3D Modeling
Parametric Curves & Surfaces

Shandong University
Spring 2013
3D Object Representations

- **Raw data**
  - Point cloud
  - Range image
  - Polygon soup

- **Surfaces**
  - Mesh
  - Subdivision
  - **Parametric**
  - Implicit

- **Solids**
  - Voxels
  - BSP tree
  - CSG
  - Sweep

- **High-level structures**
  - Scene graph
  - Skeleton
  - Application specific
Parametric Surfaces

• Applications
  – Design of smooth surfaces in cars, ships, etc.
Continuity

- When two curves are joined, we typically want some degree of continuity across the boundary (the knot)

- **Geometric Continuity**
  - $G^0$: The curves touch at the join point.
  - $G^1$: The curves also share a common tangent direction at the join point.
  - $G^2$: The curves also share a common center of curvature at the join point.
Continuity

• *Parametric Continuity*
  – $C^{-1}$: curves include discontinuities
  – $C^{0}$: curves are joined
  – $C^{1}$: first derivatives are continuous
  – $C^{2}$: first and second derivatives are continuous
  – $C^{n}$: first through $n^{th}$ derivatives are continuous
Parametric (red) and Geometric (black) Continuity comparison
Parametric Curves

- Boundary defined by parametric functions:
  \[ x = f_x(u) \]
  \[ y = f_y(u) \]

- Example: line segment

\[
\begin{align*}
x(u) &= (1 - u)x_0 + ux_1 \\
y(u) &= (1 - u)y_0 + uy_1
\end{align*}
\]
Parametric Curves

- Boundary defined by parametric functions:
  \[ x = f_x(u) \]
  \[ y = f_y(u) \]

- Example: ellipse

  \[ f_x(u) = r_x \cos u \]
  \[ f_y(u) = r_y \sin u \]
Parametric Curves

• How can we define arbitrary curves?

\[ x = f_x(u) \]
\[ y = f_y(u) \]
Parametric Curves

- How can we define arbitrary curves?
  \[ x = f_x(u) \]
  \[ y = f_y(u) \]

- Use functions that “blend” control points
  \[ x = f_x(u) = (1 - u)V_{0x} + uV_{1x} \]
  \[ y = f_y(u) = (1 - u)V_{0y} + uV_{1y} \]
Parametric Curves

- More generally:

\[ x(u) = \sum_{i=0}^{n} B_i(u) \cdot V_{ix} \]

\[ y(u) = \sum_{i=0}^{n} B_i(u) \cdot V_{iy} \]
Parametric Curves

- What $B(u)$ functions should we use?

\[
x(u) = \sum_{i=0}^{n} B_i(u) \cdot V_{ix} \\
y(u) = \sum_{i=0}^{n} B_i(u) \cdot V_{iy}
\]
Parametric Curves

- What $B(u)$ functions should we use?

$$x(u) = \sum_{i=0}^{n} B_i(u) \cdot V_{idx}$$

$$y(u) = \sum_{i=0}^{n} B_i(u) \cdot V_{idy}$$
Parametric Polynomial Curves

• Polynomial blending functions:

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

• Advantages of polynomials
  – Easy to compute
  – Easy differentiation
  – Easy to derive curve properties
Parametric Polynomial Curves

• Polynomial blending functions:

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

• What degree polynomial?
  – Easy to compute
  – Easy to control
  – Expressive
Parametric Cubic Curves

• Why cubic?
  – lower-degree polynomials give too little flexibility in controlling the shape of the curve
  – higher-degree polynomials can introduce unwanted wiggles and require more computation
  – lowest degree that allows specification of endpoints and their derivatives
  – lowest degree that is not planar in 3D
Parametric Cubic Curves

- General form:

\[
\begin{align*}
 x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\
 y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\
 z(u) &= a_z u^3 + b_z u^2 + c_z u + d_z \\
 C &= \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \\
 T &= [u^3 \quad u^2 \quad u \quad 1] \\
 Q(u) &= [x(u) \quad y(u) \quad z(u)] = T \cdot C
\end{align*}
\]
Major Types of Parametric Cubic Curves

• Cubic Bézier
  – defined by two endpoints and two other points that control the endpoint tangent vectors

• Hermite
  – defined by two endpoints and two tangent vectors

• Splines
  – several kinds, each defined by four points
  – uniform B-splines, non-uniform B-splines, ß-splines
Bézier Curves

- In 1962, Pierre Bézier, an engineer of French Renault Car company, proposed a new kind of curve representation, and finally developed a system UNISURF for car surface design in 1972.
Bézier Curves

- Two contributors
  - Pierre Bézier (at Renault)
  - Paul de Casteljau (at Citroen)

- Curve $Q(u)$ is defined by nested interpolation:
  
  $V_i$’s are control points
  \{ $V_0, V_1, \ldots, V_n$ \} is control polygon
Basic properties of Bézier curves

