52
- mesh, 50
- mesh\_adjface, 50
- mesh\_bilateral\_filter, 52
- mesh\_calc\_edges, 53
- mesh\_calc\_face\_adjacency, 53
- mesh\_calc\_face\_normal, 55
- mesh\_calc\_face\_normals, 55
- mesh\_calc\_triangle\_area, 56
- mesh\_calc\_vertex\_adjacency, 57
- mesh\_calc\_vertex\_normals, 58
- mesh\_clone\_mesh, 59
- mesh\_color, 50
- mesh\_combine\_mesh, 60
- mesh\_count\_words\_in\_line, 60
- mesh\_create\_mesh\_new, 60
- mesh\_create\_mesh\_new\_cone, 61
- mesh\_create\_mesh\_new\_cuboid, 62
- mesh\_create\_mesh\_new\_cylinder, 62
- mesh\_create\_mesh\_new\_ellipsoid, 63
- mesh\_cross\_normal, 63
- mesh\_cross\_vector3, 64
- mesh\_draw\_mesh, 64
- mesh\_draw\_mesh\_smooth, 64
- mesh\_edge, 50
- mesh\_error, 66
- mesh\_face, 50
- mesh\_fface, 51
- mesh\_find, 67
- mesh\_find2, 67
- mesh\_find3, 68
- mesh\_free\_mesh, 68
- mesh\_go\_next\_word, 68
- mesh\_isnumeric, 69
- mesh\_laplacian\_filter, 69
- mesh\_load\_file, 69
- mesh\_load\_off, 70
- mesh\_load\_ply, 70
- mesh\_load\_xyz, 72
- mesh\_normal, 51
- mesh\_read\_word, 73
- mesh\_read\_word\_only, 73
- mesh\_remove\_boundary\_faces, 73
- mesh\_remove\_boundary\_vertices, 73
- mesh\_remove\_close\_vertices, 74
- mesh\_remove\_ear\_faces, 74
- mesh\_remove\_triangles\_with\_small\_area, 75
- mesh\_remove\_unreferenced\_vertices, 75
- mesh\_remove\_zero\_area\_faces, 76
- mesh\_restricted\_laplacian\_filter, 77
- mesh\_rotate, 77
- mesh\_rotation, 51
- mesh\_rotation\_create, 78
- mesh\_rotation\_free, 78
- mesh\_rotation\_set\_angleaxis, 79
- mesh\_rotation\_set\_matrix, 79
- mesh\_scale, 79
- mesh\_skip\_line, 81
- mesh\_struct, 51
- mesh\_struct2, 51
- mesh\_struct3, 51

- mesh\_transform, 52
- mesh\_translate, 81
- mesh\_translate\_vector, 81
- mesh\_upsample, 82
- mesh\_vector3, 52
- mesh\_vertex, 52
- mesh\_vertex\_rotate, 82
- mesh\_vface, 52
- mesh\_write\_file, 83
- mesh\_write\_off, 83
- mesh\_write\_ply, 84
- mesh\_write\_xyz, 85
- meshload.c, 85
  - mesh\_load\_file, 86
  - mesh\_load\_off, 87
  - mesh\_load\_ply, 88
  - mesh\_load\_xyz, 88
- meshops.c, 89
  - mesh\_clone\_mesh, 90
  - mesh\_combine\_mesh, 91
- meshtext.c, 91
  - mesh\_count\_words\_in\_line, 92
  - mesh\_go\_next\_word, 93
  - mesh\_isnumeric, 93
  - mesh\_read\_word, 93
  - mesh\_read\_word\_only, 93
  - mesh\_skip\_line, 94
- meshtransform.c, 94
  - mesh\_rotate, 95
  - mesh\_rotation\_create, 95
  - mesh\_rotation\_free, 96
  - mesh\_rotation\_set\_angleaxis, 96
  - mesh\_rotation\_set\_matrix, 97
  - mesh\_scale, 97
  - mesh\_translate, 98
  - mesh\_translate\_vector, 98
  - mesh\_vertex\_rotate, 98
- meshwrite.c, 100
  - mesh\_write\_file, 101
  - mesh\_write\_off, 101
  - mesh\_write\_ply, 102
  - mesh\_write\_xyz, 103
- num\_edges
  - mesh, 9
- num\_faces
  - mesh, 9
  - mesh\_adjface, 10
- num\_items
  - mesh\_struct, 13
  - mesh\_struct2, 13
  - mesh\_struct3, 13
- num\_vertices
  - mesh, 9
  - mesh\_face, 12
- origin\_type
  - mesh, 9
- r
  - mesh\_color, 11
- vcolors
  - mesh, 10
- vertices
  - mesh, 10
  - mesh\_edge, 11
  - mesh\_face, 12
- vfaces
  - mesh, 10
- vnormals
  - mesh, 10
- x
  - mesh\_vector3, 14
- y
  - mesh\_vector3, 14
- z
  - mesh\_vector3, 14