ADMesh Documentation

Release 0.98.1

ADMesh contributors

April 15, 2015

Contents

CHAPTER 1

Contents

1.1 Triangular mesh and the STL format

FIXME

1.2 ADMesh command line tool

ADMesh command line tool is executed as follows:

```
admesh [OPTION]... file
```

By default, ADMesh performs all of the mesh checking and repairing options on the input file. This means that is checks exact, nearby, remove-unconnected, fill-holes, normal-directions, and normal-values. The file type (ASCII or binary) is automatically detected. The input file is not modified unless it is specified by the `--write` option. If the following command line was used:

```
admesh sphere.stl
```

the file `sphere.stl` would be opened and read, it would be checked and fixed if necessary, and the results of processing would be printed out. The results would not be saved.

If any of the options `--exact`, `--nearby`, `--remove-unconnected`, `--fill-holes`, `--normal-directions`, `--reverse-all`, `--normal-values`, or `--no-check` are given, then no other checks besides that one will be done unless they are specified or unless they are required by ADMesh before the specified check can be done. For example the following invocation:

```
admesh --remove-unconnected sphere.stl
```

would first do an exact check because it is required, and then the unconnected facets would be removed. The results would be printed and no other checks would be done.

1.2.1 Examples

To perform all checks except for nearby, the following command line would be used:

```
admesh --exact --remove-unconnected --fill-holes \
       --normal-directions --normal-values sphere.stl
```

Actually, since the `--exact` check is required by ADMesh before `--remove-unconnected`, and `--remove-unconnected` is required before `--fill-holes`, the above command line could be shortened as follows with the same results:

```
admesh --fill-holes --normal-directions --normal-values sphere.stl
```

And again the same results could be achieved using the short options:

```
admesh -fudev sphere.stl
```

or:

```
admesh -fdv sphere.stl
```

The following command lines do the same thing:

```
admesh sphere.stl
admesh -fundev sphere.stl
admesh -f -u -n -d -e -v sphere.stl
```

since the `-fundev` options are implied by default. To eliminate one of the checks, just remove the letter of the check to eliminate from the "word" `fundev`.

1.2.2 Options

ADMesh supports the following options, grouped by type.

Mesh Transformation and Manipulation Options

```
--x-rotate=angle     Rotate CCW about x-axis by angle degrees
--y-rotate=angle     Rotate CCW about y-axis by angle degrees
--z-rotate=angle     Rotate CCW about z-axis by angle degrees
--xy-mirror          Mirror about the xy plane
--yz-mirror          Mirror about the yz plane
--xz-mirror          Mirror about the xz plane
--scale=factor       Scale the file by factor (multiply by factor)
--translate=x,y,z    Translate the file to x, y, and z
--merge=name         Merge file called name with input file
```

Mesh Checking and Repairing Options

```
-e, --exact              Only check for perfectly matched edges
-n, --nearby             Find and connect nearby facets. Correct bad facets
-t, --tolerance=tol      Initial tolerance to use for nearby check = tol
-i, --iterations=i       Number of iterations for nearby check = i
-m, --increment=inc      Amount to increment tolerance after iteration=inc
-u, --remove-unconnected Remove facets that have 0 neighbors
-f, --fill-holes         Add facets to fill holes
-d, --normal-directions  Check and fix direction of normals (ie. CW, CCW)
    --reverse-all        Reverse the directions of all facets and normals
-v, --normal-values      Check and fix normal values
-c, --no-check           Don't do any check on input file
```

The default value for tolerance is the length of the shortest edge of the mesh. The default number of iterations is 2, and the default increment is 0.01% of the diameter of a sphere that encloses the entire mesh.

File Output Options

```
-b, --write-binary-stl=name   Output a binary STL file called name
-a, --write-ascii-stl=name    Output an ASCII STL file called name
    --write-off=name          Output a Geomview OFF format file called name
    --write-dxf=name          Output a DXF format file called name
    --write-vrml=name         Output a VRML format file called name
```

The input file is not modified by ADMesh so the only way to preserve any modifications that have been made to the input file is to use one of the `--write` options.

