

**MeshLib**

**1.3.0.0**

Generated by Doxygen 1.8.1.1

Mon Jun 8 2015 09:27:47



# 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	faces . . . . .	8
4.1.1.3	fareas . . . . .	8
4.1.1.4	fcolors . . . . .	8
4.1.1.5	fnormals . . . . .	8
4.1.1.6	is_faces . . . . .	8
4.1.1.7	is_fareas . . . . .	8
4.1.1.8	is_fcolors . . . . .	8
4.1.1.9	is_fnormals . . . . .	8
4.1.1.10	is_loaded . . . . .	8
4.1.1.11	is_trimesh . . . . .	8
4.1.1.12	is_vcolors . . . . .	9
4.1.1.13	is_vertices . . . . .	9
4.1.1.14	is_vfaces . . . . .	9
4.1.1.15	is_vnormals . . . . .	9
4.1.1.16	num_faces . . . . .	9
4.1.1.17	num_vertices . . . . .	9
4.1.1.18	origin_type . . . . .	9

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

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

5.6.1	Detailed Description	35
5.6.2	Function Documentation	35
5.6.2.1	mesh_bilateral_filter	35
5.6.2.2	mesh_laplacian_filter	36
5.6.2.3	mesh_restricted_laplacian_filter	36
5.7	meshlib.h File Reference	37
5.7.1	Detailed Description	41
5.7.2	Macro Definition Documentation	41
5.7.2.1	_CRT_SECURE_NO_DEPRECATED	41
5.7.2.2	FLOATDATA	41
5.7.2.3	INTDATA	41
5.7.2.4	MESH_ERR_FNOTOPEN	41
5.7.2.5	MESH_ERR_MALLOC	41
5.7.2.6	MESH_ERR_SIZE_MISMATCH	41
5.7.2.7	MESH_ERR_UNKNOWN	41
5.7.2.8	MESH_FLOATDATA_TYPE	41
5.7.2.9	MESH_INTDATA_TYPE	41
5.7.2.10	MESH_ORIGIN_TYPE_BUILD	42
5.7.2.11	MESH_ORIGIN_TYPE_COFF	42
5.7.2.12	MESH_ORIGIN_TYPE_NCOFF	42
5.7.2.13	MESH_ORIGIN_TYPE_NOFF	42
5.7.2.14	MESH_ORIGIN_TYPE_OFF	42
5.7.2.15	MESH_ORIGIN_TYPE_PLY_ASCII	42
5.7.2.16	MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN	42
5.7.2.17	MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN	42
5.7.2.18	MESH_ORIGIN_TYPE_XYZ	42
5.7.2.19	MESH_PI	42
5.7.3	Typedef Documentation	42
5.7.3.1	FILEPOINTER	42
5.7.3.2	INTDATA2	42
5.7.3.3	mesh	43
5.7.3.4	MESH	43
5.7.3.5	mesh_color	43
5.7.3.6	MESH_COLOR	43
5.7.3.7	mesh_face	43
5.7.3.8	MESH_FACE	43
5.7.3.9	mesh_normal	43
5.7.3.10	MESH_NORMAL	43
5.7.3.11	mesh_rotation	43
5.7.3.12	MESH_ROTATION	43

5.7.3.13	<a href="#">mesh_struct</a>	43
5.7.3.14	<a href="#">MESH_STRUCT</a>	43
5.7.3.15	<a href="#">mesh_struct2</a>	43
5.7.3.16	<a href="#">MESH_STRUCT2</a>	44
5.7.3.17	<a href="#">mesh_transform</a>	44
5.7.3.18	<a href="#">MESH_TRANSFORM</a>	44
5.7.3.19	<a href="#">mesh_vector3</a>	44
5.7.3.20	<a href="#">MESH_VECTOR3</a>	44
5.7.3.21	<a href="#">mesh_vertex</a>	44
5.7.3.22	<a href="#">MESH_VERTEX</a>	44
5.7.3.23	<a href="#">mesh_vface</a>	44
5.7.3.24	<a href="#">MESH_VFACE</a>	44
5.7.4	<a href="#">Function Documentation</a>	44
5.7.4.1	<a href="#">mesh_bilateral_filter</a>	44
5.7.4.2	<a href="#">mesh_calc_face_normal</a>	45
5.7.4.3	<a href="#">mesh_calc_face_normals</a>	45
5.7.4.4	<a href="#">mesh_calc_triangle_area</a>	46
5.7.4.5	<a href="#">mesh_calc_vertex_adjacency</a>	47
5.7.4.6	<a href="#">mesh_calc_vertex_normals</a>	48
5.7.4.7	<a href="#">mesh_count_words_in_line</a>	49
5.7.4.8	<a href="#">mesh_create_mesh_new</a>	49
5.7.4.9	<a href="#">mesh_create_mesh_new_cone</a>	50
5.7.4.10	<a href="#">mesh_create_mesh_new_cuboid</a>	50
5.7.4.11	<a href="#">mesh_create_mesh_new_cylinder</a>	51
5.7.4.12	<a href="#">mesh_create_mesh_new_ellipsoid</a>	51
5.7.4.13	<a href="#">mesh_cross_normal</a>	52
5.7.4.14	<a href="#">mesh_cross_vector3</a>	52
5.7.4.15	<a href="#">mesh_draw_mesh</a>	53
5.7.4.16	<a href="#">mesh_error</a>	53
5.7.4.17	<a href="#">mesh_find</a>	54
5.7.4.18	<a href="#">mesh_find2</a>	55
5.7.4.19	<a href="#">mesh_free_mesh</a>	55
5.7.4.20	<a href="#">mesh_go_next_word</a>	55
5.7.4.21	<a href="#">mesh_isnumeric</a>	55
5.7.4.22	<a href="#">mesh_laplacian_filter</a>	56
5.7.4.23	<a href="#">mesh_load_file</a>	56
5.7.4.24	<a href="#">mesh_load_off</a>	56
5.7.4.25	<a href="#">mesh_load_ply</a>	57
5.7.4.26	<a href="#">mesh_load_xyz</a>	58
5.7.4.27	<a href="#">mesh_read_word</a>	58

5.7.4.28	<a href="#">mesh_remove_boundary_faces</a>	59
5.7.4.29	<a href="#">mesh_remove_boundary_vertices</a>	59
5.7.4.30	<a href="#">mesh_remove_close_vertices</a>	59
5.7.4.31	<a href="#">mesh_remove_ear_faces</a>	60
5.7.4.32	<a href="#">mesh_remove_triangles_with_small_area</a>	60
5.7.4.33	<a href="#">mesh_remove_unreferenced_vertices</a>	61
5.7.4.34	<a href="#">mesh_remove_zero_area_faces</a>	62
5.7.4.35	<a href="#">mesh_restricted_laplacian_filter</a>	62
5.7.4.36	<a href="#">mesh_rotate</a>	63
5.7.4.37	<a href="#">mesh_rotation_create</a>	63
5.7.4.38	<a href="#">mesh_rotation_free</a>	64
5.7.4.39	<a href="#">mesh_rotation_set_angleaxis</a>	64
5.7.4.40	<a href="#">mesh_rotation_set_matrix</a>	65
5.7.4.41	<a href="#">mesh_scale</a>	65
5.7.4.42	<a href="#">mesh_skip_line</a>	66
5.7.4.43	<a href="#">mesh_translate</a>	66
5.7.4.44	<a href="#">mesh_translate_vector</a>	66
5.7.4.45	<a href="#">mesh_upsample</a>	67
5.7.4.46	<a href="#">mesh_vertex_rotate</a>	67
5.7.4.47	<a href="#">mesh_write_file</a>	68
5.7.4.48	<a href="#">mesh_write_off</a>	68
5.7.4.49	<a href="#">mesh_write_ply</a>	69
5.7.4.50	<a href="#">mesh_write_xyz</a>	70
5.8	<a href="#">meshload.c File Reference</a>	70
5.8.1	<a href="#">Detailed Description</a>	71
5.8.2	<a href="#">Function Documentation</a>	71
5.8.2.1	<a href="#">mesh_load_file</a>	71
5.8.2.2	<a href="#">mesh_load_off</a>	72
5.8.2.3	<a href="#">mesh_load_ply</a>	73
5.8.2.4	<a href="#">mesh_load_xyz</a>	73
5.9	<a href="#">meshtext.c File Reference</a>	74
5.9.1	<a href="#">Detailed Description</a>	75
5.9.2	<a href="#">Function Documentation</a>	75
5.9.2.1	<a href="#">mesh_count_words_in_line</a>	75
5.9.2.2	<a href="#">mesh_go_next_word</a>	75
5.9.2.3	<a href="#">mesh_isnumeric</a>	76
5.9.2.4	<a href="#">mesh_read_word</a>	76
5.9.2.5	<a href="#">mesh_skip_line</a>	76
5.10	<a href="#">meshttransform.c File Reference</a>	76
5.10.1	<a href="#">Detailed Description</a>	77



5.10.2	Function Documentation	78
5.10.2.1	mesh_rotate	78
5.10.2.2	mesh_rotation_create	78
5.10.2.3	mesh_rotation_free	79
5.10.2.4	mesh_rotation_set_angleaxis	79
5.10.2.5	mesh_rotation_set_matrix	80
5.10.2.6	mesh_scale	80
5.10.2.7	mesh_translate	80
5.10.2.8	mesh_translate_vector	81
5.10.2.9	mesh_vertex_rotate	81
5.11	meshwrite.c File Reference	81
5.11.1	Detailed Description	82
5.11.2	Function Documentation	82
5.11.2.1	mesh_write_file	82
5.11.2.2	mesh_write_off	83
5.11.2.3	mesh_write_ply	84
5.11.2.4	mesh_write_xyz	84



# 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_color</a>	10
<a href="#">mesh_face</a>	10
<a href="#">mesh_rotation</a>	11
<a href="#">mesh_struct</a>	11
<a href="#">mesh_struct2</a>	12
<a href="#">mesh_transform</a>	12
<a href="#">mesh_vector3</a>	12
<a href="#">mesh_vface</a>	13



## 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 . . . . .	15
<a href="#">meshclean.c</a>	This file contains functions pertaining to different mesh cleaning algorithms . . . . .	21
<a href="#">meshcreate.c</a>	This file contains functions pertaining to mesh creation and freeing . . . . .	26
<a href="#">meshdraw.c</a>	This file contains functions pertaining to mesh drawing in OpenGL . . . . .	30
<a href="#">mesherror.c</a>	This file contains functions pertaining to handling errors . . . . .	32
<a href="#">meshfilter.c</a>	This file contains functions pertaining to different mesh filtering algorithms . . . . .	34
<a href="#">meshlib.h</a>	This header file contains declarations of all functions of meshlib . . . . .	37
<a href="#">meshload.c</a>	This file contains functions pertaining to loading different mesh file types . . . . .	70
<a href="#">meshtext.c</a>	This file contains functions pertaining to different text routines . . . . .	74
<a href="#">meshtransform.c</a>	This file contains functions pertaining to different mesh transformations . . . . .	76
<a href="#">meshwrite.c</a>	This file contains functions pertaining to writing different mesh file types . . . . .	81





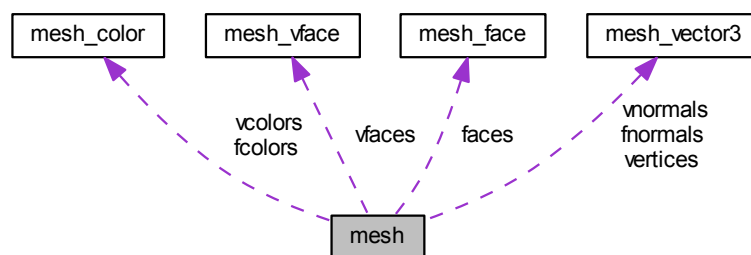
## Chapter 4

# Data Structure Documentation

### 4.1 mesh Struct Reference

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

Collaboration diagram for mesh:



#### Data Fields

- `uint8_t origin_type`
- `uint8_t is_loaded`
- `uint8_t is_vertices`
- `uint8_t is_faces`
- `uint8_t is_vnormals`
- `uint8_t is_fnormals`
- `uint8_t is_vcolors`
- `uint8_t is_fcolors`
- `uint8_t is_vfaces`
- `uint8_t is_fareas`
- `INTDATA num_vertices`
- `INTDATA num_faces`
- `MESH_VERTEX vertices`
- `MESH_FACE faces`
- `MESH_NORMAL vnormals`
- `MESH_NORMAL fnormals`

- [MESH\\_COLOR](#) vcolors
- [MESH\\_COLOR](#) fcolors
- [MESH\\_VFACE](#) vfaces
- [FLOATDATA](#) \* fareas
- [uint8\\_t](#) is\_trimesh
- [uint8\\_t](#) dummy

#### 4.1.1 Field Documentation

##### 4.1.1.1 `uint8_t` dummy

##### 4.1.1.2 `MESH_FACE` faces

Pointer to faces

##### 4.1.1.3 `FLOATDATA`\* fareas

Pointer to face areas

##### 4.1.1.4 `MESH_COLOR` fcolors

Pointer to face colors

##### 4.1.1.5 `MESH_NORMAL` fnormals

Pointer to face normals

##### 4.1.1.6 `uint8_t` is\_faces

Has faces?

##### 4.1.1.7 `uint8_t` is\_fareas

Has face areas?

##### 4.1.1.8 `uint8_t` is\_fcolors

Has face colors?

##### 4.1.1.9 `uint8_t` is\_fnormals

Has face normals?

##### 4.1.1.10 `uint8_t` is\_loaded

Is loaded?

##### 4.1.1.11 `uint8_t` is\_trimesh

Is trimesh?

#### 4.1.1.12 `uint8_t is_vcolors`

Has vertex colors?

#### 4.1.1.13 `uint8_t is_vertices`

Has vertices?

#### 4.1.1.14 `uint8_t is_vfaces`

Has vertex adjacent faces?

#### 4.1.1.15 `uint8_t is_vnormals`

Has vertex normals?

