CS 430
Computer Graphics

B-Splines and NURBS

Week 5, Lecture 9

David Breen, William Regli and Maxim Peysakhov
Department of Computer Science
Drexel University

Outline

• Types of Curves
  – Splines
  – B-splines
  – NURBS
• Knot sequences
• Effects of the weights

Splines

• Popularized in late 1960s in US Auto industry (GM)
  – R. Riesenfeld (1972)
  – W. Gordon
• Origin: the thin wood or metal strips used in building/ship construction
• Goal: define a curve as a set of piecewise simple polynomial functions connected together

Natural Splines

• Mathematical representation of physical splines
• $C^2$ continuous
• Interpolate all control points
• Have Global control (no local control)

B-splines: Basic Ideas

• Similar to Bézier curves
  – Smooth blending function times control points
• But:
  – Blending functions are non-zero over only a small part of the parameter range (giving us local support)
  – When nonzero, they are the “concatenation” of smooth polynomials. (They are piecewise!)

B-spline: Benefits

• User defines degree
  – Independent of the number of control points
• Produces a single piecewise curve of a particular degree
  – No need to stitch together separate curves at junction points
• Continuity comes for free!
B-splines

- Defined similarly to Bézier curves
  - \( p_i \) are the control points
  - Computed with basis functions (Basis-splines)
    - B-spline basis functions are blending functions
  - Each point on the curve is defined by the blending of the control points
    \( B_i \) is the \( i \)-th B-spline blending function

\[
p(t) = \sum_{i=0}^{m} B_{i,d}(t) p_i
\]

- \( B_i \) is zero for most values of \( t \)

B-spline Blending Functions

\( B_{k,0}(t) \) is a step function that is 1 in the interval
\( B_{k,1}(t) \) spans two intervals and is a piecewise linear function that goes from 0 to 1 (and back)
\( B_{k,2}(t) \) spans three intervals and is a piecewise quadratic that grows from 0 to 1/4, then up to 3/4 in the middle of the second interval, back to 1/4, and back to 0
\( B_{k,3}(t) \) is a cubic that spans four intervals growing from 0 to 1/6 to 2/3, then back to 1/6 and to 0

B-spline: Cox-deBoor Recursion

- Cox-deBoor Algorithm: defines the blending functions for spline curves (not limited to deg 3)
  - curves are weighted avgs of lower degree curves
- Let \( B_{i,d}(t) \) denote the \( i \)-th blending function for a B-spline of degree \( d \), then:

\[
B_{i,d}(t) = \begin{cases} 
1 & \text{if } t_k \leq t < t_{k+1} \\
0 & \text{otherwise}
\end{cases}
\]

\[
B_{i,d}(t) = \frac{t - t_k}{t_{k+d} - t_k} B_{i,d-1}(t) + \frac{t_{k+d} - t}{t_{k+d} - t_{k+1}} B_{i+1,d-1}(t)
\]

B-spline Blending Functions: Example for 2\(^{nd}\) Degree Splines

- Note: can’t define a polynomial with these properties (both 0 and non-zero for ranges)
- Idea: subdivide the parameter space into intervals and build a piecewise polynomial
  - Each interval gets different polynomial function

Transitions at Knots

- As one blending function goes to zero, another smoothly becomes non-zero
Example: Creating a B-spline Curve Segment

Uniform B-splines: Setting the Options

• Specified by
  - \( m+1 \) control points, \( P_0 \ldots P_m \)
  - \( m+1 \) control points, \( P_0 \ldots P_m \)
  - \( m \) cubic polynomial curve segments, \( Q_0 \ldots Q_m \)
  - \( m-1 \) knot points, \( t_2 \ldots t_{m+1} \)
  - segments \( Q_i \) of the B-spline curve are
    • defined over a knot interval \([t_i, t_{i+1}]\)
    • defined by 4 of the control points, \( P_{i-3} \ldots P_i \)
  - segments \( Q_i \) of the B-spline curve are blended together into smooth transitions via (the new & improved) blending functions

B-splines: Knot Selection

• Instead of working with the parameter space \( 0 \leq t \leq 1 \), use \( t_{min} \leq t \leq t_{max} \)
• The knot points
  – joint points between curve segments, \( Q_i \)
  – Each has a knot value
  – \( m-1 \) knots for \( m+1 \) points

