Wavefront and Java3D *.obj* Format

The Wavefront .obj file format is a standard 3D object file format created for use with Wavefront's Advanced Visualizer™ and available for purchase from Viewpoint DataLabs, as well as other 3D model companies. Object Files are text based files supporting both polygonal and free-form geometry (curves and surfaces). The Java 3D .obj file loader supports a subset of the file format, but it is enough to load almost all commonly available Object Files. Free-form geometry is not supported.
The following text is close to the description of the *.obj* file format in the Sun Java3D documentation.
The Object File tokens currently supported by the JavaView loader are listed below. Unknown tokens are skipped without affecting the reading process.

some text

Line is a comment until the end of the line

v *float float float*

A single vertex's geometric position in space. The first vertex listed in the file has index 1, and subsequent vertices are numbered sequentially.

vn *float float float*

A normal. The first normal in the file is index 1, and subsequent normals are numbered sequentially.

vt *float float*

A texture coordinate. The first texture coordinate in the file is index 1, and subsequent textures are numbered sequentially.

f *int int int* ...

or

f *int/int int/int int/int* . . .

or

f *int/int/int int/int/int int/int/int* ...

A polygonal face. The numbers are indexes into the arrays of vertex positions, texture coordinates, and normals respectively. A number may be omitted if, for example, texture coordinates are not being defined in the model.
There is no maximum number of vertices that a single polygon may contain. The .obj file specification says that each face must be flat and convex. In JavaView polygonal face may be triangulated.

OBJ File Format

Poser 3 only uses a subset of the full OBJ file format. Here's everything you need to know about OBJ files if your just using Poser. This document leaves out all non-Poser usable commands and options so you aren't confused about things you'll never need. I recommend reading this tutorial, then opening up a couple OBJ files and checking them out, then rereading this tutorial, and things should start to fall in place.

The OBJ file format is a text file format, which means you can edit OBJ files in a text editor if you are hard-core. Unfortunately, the original specification didn't seem to state what the end of line character should be, so some tools use carriage-returns and some use linefeeds. You may have to convert the end of line characters depending on which tools you are reading the OBJ file in from. (Most notably the OBJIMP plugin for 3DS MAX will crash when reading in any of the OBJ files that shipped with Poser.) Also your Windows text editors may think the files are binary because of this.

The first character of each line specifies the type of command. If the first character is a pound sign, #, the line is a comment and the rest of the line is ignored. Any blank lines are also ignored. The file is read in by a tool and parsed from top to bottom just like you would read it. In the descriptions that follow, the first character is a command, followed by any arguments. Anything shown in square brackets is optional.

 # a comment line

These are always ignored. Usually the first line of every OBJ file will be a comment that says what program wrote the file out. Also, its quite common for comments to contain the number of verticies and/or faces an object used.

v x y z

The vertex command, this specifies a vertex by its three coordinates. The vertex is implicitly named by the order it is found in the file. For example, the first vertex in the file is referenced as '1', the second as '2' and so on. None of the vertex commands actually specify any geometry, they are just points in space.

vt u v [w]

The vertex texture command specifies the UV (and optionally W) mapping. These will be floating point values between 0 and 1 which say how to map the texture. They really don't tell you anything by themselves, they must be grouped with a vertex in a 'f' face command.

vn x y z

The vertex normal command specifies a normal vector. A lot of times these aren't used, because the 'f' face command will use the order the 'v' commands are given

to determine the normal instead. Like the 'vt' commands, they don't mean anything until grouped with a vertex in the 'f' face command.

f v1[/vt1][/vn1] v2[/vt2][/vn2] v3[/vt3][/vn3] ...

The face command specifies a polygon made from the verticies listed. You may have as many verticies as you like. To reference a vertex you just give its index in the file, for example 'f 54 55 56 57' means a face built from vertecies 54 - 57. For each vertex, there may also be an associated vt, which says how to map the texture at this point, and/or a vn, which specifies a normal at this point. If you specify a vt or vn for one vertex, you must specify one for all. If you want to have a vertex and a vertex normal, but no vertex texture, it will look like: 'f v1//vt1'. The normal is what tells it which way the polygon faces. If you don't give one, it is determined by the order the verticies are given. They are assumed to be in counter-clockwise direction. If you aren't using vn's to specify the normal and you wanted to 'flip the normal' you would reverse the order of the verticies. In most 3D programs, if the normal points the wrong way, there will appear to be a hole in the object. Luckily, Poser 3 renders both sides of a polygon, so even if you are editing something in another program that looks like it has holes, they will effectively go away inside Poser (NOTE: I read that somewhere, but haven't personally confirmed that fact.) One more thing, if you ever see a negative v, vt, or vn, that is a relative offset. It means go back that many verticies from where you are now in the file to find the vertex. This is part of the OBJ file spec, but I haven't seen any Poser OBJs that use it.

g name

The group name command specifies a sub-object grouping. All 'f' face commands that follow are considered to be in the same group.

usemtl name

The use material command lets you name a material to use. All 'f' face commands that follow will use the same material, until another usemtl command is encountered. For all of the Poser OBJ files I've seen, all 'g' commands should be followed by a 'usemtl' command.

Remember that for verticies, they can be interspersed throughout the file, only the order they appear makes a difference. The faces can also be spread throughout the file, except they must follow the verticies they use (I think), and they will be part of whichever group and/or material they follow. That said, most OBJ files follow a consistant layout. Now the 'normal' layout of the file will be:

 # comment about what application generated this file.
 # all of the 'v' commands will be listed
v x y z
v ...
 # all of the 'vn' commands will be listed, although most Poser OBJ files
 # do not use the 'vn' commands
vn x y z

vn ...
 # all of the 'vt' commands will be listed
vt x y z
vt ...
 # the object and its material will be set
g object
usemtl material
 # all of the 'f' commands are listed
f 1/1 2/2 3/3 4/4
f ....

Additional Items:

If you had two OBJ files and wanted to merge them, you can cut all the 'v', 'vt', and 'vn' commands from the second file and paste them at the end of the first. However, you cannot just copy over the 'f' commands, because they will have to have all their verticies offset. I'm thinking there has to be a tool out there somewhere that will do this for you, but I don't know of any.

Hair objects with separate pieces like the ponytails that have a hairtie that can be colored separately, are actually a single group. They use additional usemtl commands to perform their magic. You can setup the additional material parameters in the .hr2 file (MAC). Example OBJ:

v x y z
v ...
g hair
usemtl hair
f 1 2 3 4
f ...
 # there is NOT another g command here
usemtl hairTie
f 10 11 12 13
f ...

s groupNumber

Poser 2 hair objects sometimes use the 's' command to set a smoothing group. The groupNumber is used to make separate groups. All subsequent 'f' commands are in the same smoothing group until another 's' command is encountered. These also have some kind of interaction with vertex normals, but I haven't explored it fully because apparently as of Poser 3, it doesn't use the 's' command anymore.