Lecture 8: Curved Surfaces
Subdivision Surfaces

Rui Wang
Overview

- Parametric Surfaces
  - Bezier patch
  - B-Spline patch
  - Drawing surfaces
- Surfaces from Curves
  - Surface of revolution
  - Generalized cylinder
- Subdivision Surfaces
Curved Surfaces

- Desired Attributes:
  - Predictable control
  - Analytic representation
  - Guaranteed continuity
  - Local control
  - Fast to compute

- Approach
  - Parametric
  - Polynomials
  - Patches

H&B Figure 10.46
Parametric Surfaces

- Surface points defined by parametric functions

\[ Q(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} \]

- Example: Ellipsoid

\[ Q(\theta, \phi) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r_x \cos \theta \cos \phi \\ r_y \cos \theta \sin \phi \\ r_z \sin \theta \end{bmatrix} \]
Parametric Surfaces

- Polynomial (degree n)

\[ f(u, v) = a_{00} + a_{10}u + a_{01}v + a_{20}u^2 + a_{11}uv + a_{02}v^2 + \ldots \]

- Generalized form

\[ [u, v] \in [0, 1] \times [0, 1] \]

\[ Q(u, v) = \sum_{i=0}^{k} P_i b_i(u, v) \]

- Derivatives

\[ \frac{\partial Q(u, v)}{\partial u}, \frac{\partial Q(u, v)}{\partial v} \]

Basis polynomial
Parametric Patches

- Each patch is defined by nxn control points (control point grid)
Parametric Patches

- Generated patch as a surface mesh
Parametric Patches

- Each column of control points define a curve (Bezier curve in this example)
Parametric Patches

Points on the same vertical lines form a new set of control points
Parametric Patches

This new set of control points define a curve on the final mesh
Parametric Patches

☐ Putting everything together

\[ Q(u, v) = U \times M \times \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} \times M^T \times V^T \]

\[ U = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \quad V = \begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix} \]

M is the center matrix describing the blending functions for a parametric cubic curve (e.g., Bezier, B-spline, etc.)
Beziers Patches

\[ Q(u, v) = U \times M \times \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} \times M^T \times V^T \]

\[ M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \]
Bezier Patches

Properties

- Interpolate four corner points
- Convex hull
- Local control
- Tangent at corner points?
Piecewise Parametric Surfaces

- Surface is formed by connecting multiple patch pieces:
Bezzer Surface

- Continuity constraints are similar to the constraints for Bezier splines
Beziers Surface

- $C^0$ continuity requires aligning boundary control points
Bezier Surface

- $C^1$ continuity requires aligning boundary control points and tangents.
B-Spline Patches

\[ Q(u, v) = U \times M \times \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} \times M^T \times V^T \]

\[ M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \]
**Drawing Surfaces**

- Simple approach is to loop through uniformly spaced increments of \( u \) and \( v \).

```c
DrawSurface(void) {
    for (int i = 0; i < imax; i++) {
        float u = i / float(imax);
        for (int j = 0; j < jmax; j++) {
            float v = j / float(jmax);
            DrawQuadrilateral(...);
        }
    }
}
```
Drawing Surfaces in OpenGL

- `glMap` defines the set of control points
- `glMapGrid` defines how finely to evaluate the surface
- `glEvalCoord/glEvalMesh` cause the mesh to be drawn
Surfaces from Curves

- Surface of Revolution
Conclusion of Parametric Surfaces

- **Advantages:**
  - Analytic representation
  - Easy to control and enumerate points
  - Possible to describe complex shapes

- **Disadvantages:**
  - Control mesh must be quadrilaterals
  - Continuity constraints difficult to maintain
  - Hard to work with arbitrary topology
Subdivision Surfaces
Representation of Surface

- Polygonal Mesh
  - Vertex data describe the x, y, z coordinates

Vertex Data

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x0</td>
<td>y0</td>
<td>z0</td>
</tr>
<tr>
<td>x1</td>
<td>y1</td>
<td>z1</td>
</tr>
<tr>
<td>x2</td>
<td>y2</td>
<td>z2</td>
</tr>
<tr>
<td>x3</td>
<td>y3</td>
<td>z3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Representation of Surface

- **Polygonal Mesh**
  - Face data describe connectivity between vertices.

