Subdivision Surfaces

Adam Finkelstein
Princeton University
COS 426, Spring 2003

Course Syllabus
I. Image processing
II. Rendering
III. Modeling
IV. Animation

Modeling
• How do we ...
  o Represent 3D objects in a computer?
  o Construct 3D representations quickly/easily?
  o Manipulate 3D representations efficiently?

Equivalence of Representations
• Thesis:
  o Each fundamental representation has enough expressive power to model the shape of any geometric object
  o It is possible to perform all geometric operations with any fundamental representation!

  • Analogous to Turing-Equivalence:
    o All computers today are turing-equivalent, but we still have many different processors

3D Object Representations
• Raw data
  o Voxels
  o Point cloud
  o Range image
  o Polygons

• Surfaces
  o Mesh
  o Subdivision
  o Parametric
  o Implicit

• Solids
  o Octree
  o BSP tree
  o CSG
  o Sweep

• High-level structures
  o Scene graph
  o Skeleton
  o Application specific

Different representations for different types of objects
### Computational Differences

- **Efficiency**
  - Combinatorial complexity
  - Space/time trade-offs
  - Numerical accuracy/stability

- **Simplicity**
  - Ease of acquisition
  - Hardware acceleration
  - Software creation and maintenance

- **Usability**
  - Designer interface vs. computational engine

### 3D Object Representations

- **Raw data**
  - Voxels
  - Point cloud
  - Range image
  - Polygons

- **Solids**
  - Octree
  - BSP tree
  - CSG
  - Sweep

- **Surfaces**
  - Mesh
  - Subdivision
  - Parametric
  - Implicit

- **High-level structures**
  - Scene graph
  - Skeleton
  - Application specific

### 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 Surfaces

- Properties:
  - 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

---

Pixar

H&B Figure 18.46

H&B Figure 18.46
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

Loop Subdivision Scheme

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

What if vertex does not have degree 6?

Loop Subdivision Scheme

- Rules for extraordinary vertices and boundaries:

Loop

- How to choose β?
  - Analyze properties of limit surface
  - Interested in continuity of surface and smoothness
  - Involves calculating eigenvalues of matrices
    - Original Loop
      \[ \beta = \frac{1}{4} \left( 1 + \frac{n}{3} \cos \frac{2\pi}{3} \right) \]
    - Warren
      \[ \beta = \begin{cases} 
      \frac{1}{4} & n > 3 \\
      \frac{1}{10} & n = 3 
      \end{cases} \]

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

<table>
<thead>
<tr>
<th>Face split</th>
<th>Triangular meshes</th>
<th>Quad. meshes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approximating</td>
<td>Loop (C1)</td>
<td>Cannell-Clark (C2)</td>
</tr>
<tr>
<td>Decimating</td>
<td>Med. Butterfly (C3)</td>
<td>Kobbelt (C4)</td>
</tr>
</tbody>
</table>

Key Questions

• How refine mesh?
  ◦ Aim for properties like smoothness

• How store mesh?
  ◦ Aim for efficiency for implementing subdivision rules

Polygon Meshes

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

Independent Faces

• Each face lists vertex coordinates
  ◦ Redundant vertices
  ◦ No topology information
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
  - 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:

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

Assignment 3

• 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

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

• Difficulties:
  ◦ Intuitive specification
  ◦ Parameterization
  ◦ Intersections