NAME

tmesh2rad - convert a triangular mesh to a RADIANCE scene description

SYNOPSIS

tmesh2rad [**−o obj**][**−m mat**][**−p pat**] [**input ..**]

DESCRIPTION

Tmesh2rad converts one or more triangle-mesh files to a RADIANCE scene description. The *−o* option may be used to assign a default object name. The single letter "T" is used if no name is given on the command line or in the file. The *−m* option may be used to assign a default material name. The non-material "void" is used as a default if none is given on the command line or in the file. The *−p* option may be used to assign a default picture for a surface pattern. If none is given on the command line or in the file, the surface will not have an associated pattern.

FILE FORMAT

A triangle-mesh is a free-format ASCII file composed of the following eight primitive types. Each primitive is begun with a single, white-space-delimited letter:

 # Comment

Whatever follows up until the end of line is passed as a comment to the output. Note that there must be at least one space or tab following the pound-sign.

o *name* The white-space-delimited string *name* is used as a prefix for all following output triangles.

m *material* The white-space-delimited string *material* is used as the modifier name for all following output triangles.

p *picture* The white-space-delimited string *picture* is used as the name of the RADIANCE picture file to be used as a pattern for all following output triangles with properly defined vertices. (See *i* primitive below.)

v *id x y z* Defines the vertex *id* with 3-dimensional coordinates *x, y* and *z*. The identifier, *id* must be some small, non-negative integer value. If the same integer is used for a later vertex definition, this definition will be lost, though any triangles using the vertex prior to its redefinition will be unaffected.

n *nx ny nz* Defines a surface normal vector with the 3-dimensional components *nx, ny* and *nz*. This vector will be associated with the most recently defined vertex, and is often placed on the same line as the vertex definition for clarity. The vector need not be normalized.

i *u v* Defines a picture index for the most recently defined vertex. The *u* value will be used to lookup the horizontal pixel coordinate in the currently defined picture. The *v* value will be used to lookup the vertical pixel coordinate. (See the RADIANCE reference manual for details on picture coordinate values.) As with associated surface normals, picture indices are interpolated using barycentric coordinates based on the triangle vertices. If these coordinates are calculated correctly, this should result in a smooth mapping of a pattern onto the surface mesh.

t *id1 id2 id3*

Create a triangle connecting the three vertices identified by *id1, id2* and *id3*. The right-hand rule is used to determine the default surface normal orientation, and this should not be too far from the associated vertex normals (if any). All three vertices must have an associated normal if the triangle is to be smoothed. If a picture file is defined and all three vertices have pattern indices associated with them, then this picture will be used as a pattern to modify the triangle's color.

We realize there are many similar T-mesh file formats in existence, and that it would have been just as easy to support one of these formats directly. The disadvantage to supporting an existing format is that conversion from other formats might prove difficult. It was our hope to provide a "greatest common multiple" format that would support all similar T-mesh formats, rather than supporting WaveFront's .OBJ format (for example) and being unable to associate a pattern with an object. Converting from other formats should be relatively straightforward. In many cases, an *awk(1), rcalc(1)* or even a *sed(1)* script should be sufficient.

EXAMPLE

Here is an example T-mesh file:

```
# Our object name:
o test_object
# Our material:
m puce
# Our vertices:
v 1       10      15      5
v 2       10      –15     5
v 3       0       –15     0
v 4       –10     15      –5
# Two triangles joined together:
t 1 2 3
t 2 3 4
```

Which generates the following output:

```
## T-mesh read from: <stdin>

# Our material:

# Our vertices:

# Two triangles joined together:

puce polygon test_object.1
0
0
9
        10          15          5
        10          –15          5
         0          –15         0

puce polygon test_object.2
0
0
9
        10          –15          5
         0          –15         0
        –10          15          –5
```

Here is another example:

```
# A partial cylinder:
m BluePlastic
v 1 –14.673 –3.119    50 n –0.95677 –0.203374 1.17936e–10
v 2 –12.136 –8.817   –50 n –0.791363 –0.574922 4.84915e–10
v 3 –12.136 –8.817    50 n –0.791363 –0.574922 4.84915e–10
t 1 2 3
m OrangePlastic
v 1 –7.501 –12.991    50 n –0.549094 –0.812427 –1.45812e–09
v 2 –12.136 –8.817    50 n –0.791363 –0.574922 4.84915e–10
v 3 –12.136 –8.817   –50 n –0.791363 –0.574922 4.84915e–10
```

```
t 1 2 3
m BluePlastic
v 1 –1.568 –14.918    50 n –0.171094 –0.965568 –5.69788e–09
v 2 –7.501 –12.991    50 n –0.549094 –0.812427 –1.45812e–09
v 3 –7.501 –12.991   –50 n –0.429001 –0.881759 –3.6502e–09
t 1 2 3
```

Note that the same three vertices were used repeatedly, and intermingled with the triangle definitions.

AUTHOR

Greg Ward

BUGS

Triangle smoothing doesn't work very well for glass or trans material types in Radiance, since textures cause distorted transmission through these materials. It is best to use the dielectric material type if smooth transmission is desired.

SEE ALSO

arch2rad(1), awk(1), ies2rad(1), thf2rad(1), oconv(1), rcalc(1), sed(1), xform(1)