Today

Curves
  • NURBS

Surfaces
  • Parametric surfaces
  • Bilinear patch
  • Bicubic Bézier patch
  • Advanced surface modeling
Piecewise Bézier curves

- Each segment spans four control points
- Each segment contains four Bernstein polynomials
- Each control point belongs to one Bernstein polynomial

\[
\begin{align*}
\text{Segment} & \quad \text{Bernstein} \quad \text{polynomials} \\
\begin{array}{c}
\begin{array}{c}
\bullet \\
\bullet \\
\bullet \\
\bullet
\end{array}
\end{array} & \quad \begin{array}{c}
\begin{array}{c}
x_0(t) \\
x_1(t) \\
x_2(t) \\
x_3(t)
\end{array}
\end{array}
\end{align*}
\]
B-splines

• Same idea, but different polynomial blending functions

• **Uniform** B-splines have only one type of blending function: B-spline (basis) function $b_i$

• B-spline function of degree $n$ is $C^{n-1}$ continuous

• **Local support**, at each point $u$ exactly $n+1$ functions are non-zero
B-splines

- Weighted average of control points $p_i$ using B-spline functions $b_i(u)$

$$x(u) = \sum b_i(u)p_i$$

- Positive, partition of unity $\Rightarrow$ convex hull property

- Matrix form (note different basis matrix; caution: last lecture, matrices were transposed)

$$x(u) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_i \\ p_{i+1} \\ p_{i+2} \\ p_{i+3} \end{bmatrix}$$

where $t = u - i$ and $i = \left\lfloor u \right\rfloor$
B-splines

- Widely used for curve and surface modeling

- Advantages over Bézier curves
  - Built-in continuity
  - Local support: curve only affected by nearby control points
Generalization: NURBS

http://en.wikipedia.org/wiki/Non-uniform_rational_B-spline

• Non-Uniform Rational B-splines

• Interactive explanation
  http://www.ibiblio.org/e-notes/Splines/nurbs.html

• NOTE: notation now uses $t$ instead of $u$ for curve parameter
Non-uniform B-splines

Knot vector

- Defines B-spline bases functions
- Uniform B-spline bases and Bernstein polynomials are *special cases* for specific knot vectors
Knot vector

- Knot vector is vector of locations \( \{ t_j \} \) on the \( t \) axis
  - B-spline function of degree \( n \) uses \( n+2 \) knots
- (Uniform) B-splines use a uniform knot vector \( t_j = j \)
- Nonuniform B-splines use an arbitrary knot vector

http://ibiblio.org/e-notes/Splines/basis.html
Nonuniform B-spline bases

Construction using knot vector

- Recursive

- Generate higher order bases step by step from lower order bases

- Can prove
  - Partition of unity (i.e., convex hull property)
  - Built-in continuity
Recursive construction

Nonuniform, linear B-spline bases

Linear weighting function

Multiply & add

Quadratic B-spline basis

Nonuniform knot vector
Recursive construction

Recipe

• Input: two neighboring basis functions of degree $n$
  - Multiply basis functions with linear weighting functions (one increasing, one decreasing)
  - Add

• Output: one basis function of degree $n+1$
For your reference...

- Recursive definition of non-uniform B-spline basis functions $b_{j,n}$
  - Function $b_{j,n}$ has degree $n$
  - Knot vector \{t_j\}

\[
b_{j,0}(t) := \begin{cases} 
1 & \text{if } t_j \leq t < t_{j+1} \\
0 & \text{otherwise}
\end{cases}
\]

Basis functions of degree 0

\[
b_{j,n}(t) := \frac{t - t_j}{t_{j+n} - t_j} b_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} b_{j+1,n-1}(t).
\]

Recursive definition of higher order functions

http://en.wikipedia.org/wiki/B-spline
Special cases

- **Uniform** B-splines have knot vector $t_j=j$
- **Cubic Bézier curves** $\{t_j\} = [0,0,0,0,1,1,1,1,1]$
  - Can make corners ($C^1$ discontinuity)
  - Allows mixing interpolating (e.g. at endpoints) and approximating

Bézier curve as B-spline with nonuniform knot vector

http://www.ibiblio.org/e-notes/Splines/basis.html
Generalization: NURBS

http://en.wikipedia.org/wiki/Non-uniform_rational_B-spline

• **Non-Uniform Rational B-splines**

• Interactive explanation
  
  http://www.ibiblio.org/e-notes/Splines/nurbs.html

• **NOTE:** notation now uses $t$ instead of $u$ for curve parameter
Rational curves

• Big drawback of all polynomial curves
  - Can’t make circles, ellipses, nor arcs, nor conic sections

• Rational B-spline
  - Add a weight to each control point $i$
  - Control points with homogeneous coordinates $w_i$

\[
x(u) = \sum_i b_i(u) p_i
\]

