Polygon Model File Formats

There are two common file formats used for the storage of polygon models, at least for use in university courses and computer graphics research. There are certainly many other file formats for storing models that are tied to either open source of commercial modeling software. Other file formats may store other kinds of data, such as materials, shaders, textures, animations, or other kinds of data. For this class we are concerned primarily with the geometry and then we'll do computations over the provided geometry.

OBJ File Format

PLY File Format

References:

- https://en.wikipedia.org/wiki/PLY_(file_format)
- http://paulbourke.net/dataformats/ply/

Polygon File Format, also known as the Stanford Triangle Format. The file can be either ASCII or binary format. I will only describe the ASCII formatting for this class.

This is a file format developed by Greg Turk while working in the Stanford graphics lab. Its development was inspired from the Wavefront obj file format, while also wanting to include extensibility (which obj lacks).

There are two primary sections to the file, a header, and then the elements. The header contains a description of which elements to expect in the second part of the file. The elements are the details of the model, the vertices, normals, indices, etc.

Header

- Line 1: The magic number, `ply`
- Line 2: `format ascii 1.0`
  - can be other formats, but we are only looking at the ASCII format
- A comment can be placed anywhere in the header by starting the line with the word `comment`
- The `element` keyword is used to being a section that describes the properties to be associated with each element (vertex). Begin the section with the keywords `element vertex`, followed by how many vertices are in the elements section: `element vertex 4`
- The next lines are the properties for each vertex. For example…
  - `property float x`
  - `property float y`
  - `property float z`
  - `property float u`
  - `property float v`
- The property section is closed when the `element face` line is encountered. This line indicates how many polygon faces are expected in the elements section. For example: `property list uint8 int32 vertex_indices` The uint8 indicates the data type for the first value of the vertex indices lines (in the elements section). The int32 indicates the data type for the vertex index itself, in the vertex indices lines. It'll make more sense when you see the elements section.

- The header section is terminated with `end_header`

The following is an example header section:

```
ply
format ascii 1.0
element vertex 4
property float x
property float y
property float z
element face 2
property list uint8 int32 vertex_indices
end_header
```

Elements

Following the header section, the elements comes in two parts. The first part is the list of vertices with each of the properties indicated in the header. The second part is the list of faces, also indicated in the header.

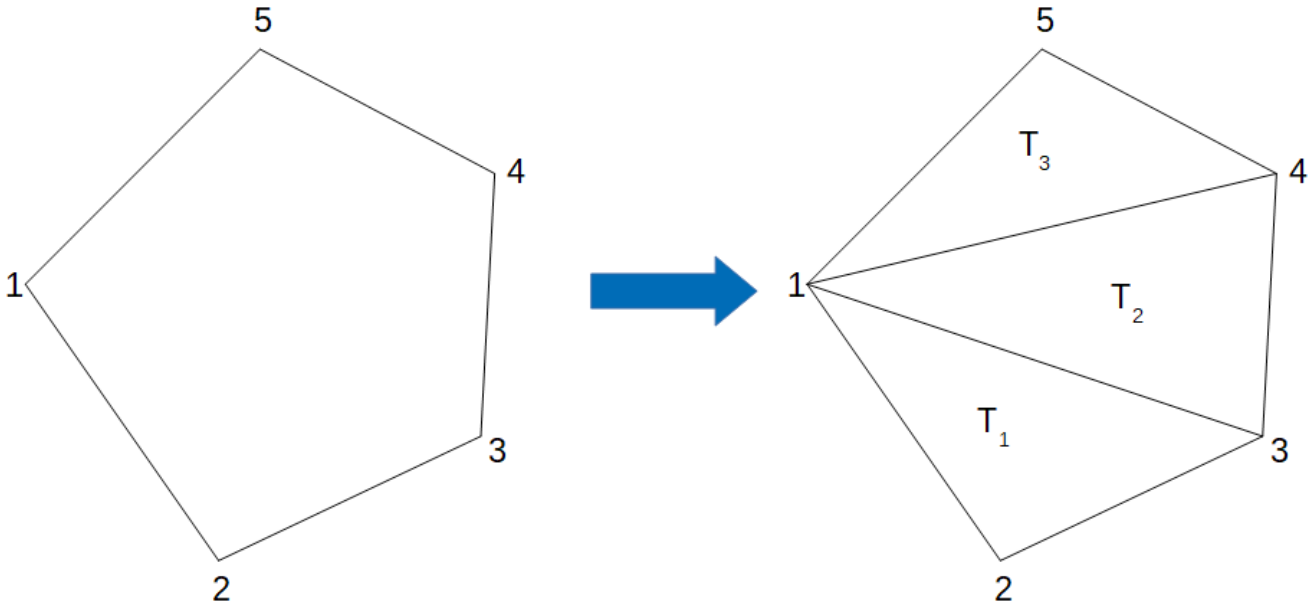
Vertex List

Each line in the vertex list has a column for the properties identified in the header. For the example header shown above, a vertex line might be: `-0.5 0.5 0.5` Where the values are the x, y, and z positions of the vertex. It is that simple.

Index List

Each line in the index list is the description of a polygon face. The first column is the number of vertices in the polygon. The remaining columns are the 0-based indices into the list of vertices. It is that simple.

Because our programs process triangles, rather than generalized polygons, when a face is encountered with more than 3 vertex indices, it must be split into triangles. This isn't too bad. Here is what it looks like.



The numbers at the vertices are the indices into the vertex list. With that, the three triangles formed from this face are:

- T_1 = { 1, 2, 3 }
- T_2 = { 1, 3, 4 }
- T_3 = { 1, 4, 5 }

For any face with N vertices, there will be N – 2 triangles, following the pattern of this example.