• Endpoint interpolation:
  \[ Q(0) = V_0 \]
  \[ Q(1) = V_n \]

• Convex hull:
  – Curve is contained within convex hull of control polygon

• Transformational invariance

• Symmetry
  \[ Q(u) \text{ defined by } \{V_0, \ldots, V_n\} \equiv Q(1 - u) \text{ defined by } \{V_n, \ldots, V_0\} \]
More Properties

- General case: Bernstein polynomials
  \[ Q(u) = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} V_i \]

- Degree: polynomial of degree n

- Tangents:
  \[ Q'(0) = n(V_1 - V_0) \]
  \[ Q'(1) = n(V_n - V_{n-1}) \]
Some Bézier Curves

(a) 

(b) 

(c) 

(d) 

(e)
Some Bézier Curves

Animation of a quadratic Bézier curve, $t$ in $[0,1]$  
Animation of a cubic Bézier curve, $t$ in $[0,1]$
Cubic Bézier Curves

\[ Q(u) = \sum_{i=0}^{n} \binom{n}{i} u^i (1 - u)^{n-i} V_i \]

\[ = (1 - u)^3 V_0 + 3u(1 - u)^2 V_1 + 3u^2(1 - u) V_2 + u^3 V_3 \]

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

- Given: two points and two tangent vectors
  - Similarity to cubic Bézier curves
  - Other two Bézier control points along those tangents
- Call the points $P_1$ and $P_2$, and the tangents $R_1$ and $R_2$

- So given two points and vectors, find the coefficients of $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$ etc.
Hermite Curves

- We can treat \( x \) in the mapping as a vector. Its components can be explicitly written as

\[
\begin{bmatrix}
  x \\
  y \\
  z
\end{bmatrix} =
\begin{bmatrix}
  x_1 & x_0 & x'_1 & x'_0 \\
  y_1 & y_0 & y'_1 & y'_0 \\
  z_1 & z_0 & z'_1 & z'_0
\end{bmatrix}
\begin{bmatrix}
  -2 & 3 & 0 & 0 \\
  2 & -3 & 0 & 1 \\
  1 & -1 & 0 & 0 \\
  1 & -2 & 1 & 0
\end{bmatrix} \begin{bmatrix}
  u^3 \\
  u^2 \\
  u \\
  1
\end{bmatrix}
\]
Longer Curves

• A single cubic Bezier or Hermite curve can only capture a small class of curves
  – At most 2 inflection points

• One solution is to raise the degree
  – Allows more control, at the expense of more control points and higher degree polynomials
  – Control is not local, one control point influences entire curve

• Alternate, most common solution is to join pieces of cubic curve together into piecewise cubic curves
  – Total curve can be broken into pieces, each of which is cubic
  – Local control: Each control point only influences a limited part of the curve
  – Interaction and design is much easier
Piecewise Bézier Curve
• Question:
  – How do we ensure that two Hermite curves are $C^1$ across a knot?

• Question:
  – How do we ensure that two Bézier curves are $C^0$, or $C^1$, or $C^2$ across a knot?
Achieving Continuity

• For Hermite curves, the user specifies the derivatives, so $C^1$ is achieved simply by sharing points and derivatives across the knot.

• For Bézier curves:
  – They interpolate their endpoints, so $C^0$ is achieved by sharing control points.
  – The parametric derivative is a constant multiple of the vector joining the first/last 2 control points.
  – So $C^1$ is achieved by setting $P_{0,3}=P_{1,0}=J$, and making $P_{0,2}$ and $J$ and $P_{1,1}$ collinear, with $J-P_{0,2}=P_{1,1}-J$.
  – $C^2$ comes from further constraints on $P_{0,1}$ and $P_{1,2}$. 
Bézier Continuity
B-Spline Curves

Why to introduce B-Spline?

- Bezier curve/surface has many advantages, but they have two main shortcomings:
  - Bezier curve/surface cannot be modified locally
  - It is very complex to satisfy geometric continuity conditions for Bezier curves or surfaces joining.

Why not use lower degree piecewise polynomial with continuous joining?

- that’s Spline
B-Spline Curves

- Formula of B-Spline Curve.

\[ P(t) = \sum_{i=0}^{n} P_i N_{i,k}(t) \]

- \( P_i \) (i=0,1,\ldots,n) are control points.
- \( N_{i,k}(t) \) (i=0,1,\ldots,n) are the i-th B-Spline basis function of order k.
- B-Spline basis function is an order k (degree k -1) piecewise polynomial.
B-Spline Curves

- Definition of B-Spline Basis Function

- de Boor-Cox recursion formula:

\[ N_{i,1}(t) = \begin{cases} 
1 & t_i < x < t_{i+1} \\
0 & Otherwise 
\end{cases} \]

\[ N_{i,k}(t) = \frac{t-t_i}{t_{i+k-1}-t_i} N_{i,k-1}(t) + \frac{t_{i+k}-t}{t_{i+k}-t_{i+1}} N_{i+1,k-1}(t) \]

