Lecture 18 – 3D Modeling & Polygons

Michael Gleicher
November 8, 2005
Used as notes, not projected as lecture

Modeling 3D Shapes

- Modeling = process of describing an object
  - Representation
- Can model shape, physical properties, behvaior, …
- Many uses of (geometric) models
  - Graphics – make a picture
  - CAD – represent for manufacture

Types of Shape Models in 3D

- Points
- Curves
- Surfaces and Solids
- Volumes

Surface vs. Volume

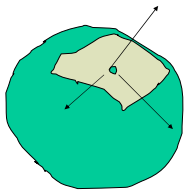
- Cube
  - Volume = space inside 0 <= x,y,z < 1
  - Surface = 6 squares (0,0,0)(0,0,1)(0,1,1)…
- Surface can be a boundary
  - But might not be
- Graphics often only need surfaces

When might we care about Volumes?

- Engineering / Manufacturing / Design
  - Can't be non-physical
- Some kinds of data has "insides"
  - Medical data (scanned)
- Some operations make sense
  - Constructive solid geometry
  - Cut / Join / Subtract / Union
    - Makes less sense on surfaces

Surface Basics

- Locally flat
- At any point
  - Normal
  - Tangent Plane
  - Tangent vectors in plane

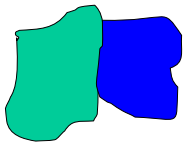


Surfaces

- Generally what we use in graphics
  - Hard enough!
- Similar issues to curves, but worse
- Named vs. Free-Form
- Build out of little pieces
- Linear pieces (polygons) – analogy to lines

Basic Strategy

- Break complicated surfaces into pieces
- Need to choose good pieces
- Need to make sure that the pieces connect
- Connections are more complicated



Polygons

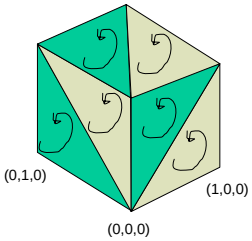
- Or triangles
- Need to have a front/back
- Outward facing normal
- Be consistent in orientation (e.g. CCW)

Polygon Soup

- Random Assortment
- Unstructured
  - At least get ordering right
- Tells little about how polygons connect
- Lots of redundancy

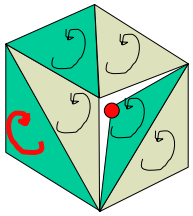
Cube Soup

```
struct Triangle Cube[12] =
    {{{1,1,1},{1,0,0},{1,1,0}},
     {{1,1,1},{1,0,1},{1,0,0}},
     {{0,1,1},{1,1,1},{0,1,0}},
     {{1,1,1},{1,1,0},{0,1,0}},
     ...
    };
```



Polygon Soup

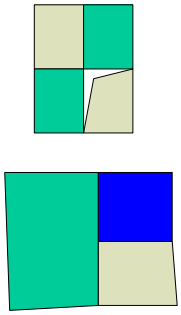
- Advantages
  - Easy
- Problems
  - Redundancy
  - No global info
  - No open/closed info
  - Hard to edit
  - Hard to prevent degeneracies



Cracks / Cracking

- Gaps in the surface
- Prevents from being solid
- Can be ugly
- Airtight / Watertight
  - No cracks

- Beware edge/vertex
  - Numerical errors cause cracks



Mesh

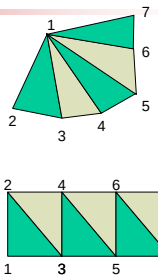
- Share vertices
  - Indirection to vertex table
  - Prevents cracking
  - More efficient (lots of info at vertex)

- Store Polygons as vertex lists
- Store Edges – Faces are lists of edges
  - Every edge borders 2 faces

Regular Meshes

- Often have meshes with uniform patterns
- Grids, fans, strips

- Connectivity is implicit
- Very efficient
- Processing is easy
- Avoid redundant transforms



Vertex Arrays

- Hardware caches vertices (after transform)

- Give vertex list and connectivity
- Do in an order to get cache performance
  - Groups of n vertices
- Hardware specific trick

- Best way to draw triangles in opengl

Normals

- Per Face
  - Can be computed (assume polygon, order)
- Per Vertex
  - Assumes we're approximated smooth surface

- Per Face/Vertex
  - If you want discontinuous normals

Smooth Surfaces

- Approximate with polygons

- Consider Cylinder
  - Number of faces
  - Better looking with smooth shading
  - Even better with Phong Shading
- Tradeoff: more polygons = smoother