If the user wants to modify (overwrite) the input file, then the input file can also be specified for the `--write` option. For example, to convert an input ASCII STL file called `sphere.stl` to a binary STL file, overwriting the original file, and performing no checks, the following command line would be used:

```
admesh --write-binary-stl=sphere.stl --no-check sphere.stl
```

Miscellaneous Options

```
--help                Display the help and exit
--version             Output version information and exit
```

1.2.3 Mesh Transformation and Manipulation Options

```
--x-rotate=angle
--y-rotate=angle
--z-rotate=angle
```

Rotate the entire mesh about the specified axis by the given number of degrees. The rotation is counter-clockwise about the axis as seen by looking along the positive axis towards the origin.

```
--xy-mirror
--yz-mirror
--xz-mirror
```

Mirror the mesh about the specified plane. Mirroring involves reversing the sign of all of the coordinates in a particular axis. For example, to mirror a mesh about the xy plane, the signs of all of the z coordinates in the mesh are reversed.

```
--scale=factor
```

Scale the mesh by the given factor. This multiplies all of the coordinates by the specified number. This option could be used to change the "units" (there are no units explicitly specified in an STL file) of the mesh. For example, to change a part from inches to millimeters, just use the –scale=25.4 option.

```
--translate=x,y,z
```

Translate the mesh to the position x,y,z. This moves the minimum x, y, and z values of the mesh to the specified position. For example, given a mesh that has the following initial minimum and maximum coordinate values:

```
Min X =  4.000000, Max X =  5.000000
Min Y =  1.000000, Max Y =  3.000000
Min Z = -7.000000, Max Z = -2.000000
```

if the option `--translate=1,2,3` is specified, the final values will be:

```
Min X =  1.000000, Max X =  2.000000
Min Y =  2.000000, Max Y =  4.000000
Min Z =  3.000000, Max Z =  8.000000
```

The translate option is often used to translate a mesh with arbitrary minimum and maximum coordinates to 0,0,0. Usually, translation is also required when merging two files.

```
merge=name
```

Merge the specified file with the input file. No translation is done, so if, for example, a file was merged with itself, the resulting file would end up with two meshes exactly the same, occupying exactly the same space. So generally, translations need to be done to the files to be merged so that when the two meshes are merged into one, the two resulting parts are properly spaced. If you know the nature of the parts to be merged, it is possible to "nest" one part inside the other. Note, however, that no warnings will be given if one part intersects with the other.

It is possible to place one part against another, with no space in between, but you will still end up with two separately defined parts. If such a mesh was made on a rapid-prototyping machine, the result would depend on the nature of the machine. Machines that use a photopolymer would produce a single solid part because the two parts would be "bonded" during the build process. Machines that use a cutting process would yield two or more parts.

A copy of a mesh can be made by using the `--merge` and `--translate` options at the same time. For example, given a file called `block.stl` with the following size:

```
Min X =  0.000000, Max X =  2.000000
Min Y =  0.000000, Max Y =  2.000000
Min Z =  0.000000, Max Z =  2.000000
```

to create a file called `2blocks.stl` that contains two of the parts separated by 1 unit in the x direction, the following command line would be used:

```
admesh --translate=3,0,0 --merge=block.stl --write-binary=2blocks.stl block.stl
```

This would yield a binary STL file called 2blocks.stl with the following size:

```
Min X =  0.000000, Max X =  5.000000
Min Y =  0.000000, Max Y =  2.000000
Min Z =  0.000000, Max Z =  2.000000
```

1.2.4 Mesh Checking and Repairing Options

```
-e, --exact
```

Check each facet of the mesh for its 3 neighbors. Since each facet is a triangle, there should be exactly 3 neighboring facets for every facet in the mesh. Since the mesh defines a solid, there should be no unconnected edges in the mesh. When this option is specified, the 3 neighbors of every facet are searched for and, if found, the neighbors are added to an internal list that keeps track of the neighbors of each facet. A facet is only considered a neighbor if two of its vertices EXACTLY match two of the vertices of another facet. That means that there must be 0 difference between the x, y, and z coordinates of the two vertices of the first facet and the two vertices of the second facet.

