More Curves and Surfaces

Jason Lawrence
Princeton University
COS 426, Spring 2005

Types of Spline Curves

- Splines covered in last lecture
  - Hermite
  - Bezier
  - Catmull-Rom
  - B-Spline

Each has different blending functions resulting in different properties

Uniform Cubic B-Splines

- Properties:
  - Local control
  - $C^2$ continuity
  - Approximating

B-Spline Blending Functions

- Properties imply blending functions:
  - Cubic polynomials
  - Four control vertices affect each point
  - $C^2$ continuity
B-Spline Blending Functions

- How derive blending functions?
  - Cubic polynomials
  - Local control
  - C² continuity

V₀ V₁ V₂ V₃ V₄ V₅

b₀ b₁ b₂ b₃

u

B-Spline Blending Functions

• Four cubic polynomials for four vertices
  - 16 variables (degrees of freedom)
  - Variables are $a_i, b_i, c_i, d_i$ for four blending functions

\[
\begin{align*}
    b_0(u) &= a_0u^3 + b_0u^2 + c_0u + d_0 \\
    b_1(u) &= a_1u^3 + b_1u^2 + c_1u + d_1 \\
    b_2(u) &= a_2u^3 + b_2u^2 + c_2u + d_2 \\
    b_3(u) &= a_3u^3 + b_3u^2 + c_3u + d_3
\end{align*}
\]

B-Spline Blending Functions

• C² continuity implies 15 constraints
  - Position of two curves same
  - Derivative of two curves same
  - Second derivatives same

V₀ V₁ V₂ V₃ V₄ V₅

Fifteen continuity constraints:

\[
\begin{align*}
    0 &= b_0(0) & 0 &= b_0'(0) & 0 &= b_0''(0) \\
    b_0(1) &= b_1(0) & b_0'(1) &= b_1'(0) & b_0''(1) &= b_1''(0) \\
    b_1(1) &= b_2(0) & b_1'(1) &= b_2'(0) & b_1''(1) &= b_2''(0) \\
    b_2(1) &= b_3(0) & b_2'(1) &= b_3'(0) & b_2''(1) &= b_3''(0) \\
    b_3(1) &= 0 & b_3'(1) &= 0 & b_3''(1) &= 0
\end{align*}
\]

One more convenient constraint:

\[
b_0(0) + b_1(0) + b_2(0) + b_3(0) = 1
\]
B-Spline Blending Functions

- Solving the system of equations yields:

\[
\begin{aligned}
  b_{-3}(u) &= -\frac{1}{6}u^3 + \frac{1}{2}u^2 - \frac{1}{2}u + \frac{1}{6} \\
  b_{-2}(u) &= \frac{1}{2}u^3 - u^2 + \frac{5}{6} \\
  b_{-1}(u) &= -\frac{1}{2}u^3 + \frac{1}{2}u^2 + \frac{1}{2}u + \frac{1}{6} \\
  b_0(u) &= \frac{1}{6}u^3 
\end{aligned}
\]

In matrix form:

\[
\begin{bmatrix}
  -\frac{1}{6} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{6} \\
  \frac{1}{2} & -1 & \frac{5}{6} \\
  -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} \\
  \frac{1}{6} & \frac{1}{2} & -1 & \frac{5}{6} \\
\end{bmatrix}
\begin{bmatrix}
  V_0 \\
  V_1 \\
  V_2 \\
  V_3 \\
\end{bmatrix}
\]

In plot form:

\[
B(u) = \sum_{j=0}^{m} a_j B_j(u)
\]

- Blending functions imply properties:
  - Local control
  - Approximating
  - C^2 continuity
  - Convex hull

\[
Q(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix}
  -1 & 3 & -3 & 1 \\
  3 & -6 & 3 & 0 \\
 -3 & 0 & 3 & 0 \\
  1 & 4 & 1 & 0 \\
\end{bmatrix} \begin{bmatrix}
  V_0 \\
  V_1 \\
  V_2 \\
  V_3 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  V_0 \\
  V_1 \\
  V_2 \\
  V_3 \\
\end{bmatrix}
\]
Curved Surfaces

- Motivation
  - Exact boundary representation for some objects
  - More concise representation than polygonal mesh

Curved Surface Representations

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

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

Parametric Surfaces

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

- Example: ellipsoid
  - $x = r_z \cos \phi \cos \theta$
  - $y = r_z \cos \phi \sin \theta$
  - $z = r_z \sin \phi$
Parametric Surfaces

• Advantages:
  o Easy to enumerate points on surface

• Problem:
  o Need piecewise-parametric surfaces to describe complex shapes

Parametric Patches

• Each patch is defined by blending control points

Same ideas as parametric curves!

Piecewise Parametric Surfaces

• Surface is partitioned into parametric patches:

Same ideas as parametric splines!

Parametric Patches

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

Watt Figure 6.25

Watt Figure 11.42

Watt Figure 6.21
Parametric Bicubic Patches

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

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

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

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

Bezier Patches

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

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

Bezier Patches

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

B-Spline Patches

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

\[
M_{\text{Spline}} = \begin{bmatrix} 1/6 & 1/2 & -1/2 & 1/6 \\ 1/2 & -1 & 1/2 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 1/6 & 1/3 & 1/6 & 0 \end{bmatrix}
\]
Bezier Surfaces

- Continuity constraints are similar to the ones for Bezier splines

---

Bezier Surfaces

- \( C^0 \) continuity requires aligning boundary curves

---

Drawing Bezier 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 = umin + i * ustep;
        for (int j = 0; j < jmax; j++) {
            float v = vmin + j * vstep;
            DrawQuadrilateral(...);
        }
    }
}
```
**Drawing Bezier Surfaces**

• Better approach is to use adaptive subdivision:

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

**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

**Drawing Bezier Surfaces**

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

Avoid these cracks by adding extra vertices and triangulating quadrilaterals whose neighbors are subdivided to a finer level

**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
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 Surfaces (H&B Figure 10.10)

Implicit surface examples

- MaxMan Blobby Object
- Skin [Markosian99]

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 specific complex shapes
  - Hard to enumerate points on surface
<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>