**CSE167**  
Introduction to Computer Graphics  
Matthias Zwicker  
University of California, San Diego  
Spring 2008

---

### NURBS

- Non uniform rational B-splines
- Generalization of Bézier curves
  - Easier to guarantee smoothness of curve
  - Can represent conic sections (circles, ellipses)

### Rational curves

- Weight causes point to “pull” more (or less)
- With proper points & weights, can do circles

### Curved surfaces

- Described by a 1D series of control points
- A function $x(t)$
- Segments joined together to form a longer curve

---

### Piecewise Cubic Bézier curve

- Parameter in $0 \leq u \leq 3N$

$$ x(u) = \begin{cases} x_0 (1 - 3u), & 0 \leq u \leq 3 \\ x_1 (1 - 3u + 3u^2 - u^3), & 3 \leq u \leq 6 \\ \vdots & \\ x_{N-1} (1 - 3u + 3N - 3N^2 - N^3), & 3N - 3 \leq u \leq 3N \\ x_N (1 - 3u + 3N - 3N^2 + N^3), & 3N \leq u \leq 3N \\ \end{cases} $$

$$ x(u) = \sum_{i=0}^{N} b_i(u) w_i p_i $$

**Rational curve**

**B-spline blending functions** $b_i(u)$

---

### Today

**Surfaces**
- Bilinear patch
- Bicubic Bézier patch
- Advanced surface modeling

---

**Curved surfaces**

**Curves**
- Described by a 1D series of control points
- A function $x(t)$
- Segments joined together to form a longer curve

**Surfaces**
- Described by a 2D mesh of control points
- Parameters have two dimensions (two dimensional parameter domain)
- A function $x(u,v)$
- **Patches** joined together to form a bigger surface
Parametric surface patch

- $x(u,v)$ describes a point in space for any given $(u,v)$ pair
- $u,v$ each range from 0 to 1

2D parameter domain

Parametric surface patch

- $x(u,v)$ describes a point in space for any given $(u,v)$ pair
- $u,v$ each range from 0 to 1

2D parameter domain

Parametric curves

- For fixed $u_0$, have a $v$ curve $x(u_0,v)$
- For fixed $v_0$, have a $u$ curve $x(u,v_0)$
- For any point on the surface, there are a pair of parametric curves that go through point

Tangents

- The tangent to a parametric curve is also tangent to the surface
- For any point on the surface, there are a pair of (parametric) tangent vectors
- Note: not necessarily perpendicular to each other

Surface Normal

- Cross product of the two tangent vectors
- Order matters!

Tangents

- Notation:
  - The tangent along a $u$ curve, AKA the tangent in the $u$ direction, is written as: $\frac{\partial x}{\partial u}(u,v)$ or $\frac{\partial x}{\partial u}(u,v)$ or $x_u(u,v)$
  - The tangent along a $v$ curve, AKA the tangent in the $v$ direction, is written as: $\frac{\partial x}{\partial v}(u,v)$ or $\frac{\partial x}{\partial v}(u,v)$ or $x_v(u,v)$
- Note that each of these is a vector-valued function:
  - At each point $x(u,v)$ on the surface, we have tangent vectors $\frac{\partial x}{\partial u}(u,v)$ and $\frac{\partial x}{\partial v}(u,v)$

Surface Normal

- Cross product of the two tangent vectors
- Order matters!

Bilinear patch

- Control mesh with four points $p_0$, $p_1$, $p_2$, $p_3$
- Compute $x(u,v)$ using a two-step construction
Bilinear patch (step 1)

- For a given value of $u$, evaluate the linear curves on the two $u$-direction edges
- Use the same value $u$ for both:
  \[
  q_0 = \text{Lerp}(u, p_0, p_1) \\
  q_1 = \text{Lerp}(u, p_2, p_3)
  \]

Bilinear patch (step 2)

- Consider that $q_0, q_1$ define a line segment
- Evaluate it using $v$ to get $x$
  \[
  x = \text{Lerp}(v, q_0, q_1)
  \]

Bilinear patch

- Combining the steps, we get the full formula
  \[
  x(u, v) = \text{Lerp}(v, \text{Lerp}(u, p_0, p_1), \text{Lerp}(u, p_2, p_3))
  \]

Bilinear patch

- Try the other order
- Evaluate first in the $v$ direction
  \[
  r_0 = \text{Lerp}(v, p_0, p_2) \\
  r_1 = \text{Lerp}(v, p_1, p_3)
  \]

Bilinear patch

- Consider that $r_0, r_1$ define a line segment
- Evaluate it using $u$ to get $x$
  \[
  x = \text{Lerp}(u, r_0, r_1)
  \]

Bilinear patch

- The full formula for the $v$ direction first:
  \[
  x(u, v) = \text{Lerp}(u, \text{Lerp}(v, p_0, p_1), \text{Lerp}(v, p_2, p_3))
  \]
Bilinear patch

- It works out the same either way!

\[ x(u, v) = \text{Lerp}(v, \text{Lerp}(u, p_0, p_1), \text{Lerp}(u, p_0, p_1)) \]

\[ x(u, v) = \text{Lerp}(u, \text{Lerp}(v, p_0, p_1), \text{Lerp}(v, p_0, p_1)) \]

Bilinear patches

- Weighted sum of control points

\[ x(u, v) = (1 - u)(1 - v)p_0 + u(1 - v)p_1 + (1 - u)v + uvp_3 \]

- Bilinear polynomial

\[ x(u, v) = (p_0 - p_1,d_1)d_2 + (p_2 - p_3)d_3 + (p_4 - p_5)d_4 + d_5 \]

