Subdivision Surfaces

Greg Humphreys
University of Virginia
CS 445, Fall 2003

Modeling

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

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
  - 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

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

- Surfaces
  - Mesh
  - Subdivision
  - Parametric
  - Implicit

- Solids
  - Octree
  - BSP tree
  - CSG
  - Sweep

- 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

Base Mesh

Limit Surface
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*:
  - a. Make for odd vertices
  - b. Make for even vertices
Loop

• How to choose $\beta$?
  - Analyze properties of limit surface
  - Interested in continuity of surface and smoothness
  - Involves calculating eigenvalues of matrices
    - Original Loop
    - Warren

$$\beta = \frac{1}{n} \left( \frac{1}{2} - \frac{3}{n} \cos \frac{2\pi}{n} \right)^2$$

$$\beta = \left\{ \begin{array}{ll}
\frac{1}{16} & n > 3 \\
\frac{3}{16} & n = 3 
\end{array} \right.$$
Subdivision Schemes

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

FACE TABLE

- 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:**

```
\begin{align*}
\text{Vertex Table} &: \begin{array}{c|ccc}
V_1 & X_1 & Y_1 & Z_1 \\
V_2 & X_2 & Y_2 & Z_2 \\
V_3 & X_3 & Y_3 & Z_3 \\
V_4 & X_4 & Y_4 & Z_4 \\
\end{array} \\
\text{Edge Table} &: \begin{array}{cccc}
\varepsilon_1 & \varepsilon_2 & \varepsilon_3 & \varepsilon_4 \\
V_1 & V_2 & V_3 & V_4 \\
\end{array} \\
\text{Face Table} &: \begin{array}{cccc}
\Phi_1 & \Phi_2 & \Phi_3 & \Phi_4 \\
F_1 & F_2 & F_3 & F_4 \\
\end{array}
\end{align*}
```

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

Summary

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

- **Difficulties:**
  - Intuitive specification
  - Parameterization
  - Intersections