Polynomial curve (b-spline, Bézier)

\[
x(u) = \frac{\sum_i b_i(u) w_i p_i}{\sum_i b_i(u) w_i}
\]

Rational curve
Not polynomial any more!
Rational curves

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

- Can generate curves for conic sections (circles, ellipses, etc.) with appropriate weights

- Need extra user interface to adjust the weights

- Often, hand-drawn curves are unweighted

http://en.wikipedia.org/wiki/Conic_section
NURBS

- Math is more complicated
  - Knot vectors
  - Rational functions

- Very widely used for curve and surface modeling
  - Supported by virtually all 3D modeling tools
  - Open source modeling tool: [http://www.blender.org](http://www.blender.org)

- Techniques for cutting, inserting, merging, revolving, etc...

- Applets
  - [http://ibiblio.org/e-notes/Splines/Intro.htm](http://ibiblio.org/e-notes/Splines/Intro.htm)
  - [http://www.gris.uni-tuebingen.de/edu/projects/grdev/doc/html/etc/AppletIndex_en.html](http://www.gris.uni-tuebingen.de/edu/projects/grdev/doc/html/etc/AppletIndex_en.html)
Today

Curves
- NURBS

Surfaces
- Parametric 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

- 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 is one 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
Tangents

Notation

• Tangent along u direction

\[ \frac{\partial x}{\partial u}(u, v) \quad \text{or} \quad \frac{\partial}{\partial u}x(u, v) \quad \text{or} \quad x_u(u, v) \]

• Tangent along v direction

\[ \frac{\partial x}{\partial v}(u, v) \quad \text{or} \quad \frac{\partial}{\partial v}x(u, v) \quad \text{or} \quad x_v(u, v) \]

• Tangents are vector valued functions, i.e., vectors!
Surface normal

- Cross product of the two tangent vectors
  \[ \mathbf{x}_u(u, v) \times \mathbf{x}_v(u, v) \]
- Order matters (determines normal orientation)
- Usually, want unit normal
  - Need to normalize by dividing through length
Today

Curves
  • NURBS

Surfaces
  • Parametric surfaces
  • Bilinear patch
  • Bicubic Bézier patch
  • Advanced surface modeling
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

$$r_0 = \text{Lerp}(v, p_0, p_2) \quad r_1 = \text{Lerp}(v, p_1, p_3)$$
**Bilinear patch**

- Consider that \( r_0, r_1 \) define a line segment
- Evaluate it using \( u \) to get \( x \)

\[
x = Lerp(u, r_0, r_1)
\]
Bilinear patch

• The full formula for the $v$ direction first:

$$x(u, v) = \text{Lerp}(u, \text{Lerp}(v, p_0, p_2), \text{Lerp}(v, p_1, p_3))$$
Bilinear patch

- It works out the same either way!

\[ x(u, v) = \text{Lerp}(v, \text{Lerp}(u, p_0, p_1), \text{Lerp}(u, p_2, p_3)) \]

\[ x(u, v) = \text{Lerp}(u, \text{Lerp}(v, p_0, p_2), \text{Lerp}(v, p_1, p_3)) \]
Bilinear patch

• Visualization
Bilinear patches

• Weighted sum of control points

\[ x(u, v) = (1-u)(1-v)p_0 + u(1-v)p_1 + (1-u)v p_2 + uv p_3 \]

• Bilinear polynomial

\[ x(u, v) = (p_0 - p_1 - p_2 + p_3)uv + (p_1 - p_0)u + (p_2 - p_0)v + p_0 \]

• 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

Curves

• NURBS

Surfaces

• Parametric 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 \( u \)
  - \( p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7; p_8, p_9, p_{10}, p_{11}; p_{12}, p_{13}, p_{14}, p_{15} \)
- Four columns define Bézier curves along \( v \)
  - \( p_0, p_4, p_8, p_{12}; p_1, p_5, p_9, p_{13}; p_2, p_6, p_{10}, p_{14}; p_3, p_7, p_{11}, p_{15} \)
Bicubic Bézier patch (step 1)

- Evaluate four $u$-direction Bézier curves at $u$
- Get intermediate points $q_0 \ldots q_3$

$$q_0 = \text{Bez}(u, p_0, p_1, p_2, p_3)$$
$$q_1 = \text{Bez}(u, p_4, p_5, p_6, p_7)$$
$$q_2 = \text{Bez}(u, p_8, p_9, p_{10}, p_{11})$$
$$q_3 = \text{Bez}(u, p_{12}, p_{13}, p_{14}, p_{15})$$
Bicubic Bézier patch (step 2)

