Alias/WaveFront Object (.obj) File Format

The following documentation of the .obj file format is from: http://www.fileformat.info/format/wavefrontobj/

In Wavefront's 3D software, geometric object files may be stored in ASCII format (using the ".obj" file extension) or in binary format (using the .MOD extension). The binary format is proprietary and undocumented, so only the ASCII format is described here.

The OBJ file format supports lines, polygons, and free-form curves and surfaces. Lines and polygons are described in terms of their points, while curves and surfaces are defined with control points and other information depending on the type of curve. The format supports rational and non-rational curves, including those based on Bezier, B-spline, Cardinal (Catmull-Rom splines), and Taylor equations.

File Organization

OBJ files do not require any sort of header, although it is common to begin the file with a comment line of some kind. Comment lines begin with a hash mark (#). Blank space and blank lines can be freely added to the file to aid in formatting and readability. Each non-blank line begins with a keyword and may be followed on the same line with the data for that keyword. Lines are read and processed until the end of the file. Lines can be logically joined with the line continuation character (\ ) at the end of a line.

The following keywords may be included in an OBJ file. In this list, keywords are arranged by data type, and each is followed by a brief description and the format of a line.

Vertex data:

`v`	Geometric vertices:	`v` *`x y z`*
`vt`	Texture vertices:	`vt` *`u v`*
`vn`	Vertex normals:	`vn` *`dx dy dz`*

Elements:

`p`	Point:	`p` v_1
`l`	Line:	`l` v_1 v_2 ... v_n
`f`	Face:	`f` v_1 v_2 ... v_n

Grouping:

`g`	Group name:	`g` *`groupname`*

Display/render attributes:

`usemtl`	Material name:	`usemtl` *`materialname`*
`mtllib`	Material library:	`mtllib` *`materiallibname`*`.mtl`

File Details

The most commonly encountered OBJ files contain only polygonal faces. To describe a polygon, the file first describes each point with the "v" keyword, then describes the face with the "f" keyword. The line of a face command contains the enumerations of the points in the face, as 1-based indices into the list of points, in the order they occurred in the file. For example, the following describes a simple triangle:

```
# Simple Wavefront file
v 0.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 0.0 0.0
f 1 2 3
```

It is also possible to reference points using negative indices, where the indices are specified relative to the current maximum vertex position (-1 references the last vertex defined). This makes it easy to describe the points in a face, then the face, without the need to store a large list of points and their indexes. In this way, "v" commands and "f" commands can be interspersed.

```
v -0.500000 0.000000 0.400000
v -0.500000 0.000000 -0.800000
v -0.500000 1.000000 -0.800000
v -0.500000 1.000000 0.400000
f -4 -3 -2 -1
```

OBJ files do not contain color definitions for faces, although they can reference materials that are stored in a separate material library file. The material library can be loaded using the "mtllib" keyword. The material library contains the definitions for the RGB values for the material's diffuse, ambient, and specular colors, along with other characteristics such as specularity, refraction, transparency, etc.

The OBJ file references materials by name with the "usemtl" keyword. All faces that follow are given the attributes of this material until the next "usemtl" command is encountered.

Faces and surfaces can be assigned into named groups with the "g" keyword. This is used to create convenient sub-objects to make it easier to edit and animate 3D models. Faces can belong to more than one group.

The following demonstrate two examples of material assignment, and grouping.

Cube with Materials

```
# This cube has a different material
# applied to each of its faces.
    mtllib master.mtl
    v    0.000000    2.000000    2.000000
    v    0.000000    0.000000    2.000000
    v    2.000000    0.000000    2.000000
    v    2.000000    2.000000    2.000000
    v    0.000000    2.000000    0.000000
    v    0.000000    0.000000    0.000000
    v    2.000000    0.000000    0.000000
    v    2.000000    2.000000    0.000000
    # 8 vertices
    g front
    usemtl red
    f 1 2 3 4
    g back
    usemtl blue
    f 8 7 6 5
    g right
    usemtl green
    f 4 3 7 8
    g top
    usemtl gold
    f 5 1 4 8
    g left
    usemtl orange
    f 5 6 2 1
    g bottom
    usemtl purple
    f 2 6 7 3
    # 6 elements
```

Texture-Mapped Square

```
# A 2 x 2 square mapped with a 1 x 1 square
# texture stretched to fit the square exactly.
mtllib master.mtl
v  0.000000 2.000000 0.000000
v  0.000000 0.000000 0.000000
v  2.000000 0.000000 0.000000
v  2.000000 2.000000 0.000000
vt 0.000000 1.000000 0.000000
vt 0.000000 0.000000 0.000000
vt 1.000000 0.000000 0.000000
vt 1.000000 1.000000 0.000000
# 4 vertices
usemtl wood
# The first number is the point,
# then the slash,
```

```
# and the second is the texture point
f 1/1 2/2 3/3 4/4
# 1 element
```