Example: Creating a B-spline

\( p(t) = \sum_{i=0}^{m} B_{i,d}(t) P_i \)

- \( m = 9 \)
- 10 control points
- 8 knot points
- 7 segments

B-spline: Knot Sequences

• Even distribution of knots
  – uniform B-splines
  – Curve does not interpolate end points
    • first blending function not equal to 1 at \( t=0 \)
• Uneven distribution of knots
  – non-uniform B-splines
  – Allows us to tie down the endpoints by repeating knot values
    (in Cox-deBoor, \( 0/0=0 \))
  – If a knot value is repeated, it increases the effect (weight) of the blending function at that point
  – If knot is repeated \( d \) times, blending function converges to 1 and the curve interpolates the control point

B-splines: Cox-deBoor Recursion

• Cox-deBoor Algorithm: defines the blending functions for spline curves (not limited to deg 3)
  – curves are weighted avgs of lower degree curves
• Let \( B_{i,d}(t) \) denote the \( i \)-th blending function for a B-spline of degree \( d \), then:

\[
B_{i,d}(t) = \begin{cases} 
1, & \text{if } t_{i} < t < t_{i+1} \\
0, & \text{otherwise}
\end{cases}
\]

\[
B_{i,d}(t) = \begin{cases} 
\frac{t-t_{i}}{t_{i+d}-t_{i}} B_{i,d+1}(t) + \frac{t_{i+d+1}-t}{t_{i+1}-t_{i+1}} B_{i+1,d+1}(t), & \text{if } t_{i} < t < t_{i+1} \\
0, & \text{otherwise}
\end{cases}
\]
Creating a Non-Uniform B-spline: Knot Selection

- Given curve of degree $d=3$, with $m+1$ control points $p_0, \ldots, p_m$
  - first, create $m+d$ knot values
  - use knot values $(0, 0, 0, 1, 2, \ldots, m-2, m-1, m-1, m-1)$ (adding two extra 0's and $m-1$'s)
  - Note
    - Causes Cox-deBoor to give added weight in blending to the first and last points when $t$ is near $t_{min}$ and $t_{max}$

B-splines: Multiple Knots

- Knot Vector $(0.0, 0.0, 3.0, 4.0, 5.0, 6.0, 7.0)$
- Several consecutive knots get the same value
- Changes the basis functions!

B-spline Summary

$p(t) = \sum_{j=0}^{m} B_{i,j}(t) p_j$

Watching Effects of Knot Selection

- 9 knot points (initially)
  - Note: knots are distributed parametrically based on $t$, hence why they "move"
- 10 control points
- Curves have as many segments as they have non-zero intervals in $u$

B-splines: Local Control Property

- Local Control
  - polynomial coefficients depend on a few points
  - moving control point ($p_4$) affects only local curve
  - Why: Based on curve def'n, affected region extends at most 2 knot points away

B-splines: Local Control Property

- Knot
- Control point $x(t)$
**B-splines: Convex Hull Property**

- The effect of multiple control points on a uniform B-spline curve

**B-splines: Continuity**

- Derivatives are easy for cubics
  
  \[ p(u) = \sum c_k u^k \]

- Derivative:
  
  \[ p'(u) = c_1 + 2c_2 u + 3c_3 u^2 \]

  Easy to show \( C^0, C^1, C^2 \)

**B-splines: Setting the Options**

- How to space the knot points?
  - Uniform
    - equal spacing of knots along the curve
  - Non-Uniform

- Which type of parametric function?
  - Rational
    - \( x(t), y(t), z(t) \) defined as ratio of cubic polynomials
  - Non-Rational

**NURBS**

- At the core of several modern CAD systems
  - I-DEAS, Pro/E, Alpha_1
- Describes analytic and freeform shapes
- Accurate and efficient evaluation algorithms
- Invariant under affine and perspective transformations

**Benefits of Rational Spline Curves**

- Invariant under rotation, scale, translation, perspective transformations
  - transform just the control points, then regenerate the curve
  - (non-rationals only invariant under rotation, scale and translation)
- Can precisely define the conic sections and other analytic functions
  - conics require quadratic polynomials
  - conics only approximate with non-rationals