Degenerate facets (facets with two or more vertices equal to each other) are removed during the exact check. No other changes are made to the mesh. An exact check is always done before any of the other checking and repairing options even if `--exact` isn't specified. There is one exception to this rule; no exact check needs to be done before the `--normal-values` option.

```
-n, --nearby
-t, --tolerance=tol
-i, --iterations=i
-m, --increment=inc
```

Checks each unconnected facet of the mesh for facets that are almost connected but not quite. Due to round-off errors and other factors, it is common for a mesh to have facets with neighbors that are very close but don't match exactly. Often, this difference is only in the 8th decimal place of the vertices, but these facets will not show up as neighbors during the exact check. This option finds these nearby neighbors and it changes their vertices so that they match exactly. The exact check is alway done before the nearby check, so only facets that remain unconnected after the exact check are candidates for the nearby check.

The `--tolerance=tol` option is used to specify the distance that is searched for the neighboring facet. By default, this value is set automatically by ADMesh to be the length of the shortest edge of the mesh. This value is used because it makes it unlikely for a facet that shouldn't be a neighbor to be found and matched as a neighbor. If the tolerance is too big, then some facets could end up connected that should definitely not be connected. This could create a "mobius part" that is not a valid solid. If this occurs, it can be seen by checking the value of *Backwards edges* that is printed after processing. (The number of backwards edges should be 0 for a valid solid.)

The `--iterations=i` and `--increment=inc` options are used together to gradually connect nearby facets using progressively larger tolerances. This helps to prevent incorrect connects but can also allow larger tolerances to be used. The `--iterations` option gives the number of times that facets are checked for nearby facets, each time using a larger tolerance. The `--increment=inc` option gives the amount that the tolerance is increased after each iteration. The number specified by `inc` is added to the tolerance that was used in the previous iteration. If all of the facets are connected, no further nearby checks will be done.

```
-f, --fill-holes
```

Fill holes in the mesh by adding facets. This is done after the exact check and after nearby check (if any nearby check is done). If there are still unconnected facets, then facets will be added to the mesh, connecting the unconnected facets, until all of the holes have been filled. This is guaranteed to completely fix all unconnected facets. However, the resulting mesh may or may not be what the user expects.

```
-d, --normal-directions
```

Check and fix if necessary the directions of the facets. This only deals with whether the vertices of all the facets are oriented clockwise or counterclockwise, it doesn't check or modify the value of the normal vector. Every facet should have its vertices defined in a counterclockwise order when looked at from the outside of the part. This option will orient all of the vertices so that they are all facing in the same direction. However, it it possible that this option will make all of the facets facet inwards instead of outwards. The algorithm tries to get a clue of which direction is inside and outside by checking the value of the normal vector so the chance is very good that the resulting mesh will be correct. However, it doesn't explicitly check to find which direction is inside and which is outside.

```
--reverse-all
```

Reverses the directions of all of the facets and normals. If the `--normal-directions` option ended up making all of the facets facing inwards instead of outwards, then this option can be used to reverse all of the facets. It is up to the user to determine if the facets are facing inwards and if they need reversing. This option also fixes and updates the normal vector for each facet.

```
-v, --normal-values
```

Checks and fixes if necessary the normal vectors of every facet. The normal vector will point outward for a counter-clockwise facet. The length of the normal vector will be 1.

```
-c, --no-check
```

Don't do any checks or modifications to the input file. By default, ADMesh performs all processes (exact, nearby, remove_unconnected, fill-holes, normal-directions, and normals-values) on the input file. If the `--no-check` option is specified, no checks or modifications will be made on the input file. This could be used, for example, to translate an ASCII STL file to a binary STL file, with no modifications made. A command line such as the following might be used:

```
admesh --no-check --write-binary-stl=newblock.stl --translate=0,0,0 block.stl
```

This would open the file `block.stl`, would translate it to 0,0,0 no checks would be performed and a binary STL file of the translated mesh would be written to `newblock.stl`.

