Subdivision Surfaces

Adam Finkelstein
Princeton University
COS 426, Fall 2001

Course Syllabus

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

Image Processing
(Michael Bostock, CS426, Fall 99)
Rendering
(Michael Bostock, CS426, Fall 99)
Modeling
(Fessen Zem, CalTech)
Animation
(Angel, Plate 1)

Modeling

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

Different representations for different types of objects

3D Object Representations

• Raw data
  ▫ Volumes
  ▫ Point cloud
  ▫ Range image
  ▫ Polygons

• Solids
  ▫ Octree
  ▫ BSP tree
  ▫ CSG
  ▫ Sweep

• Surfaces
  ▫ Mesh
  ▫ Subdivision
  ▫ Parametric
  ▫ Implicit

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

Equivalence of Representations

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

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

- **Efficiency**
  - Combinatorial complexity (e.g. $O(n \log n)$)
  - Space/time trade-offs (e.g. z-buffer)
  - Numerical accuracy/stability (degree of polynomial)

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

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

3D Object Representations

- **Raw data**
  - Volumes
  - 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

- 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

Loop Subdivision Scheme

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

Loop Subdivision Scheme

• Rules for extraordinary vertices and boundaries:

Loop

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

Loop Subdivision Scheme

• 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

![Face split for triangles](image1.png)
![Face split for quads](image2.png)
**Key Questions**

- **How refine mesh?**
  - Aim for properties like smoothness

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

**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 4

- Interactive editing of subdivision surfaces
  - Loop subdivision scheme
  - Partial adjacency list mesh representation
  - Interactive vertex dragging
Assignment 4

- Edit coarse mesh while display subdivided mesh

Subdivision Surfaces

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

Summary

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

- Difficulties:
  - Intuitive specification
  - Parameterization
  - Intersections