Appendix A:

OBJ File Format

An overview of the subset of OBJ commands as used by Poser 3.

Compiled March 1999 by Dave Hill
www.aarrgghh.com/poser
dave@aarrgghh.com

The OBJ file format is a program in text file format that can be opened by any text editor and rewritten line by line if need be. The program language is very simple, consisting of a list of commands and parameters. There are no type declarations, subroutines, logical branching or other such complex programming structures. There is only one command for each line of text*, and the file is parsed by an interpreter—usually some form of 3d file modeller or viewer—line by line, from top to bottom, just as you or I might read it.

The bulk of the file is a sequential list of the coordinates of the 3d object's vertices and facets, which can easily run into many tens of thousands of lines of obtuse code. (I call the code obtuse only because of the difficulty most humans have visualizing a 3d object from a raw list of its vertices.) For this reason manual editing is usually restricted to the grouping and materials commands, which segregate the facets and designate, by name, the colors and/or textures to be used on the facets. These commands are intersperced throughout the section where the facets are defined.

Below is a sample of truncated output from the OBJ Exporter used by Ray Dream Studio 5:

```
# Created using Ray Dream Designer(R) 5

v 0.34316099 0.59390599 -0.040665001
v 0.342087 0.59566301 -0.041138999
v 0.342419 0.596699 -0.040672
...

vn -0.04851243 0.00823834 0.00886972
vn -0.04933765 0.00547195 0.00598770
vn -0.04951001 0.00626423 0.00308482
...

vt 0.01534 0.93879497
vt 0.016168 0.92733002
vt 0.016718 0.190071
...

g chest
usemtl skin
f 10159/6670 10160/6593 10161/6660 10162/6758
f 10160/6593 10159/6670 4574/6598 4576/6521
f 10160/6593 10163/6515 10164/6576 10161/6660
...
```

**Because the OBJ format doesn't specify the end-of-line character, some exporters insert carriage-returns and others use linefeeds. Depending on your platform or interpreter, these characters may need to be converted. (For example, the OBJIMP plugin for 3DS MAX crashes when importing Poser's OBJ files,) On the Macintosh a carriage-return is used at the end of lines; on Windows, a carriage-return and a linefeed; on Unix, a linefeed. Notably, BBEdit by Bare Bones Software allows users to specify end-of-line characters.*

The file contains four different sections of related commands, and this is essentially all there is to most OBJ files. The command sections follow a strict order. Poser 3 uses only a subset of the commands available to the format, which include:

OBJ Command *Parameter(s)*[1]	**Definition**
# *comment*	programmer's comment; ignored by interpreter
v *x y z* **v ...**	vertex command; designates location in 3d space
vn *x y z* **vn ...**	vertex normal command; designates orientation
vt *u v* [*w*] **vt ...**	vertex texture command; designates orientation of textures
g *groupName*	group facets command; designates contiguous facets
usemtl *textureName*	use material command; designates named texture
f *v1*[*/vt1/vn1*] *v2*[*/vt2/vn2*] ... **f ...**	facet command; designates polygon vertices by index number, and optional orientations of textures and normals

[1]*Parameters within square brackets are optional.*

The Comment Command. The first word of each line specifies the type of command, which is followed by its parameters. The first line in most OBJ files is the comment command, designated by a pound sign (#), whose parameters are always ignored by the interpreter. (The interpreter also ignores blank lines.) The first comment usually identifies the program that output the file, as in the Ray Dream sample above:

```
# Created using Ray Dream Designer(R) 5
```

The opening comment(s) may also contain information about the number of verticies and/or facets in the object, or the object's creator. They may be inserted anywhere in the file.

The Vertex Command. Because the comment line is ignored, the OBJ file really begins with the first vertex command, designated by the command **v**, whose parameters are the three coordinates of the point. Each parameter is a floating point value between 0 and 1.

```
v x y z
```

The vertex command section is composed of the list of vertex commands identifying every point in the object. Note that these commands specify only locations in 3d space, not geometry.

Each vertex is implicitly indexed by its order in the command list. The first vertex is indexed as '1', the second as '2', the third as '3', and so forth. This indexing is utilized by some of the other commands to follow.