**NURBS**

*Non-uniform Rational B-splines: NURBS*

- Basic idea: four dimensional non-uniform B-splines, followed by normalization via homogeneous coordinates
  - If \( P_i = (x, y, z, 1) \), results are invariant wrt perspective projection
- Also, recall in Cox-deBoor, knot spacing is arbitrary
  - knots are close together, influence of some control points increases
  - Duplicate knots can cause points to interpolate
  - e.g. Knots \( \{0, 0, 0, 0, 1, 1, 1, 1\} \) create a Bézier curve
Rational Functions

- Cubic curve segments
  \[ x(t) = \frac{X(t)}{W(t)}, \quad y(t) = \frac{Y(t)}{W(t)}, \quad z(t) = \frac{Z(t)}{W(t)} \]
  where \(X(t), Y(t), Z(t), W(t)\) are all cubic polynomials with control points specified in homogenous coordinates, \([x, y, z, w]\)
- Note: for 2D case, \(Z(t) = 0\)

Rational Functions: Example

- Example:
  - rational function: a ratio of polynomials
  - a rational parameterization in \(u\) of a unit circle in xy-plane:
    \[ x(u) = \frac{1 - u^2}{1 + u^2}, \quad y(u) = \frac{2u}{1 + u^2}, \quad z(u) = 0 \]
  - a unit circle in 3D homogeneous coordinates:
    \[ x(u) = 1 - u^2, \quad y(u) = 2u, \quad z(u) = 0, \quad w(u) = 1 + u^2 \]

NURBS: Notation Alert

- Depending on the source/reference
  - Blending functions are either \(B_{i,d}(u)\) or \(N_{i,d}(u)\)
  - Parameter variable is either \(u\) or \(t\)
  - Curve is either \(C\) or \(P\) or \(Q\)
  - Control Points are either \(P_i\) or \(B_i\)
  - Variables for order, degree, number of control points etc are frustratingly inconsistent
    - \(k, i, j, m, n, p, L, d, \ldots\)

NURBS

- A \(d\)-th degree NURBS curve \(C\) is def'd as:
  \[ C(u) = \frac{\sum_{i=0}^{n} w_i B_{i,d}(u) P_i}{\sum_{i=0}^{n} w_i B_{i,d}(u)} \]
  Where
  - control points, \(P_i\)
  - \(d\)-th degree B-spline blending functions, \(B_{i,d}(u)\)
  - the weight, \(w_i\), for control point \(P_i\)
  (when all \(w_i = 1\), we have a B-spline curve)

Observe: Weights Induce New Rational Basis Functions, \(R\)

- Setting:
  \[ R_i(u) = \frac{w_i B_{i,d}(u)}{\sum_{j=0}^{n} w_j B_{j,d}(u)} \]
  Allows us to write:
  \[ C(u) = \sum_{i=0}^{n} R_i(u) P_i \]
  Where \(R_i(u)\) are rational basis functions
  - piecewise rational basis functions on \(u \in [0, 1]\)
  - weights are incorporated into the basis fctns
Geometric Interpretation of NURBS

- With Homogeneous coordinates, a rational \(n\)-D curve is represented by polynomial curve in \((n+1)\)-D
- Homogeneous 3D control points are written as: 
  \[ P_i^w = w_i x_i, w_i y_i, w_i z_i, w_i \]
  in 4D where \( w \neq 0 \)
- To get \( P_i \), divide by \( w_i \)
  - a perspective transform with center at the origin
- Note: weights can allow final curve shape to go outside the convex hull (i.e. negative \( w \))

NURBS: Examples

- Unif. Knot Vector
- Non-Unif. Knot Vector

The Effects of the Weights

- \( w_i \) of \( P_i \) effects only the range \([u_i, u_{i+k+1}]\)
- If \( w_i = 0 \) then \( P_i \) does not contribute to \( C \)
- If \( w_i \) increases, point \( B \) and curve \( C \) are pulled toward \( P_i \) and pushed away from \( P_j \)
- If \( w_i \) decreases, point \( B \) and curve \( C \) are pushed away from \( P_i \) and pulled toward \( P_j \)
- If \( w_i \) approaches infinity then \( B \) approaches 1
  and \( B_i \to P_i \), if \( u \) in \([u_i, u_{i+k+1}]\)
Programming assignment 3

• Input PostScript-like file containing polygons
• Output B/W PBM
• Implement viewports
• Use Sutherland-Hodgman intersection for polygon clipping
• Implement scanline polygon filling. (You cannot use flood filling)