Surfaces

Adam Finkelstein
Princeton University
COS 426, Fall 2001

Curved Surfaces

• Motivation
  o Exact boundary representation for some objects
  o 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:
  o \( x = f_x(u,v) \)
  o \( y = f_y(u,v) \)
  o \( z = f_z(u,v) \)

• Example: ellipsoid
  \[
  x = r_e \cos \phi \cos \theta \\
  y = r_e \cos \phi \sin \theta \\
  z = r_e \sin \phi
  \]
Surface of revolution

- Idea: take a curve and rotate it about an axis

Swept surface

- Idea: sweep one curve along path of another curve

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:

Parametric Patches

- Each patch is defined by blending control points

Parametric Patches

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

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

$$
\mathbf{Q}(u,v) = \mathbf{U} \mathbf{M} \mathbf{V}^T
$$

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

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

B-Spline Patches

$$
\mathbf{Q}(u,v) = \mathbf{U} \mathbf{M}_b \mathbf{V}^T
$$

$$
\mathbf{M}_b = \begin{bmatrix}
-\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \\
\frac{1}{2} & -1 & \frac{1}{2} & 0 \\
-\frac{1}{2} & \frac{1}{2} & 0 & 0 \\
\frac{1}{6} & -\frac{1}{2} & 0 & 0
\end{bmatrix}
$$

Bezier Patches

$$
\mathbf{Q}(u,v) = \mathbf{U} \mathbf{M}_b \mathbf{V}^T
$$

$$
\mathbf{M}_b = \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 3 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
$$

Bezier Surfaces

- Continuity constraints are similar to the ones for Bezier splines

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

- $C^1$ continuity requires aligning boundary curves and derivatives

\[
\text{Watt Figure 6.26b}
\]

Drawing Bezier Surfaces

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

\[
\text{DrawSurface}(\text{void}) \\
\{ \\
\quad \text{for (int } i = 0; i < \text{imax}; i++) \\
\quad \quad \text{float } u = \text{umin} + i * \text{ustep}; \\
\quad \quad \text{for (int } j = 0; j < \text{jmax}; j++) \\
\quad \quad \quad \text{float } v = \text{vmin} + j * \text{vstep}; \\
\quad \quad \quad \text{DrawQuadrilateral(...)}; \\
\quad \}\n\]

\[
\text{Watt Figure 6.32}
\]

Drawing Bezier Surfaces

- Better approach is to use adaptive subdivision:

\[
\text{DrawSurface}(\text{surface}) \\
\{ \\
\quad \text{if Flat (surface, } \epsilon) \{ \\
\quad \quad \text{DrawQuadrilateral(surface);} \\
\quad \} \text{ else } \\
\quad \quad \text{SubdivideSurface(surface, ...);} \\
\quad \quad \text{DrawSurface(surfaceLL);} \\
\quad \quad \text{DrawSurface(surfaceLR);} \\
\quad \quad \text{DrawSurface(surfaceRL);} \\
\quad \quad \text{DrawSurface(surfaceRR);} \\
\quad \}\n\]

\[
\text{Watt Figure 6.32}
\]

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)
  \[ ax + by + cz + d = 0 \]

**Example: quadric**

\[ f(x, y, z) = ax^2 + by^2 + cz^2 + 2dxy + 2eyz + 2fxz + 2gx + 2hy + 2jz + k \]

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

**Implicit surface examples**

- MaxMan Blobby Object
- Skin [Markowitz99]

**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 specific 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>

**Blender**

Blender (www.blender.nl)