- Matrix form exists, too

Properties

- Interpolates the control points
- The boundaries are straight line segments
- If all 4 points of the control mesh are co-planar, the patch is flat
- If the points are not coplanar, get a curved surface
  - saddle shape, AKA hyperbolic paraboloid
- The parametric curves are all straight line segments!
  - a (doubly) ruled surface: has (two) straight lines through every point

- Not terribly useful as a modeling primitive

Today

Surfaces
- Bilinear patch
- Bicubic Bézier patch
- Advanced surface modeling

Bicubic Bézier patch

- Grid of 4x4 control points, \( p_0 \) through \( p_{15} \)
- Four rows of control points define Bézier curves along \( u \)
  \( p_0, p_2, p_4, p_6; p_1, p_3, p_5, p_7; p_2, p_4, p_6, p_8; p_3, p_5, p_7, p_9 \)
- Four columns define Bézier curves along \( v \)
  \( p_0, p_4, p_8, p_{12}; p_1, p_5, p_9, p_{13}; p_2, p_6, p_{10}, p_{14}; p_3, p_7, p_{11}, p_{15} \)
**Bézier patch (step 1)**
- Evaluate four $u$-direction Bézier curves at $u$
- Get points $q_0, q_1$

$$
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
$$

**Bézier patch (step 2)**
- Points $q_0, q_1$ define a Bézier curve
- Evaluate it at $v$

$$
x(u, v) = \text{Bez}(v, q_0, q_1, q_2)
$$

**Bézier patch**
- Same result in either order (evaluate $u$ before $v$ or vice versa)

$$
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
q_u = \text{Bez}(p_0, p_1, p_2, p_3)
\text{x}(u, v) = \text{Bez}(v, q_0, q_1, q_2)
\text{x}(u, v) = \text{Bez}(v, q_0, q_1, q_2)
$$

**Tensor product formulation**
- Corresponds to weighted average formulation
- Construct two-dimensional weighting function as product of two one-dimensional functions

$$
x(u, v) = \sum_i \sum_j p_{ij} B_i(u) B_j(v)
$$
- Bernstein polynomials $B_i, B_j$ as for curves

**Properties**
- Convex hull: any point on the surface will fall within the convex hull of the control points
- Interpolates 4 corner points
- Approximates other 12 points, which act as “handles”
- The boundaries of the patch are the Bézier curves defined by the points on the mesh edges
- The parametric curves are all Bézier curves

**Tangents of Bézier patch**
- Remember parametric curves $x(u, v)$, $x(u_0, v)$ where $u_0, u_1$ is fixed
- Tangents to surface = tangents to parametric curves
- Tangents are partial derivatives of $x(u, v)$
- Normal is cross product of the tangents
**Tangents of Bézier patch**

\[ q_0 = \text{Bez}(u, p_0, p_1, p_2, p_3) \]
\[ q_1 = \text{Bez}(u, p_4, p_5, p_6, p_7) \]
\[ q_2 = \text{Bez}(u, p_8, p_9, p_{10}, p_{11}) \]
\[ q_3 = \text{Bez}(u, p_{12}, p_{13}, p_{14}, p_{15}) \]

\[ \frac{\partial x}{\partial u}(u, v) = \text{Be}^{\prime}(v, q_0, q_1, q_2, q_3) \]

**Tessellating a Bézier patch**

- Uniform tessellation is most straightforward
  - Evaluate points on a grid of \( u, v \) coordinates
  - Compute tangents at each point, take cross product to get per-vertex normal
  - Draw triangle strips (several choices of direction)
- Adaptive tessellation/recursive subdivision
  - Potential for “cracks” if patches on opposite sides of an edge divide differently
  - Tricky to get right, but can be done

**Piecewise Bézier surface**

- Lay out grid of adjacent meshes of control points
- For \( C^0 \) continuity, must share points on the edge
  - Each edge of a Bézier patch is a Bézier curve based only on the edge mesh points
  - So if adjacent meshes share edge points, the patches will line up exactly
  - But we have a crease...

**C\(^1\) continuity**

- We want the parametric curves that cross each edge to have \( C^1 \) continuity
  - So the handles must be equal-and-opposite across the edge:

**Modeling with Bézier patches**

- Original Utah teapot specified as Bézier Patches

**Today**

Surfaces
- Bilinear patch
- Bicubic Bézier patch
- Advanced surface modeling
**Advanced surface modeling**

- B-spline/NURBS patches
- For the same reason as using B-spline/NURBS curves
  - More flexible (can model spheres)
  - Better mathematical properties, **continuity**

**Trim curves:** cut away part of surface
- Implement as part of tessellation/rendering

**Modeling headaches**

- Original Teapot isn’t “watertight”
  - spout & handle intersect with body
  - no bottom
  - hole in spout
  - gap between lid and body

**NURBS surfaces are flexible**
- Conic sections
- Can blend, merge, trim...

*But*
- Any surface will be made of quadrilateral patches (quadrilateral topology)

**Quadrilateral topology**

Makes it hard to
- join or abut curved pieces
- build surfaces with awkward topology or structure

**Subdivision surfaces**

- Arbitrary mesh of control points, not quadrilateral topology
  - No global \( u, v \) parameters
- Can make surfaces with arbitrary topology or connectivity
- Work by recursively subdividing mesh faces
  - Per-vertex annotation for weights, corners, creases
- Used in particular for character animation
  - One surface rather than collection of patches
  - Can deform geometry without creating cracks
Next time

- Advanced shader programming