CS 536
Computer Graphics

NURBS Drawing
Week 3, Lecture 6

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

• Conic Sections via NURBS
• Knot insertion algorithm
• The de Boor’s algorithm
  – for B-Splines
  – for NURBS
• Oslo Algorithm
• Barycentric Coordinates
• Discussion of homework #2
Conic Sections via NURBS

- Obtained via projection of the 3D parabola onto a plane

- **Note:**
  - 3D Case: rational curve is a 4D object
  - 2D Case: rational curve is a 3D object
  - assign \( w \) to each control point
Conic Sections via NURBS

- Define the curve with three control points
- Weights of first/last control points are 1
- For center control point
  - \( w<1 \) gives an ellipse
  - \( w>1 \) gives a hyperbola
  - \( w=1 \) gives a parabola
  - Knot vector is \( \{0.0, 0.0, 0.0, 1.0, 1.0, 1.0\} \)

Conic Sections via NURBS: A Circular Arc

• The two sides of the control polygon are of equal length
• The chord connecting the first and last control points meets each leg at an angle $\theta$ equal to half the angular extent of the desired arc (for instance, 30° for a 60° arc)
• The weight of the inner control point is equal to the cosine of $\theta$
• Knot vector is $\{0.0, 0.0, 0.0, 1.0, 1.0, 1.0\}$

Conic Sections via NURBS: A Circle

• What if we need an arc of >180°?

• Idea:
  – Use multiple 90° or 120° arcs
  – Stitch them together with knots

Example:
3 arcs of 120°

Conic Sections via NURBS

Example:
4 arcs of 90°

\[ B_3 = \{-1,1,\frac{\sqrt{2}}{2}\} \]
\[ B_4 = \{-1,0,1\} \]
\[ B_5 = \{-1,-1,\frac{\sqrt{2}}{2}\} \]
\[ B_6 = \{0,-1,1\} \]
\[ B_7 = \{1,-1,\frac{\sqrt{2}}{2}\} \]
\[ B_2 = \{0,1,1\} \]
\[ B_1 = \{1,1,\frac{\sqrt{2}}{2}\} \]

knots = \{0,0,0,\frac{1}{4},\frac{1}{4},\frac{1}{2},\frac{1}{2},\frac{3}{4},\frac{3}{4},1,1,1\} \}

Knot Insertion

• Issue: More control points mean more control
• How do we add more points and keep same curve?

Knot Insertion

• Basic Approach
  – Decide where we’d like to tweak the curve
  – Add a new knot
  – Find affected $d-1$ control points
  – Replace it with $d$ new control points

Example:
New knot at $u=2.6$
Knot Insertion

• Given: \(n+1\) control points \((P_0, P_1, ..., P_n)\), a knot vector of \(m+1\) knots \(U = \{u_0, u_1, ..., u_m\}\) and a degree \(d\) B-spline curve \(C(u)\).

• Insert a new knot \(t\) into the knot vector without changing the shape of the curve.

• If \(t\) lies in knot span \([u_k, u_{k+1})\), only the basis functions for \((P_k, ..., P_{k-d})\) are non-zero.

• Find \(d\) new control points \(Q_k\) on edge \(P_{k-1}P_k\), \(Q_{k-1}\) on edge \(P_{k-2}P_{k-1}\), ..., and \(Q_{k-d+1}\) on edge \(P_{k-d}P_{k-d+1}\)

• All other control points are unchanged.

• Note that \(d-1\) control points of the original control polyline are removed and replaced with \(d\) new control points.