#### 4.1.1.16 `INTDATA num_faces`

Number of faces

#### 4.1.1.17 `INTDATA num_vertices`

Number of vertices

#### 4.1.1.18 `uint8_t origin_type`

Origin type

#### 4.1.1.19 `MESH_COLOR vcolors`

Pointer to vertex colors

#### 4.1.1.20 `MESH_VERTEX vertices`

Pointer to vertices

#### 4.1.1.21 `MESH_VFACE vfaces`

Pointer to vertex adjacency faces

#### 4.1.1.22 `MESH_NORMAL vnormals`

Pointer to vertex normals

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

- [meshlib.h](#)

## 4.2 mesh\_color Struct Reference

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

### Data Fields

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

### 4.2.1 Field Documentation

#### 4.2.1.1 FLOATDATA a

Alpha channel

#### 4.2.1.2 FLOATDATA b

Green channel

#### 4.2.1.3 FLOATDATA g

Blue channel

#### 4.2.1.4 FLOATDATA r

Red channel

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

- [meshlib.h](#)

## 4.3 mesh\_face Struct Reference

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

### Data Fields

- [INTDATA num\\_vertices](#)
- [INTDATA \\* vertices](#)

### 4.3.1 Field Documentation

#### 4.3.1.1 INTDATA num\_vertices

Number of vertices

#### 4.3.1.2 INTDATA\* vertices

Pointer to vertex indices

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

- [meshlib.h](#)

## 4.4 mesh\_rotation Struct Reference

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

### Data Fields

- [FLOATDATA data](#) [9]

#### 4.4.1 Field Documentation

##### 4.4.1.1 FLOATDATA data[9]

Matrix data

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

- [meshlib.h](#)

## 4.5 mesh\_struct Struct Reference

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

### Data Fields

- [INTDATA num\\_items](#)
- [INTDATA \\* items](#)

#### 4.5.1 Field Documentation

##### 4.5.1.1 INTDATA\* items

Pointer to INTDATA items

##### 4.5.1.2 INTDATA num\_items

Number of items

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

- [meshlib.h](#)

## 4.6 mesh\_struct2 Struct Reference

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

### Data Fields

- [INTDATA num\\_items](#)
- [INTDATA2 \\* items](#)

### 4.6.1 Field Documentation

#### 4.6.1.1 INTDATA2\* items

Pointer to INTDATA2 items

#### 4.6.1.2 INTDATA num\_items

Number of items

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

- [meshlib.h](#)

## 4.7 mesh\_transform Struct Reference

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

### Data Fields

- [FLOATDATA \\* data](#)

### 4.7.1 Field Documentation

#### 4.7.1.1 FLOATDATA\* data

Matrix data

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

- [meshlib.h](#)

## 4.8 mesh\_vector3 Struct Reference

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

### Data Fields

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

#### 4.8.1 Field Documentation

##### 4.8.1.1 FLOATDATA x

x co-ordinate

##### 4.8.1.2 FLOATDATA y

y co-ordinate

##### 4.8.1.3 FLOATDATA z

z co-ordinate

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

- [meshlib.h](#)

### 4.9 mesh\_vface Struct Reference

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

#### Data Fields

- [INTDATA num\\_faces](#)
- [INTDATA \\* faces](#)

#### 4.9.1 Field Documentation

##### 4.9.1.1 INTDATA\* faces

Pointer to adjacent face indices

##### 4.9.1.2 INTDATA num\_faces

Number of adjacent faces

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

- [meshlib.h](#)





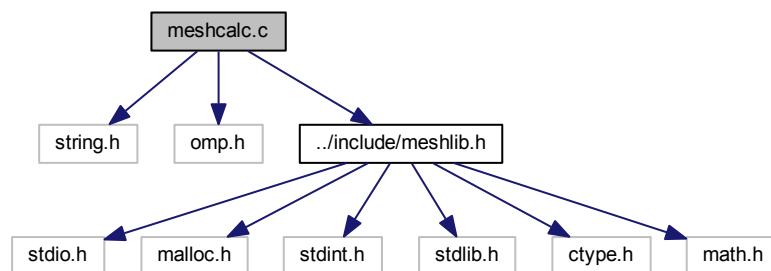
## Chapter 5

# File Documentation

### 5.1 meshcalc.c File Reference

This file contains functions pertaining to different mesh computations.

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



### Functions

- void `mesh_cross_vector3` (`MESH_VECTOR3` x, `MESH_VECTOR3` y, `MESH_VECTOR3` z)  
*Computes the cross product of two 3-d vectors.*
- void `mesh_cross_normal` (`MESH_NORMAL` x, `MESH_NORMAL` y, `MESH_NORMAL` z)  
*Computes the normalized cross product of two normals.*
- void `mesh_calc_face_normal` (`MESH_VERTEX` v1, `MESH_VERTEX` v2, `MESH_VERTEX` v3, `MESH_NORMAL` n)  
*Computes the face normal given 3 vertices.*
- int `mesh_calc_vertex_normals` (`MESH` m)  
*Computes vertex normals of a given mesh.*
- int `mesh_calc_face_normals` (`MESH` m)  
*Computes face normals of a given mesh.*
- int `mesh_calc_vertex_adjacency` (`MESH` m)  
*Computes vertex adjacent faces of a given mesh.*

- [INTDATA mesh\\_find](#) ([MESH\\_STRUCT](#) s, [INTDATA](#) q)  
*Finds an item in an INTDATA structure.*
- [INTDATA mesh\\_find2](#) ([MESH\\_STRUCT2](#) s, [INTDATA](#) q)  
*Finds an item in an INTDATA2 structure.*
- [int mesh\\_upsample](#) ([MESH](#) m, [int](#) iters)  
*Upsamples a given mesh.*
- [FLOATDATA mesh\\_calc\\_triangle\\_area](#) ([MESH\\_VERTEX](#) a, [MESH\\_VERTEX](#) b, [MESH\\_VERTEX](#) c)  
*Computes area of a triangle.*

### 5.1.1 Detailed Description

This file contains functions pertaining to different mesh computations.

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### Copyright

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

### 5.1.2 Function Documentation

5.1.2.1 `void mesh_calc_face_normal ( MESH\_VERTEX v1, MESH\_VERTEX v2, MESH\_VERTEX v3,  
MESH\_NORMAL n )`

Computes the face normal given 3 vertices.

#### Parameters

in	<a href="#">v1</a>	First vertex
in	<a href="#">v2</a>	Second vertex
in	<a href="#">v3</a>	Third vertex
out	<a href="#">n</a>	Output face normal $\mathbf{n}_f$

#### Returns

NULL

Here is the caller graph for this function:



5.1.2.2 int mesh\_calc\_face\_normals ( MESH *m* )

Computes face normals of a given mesh.

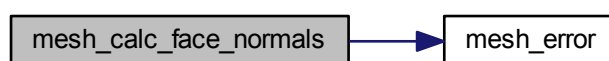
## Parameters

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

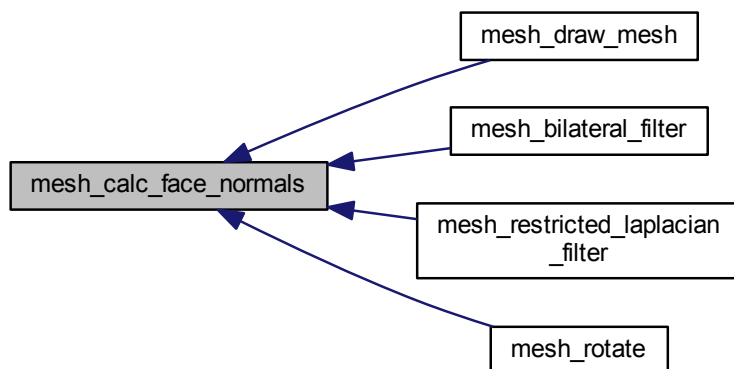
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

5.1.2.3 FLOATDATA mesh\_calc\_triangle\_area ( MESH\_VERTEX *a*, MESH\_VERTEX *b*, MESH\_VERTEX *c* )

Computes area of a triangle.

## Parameters

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

**Returns**

Area

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.1.2.4 int mesh\_calc\_vertex\_adjacency ( MESH *m* )

Computes vertex adjacent faces of a given mesh.

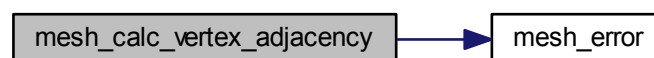
**Parameters**

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

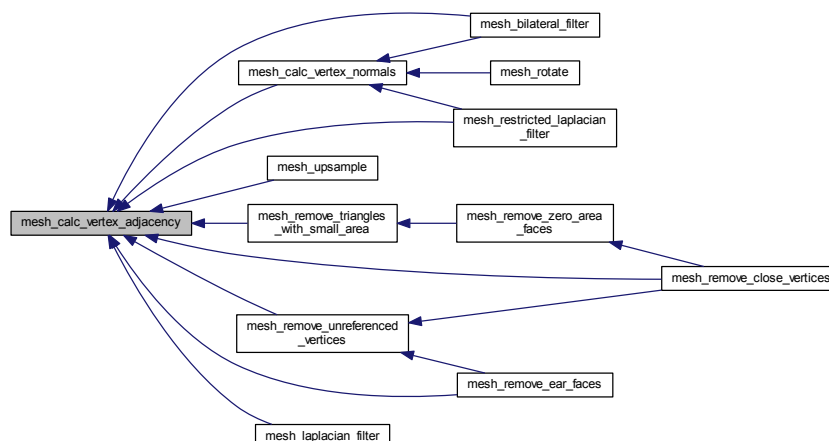
**Returns**

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



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

Computes vertex normals of a given mesh.

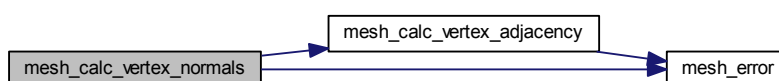
### Parameters

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

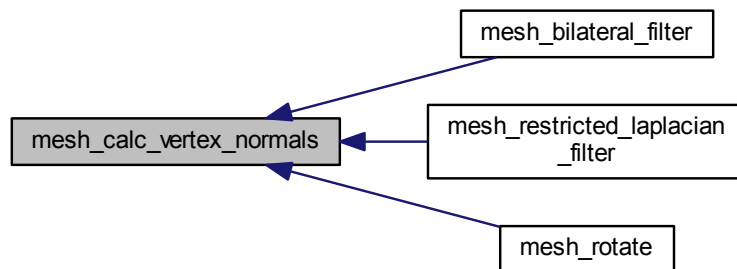
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.1.2.6 void mesh\_cross\_normal ( MESH\_NORMAL x, MESH\_NORMAL y, MESH\_NORMAL z )

Computes the normalized cross product of two normals.

##### Parameters

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

##### Returns

NULL

#### 5.1.2.7 void mesh\_cross\_vector3 ( MESH\_VECTOR3 x, MESH\_VECTOR3 y, MESH\_VECTOR3 z )

Computes the cross product of two 3-d vectors.

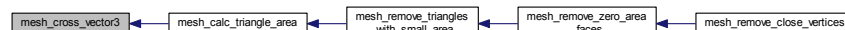
##### Parameters

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

##### Returns

NULL

Here is the caller graph for this function:



### 5.1.2.8 INTDATA mesh\_find ( MESH\_STRUCT *s*, INTDATA *q* )

Finds an item in an INTDATA structure.

#### Parameters

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

#### Returns

Index or -1

### 5.1.2.9 INTDATA mesh\_find2 ( MESH\_STRUCT2 *s*, INTDATA *q* )

Finds an item in an INTDATA2 structure.

#### Parameters

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

#### Returns

Index or -1

### 5.1.2.10 int mesh\_upsample ( MESH *m*, int *iters* )

Upsamples a given mesh.

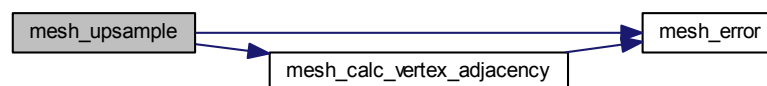
#### Parameters

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

#### Returns

Error code

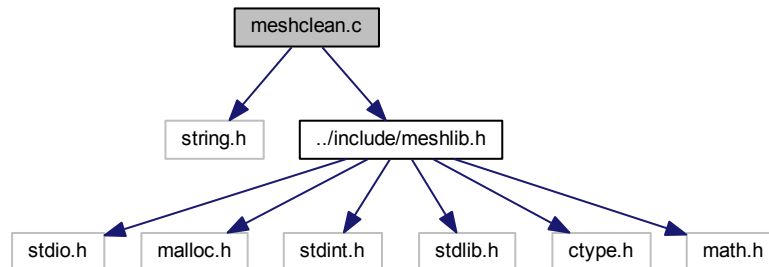
Here is the call graph for this function:



## 5.2 meshclean.c File Reference

This file contains functions pertaining to different mesh cleaning algorithms.

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



## Functions

- int [mesh\\_remove\\_boundary\\_vertices](#) ([MESH](#) m, int iters)  
*Removes boundary vertices and connecting elements.*
- int [mesh\\_remove\\_boundary\\_faces](#) ([MESH](#) m, int iters)  
*Removes boundary faces and connecting elements.*
- int [mesh\\_remove\\_triangles\\_with\\_small\\_area](#) ([MESH](#) m, [FLOATDATA](#) area)  
*Removes triangles with area smaller than a given value.*
- int [mesh\\_remove\\_zero\\_area\\_faces](#) ([MESH](#) m)  
*Removes triangles with zero area.*
- int [mesh\\_remove\\_unreferenced\\_vertices](#) ([MESH](#) m)  
*Removes unreferenced vertices.*
- int [mesh\\_remove\\_ear\\_faces](#) ([MESH](#) m, int niters)  
*Removes ear faces and connecting vertices.*
- int [mesh\\_remove\\_close\\_vertices](#) ([MESH](#) m, [FLOATDATA](#) r)  
*Removes close vertices.*

### 5.2.1 Detailed Description

This file contains functions pertaining to different mesh cleaning algorithms.

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### 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.

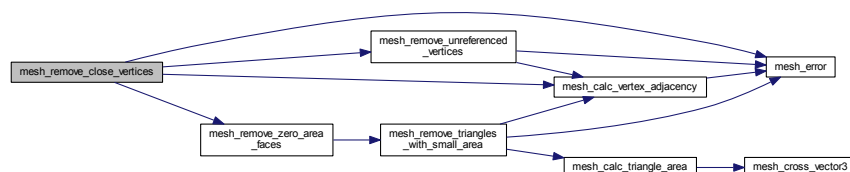
#### Parameters

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.