1.2.5 ADMesh output

After ADMesh has processed a mesh, it prints out a page of information about that mesh. The output looks like the following:

```
================= Results produced by ADMesh version 0.98 =================
Input file         : sphere.stl
File type          : Binary STL file
Header             : Processed by ADMesh version 0.98
============== Size ==============
Min X = -1.334557, Max X = 1.370952
Min Y = -1.377953, Max Y = 1.377230
Min Z = -1.373225, Max Z = 1.242838
========= Facet Status ========== Original ============ Final ====
Number of facets                 :  3656                3656
Facets with 1 disconnected edge  :    18                   0
Facets with 2 disconnected edges :     3                   0
Facets with 3 disconnected edges :     0                   0
Total disconnected facets        :    21                   0
=== Processing Statistics ===     ===== Other Statistics =====
Number of parts       :     1        Volume   :  10.889216
Degenerate facets     :     0
Edges fixed           :    24
Facets removed        :     0
Facets added          :     0
Facets reversed       :     0
Backwards edges       :     0
Normals fixed         :     0
```

Description of Output

The following describes the output information line by line.

```
Input file         : sphere.stl
```

The name of the file that was read.

```
File type          : Binary STL file
```

The type of file. Currently, the only two possibilities are Binary STL file and ASCII STL file. ADMesh automatically detect the type of input file.

```
Header             : Processed by ADMesh version 0.98
```

The first 80 characters of the STL file. The first 80 bytes of a binary STL file or the first line of an ASCII STL file can contain some text. Usually, the CAD system that has created that file, or the last program to process that file puts its name in the header. ADMesh puts its own string in the header when it saves the file.

```
============== Size ==============
Min X = -1.334557, Max X = 1.370952
Min Y = -1.377953, Max Y = 1.377230
Min Z = -1.373225, Max Z = 1.242838
```

This section gives the boundaries of the mesh. The mesh will fit just inside a box of this size.

```
========= Facet Status ========== Original ============ Final ====
Number of facets                :  3656                 3656
Facets with 1 disconnected edge  :    18                   0
Facets with 2 disconnected edges :     3                   0
Facets with 3 disconnected edges :     0                   0
Total disconnected facets        :    21                   0
```

Information about the quality of the mesh before, and after processing by ADMesh. The number of facets gives an idea about the complexity and accuracy of the mesh. Disconnected facets will fall into 3 categories. Some facets will have only one disconnected edge, some will have 2 edges disconnected, and some will have all 3 edges disconnected. Of course, for a valid solid mesh, there should be 0 disconnected facets.

```
=== Processing Statistics ===
Number of parts       :     1
```

This is the total number of separate parts in the file. This can be a very useful indication of whether your file is correct. Sometimes, the user of the CAD system that creates the mesh just puts several pieces together next to each other, and then outputs the mesh. This might not cause any problems for a rapid prototyping system that uses a photopolymer because all of the parts will be "glued" together anyway during the build. However, a rapid prototyping machine that is based on cutting will cut each one of the parts individually and the result will be many parts that need to be glued together. The number of parts is counted during `--normal-directions`, so if the `--normal-directions` check is eliminated, then the number of parts will read 0.

```
Degenerate facets     :     0
```

Number of degenerate facets in the input file. A degenerate facet is a facet that has two or more vertices exactly the same. The resulting facet is just a line (if two vertices are the same) or could even be a point (if all 3 vertices are the same). These facets add no information to the file and are removed by ADMesh during processing.

```
Edges fixed           :    24
```

The total number of edges that were fixed by moving the vertices slightly during the nearby check. This does not include facets that were added by `--fill-holes`.

```
Facets removed        :     0
```

The total number of facets removed. There are two cases where facets might be removed. First, all degenerate facets in the input file are removed. Second, if there are any completely unconnected facets (facets with 3 disconnected edges) after the exact and nearby checks, then these facets will be removed by `--remove-unconnected`.

```
Facets added          :     0
```

Number of facets that have been added by ADMesh to the original mesh. Facets are only added during `--fill-holes`. So this number represents the number of facets that had to be added to fill all of the holes, if any, in the original mesh.

```
Facets reversed       :     0
```

The number of facets that were reversed during `--normal-directions`. This only relates to the order of the vertices of the facet (CW or CCW), it has nothing to do with the value of the normal vector.

```
Backwards edges       :     0
```

