Subdivision Surfaces

Tom Funkhouser
Princeton University
COS 426, Spring 2006

3D Object Representations

• Raw data
  □ Voxels
  □ Point cloud
  □ Range image
  □ Polygons

• Surfaces
  □ Mesh
  □ Subdivision
  □ Parametric
  □ Implicit

• Solids
  □ Octree
  □ BSP tree
  □ CSG
  □ Sweep

• High-level structures
  □ Scene graph
  □ Application specific

Modeling

• How do we ...
  □ Represent 3D objects in a computer?
  □ Acquire computer representations of 3D objects?
  □ Manipulate computer representations of 3D objects?
  □ Analyze computer representations of 3D objects?

Different representations for different types of objects and operations

Surfaces

• What makes a good surface representation?
  □ Accurate
  □ Concise
  □ Intuitive specification
  □ Local support
  □ Affine invariant
  □ Arbitrary topology
  □ Guaranteed continuity
  □ Natural parameterization
  □ Efficient display
  □ Efficient intersections

Polygon Meshes

• How should we represent a mesh in a computer?
  □ Efficient traversal of topology
  □ Efficient use of memory

• Mesh Representations
  □ Independent faces
  □ Vertex and face tables
  □ Adjacency lists
  □ Winged-Edge
**Polygon Meshes**

- How should we represent a mesh in a computer?
  - Efficient traversal of topology
  - Efficient use of memory

  - Mesh Representations
    - Independent faces
    - Vertex and face tables
    - Adjacency lists
    - Winged-Edge

**Independent Faces**

- Each face lists vertex coordinates

**Vertex and Face Tables**

- Each face lists vertex references
  - Shared vertices
  - Still no topology information

**Adjacency Lists**

- Store all vertex, edge, and face adjacencies
  - Efficient topology traversal
**Adjacency Lists**
- Store all vertex, edge, and face adjacencies
  - Efficient topology traversal
  - Extra storage

**Partial Adjacency Lists**
- Can we store only some adjacency relationships and derive others?

**Winged Edge**
- Adjacency encoded in edges
  - All adjacencies in O(1) time
  - Little extra storage (fixed records)
  - Arbitrary polygons

**Winged Edge**
- Example:

**3D Object Representations**
- Raw data
  - Voxels
  - Point cloud
  - Range image
  - Polygons
- Surfaces
  - Mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Octree
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

**Gerry’s Game**
Surfaces

- What makes a good surface representation?
  - Accurate
  - Concise
  - Intuitive specification
  - Local support
  - Affine invariant
  - Arbitrary topology
  - Guaranteed continuity
  - Natural parameterization
  - Efficient display
  - Efficient intersections

Subdivision

- How do you make a smooth curve?

Subdivision Surfaces

- Coarse mesh & subdivision rule
  - Define smooth surface as limit of sequence of refinements

Key Questions

- How refine mesh?
  - Aim for properties like smoothness
- How store mesh?
  - Aim for efficiency for implementing subdivision rules

Loop Subdivision Scheme

- How refine mesh?
  - Refine each triangle into 4 triangles by splitting each edge and connecting new vertices
  - Need rules for “even / odd” (white / black) vertices

Loop Subdivision Scheme

- How position new vertices?
  - Choose locations for new vertices as weighted average of original vertices in local neighborhood

Odd: Even:
Loop Subdivision Scheme

- Rules for extraordinary vertices and boundaries:

- Limit surface has provable smoothness properties!

Subdivision Schemes

- There are different subdivision schemes
  - Different methods for refining topology
  - Different rules for positioning vertices
    - Interpolating versus approximating

Subdivision Schemes

- How to choose $\beta$?
  - Analyze properties of limit surface
  - Interested in continuity of surface and smoothness
  - Involves calculating eigenvalues of matrices
    - Original Loop
      $$\beta = \frac{1}{3} \left( \frac{1}{2} - \frac{1}{4} \cos \frac{2\pi}{n} \right)^2$$
    - Warren
      $$\beta = \begin{cases} \frac{1}{4n} & n > 3 \\ \frac{1}{n} & n = 3 \end{cases}$$
Subdivision Surfaces

- Properties:
  - Accurate
  - Concise
  - Intuitive specification
  - Local support
  - Affine invariant
  - Arbitrary topology
  - Guaranteed continuity
  - Natural parameterization
  - Efficient display
  - Efficient intersections

- Advantages:
  - Simple method for describing complex surfaces
  - Relatively easy to implement
  - Arbitrary topology
  - Local support
  - Guaranteed continuity
  - Multiresolution

- Difficulties:
  - Intuitive specification
  - Parameterization
  - Intersections

Key Questions

- How refine mesh?
  - Aim for properties like smoothness

- How store mesh?
  - Aim for efficiency for implementing subdivision rules

Triangle Meshes

- Relevant properties:
  - Exactly 3 vertices per face
  - Any number of faces per vertex

- Useful adjacency structure for Loop subdivision:
  - Do not represent edges explicitly
  - Faces store refs to vertices and neighboring faces
  - Vertices store refs to adjacent faces and vertices

Assignment 3

- Interactive editing of subdivision surfaces
  - Loop subdivision scheme
  - Partial adjacency list mesh representation
  - Interactive vertex dragging

- Edit coarse mesh while display subdivided mesh
Assignment 3

- Store hierarchy of meshes
  - Full triangle mesh at every level
  - Vertices store references to counterparts one level up and one level down
  - Enables efficient re-positioning of mesh vertices after interactive dragging

Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Polyhedral Mesh</th>
<th>Subdivision Surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accurate</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Concise</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Intuitive specification</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Local support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Affine invariant</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Arbitrary topology</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Guaranteed continuity</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Natural parameterization</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Efficient display</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Efficient intersections</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>