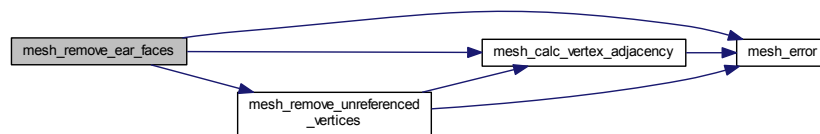
##### Parameters

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.

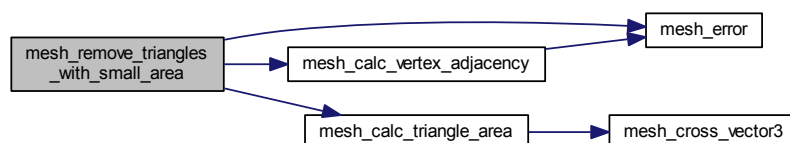
##### Parameters

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

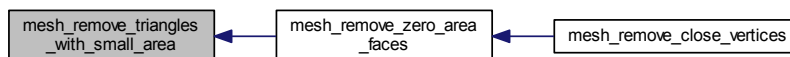
##### Returns

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.

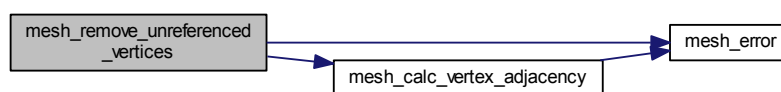
##### Parameters

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

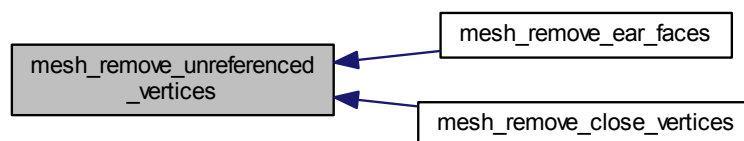
##### Returns

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.

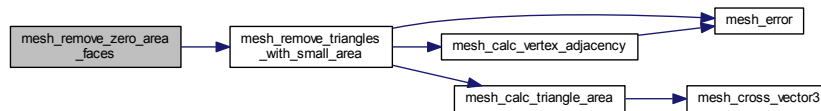
##### Parameters

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

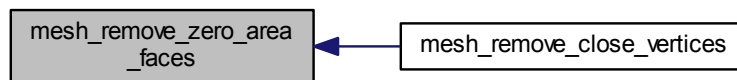
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:

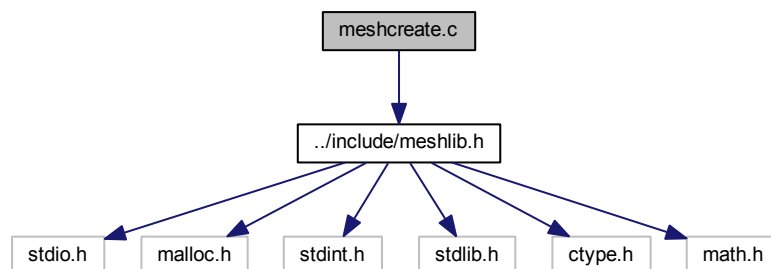


### 5.3 meshcreate.c File Reference

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 a 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

1.3.0.0

#### Copyright

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

### 5.3.2 Function Documentation

#### 5.3.2.1 `MESH mesh_create_mesh_new ( )`

Creates a new mesh.

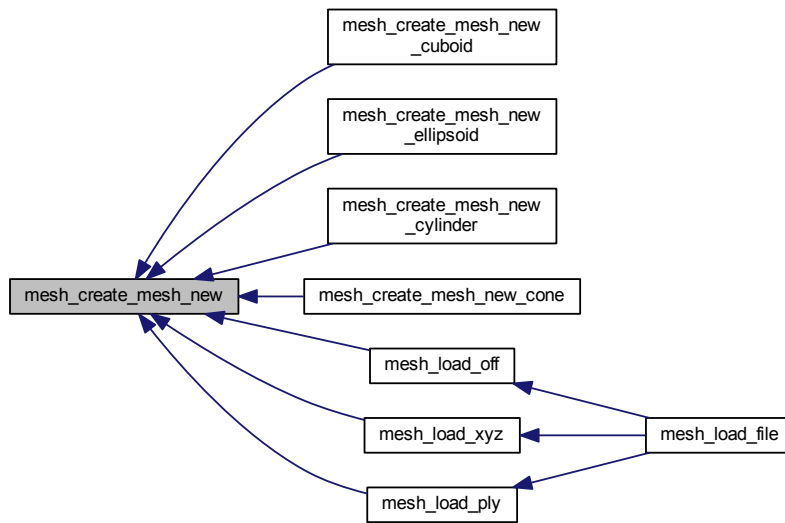
#### Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



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

Creates a cone mesh.

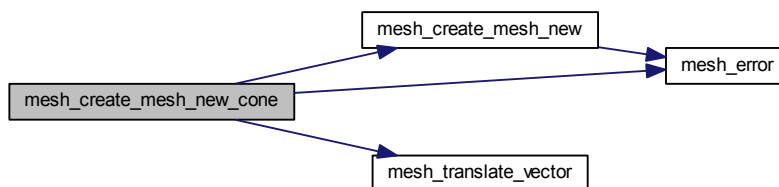
#### Parameters

in	sz	Size vector
in	pos	Position vector

#### Returns

Output mesh

Here is the call graph for this function:



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

Creates a cuboid mesh.

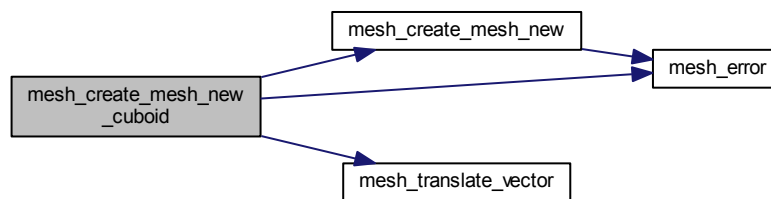
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



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

Creates a cylinder mesh.

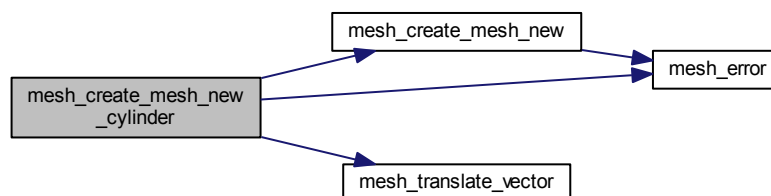
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



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

Creates a ellipsoid mesh.

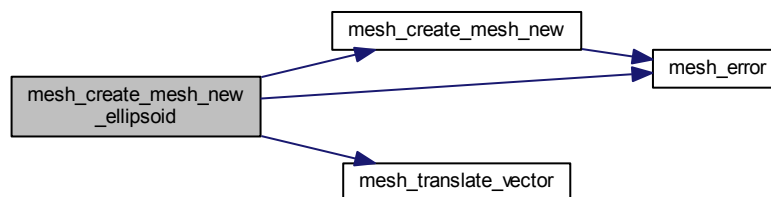
**Parameters**

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

**Returns**

Output mesh

Here is the call graph for this function:

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

Frees a mesh.

**Parameters**

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

**Returns**

NULL

**5.4 meshdraw.c File Reference**

This file contains functions pertaining to mesh drawing in OpenGL.

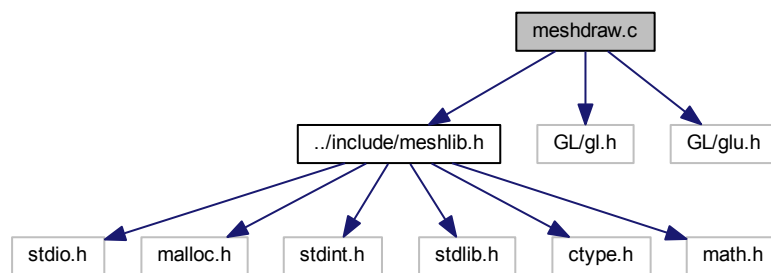
```

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

```



Include dependency graph for meshdraw.c:



## Functions

- void [mesh\\_draw\\_mesh](#) (**MESH** m)  
*Draws a given mesh in OpenGL context.*

### 5.4.1 Detailed Description

This file contains functions pertaining to mesh drawing in OpenGL.

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### Copyright

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

### 5.4.2 Function Documentation

#### 5.4.2.1 void mesh\_draw\_mesh ( **MESH** m )

Draws a given mesh in OpenGL context.

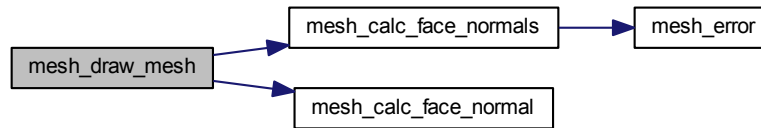
#### Parameters

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

**Returns**

NULL

Here is the call graph for this function:

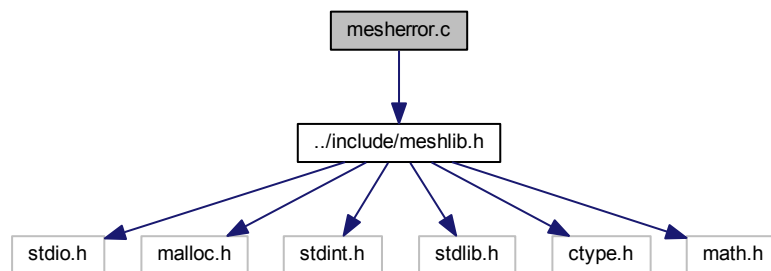


## 5.5 mesherror.c File Reference

This file contains functions pertaining to handling errors.

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

Include dependency graph for `mesherror.c`:

**Functions**

- void `mesh_error` (int type)  
*Displays error message and exits.*

### 5.5.1 Detailed Description

This file contains functions pertaining to handling errors.

**Author**

Sk. Mohammadul Haque

**Version**

1.3.0.0

## Copyright

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

## 5.5.2 Function Documentation

### 5.5.2.1 void mesh\_error ( int *type* )

Displays error message and exits.

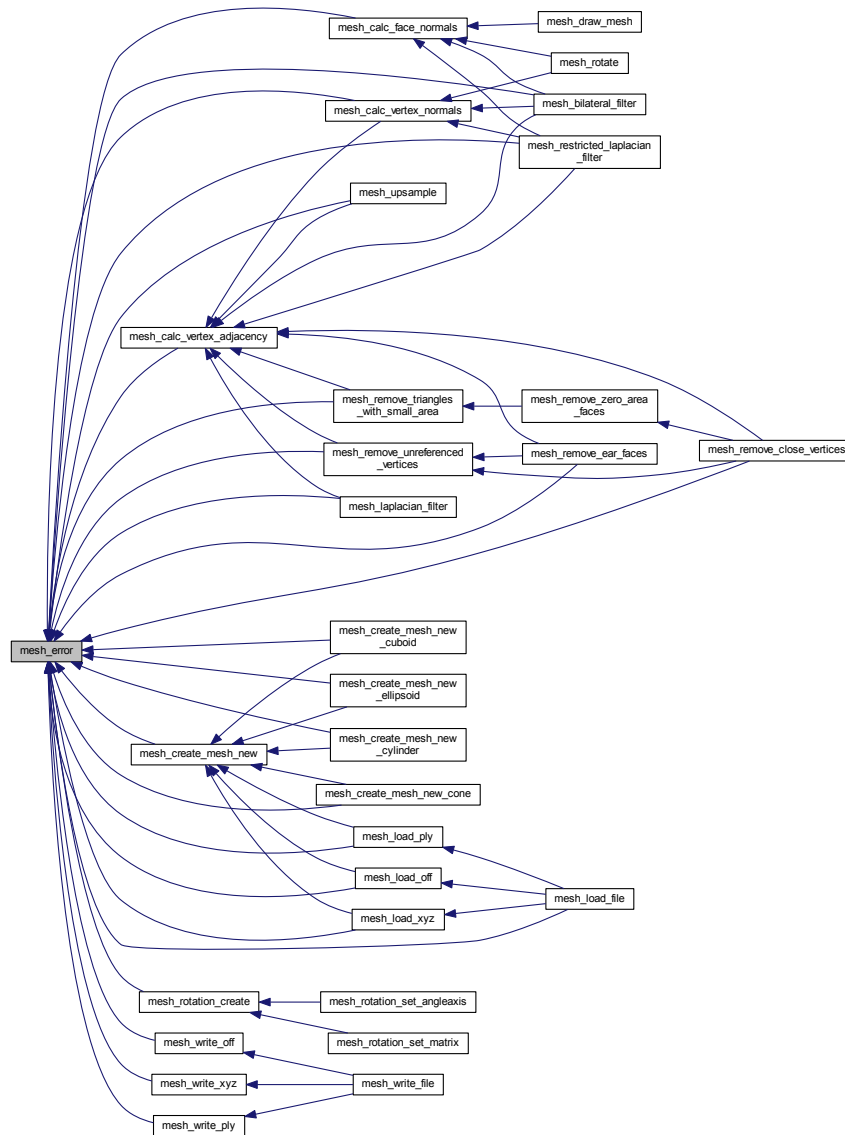
#### Parameters

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

#### Returns

NULL

Here is the caller graph for this function:

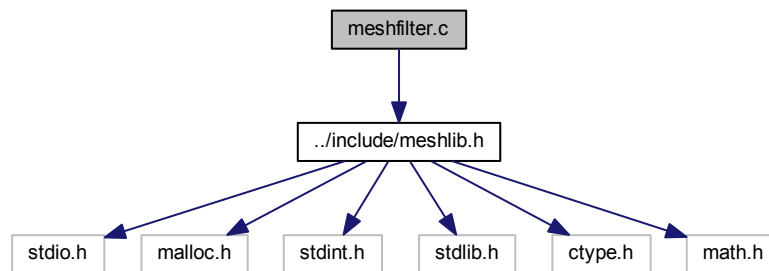


## 5.6 meshfilter.c File Reference

This file contains functions pertaining to different mesh filtering algorithms.

