Polygon Lists & 3-D File Formats

Glenn G. Chappell

CHAPPELLG@member.ams.org

U. of Alaska Fairbanks

CS 481/681 Lecture Notes

Monday, February 18, 2002

Review: Outlining Polygons

- Sometimes we wish to emphasize the angular, gem-like character of surfaces.
- One way to do this is to outline each polygon.
- How is this done?
  - In OpenGL, we can draw a surface twice, once with polygons filled, and again with polygons outlined.
  - Use **`glPolygonMode`** to set the polygon rasterization mode.
- How can we avoid stitching errors?
  - There is no perfect solution to this problem.
  - Using a polygon offset generally produces good results.
  - The relevant OpenGL command is **`glPolygonOffset`**.

Review: Describing Objects [1/2]

- We usually describe 3-D objects by describing their surfaces.
  - Usually, but not always; true 3-D descriptions are known as *volume data*.
- Object descriptions that are transmitted from one program to another are usually in files.
  - We want files to be compact, but also contain all necessary information.
  - We may want our file to be readable by commercial graphics programs.
  - Some file formats are much easier to handle than others.
- Some object description methods allow us to draw the object in detail, as smooth as we like; others do not.

Review: Describing Objects [2/2]

- Some general categories of ways to describe surfaces:
  - Via an explicit mathematical formula
  - List of polygons
  - Building out of basic shapes
    - Sphere, cylinders, cones, cubes, etc.
  - Splines
  - Isosurface (implicit description)
  - Elevation data
    - Used for terrain

Review: Surface from a Formula

- Mathematical formulas describe surfaces *implicitly* or *explicitly*.
  - An implicit description: $7x\sin(y+3z)=14\cos(xyz)$.
  - An explicit description: $x=a+3b;\ y=\cos a+\sin(a+b);\ z=ab^2.$
- Explicit descriptions have explicit formulae for *x*, *y*, and *z*.
- Explicit descriptions are also called *parametrizations*.
  - Above, variables *a* and *b* are the *parameters*.
- Explicit descriptions are generally **much easier** to handle graphically than implicit descriptions.
- To draw a parametrized surface, we need:
  - Code to draw a square (0 to 1 in both coordinates) as a fine mesh of triangles.
  - Code to compute the coordinates of points in the parametrized surface, as well as the associated normal vectors.
- Then the points and normals in the square are replaced with those for the surface.

Polygon Lists

- Regardless of how a surface is initially described, to draw it we generally convert it to a list of polygons.
  - Exception: Ray tracing.
- Thus, it is common for surfaces to be described as lists of polygons.
- Pros:
  - Easy to draw.
  - Portable; lots of programs can handle lists of polygons.
- Cons:
  - It is impossible to draw a surface in greater detail when you only know the polygons.
  - Files tend to be large.
- Surface descriptions are often placed in files. Next we discuss 3-D file formats.

File Formats: General Issues

- When designing *any* file format, the following issues need to be dealt with.
  - How will files of this type be identified?
  - Will other people/programs be using this format?
  - Might this format be extended in the future?
    - If so, will programs need to be able to read files that contain features designed after the programs were written?
  - Should the format be readable/editable using a standard text editor?
  - Is file size going to be a major issue?
  - Is there an already existing type that is good enough?

File Formats: 3-D Surface Issues [1/3]

- There are two basic ways to specify a list of polygons.
  - The file can give a list of vertices; then polygons are specified as having vertices from this list.
  - The file can simply list the polygons, with the vertex coordinates being listed with the polygons.
- An advantage of the former approach is that a program can tell how the polygons in the surface fit together, whether there are any holes in the surface, etc.
  - We say the program can determine the *topology* of the surface.
- Some surfaces are specified internally as lists of unrelated polygons. In this case, using the former approach may be inconvenient.
- A good solution is to allow both methods.

File Formats: 3-D Surface Issues [2/3]

- When specifying a surface, there is always the question of how to deal with normal vectors. There are three approaches:
  - We can specify a surface without normals.
  - We can specify a normal vector for each vertex (“vertex normals”).
  - We can specify a normal for each polygon (“facet normals”).
- Note: Facet normals are easy to compute from the list of vertices of a polygon. Good vertex normals may be harder to compute.
- A good solution is to allow (but not require) vertex normals.

File Formats: 3-D Surface Issues [3/3]

- A third issue is what sort of polygons to allow:
  - The format may allow general polygons, with any number of vertices.
  - The format may allow only triangles.
- Allowing only triangles can make reading and processing a file simpler.
- However, allowing more general polygons makes file generation easier; further, it is not hard to split up a polygon into triangles, especially if it is required than all polygons be convex.
- Allowing arbitrary (convex?) polygons is generally the best solution, since:
  - Many surfaces are naturally described using quadrilaterals or other non-triangle polygons.
  - Allowing general polygons simply means that a program may need to partition polygons into triangles before processing them; this adds little overhead.

A/W .obj Format: Introduction

- As an example of a 3-D graphical format, we will discuss the Alias/Wavefront .obj format.
- This format was developed by (you guessed it) Alias/ Wavefront for its *Advanced Visualizer*.
  - The format can also be read and written by Alias/Wavefront's *Maya*, the primary software used in ART 472 (Visualization and Animation), as well as many other professional 3-D graphics packages.
- A/W .obj files are identified by their suffix, which is (again, you guessed it) "**`.obj`**".
- The files are human-readable text.
  - As with many 3-D graphical formats, there is an associated binary format, which we will not be discussing.

A/W .obj Format: Basic Structure

- A/W .obj files are composed of lines of text.
  - Blank lines are ignored.
  - Other lines begin with a code telling what sort of data is on that line.
- There are many codes; we will discuss the following:

#	comment (line is skipped)
v	vertex coordinates
vn	vertex normal vector
f	face

- A few short sample files are on the next few slides; more will be on the web page.

A/W .obj Format: Example Files [1/3]

- Here is a complete A/W .obj file. *Italic* comments are not part of the file.

```
# This file contains a single square.
# There are no normals.
# Here are the vertices:
v 0 0 0                 This is vertex number 1.
v 1 0 0
v 1 1 0
v 0 1 0
# Here is the square itself:
f 1 2 3 4               These reference the "v" list, above.
```

A/W .obj Format: Example Files [2/3]

- Here is another .obj file. This one contains normals.
  - Before the two slashes is the vertex (“**v**”) number.
  - After the two slashes is the normal (“**vn**”) number.

```
v 0 0 0
v 1 0 0
v 1 1 0
v 0 1 0
vn 0 0 1           This is normal vector number 1.
# Two triangles
f 1//1 2//1 4//1   These reference vn 1, above.
f 2//1 3//1 4//1
```

A/W .obj Format: Example Files [3/3]

- Yet another .obj file. This has separate vertex and normal lists for each face.

```
v 0 0 0
v 1 0 0
v 0 1 0
vn 0 0 1
f -3//-1 -2//-1 -3//-1
v 1 0 0
v 1 1 0
v 0 1 0
vn 0 0 1
f -3//-1 -2//-1 -3//-1
```

-1 = most recent

-2 is the one before that, etc.