<table>
<thead>
<tr>
<th>Face Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 0 1 2</td>
</tr>
<tr>
<td>3 2 1 3</td>
</tr>
<tr>
<td>3 2 3 4</td>
</tr>
<tr>
<td>3 1 6 7</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

The order matters!
Right hand rule.
Representation of Surface

- OpenGL Polygon

```c
float v1[3] = {-1.0,-1.0,0.0};
float v2[3] = { 1.0,-1.0,0.0};
float v3[3] = { 1.0, 1.0,0.0};
float v4[3] = {-1.0, 1.0,0.0};

glBegin(GL_POLYGON);
    glVertex3fv(v1);
    glVertex3fv(v2);
    glVertex3fv(v3);
    glVertex3fv(v4);
glEnd();
```
Triangle Mesh

- Triangle \((T)\)
- Defined by three vertices
- The tree vertices define three edges
Triangle Mesh

- **Edge** \((e)\)
- **Shared by two triangles** (if this is a manifold surface)
- Defined by two vertices and connected to two other vertices
Triangle Mesh

Vertex

Connected to a number of neighboring vertices (degree or valence)
Subdivision

- Intuition
Subdivision Surfaces

- Coarse mesh & subdivision rule
  - Define smooth surface as the limit of a sequence of refinements
Subdivision Scheme

- Step 1. insert new vertices
  - Break up each triangle into 4 sub-triangles by splitting each edge and connecting new vertices
Subdivision Scheme

☐ Step 2. Define new vertex position
Subdivision Scheme

- Loop’s subdivision scheme
  - New vertex locations are computed as weighted average of neighboring vertices

This is the regular case: each vertex has 6 neighbor vertices.
Subdivision Scheme

- What about irregular (extraordinary)?
Subdivision Scheme

What about irregular (extraordinary)?

Rules for *extraordinary* vertices and boundaries:

- Masks for odd vertices
- Masks for even vertices
Loop’s Subdivision Scheme

- Choosing $\beta$
  - Different choice will result in different limiting surface
  - Limiting surface $\rightarrow$ the surface that subdivision will converge to.
Loop’s Subdivision Scheme

Loop’s Subdivision

\[ \beta = \frac{1}{k} \left[ \frac{5}{8} - \left( \frac{3}{8} + \frac{1}{4} \cos \frac{2\pi}{k} \right)^2 \right] \]

Also called ‘valence’ sometimes.
Loop Subdivision Scheme

Limit surface has provable smoothness properties!
Subdivision Schemes

- Many different subdivision schemes
  - Different methods for refining topology
  - Different rules for positioning vertices

Zorin & Schroeder, SIGGRAPH 99, Course Notes
Subdivision Examples

Loop

Butterfly

Catmull-Clark

Doo-Sabin
Subdivision Schemes

Loop
Butterfly
Catmull-Clark
Doo-Sabin
Base Mesh
Limit Surface
Summary

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

- **Difficulties:**
  - Intuitive specification
  - Parameterization
Implementation of Loop Subdiv

- Step 1: Build vertex/edge/face structure
  - For each vertex: its neighboring vertices
  - For each edge: the vertices on its neighboring faces

- Step 2: For each edge, insert a new vertex.

- Step 3: Update the positions of old and new vertices, according to Loop formula.

- Step 4: Compute new faces
Triangle Mesh

- Given an input mesh defined by vertex data and triangle data:
- How to find all the edges?
- How to find the degree of each vertex?
- How to find the two other vertices shared by an edge?
- Demo