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

Include dependency graph for meshfilter.c:



## Functions

- int [mesh\\_bilateral\\_filter](#) (MESH *m*, FLOATDATA *sigma\_c*, FLOATDATA *sigma\_s*, int *niters*)  
*Mesh bilateral filter.*
- int [mesh\\_laplacian\\_filter](#) (MESH *m*, FLOATDATA *r*)  
*Mesh Laplacian filter.*
- int [mesh\\_restricted\\_laplacian\\_filter](#) (MESH *m*, FLOATDATA *r*, FLOATDATA *ang*)  
*Restricted Mesh Laplacian filter.*

### 5.6.1 Detailed Description

This file contains functions pertaining to different mesh filtering algorithms.

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### Copyright

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

### 5.6.2 Function Documentation

5.6.2.1 int [mesh\\_bilateral\\_filter](#) ( MESH *m*, FLOATDATA *sigma\_c*, FLOATDATA *sigma\_s*, int *niters* )

Mesh bilateral filter.

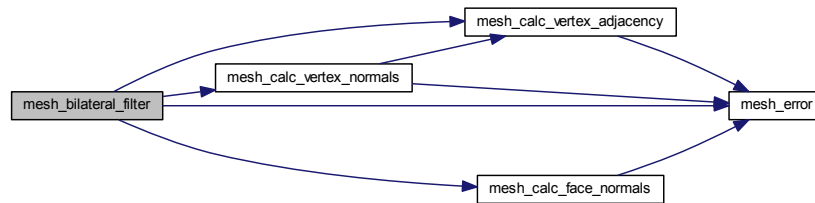
#### Parameters

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

**Returns**

Error code

Here is the call graph for this function:



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

Mesh Laplacian filter.

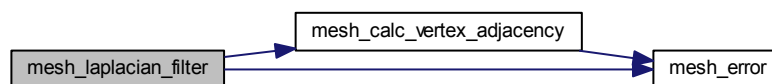
**Parameters**

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

**Returns**

Error code

Here is the call graph for this function:



### 5.6.2.3 int mesh\_restricted\_laplacian\_filter ( MESH *m*, FLOATDATA *r*, FLOATDATA *ang* )

Restricted Mesh Laplacian filter.

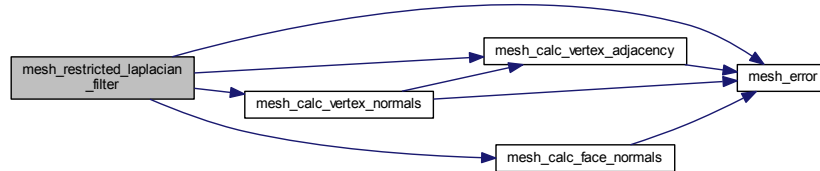
**Parameters**

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

## Returns

Error code

Here is the call graph for this function:

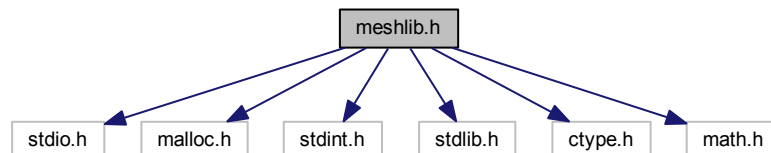


## 5.7 meshlib.h File Reference

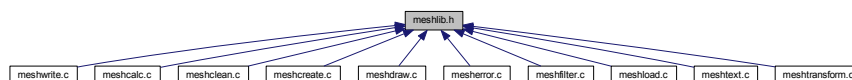
This header file contains declarations of all functions of meshlib.

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

Include dependency graph for meshlib.h:



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



## Data Structures

- struct [mesh\\_vector3](#)
- struct [mesh\\_color](#)
- struct [mesh\\_struct](#)
- struct [mesh\\_struct2](#)
- struct [mesh\\_face](#)

- struct [mesh\\_vface](#)
- struct [mesh\\_rotation](#)
- struct [mesh\\_transform](#)
- struct [mesh](#)

## Macros

- `#define \_CRT\_SECURE\_NO\_DEPRECATED`
- `#define MESH\_INTDATA\_TYPE 0`
- `#define MESH\_FLOATDATA\_TYPE 0`
- `#define INTDATA int32_t /* do not change this, careful see meshload fscanf and other functions */`
- `#define FLOATDATA float /* do not change this, careful see meshload fscanf and other functions */`
- `#define MESH\_ORIGIN\_TYPE\_BUILD 00`
- `#define MESH\_ORIGIN\_TYPE\_OFF 11`
- `#define MESH\_ORIGIN\_TYPE\_NOFF 12`
- `#define MESH\_ORIGIN\_TYPE\_COFF 13`
- `#define MESH\_ORIGIN\_TYPE\_NCOFF 14`
- `#define MESH\_ORIGIN\_TYPE\_XYZ 20`
- `#define MESH\_ORIGIN\_TYPE\_PLY\_ASCII 30`
- `#define MESH\_ORIGIN\_TYPE\_PLY\_BINARY\_LITTLE\_ENDIAN 31`
- `#define MESH\_ORIGIN\_TYPE\_PLY\_BINARY\_BIG\_ENDIAN 32`
- `#define MESH\_ERR\_MALLOC 0`
- `#define MESH\_ERR\_SIZE\_MISMATCH 1`
- `#define MESH\_ERR\_FNOTOPEN 2`
- `#define MESH\_ERR\_UNKNOWN 3`
- `#define MESH\_PI (3.14159265359)`

## Typedefs

- `typedef struct _iobuf * FILEPOINTER`
- `typedef INTDATA INTDATA2 [2]`
- `typedef struct mesh\_vector3 mesh\_vector3`
- `typedef mesh\_vector3 * MESH\_VECTOR3`
- `typedef mesh\_vector3 mesh\_vertex`
- `typedef mesh\_vertex * MESH\_VERTEX`
- `typedef mesh\_vector3 mesh\_normal`
- `typedef mesh\_normal * MESH\_NORMAL`
- `typedef struct mesh\_color mesh\_color`
- `typedef mesh\_color * MESH\_COLOR`
- `typedef struct mesh\_struct mesh\_struct`
- `typedef mesh\_struct * MESH\_STRUCT`
- `typedef struct mesh\_struct2 mesh\_struct2`
- `typedef mesh\_struct2 * MESH\_STRUCT2`
- `typedef struct mesh\_face mesh\_face`
- `typedef mesh\_face * MESH\_FACE`
- `typedef struct mesh\_vface mesh\_vface`
- `typedef mesh\_vface * MESH\_VFACE`
- `typedef struct mesh\_rotation mesh\_rotation`
- `typedef mesh\_rotation * MESH\_ROTATION`
- `typedef struct mesh\_transform mesh\_transform`
- `typedef mesh\_transform * MESH\_TRANSFORM`
- `typedef struct mesh mesh`
- `typedef mesh * MESH`



## Functions

- void `mesh_error` (int type)  
*Displays error message and exits.*
- `MESH mesh_create_mesh_new` ()  
*Creates a new mesh.*
- void `mesh_free_mesh` (`MESH m`)  
*Frees a mesh.*
- `MESH mesh_create_mesh_new_cuboid` (`MESH_VECTOR3 sz`, `MESH_VECTOR3 pos`)  
*Creates a cuboid mesh.*
- `MESH mesh_create_mesh_new_ellipsoid` (`MESH_VECTOR3 sz`, `MESH_VECTOR3 pos`)  
*Creates a 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_load_file` (const char \*fname)  
*Read a mesh from an OFF/PLY/ASC/XYZ file.*
- `MESH mesh_load_off` (const char \*fname)  
*Read 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)  
*Read a mesh from a PLY file.*
- int `mesh_write_file` (`MESH m`, const char \*fname)  
*Write a mesh to an OFF/PLY/ASC/XYZ file.*
- int `mesh_write_off` (`MESH m`, const char \*fname)  
*Write a mesh to an OFF file.*
- int `mesh_write_xyz` (`MESH m`, const char \*fname)  
*Write a mesh to an XYZ file.*
- int `mesh_write_ply` (`MESH m`, const char \*fname)  
*Write a mesh to an PLY file.*
- int `mesh_calc_vertex_normals` (`MESH m`)  
*Computes vertex normals of a given mesh.*
- int `mesh_calc_face_normals` (`MESH m`)  
*Computes face normals of a given mesh.*
- int `mesh_calc_vertex_adjacency` (`MESH m`)  
*Computes vertex adjacent faces of a given mesh.*
- int `mesh_upsample` (`MESH m`, int iters)  
*Upsamples a given mesh.*
- void `mesh_cross_vector3` (`MESH_VECTOR3 x`, `MESH_VECTOR3 y`, `MESH_VECTOR3 z`)  
*Computes the cross product of two 3-d vectors.*
- void `mesh_cross_normal` (`MESH_NORMAL x`, `MESH_NORMAL y`, `MESH_NORMAL z`)  
*Computes the normalized cross product of two normals.*
- `FLOATDATA mesh_calc_triangle_area` (`MESH_VERTEX a`, `MESH_VERTEX b`, `MESH_VERTEX c`)  
*Computes area of a triangle.*
- void `mesh_calc_face_normal` (`MESH_VERTEX v1`, `MESH_VERTEX v2`, `MESH_VERTEX v3`, `MESH_NORMAL n`)  
*Computes the face normal given 3 vertices.*
- `INTDATA mesh_find` (`MESH_STRUCT s`, `INTDATA q`)  
*Finds an item in an INTDATA structure.*

- [INTDATA mesh\\_find2](#) ([MESH\\_STRUCT2](#) s, [INTDATA](#) q)  
*Finds an item in an INTDATA2 structure.*
- [int mesh\\_remove\\_boundary\\_vertices](#) ([MESH](#) m, [int](#) iters)  
*Removes boundary vertices and connecting elements.*
- [int mesh\\_remove\\_boundary\\_faces](#) ([MESH](#) m, [int](#) iters)  
*Removes boundary faces and connecting elements.*
- [int mesh\\_remove\\_triangles\\_with\\_small\\_area](#) ([MESH](#) m, [FLOATDATA](#) area)  
*Removes triangles with area smaller than a given value.*
- [int mesh\\_remove\\_unreferenced\\_vertices](#) ([MESH](#) m)  
*Removes unreferenced vertices.*
- [int mesh\\_remove\\_zero\\_area\\_faces](#) ([MESH](#) m)  
*Removes triangles with zero area.*
- [int mesh\\_remove\\_close\\_vertices](#) ([MESH](#) m, [FLOATDATA](#) r)  
*Removes close vertices.*
- [int mesh\\_remove\\_ear\\_faces](#) ([MESH](#) m, [int](#) niters)  
*Removes ear faces and connecting vertices.*
- [int mesh\\_isnumeric](#) ([FILEPOINTER](#) fp)  
*Checks if numeric or not.*
- [int mesh\\_go\\_next\\_word](#) ([FILEPOINTER](#) fp)  
*Points to the next word.*
- [int mesh\\_read\\_word](#) ([FILEPOINTER](#) fp, [char](#) \*c\_word, [int](#) sz)  
*Reads current word.*
- [int mesh\\_count\\_words\\_in\\_line](#) ([FILEPOINTER](#) fp, [int](#) \*count)  
*Counts number of words in the current line.*
- [int mesh\\_skip\\_line](#) ([FILEPOINTER](#) fp)  
*Skips to next line.*
- [int mesh\\_bilateral\\_filter](#) ([MESH](#) m, [FLOATDATA](#) sigma\_c, [FLOATDATA](#) sigma\_s, [int](#) niters)  
*Mesh bilateral filter.*
- [int mesh\\_laplacian\\_filter](#) ([MESH](#) m, [FLOATDATA](#) r)  
*Mesh Laplacian filter.*
- [int mesh\\_restricted\\_laplacian\\_filter](#) ([MESH](#) m, [FLOATDATA](#) r, [FLOATDATA](#) ang)  
*Restricted Mesh Laplacian filter.*
- [MESH\\_ROTATION mesh\\_rotation\\_create](#) ()  
*Creates a new rotation.*
- [void mesh\\_rotation\\_free](#) ([MESH\\_ROTATION](#) r)  
*Frees a given rotation.*
- [MESH\\_ROTATION mesh\\_rotation\\_set\\_matrix](#) ([FLOATDATA](#) \*mat, [MESH\\_ROTATION](#) r)  
*Sets rotation from a matrix.*
- [MESH\\_ROTATION mesh\\_rotation\\_set\\_angleaxis](#) ([FLOATDATA](#) ang, [MESH\\_NORMAL](#) axis, [MESH\\_ROTATION](#) r)  
*Sets rotation from angle axis.*
- [int mesh\\_translate](#) ([MESH](#) m, [FLOATDATA](#) x, [FLOATDATA](#) y, [FLOATDATA](#) z)  
*Translates a mesh by x, y and z amounts.*
- [int mesh\\_translate\\_vector](#) ([MESH](#) m, [MESH\\_VERTEX](#) v)  
*Translates a mesh by a given 3-d vector.*
- [int mesh\\_scale](#) ([MESH](#) m, [FLOATDATA](#) sx, [FLOATDATA](#) sy, [FLOATDATA](#) sz)  
*Scales a mesh by x, y and z amounts.*
- [MESH\\_VERTEX mesh\\_vertex\\_rotate](#) ([MESH\\_VERTEX](#) v, [MESH\\_ROTATION](#) r)  
*Rotates a vertex by a given rotation.*
- [int mesh\\_rotate](#) ([MESH](#) m, [MESH\\_ROTATION](#) r)  
*Rotates a mesh by a given rotation.*
- [void mesh\\_draw\\_mesh](#) ([MESH](#) m)  
*Draws a given mesh in OpenGL context.*

### 5.7.1 Detailed Description

This header file contains declarations of all functions of meshlib.