The number of edges that are backwards. After ADMesh has finished all of the checks and processing, it verifies the results. If the normal-directions check has been done then the number of backwards edges should be 0. If it is not, then a "mobius part" has been created which is not a valid solid mesh. In this case the mesh can be processed again, but a smaller tolerance on the nearby check should be used or no nearby check should be done.

```
Normals fixed          :      0
```

The number of normal vectors that have been fixed. During the normal-values check, ADMesh calculates the value of every facet and compares the result with the normal vector from the input file. If the result is not within a fixed tolerance, then the normal is said to be fixed. Actually, for consistency, every normal vector is rewritten with the new calculated normal, even if the original normal was within tolerance. However, the normals that were within tolerance are not counted by normals fixed.

1.3 The C library API

Defines

STL_MAX(A, B)

STL_MIN(A, B)

ABS(X)

LABEL_SIZE

NUM_FACET_SIZE

HEADER_SIZE

STL_MIN_FILE_SIZE

ASCII_LINES_PER_FACET

SIZEOF_EDGE_SORT

SIZEOF_STL_FACET

Typedefs

typedef char **stl_extra**[2]

Enums

enum stl_type
Type of STL file.

Values:

binary

ascii

inmemory

Functions

void **stl_open**(*stl_file* *stl, char *file)
Open an STL file and load it's contents.

Warning

As IO operation, this could result in an error, always check the error flag with *stl_get_error()* or use *stl_exit_on_error()* after using *stl_open()*

Parameters

- `stl` - The struct to load the file data to
- `file` - Path to the STL file

void **stl_close** (*stl_file* **stl*)
Perform cleanup on *stl_file*.

This function frees memory, always use it, when you no longer needs the *stl_file* instance

Parameters

- `stl` - What to close

void **stl_stats_out** (*stl_file* **stl*, FILE **file*, char **input_file*)
Print statistics in human readable form to some file.

Parameters

- `stl` - Where to obtain the statisctics
- `file` - Where to print the statisctics to (can be stdout)
- `input_file` - What filename to use in the human readable output

void **stl_print_edges** (*stl_file* **stl*, FILE **file*)
Print edges to some file.

Warning

This prints from edge_start array, that is never populated and thus this will never actually work

Parameters

- `stl` - From what data
- `file` - Where to print the edges to (can be stdout)

void **stl_print_neighbors** (*stl_file* **stl*, char **file*)

void **stl_put_little_int** (FILE **fp*, int *value_in*)

void **stl_put_little_float** (FILE **fp*, float *value_in*)

void **stl_write_ascii** (*stl_file* **stl*, **const** char **file*, **const** char **label*)

void **stl_write_binary** (*stl_file* **stl*, **const** char **file*, **const** char **label*)

void **stl_write_binary_block** (*stl_file* **stl*, FILE **fp*)

void **stl_check_facets_exact** (*stl_file* **stl*)

void **stl_check_facets_nearby** (*stl_file* **stl*, float *tolerance*)

void **stl_remove_unconnected_facets** (*stl_file* **stl*)

void **stl_write_vertex** (*stl_file* **stl*, int *facet*, int *vertex*)

void **stl_write_facet** (*stl_file* **stl*, char **label*, int *facet*)

void **stl_write_edge** (*stl_file* **stl*, char **label*, *stl_hash_edge* *edge*)

void **stl_write_neighbor** (*stl_file* *stl, int *facet*)

void **stl_write_quad_object** (*stl_file* *stl, char *file)

void **stl_verify_neighbors** (*stl_file* *stl)

void **stl_fill_holes** (*stl_file* *stl)

void **stl_fix_normal_directions** (*stl_file* *stl)

void **stl_fix_normal_values** (*stl_file* *stl)

void **stl_reverse_all_facets** (*stl_file* *stl)

void **stl_translate** (*stl_file* *stl, float *x*, float *y*, float *z*)

void **stl_translate_relative** (*stl_file* *stl, float *x*, float *y*, float *z*)

void **stl_scale_versor** (*stl_file* *stl, float *versor*)

