3D Object Representations

- **Points**
  - Range image
  - Point cloud

- **Surfaces**
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit

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

- **High-level structures**
  - Scene graph
  - Application specific
3D Object Representations

- Points
  - Range image
  - Point cloud

- Surfaces
  - Polygonal mesh
  - Subdivision
  - Parametric
  - Implicit

- Solids
  - Voxels
  - BSP tree
  - CSG
  - Sweep

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

• Applications
  ◦ Design of smooth surfaces in cars, ships, etc.
Parametric Surfaces

- Applications
  - Design of smooth surfaces in cars, ships, etc.
Parametric Surfaces

- Applications
  - Design of smooth surfaces in cars, ships, etc.
  - Creating characters or scenes for movies
Parametric Curves

- Applications
  - Defining motion trajectories for objects or cameras
Parametric Curves

• Applications
  ◦ Defining motion trajectories for objects or cameras
  ◦ Defining smooth interpolations of sparse data
Parametric Curves

• Applications
  ◦ Defining motion trajectories for objects or cameras
  ◦ Defining smooth interpolations of sparse data
Outline

• Parametric curves
  ◦ Cubic B-Spline
  ◦ Cubic Bézier

• Parametric surfaces
  ◦ Bi-cubic B-Spline
  ◦ Bi-cubic Bézier
Outline

- Parametric curves
  - Cubic B-Spline
  - Cubic Bézier

- Parametric surfaces
  - Bi-cubic B-Spline
  - Bi-cubic Bézier
Parametric Curves

• Defined by parametric functions:
  ○ $x = f_x(u)$
  ○ $y = f_y(u)$

• Example: line segment
  
  $f_x(u) = (1-u)x_0 + ux_1$
  $f_y(u) = (1-u)y_0 + uy_1$

  $u \in [0..1]$
Parametric Curves

- Defined by parametric functions:
  - \( x = f_x(u) \)
  - \( y = f_y(u) \)

- Example: ellipse

\[
\begin{align*}
  f_x(u) &= r_x \cos \frac{u}{2\pi} \\
  f_y(u) &= r_y \sin \frac{u}{2\pi}
\end{align*}
\]

\( u \in [0..1] \)
Parametric curves

How to easily define arbitrary curves?

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

How to easily define arbitrary curves?

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

Use functions that “blend” control points

\[ x = f_x(u) = V0_x(1 - u) + V1_xu \]
\[ y = f_y(u) = V0_y(1 - u) + V1_yu \]
Parametric curves

More generally:

\[ x(u) = \sum_{i=0}^{n} B_i(u) * Vi_x \]

\[ y(u) = \sum_{i=0}^{n} B_i(u) * Vi_y \]
Parametric curves

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

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

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

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

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

\[ y(u) = \sum_{i=0}^{n} B_i(u) \cdot V_i \cdot y \]

\[ B_0 \]

\[ B_1 \]
What $B(u)$ functions should we use?

$$x(u) = \sum_{i=0}^{n} B_i(u) * V_i_x$$

$$y(u) = \sum_{i=0}^{n} B_i(u) * V_i_y$$
**Parametric Polynomial Curves**

- Polynomial blending functions:

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

- Advantages of polynomials
  - Easy to compute
  - Infinitely continuous
  - 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
Piecewise Parametric Polynomial Curves

- **Splines:**
  - Split curve into segments
  - Each segment defined by low-order polynomial blending subset of control vertices

- **Motivation:**
  - Same blending functions for every segment
  - Prove properties from blending functions
  - Provides local control & efficiency

- **Challenges**
  - How choose blending functions?
  - How determine properties?
Cubic Splines

- Some properties we might like to have:
  - Local control
  - Continuity
  - Interpolation?
  - Convex hull?

Blending functions determine properties

Properties determine blending functions

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

• Parametric curves
  - Cubic B-Spline
    - Cubic Bézier

• Parametric surfaces
  - Bi-cubic B-Spline
  - Bi-cubic Bézier
Cubic B-Splines

- Properties:
  - Local control
  - $C^2$ continuity at joints (infinitely continuous within each piece)
  - Approximating
  - Convex hull
Blending functions:

\[ B_i(u) = \sum_{j=0}^{m} a_j u^j; \]
Cubic B-Spline Blending Functions

- How derive blending functions?
  - Cubic polynomials
  - Local control
  - $C^2$ continuity
  - Convex hull
Cubic 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

$$
b_{-0}(u) = a_0 u^3 + b_0 u^2 + c_0 u + d_0 \\
b_{-1}(u) = a_1 u^3 + b_1 u^2 + c_1 u + d_1 \\
b_{-2}(u) = a_2 u^3 + b_2 u^2 + c_2 u + d_2 \\
b_{-3}(u) = a_3 u^3 + b_3 u^2 + c_3 u + d_3
$$
Cubic B-Spline Blending Functions

- $C^2$ continuity implies 15 constraints
  - Position of two curves same
  - Derivative of two curves same
  - Second derivatives same
Cubic B-Spline Blending Functions