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### Copyright

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

### 5.7.2 Macro Definition Documentation

#### 5.7.2.1 `#define _CRT_SECURE_NO_DEPRECATED`

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

Float datatype

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

Integer datatype

#### 5.7.2.4 `#define MESH_ERR_FNOTOPEN 2`

Mesh error type - file open

#### 5.7.2.5 `#define MESH_ERR_MALLOC 0`

Mesh error type - allocation

#### 5.7.2.6 `#define MESH_ERR_SIZE_MISMATCH 1`

Mesh error type - size mismatch

#### 5.7.2.7 `#define MESH_ERR_UNKNOWN 3`

Mesh error type - unknown

#### 5.7.2.8 `#define MESH_FLOATDATA_TYPE 0`

Float datatype selector

#### 5.7.2.9 `#define MESH_INTDATA_TYPE 0`

Integer datatype selector

5.7.2.10 `#define MESH_ORIGIN_TYPE_BUILD 00`

Mesh origin type - create new

5.7.2.11 `#define MESH_ORIGIN_TYPE_COFF 13`

Mesh origin type - COFF file

5.7.2.12 `#define MESH_ORIGIN_TYPE_NCOFF 14`

Mesh origin type - NCOFF file

5.7.2.13 `#define MESH_ORIGIN_TYPE_NOFF 12`

Mesh origin type - NOFF file

5.7.2.14 `#define MESH_ORIGIN_TYPE_OFF 11`

Mesh origin type - OFF file

5.7.2.15 `#define MESH_ORIGIN_TYPE_PLY_ASCII 30`

Mesh origin type - PLY ascii file

5.7.2.16 `#define MESH_ORIGIN_TYPE_PLY_BINARY_BIG_ENDIAN 32`

Mesh origin type - PLY binary BE file

5.7.2.17 `#define MESH_ORIGIN_TYPE_PLY_BINARY_LITTLE_ENDIAN 31`

Mesh origin type - PLY binary LE file

5.7.2.18 `#define MESH_ORIGIN_TYPE_XYZ 20`

Mesh origin type - XYZ file

5.7.2.19 `#define MESH_PI (3.14159265359)`

$\pi$

### 5.7.3 Typedef Documentation

5.7.3.1 `typedef struct _iobuf* FILEPOINTER`

File pointer

5.7.3.2 `typedef INTDATA INTDATA2[2]`

2- element INTDATA

### 5.7.3.3 `typedef struct mesh mesh`

Mesh

### 5.7.3.4 `typedef mesh* MESH`

Pointer to mesh

### 5.7.3.5 `typedef struct mesh_color mesh_color`

### 5.7.3.6 `typedef mesh_color* MESH_COLOR`

Color

### 5.7.3.7 `typedef struct mesh_face mesh_face`

Face

### 5.7.3.8 `typedef mesh_face* MESH_FACE`

Pointer to face

### 5.7.3.9 `typedef mesh_vector3 mesh_normal`

Normal

### 5.7.3.10 `typedef mesh_normal* MESH_NORMAL`

Normal pointer

### 5.7.3.11 `typedef struct mesh_rotation mesh_rotation`

Rotation

### 5.7.3.12 `typedef mesh_rotation* MESH_ROTATION`

Pointer to rotation

### 5.7.3.13 `typedef struct mesh_struct mesh_struct`

INTDATA Structure

### 5.7.3.14 `typedef mesh_struct* MESH_STRUCT`

INTDATA Structure pointer

### 5.7.3.15 `typedef struct mesh_struct2 mesh_struct2`

INTDATA2 Structure

**5.7.3.16 typedef mesh\_struct2\* MESH\_STRUCT2**

INTDATA2 Structure pointer

**5.7.3.17 typedef struct mesh\_transform mesh\_transform**

Transformation

**5.7.3.18 typedef mesh\_transform\* MESH\_TRANSFORM**

Pointer to transformation

**5.7.3.19 typedef struct mesh\_vector3 mesh\_vector3**

Generic 3-d vector

**5.7.3.20 typedef mesh\_vector3\* MESH\_VECTOR3**

Generic 3-d vector pointer

**5.7.3.21 typedef mesh\_vector3 mesh\_vertex**

Vertex

**5.7.3.22 typedef mesh\_vertex\* MESH\_VERTEX**

Vertex pointer

**5.7.3.23 typedef struct mesh\_vface mesh\_vface**

Vertex adjacent faces

**5.7.3.24 typedef mesh\_vface\* MESH\_VFACE**

Pointer to vertex adjacent faces

**5.7.4 Function Documentation****5.7.4.1 int mesh\_bilateral\_filter ( MESH *m*, FLOATDATA *sigma\_c*, FLOATDATA *sigma\_s*, int *niters* )**

Mesh bilateral filter.

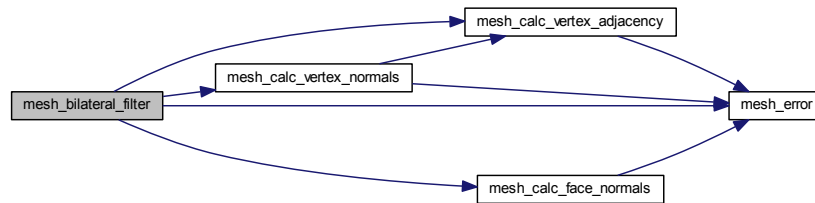
**Parameters**

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:



#### 5.7.4.2 void mesh\_calc\_face\_normal ( MESH\_VERTEX *v1*, MESH\_VERTEX *v2*, MESH\_VERTEX *v3*, MESH\_NORMAL *n* )

Computes the face normal given 3 vertices.

## Parameters

in	<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:



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

Computes face normals of a given mesh.

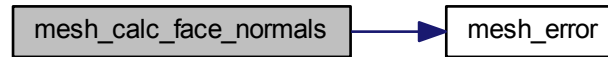
## Parameters

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

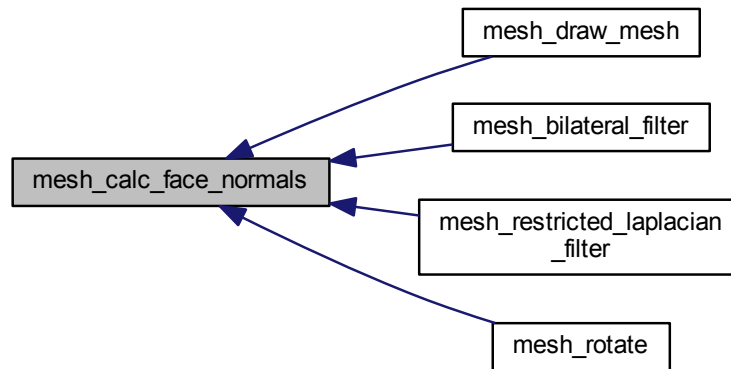
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.4 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:



#### 5.7.4.5 int mesh\_calc\_vertex\_adjacency ( MESH *m* )

Computes vertex adjacent faces of a given mesh.

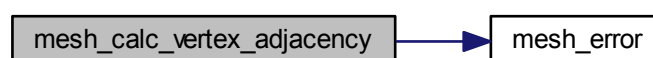
## Parameters

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

## Returns

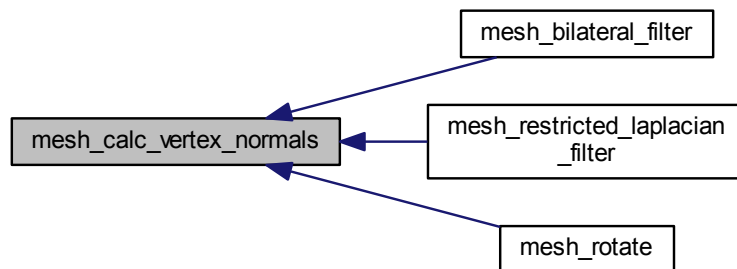
Error code

Here is the call graph for this function:





Here is the caller graph for this function:



#### 5.7.4.7 `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

#### 5.7.4.8 `MESH mesh_create_mesh_new ( )`

Creates a new mesh.

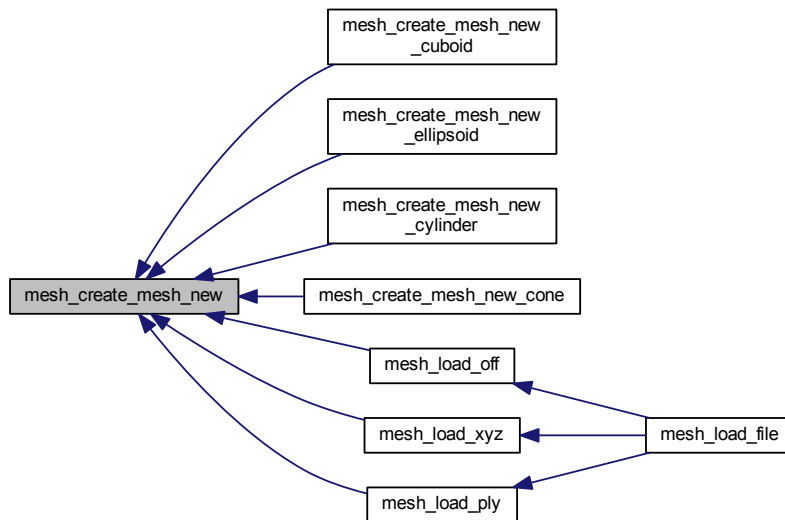
##### Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



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

Creates a cone mesh.

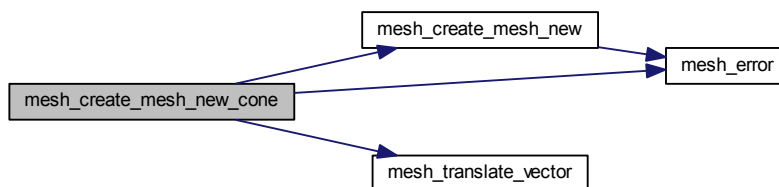
##### Parameters

in	sz	Size vector
in	pos	Position vector

##### Returns

Output mesh

Here is the call graph for this function:



#### 5.7.4.10 MESH mesh.create\_mesh\_new\_cuboid ( MESH\_VECTOR3 sz, MESH\_VECTOR3 pos )

Creates a cuboid mesh.

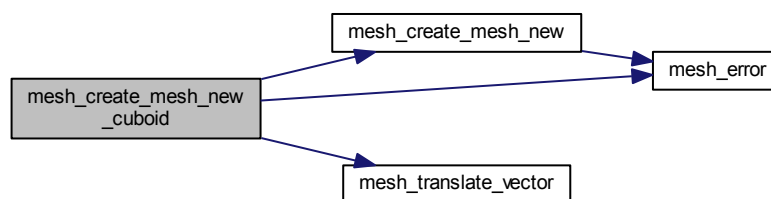
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



#### 5.7.4.11 MESH mesh.create\_mesh\_new.cylinder ( MESH\_VECTOR3 *sz*, MESH\_VECTOR3 *pos* )

Creates a cylinder mesh.

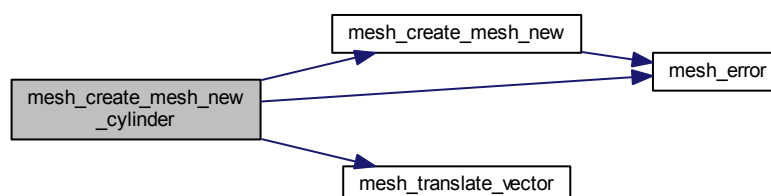
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



#### 5.7.4.12 MESH mesh.create\_mesh\_new.ellipsoid ( MESH\_VECTOR3 *sz*, MESH\_VECTOR3 *pos* )

Creates a ellipsoid mesh.

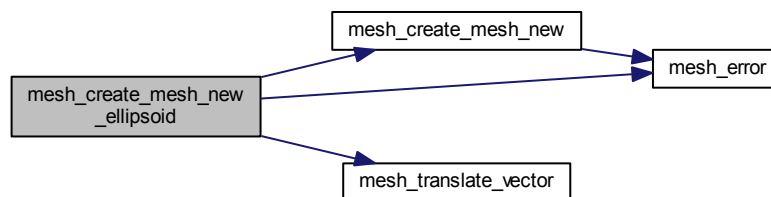
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



#### 5.7.4.13 void mesh\_cross\_normal ( MESH\_NORMAL x, MESH\_NORMAL y, MESH\_NORMAL z )

Computes the normalized cross product of two normals.

## Parameters

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

## Returns

NULL

#### 5.7.4.14 void mesh\_cross\_vector3 ( MESH\_VECTOR3 x, MESH\_VECTOR3 y, MESH\_VECTOR3 z )

Computes the cross product of two 3-d vectors.

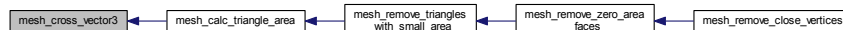
## Parameters

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

## Returns

NULL

Here is the caller graph for this function:

5.7.4.15 void mesh\_draw\_mesh ( MESH *m* )

Draws a given mesh in OpenGL context.

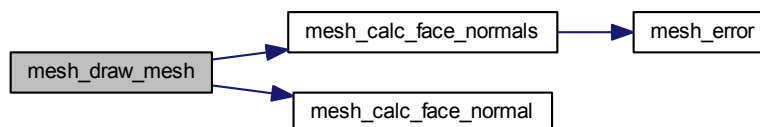
## Parameters

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

## Returns

NULL

Here is the call graph for this function:

5.7.4.16 void mesh\_error ( int *type* )

Displays error message and exits.