void **stl_scale** (*stl_file* *stl, float *factor*)

void **stl_rotate_x** (*stl_file* *stl, float *angle*)

void **stl_rotate_y** (*stl_file* *stl, float *angle*)

void **stl_rotate_z** (*stl_file* *stl, float *angle*)

void **stl_mirror_xy** (*stl_file* *stl)

void **stl_mirror_yz** (*stl_file* *stl)

void **stl_mirror_xz** (*stl_file* *stl)

void **stl_open_merge** (*stl_file* *stl, char *file)

void **stl_invalidate_shared_vertices** (*stl_file* *stl)

void **stl_generate_shared_vertices** (*stl_file* *stl)

void **stl_write_obj** (*stl_file* *stl, char *file)

void **stl_write_off** (*stl_file* *stl, char *file)

void **stl_write_dxf** (*stl_file* *stl, char *file, char *label)

void **stl_write_vrml** (*stl_file* *stl, char *file)

void **stl_calculate_normal** (float *normal*, *stl_facet* *facet)

void **stl_normalize_vector** (float *v*)

void **stl_calculate_volume** (*stl_file* *stl)

void **stl_repair** (*stl_file* *stl, int *fixall_flag*, int *exact_flag*, int *tolerance_flag*, float *tolerance*, int *increment_flag*, float *increment*, int *nearby_flag*, int *iterations*, int *remove_unconnected_flag*, int *fill_holes_flag*, int *normal_directions_flag*, int *normal_values_flag*, int *reverse_all_flag*, int *verbose_flag*)

void **stl_initialize** (*stl_file* *stl)

void **stl_count_facets** (*stl_file* *stl, char *file)

void **stl_allocate** (*stl_file* *stl)

void **stl_read** (*stl_file* *stl, int *first_facet*, int *first*)

void **stl_facet_stats** (*stl_file* *stl, *stl_facet facet*, int *first*)

void **stl_reallocate** (*stl_file* *stl)

void **stl_add_facet** (*stl_file* *stl, *stl_facet* *new_facet)

void **stl_get_size** (*stl_file* *stl)

void **stl_clear_error** (*stl_file* *stl)

int **stl_get_error** (*stl_file* *stl)

void **stl_exit_on_error** (*stl_file* *stl)

struct stl_vertex
#include <admesh/stl.h> Vertex of a facet, defined by 3D coordinates.

Public Members

float **x**

float **y**

float **z**

struct stl_normal
#include <admesh/stl.h> Normal vector of a facet, defined by 3D coordinates.

Public Members

float **x**

float **y**

float **z**

struct stl_facet
#include <admesh/stl.h> Facet, one triangle of the mesh.

Public Members

stl_normal **normal**
normal vector

stl_vertex **vertex**[3]
3 vertices

stl_extra **extra**
extra data

struct stl_edge
#include <admesh/stl.h> Edge between two vertices.

Public Members

stl_vertex **p1**
start vertex

stl_vertex **p2**
end vertex

int **facet_number**
id of facet this edge belongs to

struct stl_hash_edge
#include <admesh/stl.h>

Public Members

unsigned **key**[6]

int **facet_number**

int **which_edge**

struct *stl_hash_edge* ***next**

struct stl_neighbors
#include <admesh/stl.h>

Public Members

int **neighbor**[3]

char **which_vertex_not**[3]

struct v_indices_struct
#include <admesh/stl.h>

Public Members

int **vertex**[3]

struct stl_stats
#include <admesh/stl.h> Statistics about the STL mesh.

Some of them are populated on *stl_open()* and after some operations, others, such as volume, have to be calculated by appropriate functions.