The Normal Command. The next section is composed of the vertex normal commands, designated by the command **vn**, whose parameters specify the direction in which each vertex faces. Each parameter is a floating point value between 0 and 1.

```
vn x y z
```

Like each vertex, each normal is implicitly indexed by its order in the command list. But often the normal commands are omitted because the facet command will instead use the indexing of the 'v' commands to determine the direction of the normal.

The normal command really doesn't tell you anything by itself, but must be associated with a vertex in a facet command, described below.

The Texture Command. The next section is composed of the texture orientation commands, designated by the command **vt**, whose parameters specify the UV (and sometimes W) mapping values. This command tells the interpreter how to orient textures applied to the vertex. Each parameter is a floating point value between 0 and 1.

```
vt u v [w]
```

Like each vertex, each texture is implicitly indexed by its order in the command list, and like each normal,

the texture command doesn't tell you anything by itself and must be associated with a vertex in a facet command.

The Facet Command. The final section is composed of the facet commands, designated by the command `f`, which specifies a polygon (facet) made from the verticies listed in the parameters. Each parameter, an integer, references the vertex command index number of the vertex. Forward slashes (/) may be used to associate the vertex index number with the index numbers of normals and/or textures defined earlier in their respective sections, in strict order.

f *v1*[*/vt1*][*/vn2*] *v2*[*/vt2*][*/vn2*] *v3*[*/vt3*][*/vn3*] **...**

For example,

f 1 2 3 4

describes a facet built from verticies created with vertex commands 1 through 4. To associate a normal and/or a texture, concatenate the index numbers with a slash. For example,

f 1/1 2/2 3/3 4/4

describes a facet built from verticies created with vertex commands 1 through 4, and associates each vertex with texture commands 1 through 4.

If you associate a normal index and/or a texture index for one vertex in a facet command, you must associate one for all. Because of the strict order of the associations, a placeholder slash for the texture must be inserted between the vertex and the normal when associating them without a texture. For example,

f 1//1 2//2 3//3 4//4

describes a facet built from verticies created with vertex commands 1 through 4, and associates each vertex with normal commands 1 through 4.

As mentioned earlier, the direction a polygon faces is determined by the order of the vertex commands if no normal commands are issued. The order assumes a counter-clockwise direction. By reversing the order of the parameters of the facet command, you can 'flip' the facet in the opposite direction. (When a normal points the wrong way, 'holes' may appear in the object if the interpreting program renders polygons as single-sided. With programs that render polygons as double-sided, such as the lastest version of Poser (3.1), the 'holes' and the problem disappear.)

The Group and Material Commands. These two commands may be considered 'sub-commands' of the facet command because they only affect facet commands and therefore only appear in the facet command section. The group command, designated by the command `g`, specifies the facets that follow as a contiguous group. Its single parameter is a text string, the name of the group.

g *groupName*

The group command breaks up the model into its 'parts', and when group commands are not used or are removed from an OBJ file, the model becomes a single polymesh.

The material command, designated by the lengthy command `usemtl` , specifies the color or texture used by facets that follow it. Its single parameter is also a text string, the name of the material, which is not defined as part of the OBJ file.

usemtl *materialName*

The material command normally follows the group command, as in our Ray Dream sample, since it makes sense to organize the parts of an object by the materials intended to be applied to it. A list of facets falling under a single group command may actually encompass more than one material command—and a different material command may be issued for each facet command in the group!

If no material command is issued, the interpreter will apply to the entire model whatever default texture it is programmed to render.

Both the group command and the material command remain in effect to the end of the file unless respective new commands are encountered.

■

Other documents and resources in this series include:

Primer on New Models

Preparing a Base Model

Building Bat-Guy

Appendix A: OBJ File Format

Appendix B: Poser Resource File Format

Appendix C: Joint Parameters Tables

Appendix D: Spherical Falloff Zones

Base Model Geometry and Resource Files

Bat-Guy Geometry and Resource Files

Final Touches: Adding the Cape

Cape Geometry & Morph Targets