Curved Surfaces

Thomas Funkhouser
Princeton University
COS 426, Fall 1999

• Motivation
  ◦ Exact boundary representation for some objects
  ◦ More concise representation than polygonal mesh
Curved 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

Curved Surface Representations

• Polygonal meshes
• Subdivision surfaces
• Parametric surfaces
• Implicit surfaces
Curved Surface Representations

- Polygonal meshes
- Implicit surfaces
- Parametric surfaces
- Subdivision surfaces

Parametric Surfaces

- Boundary defined by parametric functions:
  - \( x = f_x(u,v) \)
  - \( y = f_y(u,v) \)
  - \( z = f_z(u,v) \)

- Example: ellipsoid
  \[
  x = r_x \cos \phi \cos \theta \\
  y = r_y \cos \phi \sin \theta \\
  z = r_z \sin \phi
  \]
Parametric Surfaces

- Advantages:
  - Easy to enumerate points on surface

- Problem:
  - Need piecewise-parametrics surfaces to describe complex shapes

Piecewise Parametric Surfaces

- Surface is partitioned into parametric patches:

Same ideas as parametric splines!
Parametric Patches

- Each patch is defined by blending control points

Same ideas as parametric curves!

FvDFH Figure 11.44

Parametric Patches

- Point $Q(u,v)$ on the patch is the tensor product of parametric curves defined by the control points

Watt Figure 6.21
Parametric Bicubic Patches

- Point \(Q(u,v)\) on any patch is defined by combining control points with polynomial blending functions:

\[
Q(u,v) = UM \begin{bmatrix}
P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\
P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\
P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\
P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4}
\end{bmatrix} M^T V^T
\]

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

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

B-Spline Patches

\[
Q(u,v) = UM_{B\text{-Spline}} \begin{bmatrix}
P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\
P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\
P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\
P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4}
\end{bmatrix} M_{B\text{-Spline}}^T V
\]

\[
M_{B\text{-Spline}} = \begin{bmatrix}
-1/6 & 1/2 & -1/2 & 1/6 \\
1/2 & -1 & 1/2 & 0 \\
-1/2 & 0 & 1/2 & 0 \\
1/6 & 2 & 1/2 & 0
\end{bmatrix}
\]

Watt Figure 6.28
**Bezier Patches**

\[
Q(u, v) = U M_{\text{Bezier}} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} M_{\text{Bezier}}^T V
\]

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

**Properties:**
- Interpolates four corner points
- Convex hull
- Local control

---

FvDFH Figure 11.42

Watt Figure 6.22
Bezier Surfaces

- Continuity constraints are similar to the ones for Bezier splines

Bezier Surfaces

- $C^0$ continuity requires aligning boundary curves
Bezier Surfaces

- C¹ continuity requires aligning boundary curves and derivatives

```
DrawSurface(void) {
  for (int i = 0; i < imax; i++) {
    float u = umin + i * ustep;
    for (int j = 0; j < jmax; j++) {
      float v = vmin + j * vstep;
      DrawQuadrilateral(...);
    }
  }
}
```

Watt Figure 6.26b

Drawing Bezier Surfaces

- Simple approach is to loop through uniformly spaced increments of u and v

```
DrawSurface(void) {
  for (int i = 0; i < imax; i++) {
    float u = umin + i * ustep;
    for (int j = 0; j < jmax; j++) {
      float v = vmin + j * vstep;
      DrawQuadrilateral(...);
    }
  }
}
```

Watt Figure 6.32
Drawing Bezier Surfaces

• Better approach is to use adaptive subdivision:

```plaintext
DrawSurface(surface)
{
    if Flat (surface, epsilon) {
        DrawQuadrilateral(surface);
    } else {
        SubdivideSurface(surface, ...);
        DrawSurface(surfaceLL);
        DrawSurface(surfaceLR);
        DrawSurface(surfaceRL);
        DrawSurface(surfaceRR);
    }
}
```

Watt Figure 6.32

Drawing Bezier Surfaces

• One problem with adaptive subdivision is avoiding cracks at boundaries between patches at different subdivision levels

Watt Figure 6.33
**Parametric Surfaces**

- Advantages:
  - Easy to enumerate points on surface
  - Possible to describe complex shapes

- Disadvantages:
  - Control mesh must be quadrilaterals
  - Continuity constraints difficult to maintain
  - Hard to find intersections

**Curved Surface Representations**

- Polygonal meshes
- Subdivision surfaces
- Parametric surfaces
- Implicit surfaces
Implicit Surfaces

- Boundary defined by implicit function:
  - \( f(x, y, z) = 0 \)

- Example: linear (plane)
  - \( ax + by + cz + d = 0 \)

\[ N = (a, b, c) \]

\[ f(x, y, z) = 0 \]

Implicit Surfaces

- Example: quadric
  - \( f(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fzx + 2gx + 2hy + 2jz + k \)

- Common quadric surfaces:
  - Sphere
  - Ellipsoid
  - Torus
  - Paraboloid
  - Hyperboloid

\[ \left( \frac{x}{r_x} \right)^2 + \left( \frac{y}{r_y} \right)^2 + \left( \frac{z}{r_z} \right)^2 - 1 = 0 \]
**Implicit Surfaces**

- **Advantages:**
  - Easy to test if point is on surface
  - Easy to intersect two surfaces
  - Easy to compute z given x and y

- **Disadvantages:**
  - Hard to describe complex shapes
  - Hard to enumerate points on surface

**Summary**

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