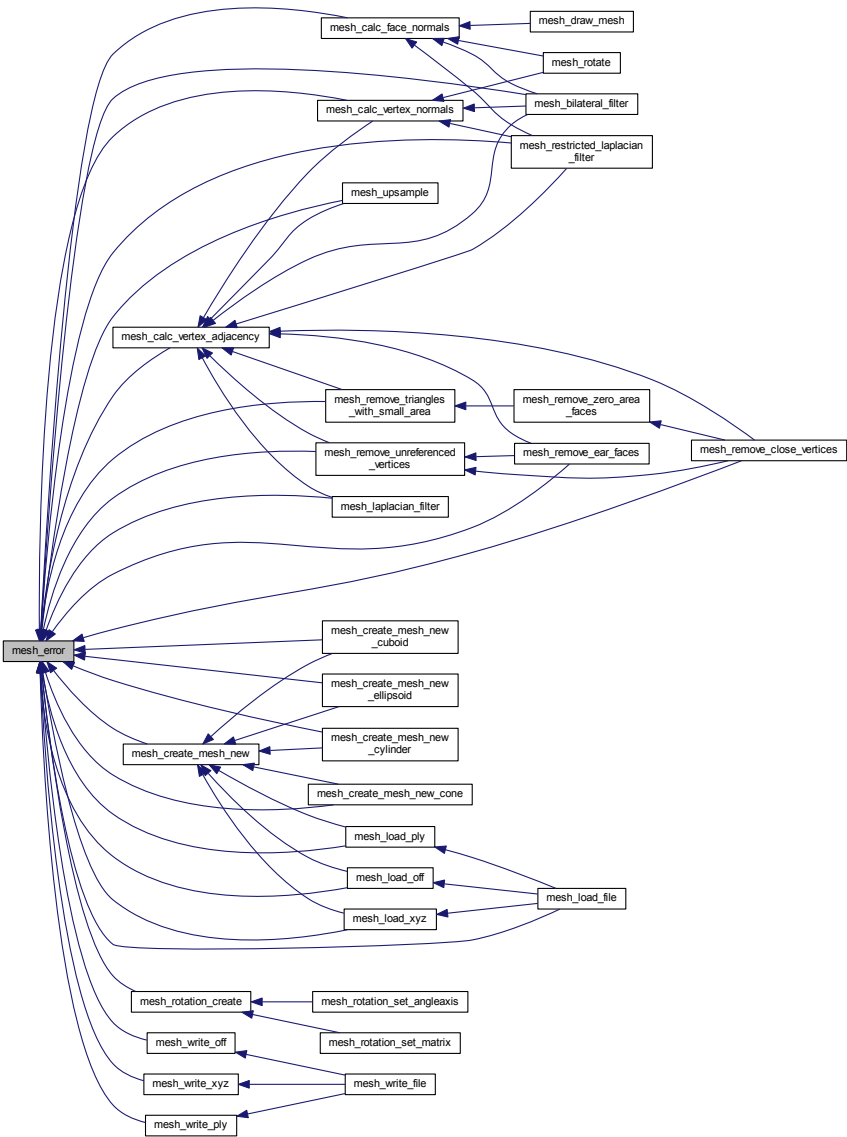
## Parameters

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

Returns

NULL

Here is the caller graph for this function:



5.7.4.17 INTDATA mesh.find ( MESH\_STRUCT s, INTDATA q )

Finds an item in an INTDATA structure.

Parameters

in	s	Input INTDATA structure
in	q	Query INTDATA



## Returns

Index or -1

**5.7.4.18 INTDATA mesh\_find2 ( MESH\_STRUCT2 *s*, INTDATA *q* )**

Finds an item in an INTDATA2 structure.

## Parameters

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

## Returns

Index or -1

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

Frees a mesh.

## Parameters

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

## Returns

NULL

**5.7.4.20 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

**5.7.4.21 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

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

Mesh Laplacian filter.

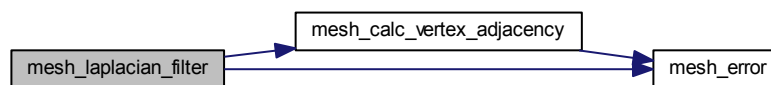
##### Parameters

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

##### Returns

Error code

Here is the call graph for this function:



#### 5.7.4.23 MESH mesh\_load\_file ( const char \* *fname* )

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

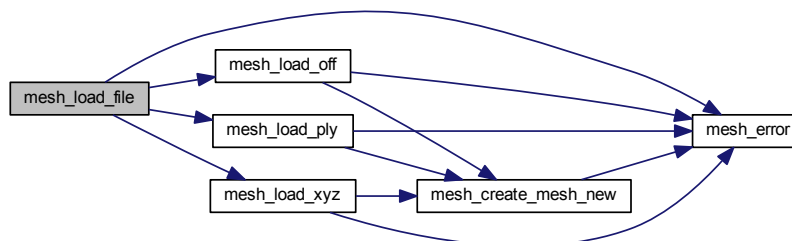
##### Parameters

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

##### Returns

Output mesh

Here is the call graph for this function:



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

Read a mesh from an OFF file.

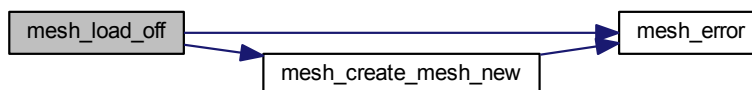
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.25 MESH mesh.load\_ply ( const char \* *fname* )

Read a mesh from a PLY file.

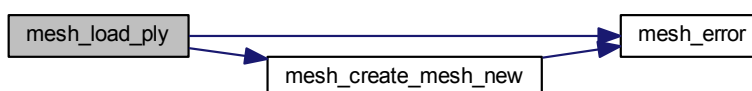
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



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

Read a mesh from an ASC/XYZ file.

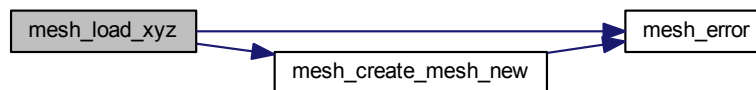
##### Parameters

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

##### Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



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

Reads current word.

## Parameters

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

## Returns

Status 0 - Normal/ 1- EOF

**5.7.4.28 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

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

Removes boundary vertices and connecting elements.

## Parameters

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

## Returns

Error code

**5.7.4.30 int mesh\_remove\_close\_vertices ( MESH *m*, FLOATDATA *r* )**

Removes close vertices.

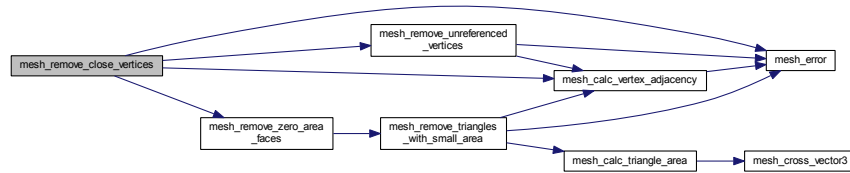
## Parameters

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

**Returns**

Error code

Here is the call graph for this function:



#### 5.7.4.31 int mesh\_remove\_ear\_faces ( MESH *m*, int *niters* )

Removes ear faces and connecting vertices.

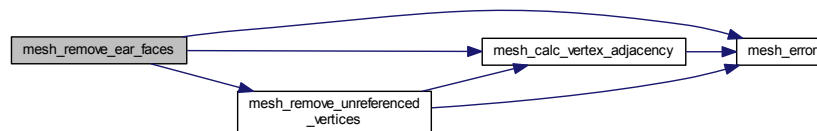
**Parameters**

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

**Returns**

Error code

Here is the call graph for this function:



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

Removes triangles with area smaller than a given value.

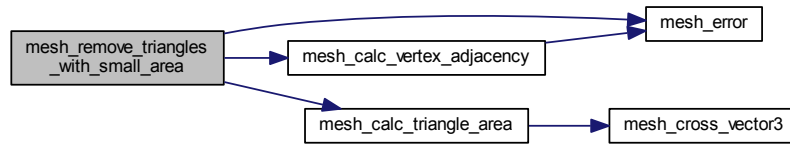
**Parameters**

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

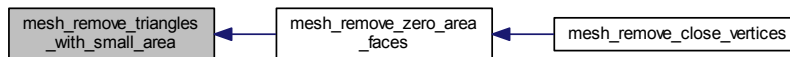
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



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

Removes unreferenced vertices.

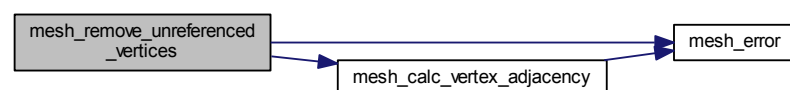
## Parameters

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

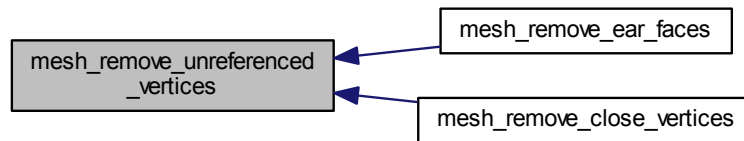
## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.34 `int mesh_remove_zero_area_faces ( MESH m )`

Removes triangles with zero area.

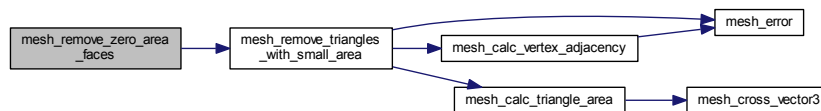
##### Parameters

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

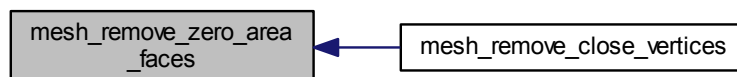
##### Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



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

Restricted Mesh Laplacian filter.



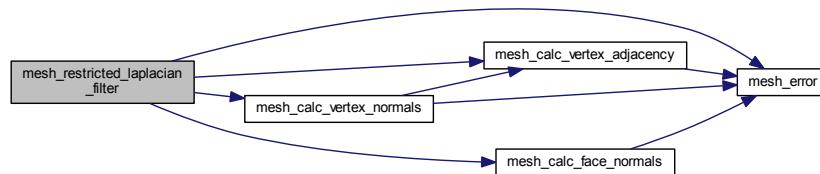
## Parameters

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

## Returns

Error code

Here is the call graph for this function:

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

Rotates a mesh by a given rotation.

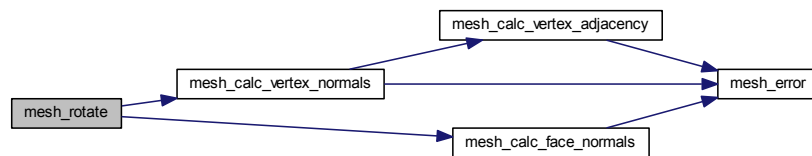
## Parameters

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

## Returns

Error code

Here is the call graph for this function:

5.7.4.37 `MESH_ROTATION mesh_rotation_create ( )`

Creates a new rotation.

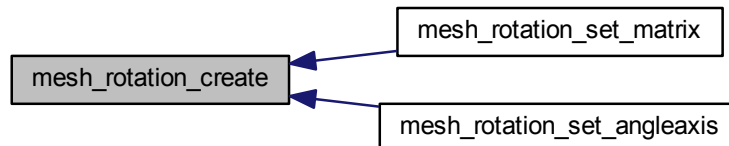
## Returns

Output rotation

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.38 void mesh\_rotation\_free ( MESH\_ROTATION *r* )

Frees a given rotation.

## Parameters

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

## Returns

NULL

#### 5.7.4.39 MESH\_ROTATION mesh\_rotation\_set\_angleaxis ( FLOATDATA *ang*, MESH\_NORMAL *axis*, MESH\_ROTATION *r* )

Sets rotation from angle axis.

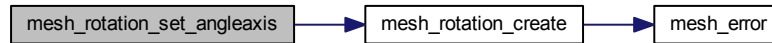
## Parameters

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:



#### 5.7.4.40 MESH\_ROTATION mesh\_rotation\_set\_matrix ( FLOATDATA \* *mat*, MESH\_ROTATION *r* )

Sets rotation from a matrix.

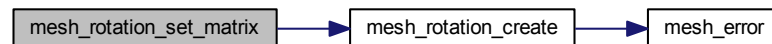
## Parameters

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

## Returns

Output rotation

Here is the call graph for this function:



#### 5.7.4.41 int mesh\_scale ( MESH *m*, FLOATDATA *sx*, FLOATDATA *sy*, FLOATDATA *sz* )

Scales a mesh by x, y and z amounts.

## Parameters

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

## Returns

Error code

**5.7.4.42 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

**5.7.4.43 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

**5.7.4.44 int mesh\_translate\_vector ( MESH *m*, MESH\_VECTOR3 *v* )**

Translates a mesh by a given 3-d vector.

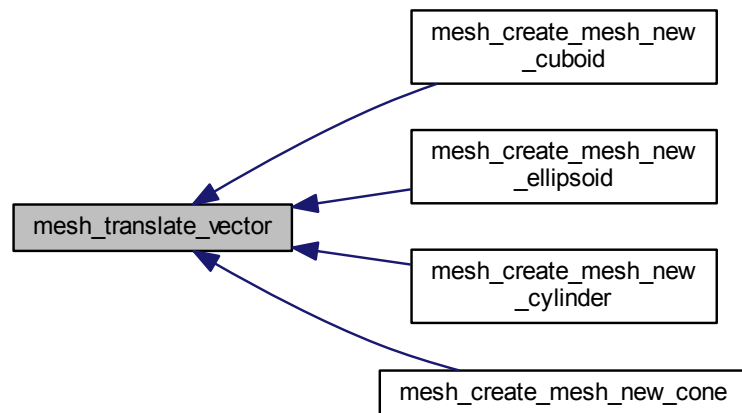
**Parameters**

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

**Returns**

Error code

Here is the caller graph for this function:



#### 5.7.4.45 int mesh\_upsample ( MESH *m*, int *iters* )

Upsamples a given mesh.

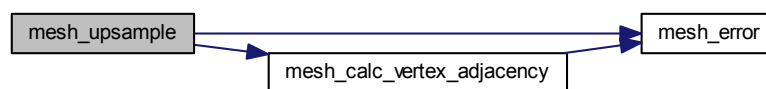
##### Parameters

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

##### Returns

Error code

Here is the call graph for this function:



#### 5.7.4.46 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

**5.7.4.47** `int mesh_write_file ( MESH m, const char * fname )`

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

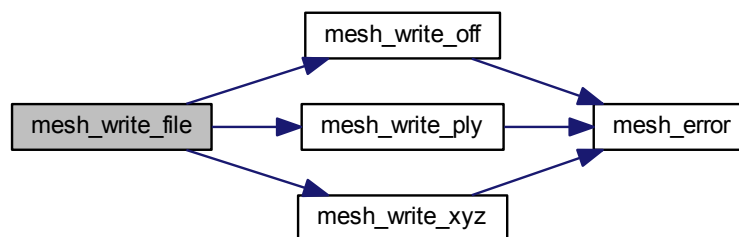
**Parameters**

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

**Returns**

Error code

Here is the call graph for this function:

**5.7.4.48** `int mesh_write_off ( MESH m, const char * fname )`

Write a mesh to an OFF file.