- Knot Vector: a sequence of non-decreasing number

\[ t_0, t_1, \ldots, t_{k-1}, t_k, \ldots, t_n, t_{n+1}, \ldots, t_{n+k-1}, t_{n+k} \]
B-Spline Curves

Cubic B-Spline \( n=3, k=4 \)

\[
p(u) = (-\frac{1}{6}p_0 + \frac{1}{2}p_1 - \frac{1}{2}p_2 + \frac{1}{6}p_3)u^3 + \\
\quad (\frac{1}{2}p_0 - p_1 + \frac{1}{2}p_2)u^2 + \\
\quad (-\frac{1}{2}p_0 + \frac{1}{2}p_2)u + \\
\quad \frac{1}{6}p_0 + \frac{2}{3}p_1 + \frac{1}{6}p_2
\]

but makes more sense as…

\[
p(u) = (-\frac{1}{6}u^3 + \frac{1}{2}u^2 - \frac{1}{2}u + \frac{1}{6})p_0 + \\
\quad (\frac{1}{2}u^3 - u^2 + \frac{2}{3})p_1 + \\
\quad (-\frac{1}{2}u^3 + \frac{1}{2}u^2 + \frac{1}{2}u + \frac{1}{6})p_2 + \\
\quad (\frac{1}{6}u^3)\ p_3
\]
B-Spline Curves

- In matrix form

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

• Examples:
B-Spline Curves

• Properties:
  – Local
  – Continuity
    • $P(t)$ is $C^{k-1-r}$ continuous at a node of repetitiveness $r$.
  – Convex hull
  – Piecewise polynomial
  – Geometry invariant
  – Affine invariant
  – Flexibility
How to Choose a Spline

• Hermite curves are good for single segments where you know the parametric derivative or want easy control of it

• Bézier curves are good for single segments or patches where a user controls the points

• B-splines are good for large continuous curves and surfaces

• NURBS are the most general, and are good when that generality is useful, or when conic sections must be accurately represented (CAD)
Parametric Surfaces

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

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

- Example: quadrilateral

\[
\begin{align*}
  f_x(u, v) &= (1 - v)((1 - u)x_0 + ux_1) + v((1 - u)x_2 + ux_3) \\
  f_y(u, v) &= (1 - v)((1 - u)y_0 + uy_1) + v((1 - u)y_2 + uy_3) \\
  f_z(u, v) &= (1 - v)((1 - u)z_0 + uz_1) + v((1 - u)z_2 + uz_3)
\end{align*}
\]
Parametric Surfaces

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

- Example: quadrilateral

\[
\begin{align*}
  f_x(u, v) &= (1 - v)(1 - u)x_0 + ux_1 + v(1 - u)x_2 + ux_3 \\
  f_y(u, v) &= (1 - v)(1 - u)y_0 + uy_1 + v(1 - u)y_2 + uy_3 \\
  f_z(u, v) &= (1 - v)(1 - u)z_0 + uz_1 + v(1 - u)z_2 + uz_3
\end{align*}
\]
Parametric Surfaces

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

• Example: ellipsoid

\[ f_x(u, v) = r_x \cos \phi \cos \theta \]
\[ f_y(u, v) = r_y \cos \phi \sin \theta \]
\[ f_z(u, v) = r_z \sin \phi \]
Parametric Surfaces

Advantage: easy to enumerate points on surface.

Disadvantage: need piecewise-parametric surface to describe complex shape.
Piecewise Polynomial Parametric Surfaces

- Surface is partitioned into parametric patches:
Parametric Patches

• Each patch is defined by blending control points

Same ideas as parametric curves!
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 $Q(u,v)$ on any patch is defined by combining control points with polynomial blending functions:

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

$$U = [u^3 \quad u^2 \quad u \quad 1], \quad V = [v^3 \quad v^2 \quad v \quad 1]$$

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

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

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

• The patch interpolates its corner points
  – Comes from the interpolation property of the underlying curves

• The tangent plane at each corner interpolates the corner vertex and the two neighboring edge vertices

• The patch lies within the convex hull of its control vertices
  – The basis functions sum to one and are positive everywhere
Bézier Patches

- A patch mesh is just many patches joined together along their edges
  - Patches meet along complete edges
  - Each patch must be a quadrilateral
Bézier Surfaces

- Continuity constraints are similar to the ones for Bézier splines
Bézier Surfaces

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

- $C^1$ continuity requires aligning boundary curves and derivatives
**B-Spline Patches**

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

\[ M_{B-Spline} = \begin{bmatrix} -1/6 & 1/2 & -1/2 & 1/6 \\ 1/2 & -1 & 1/2 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 1/6 & 2/3 & 1/6 & 0 \end{bmatrix} \]
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
## Summary

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