NAME

obj2rad - convert Wavefront .OBJ file to RADIANCE description

SYNOPSIS

obj2rad [**−n**][**−f**][**−m mapfile**][**−o objname**] [**input**]

DESCRIPTION

Obj2rad converts a Wavefront .OBJ file to a RADIANCE scene description. The material names for the surfaces will assigned based on the mapping rules file given in the *−m* option. If no mapping file is given, the identifiers given by the "usemtl" statements will be used as the material names. If no "usemtl" statements are found, the group names (given by the "g" statement) will be used instead. Failing this, the default material "white" will be used.

A mapping file contains a list of materials followed by the conditions a surface must satisfy in order to have that material. For example, if we wanted all faces in the Group "thingy" with texture Map "pine" to use the material "wood", and all other surfaces to use the material "default", we would create the following mapping file:

```
default ;
wood (Group "thingy") (Map "pine") ;
```

All faces would satisfy the first set of conditions (which is empty), but only the faces in the Group "thingy" with texture Map "pine" would satisfy the second set of conditions.

Each rule can have up to one condition per qualifier, and different translators use different qualifiers. In *obj2rad,* the valid qualifiers are *Material, Map, Group, Object* and *Face*. A condition is either a single value for a specific attribute, or an integer range of values. (Integer ranges are specified in brackets and separated by a colon, eg. [−15:27], and are always inclusive.) A semicolon is used to indicate the end of a rule, which can extend over several lines if necessary.

The semantics of the rule are such that "and" is the implied conjunction between conditions. Thus, it makes no sense to have more than one condition in a rule for a given qualifier. If the user wants the same material to be used for surfaces that satisfy different conditions, they simply add more rules. For example, if the user also wanted faces between 50 and 175 in the Group "yohey" to use "wood", they would add the following rule to the end of the example above:

```
wood (Face [50:175]) (Group "yohey") ;
```

Note that the order of conditions in a rule is irrelevant. However, the order of rules is very important, since the last rule satisfied determines which material a surface is assigned.

By convention, the identifier "void" is used to delete unwanted surfaces. A surface is also deleted if it fails to match any rule. Void is used in a rule as any other material, but it has the effect of excluding all matching surfaces from the translator output. For example, the following mapping would delete all surfaces in the Object "junk" except those with the Group name "beige", to which it would assign the material "beige_cloth", and all other surfaces would be "tacky":

```
tacky ;
void (Object "junk") ;
beige_cloth (Object "junk") (Group "beige") ;
```

The *−n* option may be used to produce a list of qualifiers from which to construct a mapping for the given .OBJ file. This is also useful for determining which materials must be defined when no mapping is used.

The *−f* option is used to flatten all faces, effectively ignoring vertex normal information. This is sometimes desirable when a smaller model or more robust rendering is desired, since interpolating vertex normals takes time and is not always reliable.

The *−o* option may be used to specify the name of this object, though it will be overriden by any "o" statements in the input file. If this option is absent, and there are no "o" statements, *obj2rad* will attempt to name surfaces based on their group associations.

If no input files are given, the standard input is read.

DETAILS

The following Wavefront statements are understood and translated by *obj2rad.*

#	A comment. This statement is passed to the output verbatim. It has no effect.
f	A polygonal face. If the vertices have associated surface normals, the face will be broken into quadrilaterals and triangles with the appropriate Radiance textures to interpolate them. Likewise, if the face is non-planar, it will be broken into triangles. Each face in the input file is assigned a number, starting with 1, and this number may be used in the material mapping rules.
g	Group association. The following faces are associated with the named group(s). These may be used in the mapping rules, where a rule is matched if there is an association with the named Group. (I.e. since there may be multiple group associations, any match is considered valid.) If a mapping file is not used and no "usemtl" statement has been encountered, the main group is used for the surface material identifier.
o	Object name. This is used to name the following faces, and may be used in the mapping rules.
usemap	A texture map (i.e. Radiance pattern) name. The name may be used in the material mapping rules, but the indexing of Radiance patterns is not yet supported.
usemtl	A material name. The name may be used in mapping rules, or will be used as the Radiance material identifier if no mapping is given.
v	A vertex, given by its x, y and z coordinates.
vn	A vertex normal, given by its x, y and z direction components. This vector will be normalized by *obj2rad,* and an error will result if it has length zero.
vt	A vertex texture coordinate. Not currently used, but will be if we ever get around to supporting Wavefront textures.

All other statement types will be ignored on the input. A final comment at the end of the Radiance output file will give some indication of how successful the translation was, since it will mention the number of statements *obj2rad* did not recognize.

EXAMPLES

To create a qualifier list for triceratops.obj:

```
obj2rad –n triceratops.obj > triceratops.qual
```

To translate triceratops.obj into a RADIANCE file using the mapping triceratops.map:

```
obj2rad –m triceratops.map triceratops.obj > triceratops.rad
```

FILES

tmesh.cal	- used for triangle normal interpolation
surf.cal	- used for quadrilateral normal interpolation

AUTHOR

Greg Ward

SEE ALSO

arch2rad(1), ies2rad(1), obj2mesh(1), objutil(1) oconv(1), thf2rad(1), xform(1)