Public Members

char **header**[81]
header of the STL file

stl_type **type**
type of the STL file

int **number_of_facets**
total number of facets

stl_vertex **max**
maximal dimensions of the mesh

stl_vertex **min**
minimal dimensions of the mesh

stl_vertex **size**
size of the bounding box

float **bounding_diameter**
diameter of the bounding box

float **shortest_edge**
length of the shortest edge

float **volume**
volume of the mesh, has to be calculated by *stl_calculate_volume()*

unsigned **number_of_blocks**
should be number of blocks, but is never set

int **connected_edges**
how many edges have been connected by ADMesh

int **connected_facets_1_edge**
how many facets are connected by at least 1 edge, get's calculated during *stl_check_facets_nearby()*

int **connected_facets_2_edge**
how many facets are connected by at least 2 edges, get's calculated during *stl_check_facets_nearby()*

int **connected_facets_3_edge**
how many facets are connected by all 3 edges, get's calculated during *stl_check_facets_nearby()*

int **facets_w_1_bad_edge**
how many facets have exactly 1 unconnected edge, get's calculated during *stl_repair()*

int **facets_w_2_bad_edge**
how many facets have exactly 2 unconnected edges, get's calculated during *stl_repair()*

int **facets_w_3_bad_edge**
how many facets have exactly 3 unconnected edges, get's calculated during *stl_repair()*

int **original_num_facets**
original number of facets when the file was loaded

int **edges_fixed**
how many edges were fixed by ADMesh

int **degenerate_facets**
number of removed degenerate facets

int **facets_removed**
number of removed degenerate facets

int **facets_added**
number of facets removed by *stl_remove_unconnected_facets()*

int **facets_reversed**
number of facets reversed by *stl_fix_normal_directions()*

int **backwards_edges**
number of edges that are backwards counted during *stl_verify_neighbors()*

int **normals_fixed**
number of normals fixed during *stl_fix_normal_values()*

int **number_of_parts**
number of parts (distinguished shells), calculated during *stl_fix_normal_directions()*

int **malloced**
how many edges have been malloced during *stl_check_facets_nearby()*

int **freed**
how many edges have been freed during *stl_check_facets_nearby()*

int **facets_malloced**
how many facets have been malloced

int **collisions**
internal collision counter for *stl_check_facets_nearby()*

int **shared_vertices**
number of shared vertices, populated by *stl_generate_shared_vertices()*

int **shared_malloced**
how many shared vertices have been malloced by *stl_generate_shared_vertices()*

struct stl_file
#include <*admesh/stl.h*> STL file.

The main structure representing the mesh. All functions take reference to this as a first argument.

Public Members

FILE ***fp**
pointer to associated file

stl_facet ***facet_start**
array of facets

stl_edge ***edge_start**
array of edges (never populated)

stl_hash_edge ****heads**
head of linked list of edges, used internally by some repairs

stl_hash_edge ***tail**
tail of linked list of edges, used internally by some repairs

int **M**
magic variable, used internally by some repairs

stl_neighbors ***neighbors_start**
array of neighbors populated by various repairs

v_indices_struct ***v_indices**
internal array used by *stl_generate_shared_vertices()*

stl_vertex ***v_shared**
vertices array used by *stl_generate_shared_vertices()*

stl_stats **stats**
statistics about the mesh

char **error**
error flag, when something went wrong, this is not 0

1.4 Python bindings for ADMesh

FIXME

CHAPTER 2

Overview

ADMesh is a program for processing triangulated solid meshes. Currently, ADMesh only reads the STL file format that is used for rapid prototyping applications, although it can write STL, VRML, OFF, and DXF files. Additional information regarding the underlying algorithms of ADMesh can be found in Anthony Martin's Masters Thesis.

ADMesh is written in ANSI C, licensed under GPLv2+. This is documentation for version 0.98.1.

2.1 Features

- Read and write binary and ASCII STL files
- Check STL files for flaws (i.e. unconnected facets, bad normals)
- Repair facets by connecting nearby facets that are within a given tolerance
- Fill holes in the mesh by adding facets
- Repair normal directions (i.e. facets should be CCW)
- Repair normal values (i.e. should be perpendicular to facet with length=1)
- Remove degenerate facets (i.e. facets with 2 or more vertices equal)
- Translate in x, y, and z directions
- Rotate about the x, y, and z axes
- Mirror about the xy, yz, and xz planes
- Scale the part by a factor
- Merge 2 STL files into one
- Write an OFF file
- Write a VRML file
- Write a DXF file
- Calculate the volume of a part
- Get other statistics about the mesh

V