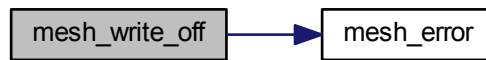
**Parameters**

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:



#### 5.7.4.49 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:



#### 5.7.4.50 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:

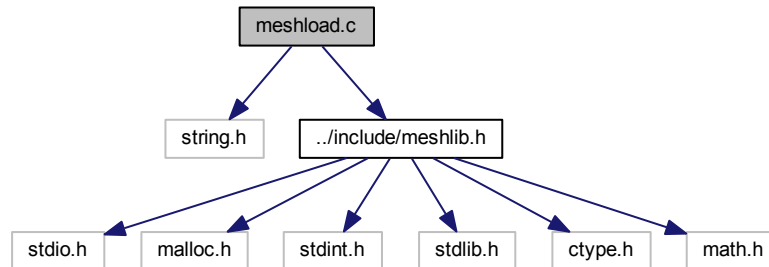


## 5.8 meshload.c File Reference

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)  
*Read a mesh from an OFF/PLY/ASC/XYZ file.*
- [MESH mesh\\_load\\_off](#) (const char \*fname)  
*Read 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)  
*Read 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

1.3.0.0

#### 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 )

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

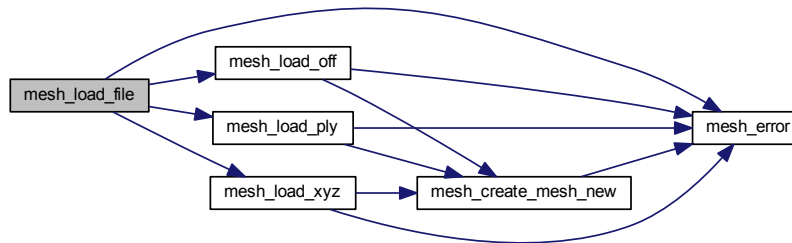
#### Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



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

Read a mesh from an OFF file.

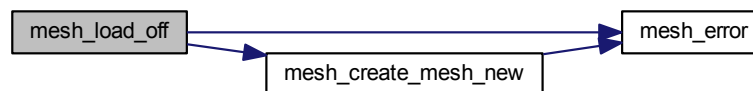
## Parameters

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

## Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.8.2.3 MESH mesh.load\_ply ( const char \* *fname* )

Read a mesh from a PLY file.

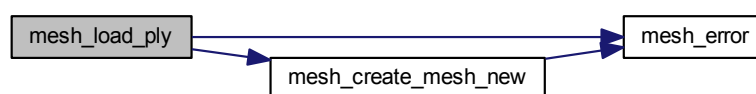
#### Parameters

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

#### Returns

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.8.2.4 MESH mesh.load\_xyz ( const char \* *fname* )

Read a mesh from an ASC/XYZ file.

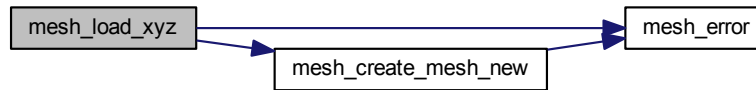
#### Parameters

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

**Returns**

Output mesh

Here is the call graph for this function:



Here is the caller graph for this function:

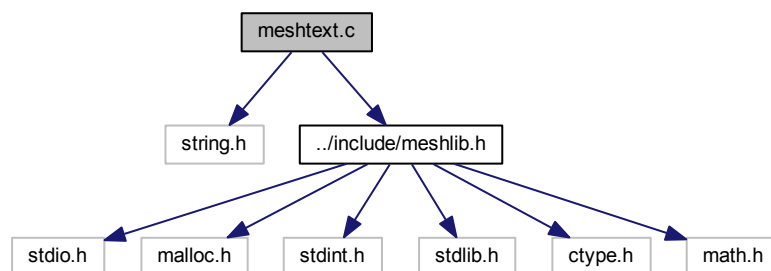


## 5.9 meshtext.c File Reference

This file contains functions pertaining to different text routines.

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

Include dependency graph for `meshtext.c`:

**Functions**

- int `mesh_isnumeric` (`FILEPOINTER` fp)  
*Checks if numeric or not.*

- int `mesh_go_next_word` (FILEPOINTER fp)  
*Points to the next word.*
- int `mesh_count_words_in_line` (FILEPOINTER fp, int \*count)  
*Counts number of words in the current line.*
- int `mesh_read_word` (FILEPOINTER fp, char \*c\_word, int sz)  
*Reads current word.*
- int `mesh_skip_line` (FILEPOINTER fp)  
*Skips to next line.*

### 5.9.1 Detailed Description

This file contains functions pertaining to different text routines.

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### Copyright

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

### 5.9.2 Function Documentation

#### 5.9.2.1 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

#### 5.9.2.2 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

### 5.9.2.3 int mesh\_isnumeric ( FILEPOINTER *fp* )

Checks if numeric or not.

#### Parameters

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

#### Returns

1 for numeric/ else - for non-numeric

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

Reads current word.

#### Parameters

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

#### Returns

Status 0 - Normal/ 1- EOF

### 5.9.2.5 int mesh\_skip\_line ( FILEPOINTER *fp* )

Skips to next line.

#### Parameters

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

#### Returns

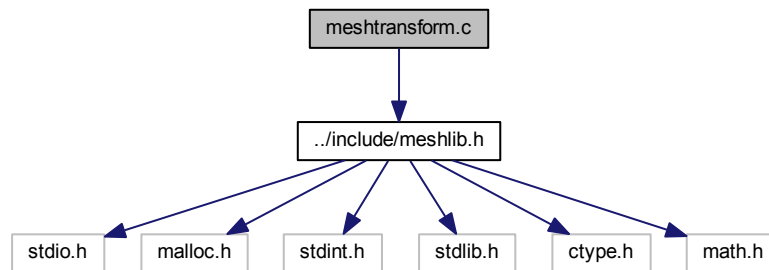
Status 0 - Normal/ 1- EOF

## 5.10 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.*

### 5.10.1 Detailed Description

This file contains functions pertaining to different mesh transformations.

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### Copyright

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

## 5.10.2 Function Documentation

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

Rotates a mesh by a given rotation.

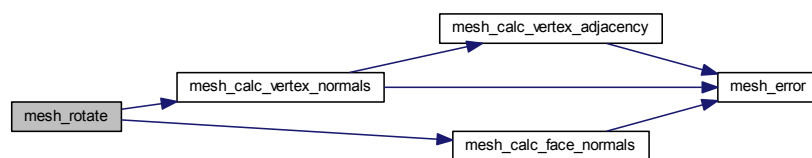
#### Parameters

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

#### Returns

Error code

Here is the call graph for this function:



### 5.10.2.2 `MESH_ROTATION mesh_rotation_create ( )`

Creates a new rotation.

#### Returns

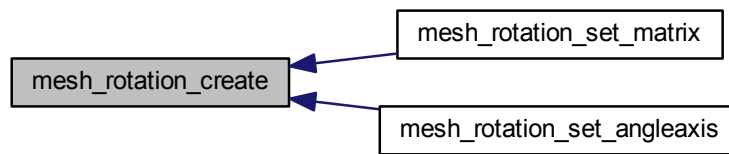
Output rotation

Here is the call graph for this function:





Here is the caller graph for this function:



#### 5.10.2.3 void mesh\_rotation\_free ( MESH\_ROTATION *r* )

Frees a given rotation.

##### Parameters

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

##### Returns

NULL

#### 5.10.2.4 MESH\_ROTATION mesh\_rotation\_set\_angleaxis ( FLOATDATA *ang*, MESH\_NORMAL *axis*, MESH\_ROTATION *r* )

Sets rotation from angle axis.

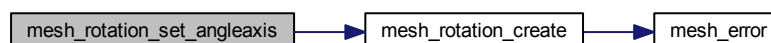
##### Parameters

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:



#### 5.10.2.5 MESH\_ROTATION mesh\_rotation\_set\_matrix ( FLOATDATA \* *mat*, MESH\_ROTATION *r* )

Sets rotation from a matrix.

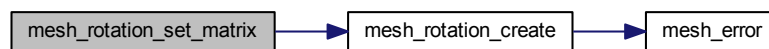
##### Parameters

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

##### Returns

Output rotation

Here is the call graph for this function:



#### 5.10.2.6 int mesh\_scale ( MESH *m*, FLOATDATA *sx*, FLOATDATA *sy*, FLOATDATA *sz* )

Scales a mesh by x, y and z amounts.

##### Parameters

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

##### Returns

Error code

#### 5.10.2.7 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

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

Translates a mesh by a given 3-d vector.

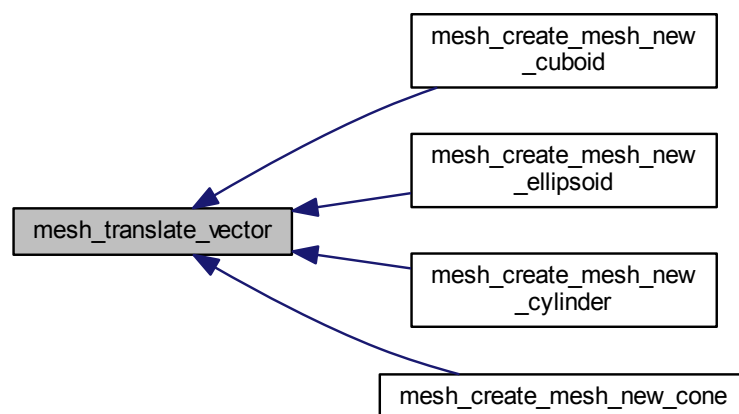
#### Parameters

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

#### Returns

Error code

Here is the caller graph for this function:



### 5.10.2.9 `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

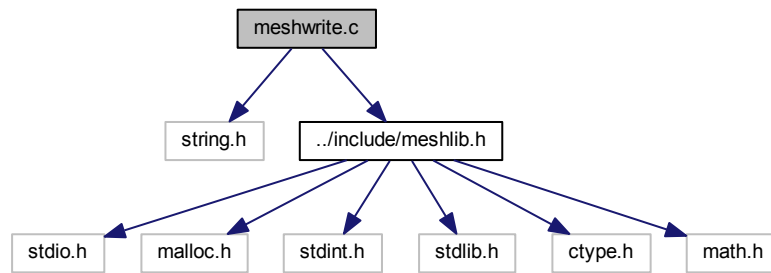
Output vertex

## 5.11 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.*

### 5.11.1 Detailed Description

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

#### Author

Sk. Mohammadul Haque

#### Version

1.3.0.0

#### Copyright

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

### 5.11.2 Function Documentation

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

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

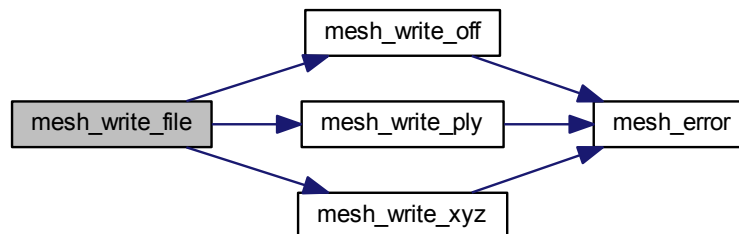
#### Parameters

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

## Returns

Error code

Here is the call graph for this function:



### 5.11.2.2 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:



### 5.11.2.3 `int mesh_write_ply ( MESH m, const char * fname )`

Write a mesh to an PLY file.

#### Parameters

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

#### Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.11.2.4 `int mesh_write_xyz ( MESH m, const char * fname )`

Write a mesh to an XYZ file.

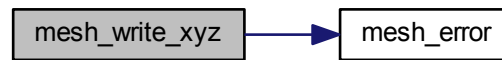
#### Parameters

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

## Returns

Error code

Here is the call graph for this function:



Here is the caller graph for this function:



# Index

a  
    mesh\_color, [10](#)

b  
    mesh\_color, [10](#)

data  
    mesh\_rotation, [11](#)  
    mesh\_transform, [12](#)

dummy  
    mesh, [8](#)

FILEPOINTER  
    meshlib.h, [42](#)

FLOATDATA  
    meshlib.h, [41](#)

faces  
    mesh, [8](#)  
    mesh\_vface, [13](#)

fareas  
    mesh, [8](#)

fcolors  
    mesh, [8](#)

fnormals  
    mesh, [8](#)

g  
    mesh\_color, [10](#)

INTDATA  
    meshlib.h, [41](#)

INTDATA2  
    meshlib.h, [42](#)

is\_faces  
    mesh, [8](#)

is\_fareas  
    mesh, [8](#)

is\_fcolors  
    mesh, [8](#)

is\_fnormals  
    mesh, [8](#)

is\_loaded  
    mesh, [8](#)

is\_trimesh  
    mesh, [8](#)

is\_vcolors  
    mesh, [8](#)

is\_vertices  
    mesh, [9](#)

is\_vfaces  
    mesh, [9](#)

is\_vnormals  
    mesh, [9](#)

items  
    mesh\_struct, [11](#)  
    mesh\_struct2, [12](#)

MESH  
    meshlib.h, [43](#)

MESH\_COLOR  
    meshlib.h, [43](#)

MESH\_ERR\_FNOTOPEN  
    meshlib.h, [41](#)

MESH\_ERR\_MALLOC  
    meshlib.h, [41](#)

MESH\_ERR\_UNKNOWN  
    meshlib.h, [41](#)

MESH\_FACE  
    meshlib.h, [43](#)

MESH\_FLOATDATA\_TYPE  
    meshlib.h, [41](#)

MESH\_INTDATA\_TYPE  
    meshlib.h, [41](#)

MESH\_NORMAL  
    meshlib.h, [43](#)

MESH\_PI  
    meshlib.h, [42](#)

MESH\_ROTATION  
    meshlib.h, [43](#)

MESH\_STRUCT  
    meshlib.h, [43](#)

MESH\_STRUCT2  
    meshlib.h, [43](#)