- Points $q_0 \ldots q_3$ define a Bézier curve
- Evaluate it at $v$

$$x(u, v) = \text{Bez}(v, q_0, q_1, q_2, q_3)$$
Bicubic Bézier patch

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

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

$r_0 = Bez(v, p_0, p_4, p_8, p_{12})$
$r_1 = Bez(v, p_1, p_5, p_9, p_{13})$
$r_2 = Bez(v, p_2, p_6, p_{10}, p_{14})$
$r_3 = Bez(v, p_3, p_7, p_{11}, p_{15})$

$x(u, v) = Bez(u, q_0, q_1, q_2, q_3)$
$x(u, v) = Bez(u, r_0, r_1, r_2, r_3)$
Tensor product formulation

- Corresponds to weighted average formulation

- Construct two-dimensional weighting function as product of two one-dimensional functions
  - Bernstein polynomials $B_i, B_j$ as for curves

$$x(u, v) = \sum_i \sum_j p_{i,j} B_i(u) B_j(v)$$

- Same tensor product construction applies to higher order Bézier and NURBS surfaces
Bicubic Bézier patch: 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 $\mathbf{x}(u,v_0)$, $\mathbf{x}(u_0,v)$ where $v_0, u_0$ is fixed
- Tangents to surface = tangents to parametric curves
- Tangents are partial derivatives of $\mathbf{x}(u,v)$
- Normal is cross product of the tangents
Tangents of Bézier patch

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

\[ \frac{\partial x}{\partial v}(u, v) = Bez'(v, q_0, q_1, q_2, q_3) \]

\[ r_0 = Bez(v, p_0, p_4, p_8, p_{12}) \]
\[ r_1 = Bez(v, p_1, p_5, p_9, p_{13}) \]
\[ r_2 = Bez(v, p_2, p_6, p_{10}, p_{14}) \]
\[ r_3 = Bez(v, p_3, p_7, p_{11}, p_{15}) \]

\[ \frac{\partial x}{\partial u}(u, v) = Bez'(u, r_0, r_1, r_2, r_3) \]
Tessellating a Bézier patch

- **Uniform tessellation** is most straightforward
  - Evaluate points on uniform 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…

Grid of control points

Piecewise Bézier surface
Want parametric curves that cross each edge to have $C^1$ continuity

- Handles must be equal-and-opposite across edge
Modeling with Bézier patches

• Original Utah teapot specified as Bézier Patches
  [Link to Wikipedia page](http://en.wikipedia.org/wiki/Utah_teapot)
Today

Curves
- NURBS

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

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

4th order NURBS patch

http://de.wikipedia.org/wiki/Non-Uniform_Rational_B-Spline
Modeling headaches

• Original Teapot is not “watertight”
  

  - Spout & handle intersect with body
  - No bottom
  - Hole in spout
  - Gap between lid and body
Modeling headaches

NURBS surfaces are flexible

- Conic sections
- Can blend, merge, trim...

...but

- Any surface will be made of quadrilateral patches (quadrilateral topology)
  - Because of tensor product formulation
  - Grid of “horizontal” and “vertical” curves
Quadrilateral topology

Makes it hard to

• join or abut curved pieces

• build surfaces with awkward topology or structure
Trim curves

- Cut away part of surface
- Define "holes" with trim curves in $u/v$ domain
- Tessellation uses trim curve to define surface
- Still hard to fit different parts together
Subdivision surfaces

• Goal
  - Create smooth surfaces from small number of control points, like splines
  - More flexibility for the topology of the control points (not restricted to quadrilateral grid)

• Idea
  - Start with initial coarse polygon mesh
  - Create smooth surface recursively by
    1. Splitting (subdividing) mesh into finer polygons
    2. Smoothing the vertices of the polygons
    3. Repeat from 1.
Subdivision surfaces

http://en.wikipedia.org/wiki/Catmull%E2%80%93Clark_subdivision_surface

Input mesh → Subdivision & smoothing → Subdivision & smoothing → Subdivision & smoothing

Limit surface
Subdivision schemes

• Various schemes available to subdivide and smooth

Doo-Sabin

http://en.wikipedia.org/wiki/Doo%E2%80%93Sabin_subdivision_surface

Loop

http://en.wikipedia.org/wiki/Loop_subdivision_surface

• All provide certain guarantees for smoothness of limit surface
Loop subdivision

- **Subdivision**
  - Split each triangle into four

- **Smoothing**
  - New vertex positions as weighted average of neighbors
  - Different cases

Cases for $\beta$:

\[ \beta = \begin{cases} 
\frac{3}{8n} & n > 3 \\
\frac{3}{16} & n = 3 
\end{cases} \]
Subdivision surfaces

• Arbitrary mesh of control points
• Arbitrary topology or connectivity
  - Not restricted to quadrilateral topology
  - No global $u,v$ parameters
• Work by recursively subdividing mesh faces
• Used in particular for character animation
  - One surface rather than collection of patches
  - Can deform geometry without creating cracks
Next time

- Implementing subdivision surfaces
- More shaders