Fifteen continuity constraints:

\[ \begin{align*}
0 &= b_{-0}(0) \\
b_{-0}(1) &= b_{-1}(0) \\
b_{-1}(1) &= b_{-2}(0) \\
b_{-2}(1) &= b_{-3}(0) \\
b_{-3}(1) &= 0 \\
0 &= b_{-0}'(0) \\
b_{-0}'(1) &= b_{-1}'(0) \\
b_{-1}'(1) &= b_{-2}'(0) \\
b_{-2}'(1) &= b_{-3}'(0) \\
b_{-3}'(1) &= 0 \\
0 &= b_{-0}''(0) \\
b_{-0}''(1) &= b_{-1}''(0) \\
b_{-1}''(1) &= b_{-2}''(0) \\
b_{-2}''(1) &= b_{-3}''(0) \\
b_{-3}''(1) &= 0 \\
\end{align*} \]

One more convenient constraint:

\[ b_{-0}(0) + b_{-1}(0) + b_{-2}(0) + b_{-3}(0) = 1 \]
Cubic B-Spline Blending Functions

• Solving the system of equations yields:

\[ b_{-3}(u) = -\frac{1}{6} u^3 + \frac{1}{2} u^2 - \frac{1}{2} u + \frac{1}{6} \]
\[ b_{-2}(u) = \frac{1}{2} u^3 - u^2 + \frac{2}{3} \]
\[ b_{-1}(u) = -\frac{1}{2} u^3 + \frac{1}{2} u^2 + \frac{1}{2} u + \frac{1}{6} \]
\[ b_{-0}(u) = \frac{1}{6} u^3 \]
Cubic B-Spline Blending Functions

- In matrix form:

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

In plot form:

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

\[ b_0 \]

\[ b_{-1} \]

\[ b_{-2} \]

\[ b_{-3} \]

\[ V_0 \]

\[ V_1 \]

\[ V_2 \]

\[ V_3 \]

\[ V_4 \]

\[ V_5 \]
Cubic B-Spline Blending Functions

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

• Parametric curves
  ◦ Cubic B-Spline
  ➢ Cubic Bézier

• Parametric surfaces
  ◦ Bi-cubic B-Spline
  ◦ Bi-cubic Bézier
Cubic Bézier

• Developed around 1960 by both
  ◦ Pierre Bézier (Renault)
  ◦ Paul de Casteljau (Citroen)

• Properties:
  ◦ Local control
  ◦ Continuity depends on control points
  ◦ Interpolating (every third)

Properties determine blending functions
Blending functions determine properties
Cubic Bézier Curves

Blending functions:

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

\[ B_{i-3} \quad B_{i-2} \quad B_{i-1} \quad B_i \]

\[ V_0 \quad V_1 \quad V_2 \quad V_3 \quad V_4 \quad V_5 \quad V_6 \]
Cubic Bézier Curves

Bézier curves in matrix form:

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

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

\[ = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} \]

\[ M_{\text{Bézier}} \]
Basic properties of Bézier Curves

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

• Convex hull:
  \( \text{Curve is contained within convex hull of control polygon} \)

• Symmetry
  \[ Q(u) \text{ defined by } \{V_0,...,V_n\} \equiv Q(1-u) \text{ defined by } \{V_n,...,V_0\} \]
Bézier Curves

- Curve $Q(u)$ can also be defined by nested interpolation:

$V_i$ are control points
$\{V_0, V_1, ..., V_n\}$ is control polygon
Enforcing Bézier Curve Continuity

- $C^0$: $V_3 = V_4$
- $C^1$: $V_5 - V_4 = V_3 - V_2$
- $C^2$: $V_6 - 2V_5 + V_4 = V_3 - 2V_2 + V_1$
Outline

• Parametric curves
  ◦ Cubic B-Spline
  ◦ Cubic Bézier

➤ Parametric surfaces
  ◦ Bi-cubic B-Spline
  ◦ Bi-cubic Bézier
Parametric Surfaces

- Defined by parametric functions:
  - $x = f_x(u,v)$
  - $y = f_y(u,v)$
  - $z = f_z(u,v)$
Parametric Surfaces

- 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

- 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

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

- Example: ellipsoid

\[
\begin{align*}
  f_x(u,v) &= r_x \cos v \cos u \\
  f_y(u,v) &= r_y \cos v \sin u \\
  f_z(u,v) &= r_z \sin v
\end{align*}
\]

H&B Figure 10.10
Parametric Surfaces

To model arbitrary shapes, 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 $Q(0,0)$, $Q(1,0)$, $Q(0,1)$, and $Q(1,1)$. 
Parametric Patches

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

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

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

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

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

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

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

Watt Figure 6.28
Bézier Patches

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

\[ \mathbf{M}_{\text{Bez}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \]
Bézier Patches

• **Properties:**
  - Interpolates four corner points
  - Convex hull
  - Local control
Surface is composition of many parametric patches
Piecewise Polynomial Parametric Surfaces

Must maintain continuity across seams

Same ideas as parametric splines!

Watt Figure 6.25
Bézier Surfaces

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

- $C^0$ continuity requires aligning boundary curves

Watt Figure 6.26a
Bézier Surfaces

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

- Properties
  - Natural parameterization
  - Guaranteed smoothness
  - Intuitive editing
  - Concise
  - Accurate
  - Efficient display
  - Easy acquisition
  - Efficient intersections
  - Guaranteed validity
  - Arbitrary topology
Parametric Surfaces

• Properties
  ☑ Natural parameterization
  ☑ Guaranteed smoothness
  ☑ Intuitive editing
  ☑ Concise
  ☑ Accurate
    • Efficient display
  ☹ Easy acquisition
  ☹ Efficient intersections
  ☹ Guaranteed validity
  ☹ Arbitrary topology