MESH\_TRANSFORM  
    meshlib.h, [44](#)

MESH\_VECTOR3  
    meshlib.h, [44](#)

MESH\_VERTEX  
    meshlib.h, [44](#)

MESH\_VFACE  
    meshlib.h, [44](#)

mesh, [7](#)  
    dummy, [8](#)  
    faces, [8](#)  
    fareas, [8](#)  
    fcolors, [8](#)  
    fnormals, [8](#)  
    is\_faces, [8](#)  
    is\_fareas, [8](#)  
    is\_fcolors, [8](#)  
    is\_fnormals, [8](#)



- is\_loaded, [8](#)
- is\_trimesh, [8](#)
- is\_vcolors, [8](#)
- is\_vertices, [9](#)
- is\_vfaces, [9](#)
- is\_vnormals, [9](#)
- meshlib.h, [42](#)
- num\_faces, [9](#)
- num\_vertices, [9](#)
- origin\_type, [9](#)
- vcolors, [9](#)
- vertices, [9](#)
- vfaces, [9](#)
- vnormals, [9](#)
- mesh\_bilateral\_filter
  - meshfilter.c, [35](#)
  - meshlib.h, [44](#)
- mesh\_calc\_face\_normal
  - meshcalc.c, [16](#)
  - meshlib.h, [45](#)
- mesh\_calc\_face\_normals
  - meshcalc.c, [16](#)
  - meshlib.h, [45](#)
- mesh\_calc\_triangle\_area
  - meshcalc.c, [17](#)
  - meshlib.h, [46](#)
- mesh\_calc\_vertex\_adjacency
  - meshcalc.c, [18](#)
  - meshlib.h, [47](#)
- mesh\_calc\_vertex\_normals
  - meshcalc.c, [19](#)
  - meshlib.h, [48](#)
- mesh\_color, [10](#)
  - a, [10](#)
  - b, [10](#)
  - g, [10](#)
  - meshlib.h, [43](#)
  - r, [10](#)
- mesh\_count\_words\_in\_line
  - meshlib.h, [49](#)
  - meshtext.c, [75](#)
- mesh\_create\_mesh\_new
  - meshcreate.c, [27](#)
  - meshlib.h, [49](#)
- mesh\_create\_mesh\_new\_cone
  - meshcreate.c, [28](#)
  - meshlib.h, [50](#)
- mesh\_create\_mesh\_new\_cuboid
  - meshcreate.c, [28](#)
  - meshlib.h, [50](#)
- mesh\_create\_mesh\_new\_cylinder
  - meshcreate.c, [29](#)
  - meshlib.h, [51](#)
- mesh\_create\_mesh\_new\_ellipsoid
  - meshcreate.c, [29](#)
  - meshlib.h, [51](#)
- mesh\_cross\_normal
  - meshcalc.c, [20](#)
  - meshlib.h, [52](#)
- mesh\_cross\_vector3
  - meshcalc.c, [20](#)
  - meshlib.h, [52](#)
- mesh\_draw\_mesh
  - meshdraw.c, [31](#)
  - meshlib.h, [53](#)
- mesh\_error
  - mesherror.c, [33](#)
  - meshlib.h, [53](#)
- mesh\_face, [10](#)
  - meshlib.h, [43](#)
  - num\_vertices, [10](#)
  - vertices, [10](#)
- mesh\_find
  - meshcalc.c, [20](#)
  - meshlib.h, [54](#)
- mesh\_find2
  - meshcalc.c, [21](#)
  - meshlib.h, [55](#)
- mesh\_free\_mesh
  - meshcreate.c, [30](#)
  - meshlib.h, [55](#)
- mesh\_go\_next\_word
  - meshlib.h, [55](#)
  - meshtext.c, [75](#)
- mesh\_isnumeric
  - meshlib.h, [55](#)
  - meshtext.c, [75](#)
- mesh\_laplacian\_filter
  - meshfilter.c, [36](#)
  - meshlib.h, [55](#)
- mesh\_load\_file
  - meshlib.h, [56](#)
  - meshload.c, [71](#)
- mesh\_load\_off
  - meshlib.h, [56](#)
  - meshload.c, [72](#)
- mesh\_load\_ply
  - meshlib.h, [57](#)
  - meshload.c, [72](#)
- mesh\_load\_xyz
  - meshlib.h, [58](#)
  - meshload.c, [73](#)
- mesh\_normal
  - meshlib.h, [43](#)
- mesh\_read\_word
  - meshlib.h, [58](#)
  - meshtext.c, [76](#)
- mesh\_remove\_boundary\_faces
  - meshclean.c, [23](#)
  - meshlib.h, [59](#)
- mesh\_remove\_boundary\_vertices
  - meshclean.c, [23](#)
  - meshlib.h, [59](#)
- mesh\_remove\_close\_vertices
  - meshclean.c, [23](#)
  - meshlib.h, [59](#)

- mesh\_remove\_ear\_faces
  - meshclean.c, 23
  - meshlib.h, 60
- mesh\_remove\_triangles\_with\_small\_area
  - meshclean.c, 24
  - meshlib.h, 60
- mesh\_remove\_unreferenced\_vertices
  - meshclean.c, 25
  - meshlib.h, 61
- mesh\_remove\_zero\_area\_faces
  - meshclean.c, 25
  - meshlib.h, 62
- mesh\_restricted\_laplacian\_filter
  - meshfilter.c, 36
  - meshlib.h, 62
- mesh\_rotate
  - meshlib.h, 63
  - meshtransform.c, 78
- mesh\_rotation, 11
  - data, 11
  - meshlib.h, 43
- mesh\_rotation\_create
  - meshlib.h, 63
  - meshtransform.c, 78
- mesh\_rotation\_free
  - meshlib.h, 64
  - meshtransform.c, 79
- mesh\_rotation\_set\_angleaxis
  - meshlib.h, 64
  - meshtransform.c, 79
- mesh\_rotation\_set\_matrix
  - meshlib.h, 65
  - meshtransform.c, 79
- mesh\_scale
  - meshlib.h, 65
  - meshtransform.c, 80
- mesh\_skip\_line
  - meshlib.h, 65
  - meshtext.c, 76
- mesh\_struct, 11
  - items, 11
  - meshlib.h, 43
  - num\_items, 11
- mesh\_struct2, 12
  - items, 12
  - meshlib.h, 43
  - num\_items, 12
- mesh\_transform, 12
  - data, 12
  - meshlib.h, 44
- mesh\_translate
  - meshlib.h, 66
  - meshtransform.c, 80
- mesh\_translate\_vector
  - meshlib.h, 66
  - meshtransform.c, 80
- mesh\_upsample
  - meshcalc.c, 21
- meshlib.h, 67
- mesh\_vector3, 12
  - meshlib.h, 44
  - x, 13
  - y, 13
  - z, 13
- mesh\_vertex
  - meshlib.h, 44
- mesh\_vertex\_rotate
  - meshlib.h, 67
  - meshtransform.c, 81
- mesh\_vface, 13
  - faces, 13
  - meshlib.h, 44
  - num\_faces, 13
- mesh\_write\_file
  - meshlib.h, 68
  - meshwrite.c, 82
- mesh\_write\_off
  - meshlib.h, 68
  - meshwrite.c, 83
- mesh\_write\_ply
  - meshlib.h, 69
  - meshwrite.c, 84
- mesh\_write\_xyz
  - meshlib.h, 70
  - meshwrite.c, 84
- meshcalc.c, 15
  - mesh\_calc\_face\_normal, 16
  - mesh\_calc\_face\_normals, 16
  - mesh\_calc\_triangle\_area, 17
  - mesh\_calc\_vertex\_adjacency, 18
  - mesh\_calc\_vertex\_normals, 19
  - mesh\_cross\_normal, 20
  - mesh\_cross\_vector3, 20
  - mesh\_find, 20
  - mesh\_find2, 21
  - mesh\_upsample, 21
- meshclean.c, 21
  - mesh\_remove\_boundary\_faces, 23
  - mesh\_remove\_boundary\_vertices, 23
  - mesh\_remove\_close\_vertices, 23
  - mesh\_remove\_ear\_faces, 23
  - mesh\_remove\_triangles\_with\_small\_area, 24
  - mesh\_remove\_unreferenced\_vertices, 25
  - mesh\_remove\_zero\_area\_faces, 25
- meshcreate.c, 26
  - mesh\_create\_mesh\_new, 27
  - mesh\_create\_mesh\_new\_cone, 28
  - mesh\_create\_mesh\_new\_cuboid, 28
  - mesh\_create\_mesh\_new\_cylinder, 29
  - mesh\_create\_mesh\_new\_ellipsoid, 29
  - mesh\_free\_mesh, 30
- meshdraw.c, 30
  - mesh\_draw\_mesh, 31
- mesherror.c, 32
  - mesh\_error, 33
- meshfilter.c, 34

- mesh\_bilateral\_filter, 35
- mesh\_laplacian\_filter, 36
- mesh\_restricted\_laplacian\_filter, 36
- meshlib.h, 37
  - FILEPOINTER, 42
  - FLOATDATA, 41
  - INTDATA, 41
  - INTDATA2, 42
  - MESH, 43
  - MESH\_COLOR, 43
  - MESH\_ERR\_FNOTOPEN, 41
  - MESH\_ERR\_MALLOC, 41
  - MESH\_ERR\_UNKNOWN, 41
  - MESH\_FACE, 43
  - MESH\_FLOATDATA\_TYPE, 41
  - MESH\_INTDATA\_TYPE, 41
  - MESH\_NORMAL, 43
  - MESH\_PI, 42
  - MESH\_ROTATION, 43
  - MESH\_STRUCT, 43
  - MESH\_STRUCT2, 43
  - MESH\_TRANSFORM, 44
  - MESH\_VECTOR3, 44
  - MESH\_VERTEX, 44
  - MESH\_VFACE, 44
- mesh, 42
  - mesh\_bilateral\_filter, 44
  - mesh\_calc\_face\_normal, 45
  - mesh\_calc\_face\_normals, 45
  - mesh\_calc\_triangle\_area, 46
  - mesh\_calc\_vertex\_adjacency, 47
  - mesh\_calc\_vertex\_normals, 48
  - mesh\_color, 43
  - mesh\_count\_words\_in\_line, 49
  - mesh\_create\_mesh\_new, 49
  - mesh\_create\_mesh\_new\_cone, 50
  - mesh\_create\_mesh\_new\_cuboid, 50
  - mesh\_create\_mesh\_new\_cylinder, 51
  - mesh\_create\_mesh\_new\_ellipsoid, 51
  - mesh\_cross\_normal, 52
  - mesh\_cross\_vector3, 52
  - mesh\_draw\_mesh, 53
  - mesh\_error, 53
  - mesh\_face, 43
  - mesh\_find, 54
  - mesh\_find2, 55
  - mesh\_free\_mesh, 55
  - mesh\_go\_next\_word, 55
  - mesh\_isnumeric, 55
  - mesh\_laplacian\_filter, 55
  - mesh\_load\_file, 56
  - mesh\_load\_off, 56
  - mesh\_load\_ply, 57
  - mesh\_load\_xyz, 58
  - mesh\_normal, 43
  - mesh\_read\_word, 58
  - mesh\_remove\_boundary\_faces, 59
  - mesh\_remove\_boundary\_vertices, 59
  - mesh\_remove\_close\_vertices, 59
  - mesh\_remove\_ear\_faces, 60
  - mesh\_remove\_triangles\_with\_small\_area, 60
  - mesh\_remove\_unreferenced\_vertices, 61
  - mesh\_remove\_zero\_area\_faces, 62
  - mesh\_restricted\_laplacian\_filter, 62
  - mesh\_rotate, 63
  - mesh\_rotation, 43
  - mesh\_rotation\_create, 63
  - mesh\_rotation\_free, 64
  - mesh\_rotation\_set\_angleaxis, 64
  - mesh\_rotation\_set\_matrix, 65
  - mesh\_scale, 65
  - mesh\_skip\_line, 65
  - mesh\_struct, 43
  - mesh\_struct2, 43
  - mesh\_transform, 44
  - mesh\_translate, 66
  - mesh\_translate\_vector, 66
  - mesh\_upsample, 67
  - mesh\_vector3, 44
  - mesh\_vertex, 44
  - mesh\_vertex\_rotate, 67
  - mesh\_vface, 44
  - mesh\_write\_file, 68
  - mesh\_write\_off, 68
  - mesh\_write\_ply, 69
  - mesh\_write\_xyz, 70
- meshload.c, 70
  - mesh\_load\_file, 71
  - mesh\_load\_off, 72
  - mesh\_load\_ply, 72
  - mesh\_load\_xyz, 73
- meshtext.c, 74
  - mesh\_count\_words\_in\_line, 75
  - mesh\_go\_next\_word, 75
  - mesh\_isnumeric, 75
  - mesh\_read\_word, 76
  - mesh\_skip\_line, 76
- meshtransform.c, 76
  - mesh\_rotate, 78
  - mesh\_rotation\_create, 78
  - mesh\_rotation\_free, 79
  - mesh\_rotation\_set\_angleaxis, 79
  - mesh\_rotation\_set\_matrix, 79
  - mesh\_scale, 80
  - mesh\_translate, 80
  - mesh\_translate\_vector, 80
  - mesh\_vertex\_rotate, 81
- meshwrite.c, 81
  - mesh\_write\_file, 82
  - mesh\_write\_off, 83
  - mesh\_write\_ply, 84
  - mesh\_write\_xyz, 84
- num\_faces
  - mesh, 9
  - mesh\_vface, 13
- num\_items

- mesh\_struct, [11](#)
  - mesh\_struct2, [12](#)
- num\_vertices
  - mesh, [9](#)
  - mesh\_face, [10](#)
- origin\_type
  - mesh, [9](#)
- r
  - mesh\_color, [10](#)
- vcolors
  - mesh, [9](#)
- vertices
  - mesh, [9](#)
  - mesh\_face, [10](#)
- vfaces
  - mesh, [9](#)
- vnormals
  - mesh, [9](#)
- x
  - mesh\_vector3, [13](#)
- y
  - mesh\_vector3, [13](#)
- z
  - mesh\_vector3, [13](#)