

Lecture 27 – Meshes, Parametric Surfaces

CS 559 Lecture Notes
Not for display
November 2007

Polygons

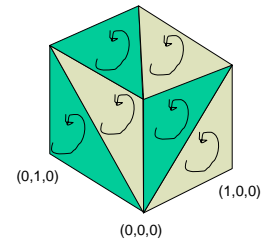
- Or triangles
- Need to have a front/back
- Outward facing normal
- Be consistent in orientation (e.g. CCW)

Polygon Soup

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

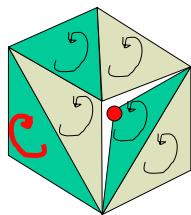
Cube Soup

```
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}},
   ...
};
```



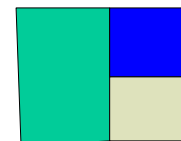
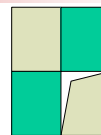
Polygon Soup

- Advantages
 - Easy
- Problems
 - Redundancy
 - No global info
 - No open/closed info
 - Hard to edit
 - Hard to prevent degeneracies
 - No non-local information
 - Is it closed?
 - Is it connected?
 - Is this an edge or internal?



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
- Simplicial Complex
 - Mathematically deep term
 - Fancy way to say “nice mesh” – all faces meet at an edge, ...

Vertex Indirection

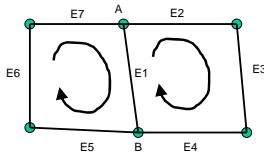


- List of vertices
- Everything is an index into this table
- Good points:
 - Sharing prevents some cracking
 - Transform/Light each vertex once
 - Data reduction

More complex Mesh Structures



- Store Edges
 - Can be handy to have
- Each edge only 2 faces – one CW one CCW (pass through edge in opposite ways)
 - Store “next” edge for each direction
 - Winged Edge Data Structure



E1: A->B

Forw: next=E5,
prev=E7

Back: next=E2,
prev=E4

Getting Meshes to Hardware Fast



- Minimize number of vertices sent down pipe
 - Old days – definitely bottleneck
 - Now – maybe not, since lots of per-pixel computation
- Vertex Buffers
 - Send small number of vertices
 - Index into this small array (since memory << model size)
 - Group into small sets (like 8 or 16) of vertices, draw all triangles between them
- Vertex Cache
 - Automatically buffer, use LIFO

Vertex Arrays

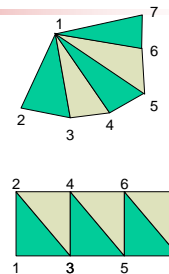


- Hardware caches vertices (after transform)
- Give vertex list and connectivity
- Do in an order to get cache performance
 - Groups of n vertices
- Hardware specific trick
- Best way to draw triangles in opengl
- Send blocks of data at once (avoid function call overhead)
 - Can be high since function call means call to low-level driver
- Possibly: store array in fast memory specific for graphics
 - On graphics card or in driver address space
- Issues with data formats

Regular Meshes



- Reduce number of vertices needed
- Reduce amount of connectivity info needed (which can be sizable!)
- Often have meshes with uniform patterns
- Grids, fans, strips
- Connectivity is implicit
- Very efficient
- Processing is easy
- Avoid redundant transforms



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
- Better: use pieces that are really curved
 - Will (almost always) draw them by tessellating
 - But – easier to model with fewer pieces, more accurate, adaptive, ...