See http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/single-insertion.html
Knot Insertion Algorithm

• Create new control point

\[ Q_j = (1 - \alpha_j)P_{j-1} + \alpha_jP_j \]

• Where \( \alpha \) is defined as

\[ \alpha_j = \frac{t - u_j}{u_{j+d} - u_j} \]

See http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/single-insertion.html
Properties of Knot Insertion

• Increasing the multiplicity of a knot decreases the number of non-zero basis functions at this knot
• At a knot of multiplicity \( d \), there will be only one non-zero basis function
• Corresponding point on the curve \( p(u) \) is affected by \textit{exactly one} control point \( p_i \)
• In fact \( p(u) \) is \( p_i \)!
The de Boor Algorithm

• Generalization of de Casteljau's algorithm
• It provides a fast and numerically stable way for finding a point on a B-spline curve
• Observation: if a knot $u$ is inserted $d$ times to a B-spline, then $p(u)$ is the point on the curve.
• Idea: We just simply insert $u$ $d$ times and the last point is $p(u)$!

See http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/B-spline/de-Boor.html
The de Boor Algorithm
The de Boor Algorithm
The de Boor Algorithm
The de Boor Algorithm
The de Boor Algorithm
De Boor’s Algorithm

If $u$ lies in $[u_k, u_{k+1})$ and $u \neq u_k$, let $h = d$
If $u = u_k$ and $u_k$ is a knot of multiplicity $s$, let $h = d - s$
Copy the affected control points $p_{k-s}$, $p_{k-s-1}$, ..., $p_{k-d+1}$, $p_{k-d}$
to a new array and rename them as $p_{k-s,0}$, $p_{k-s-1,0}$, ..., $p_{k-d+1,0}$

for $r := 1$ to $h$ do
    for $i := k-d+r$ to $k-s$ do
        {
            Let $a_{i,r} = (u - u_i) / (u_{i+d-r+1} - u_i)$
            Let $p_{i,r} = (1 - a_{i,r}) p_{i-1,r-1} + a_{i,r} p_{i,r-1}$
        }

$p_{k-s,d-s}$ is the point $p(u)$. 
De Boor’s Algorithm (cont)

\[
\text{for } u := 0 \text{ to } u_{\text{max}} \text{ do }
\{
\ldots
\text{for } r := 1 \text{ to } h \text{ do }
\text{for } i := k-p+r \text{ to } k-s \text{ do }
\{
\text{Let } a_{i,r} = \frac{(u - u_i)}{(u_i + p - r + 1 - u_i)}
\text{Let } p_{i,r} = (1 - a_{i,r}) p_{i-1,r-1} + a_{i,r} p_{i,r-1}
\}
\text{p}_{k-s,p-s} \text{ is the point } p(u).
\}
Example of de Boor’s Algorithm

Degree 3 B-spline curve \((i.e., \, d = 3)\)
Defined by seven control points \(p_0, \ldots, p_6\)
And knot vector:

<table>
<thead>
<tr>
<th>(u_0 = u_1 = u_2 = u_3)</th>
<th>(u_4)</th>
<th>(u_5)</th>
<th>(u_6)</th>
<th>(u_7 = u_8 = u_9 = u_{10})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
<td>1</td>
</tr>
</tbody>
</table>

\(u = 0.4\)

\[a_{4,1} = \frac{(u - u_4)}{(u_{4+3} - u_4)} = 0.2\]
\[a_{3,1} = \frac{(u - u_3)}{(u_{3+3} - u_3)} = \frac{8}{15} = 0.53\]
\[a_{2,1} = \frac{(u - u_2)}{(u_{2+3} - u_2)} = 0.8\]
\[p_{4,1} = (1 - a_{4,1})p_{3,0} + a_{4,1}p_{4,0} = 0.8p_{3,0} + 0.2p_{4,0}\]
\[p_{3,1} = (1 - a_{3,1})p_{2,0} + a_{3,1}p_{3,0} = 0.47p_{2,0} + 0.53p_{3,0}\]
\[p_{2,1} = (1 - a_{2,1})p_{1,0} + a_{2,1}p_{2,0} = 0.2p_{1,0} + 0.8p_{2,0}\]

\[a_{4,2} = \frac{(u - u_4)}{(u_{4+3-1} - u_4)} = 0.3\]
\[a_{3,2} = \frac{(u - u_3)}{(u_{3+3-1} - u_3)} = 0.8\]
\[p_{4,2} = (1 - a_{4,2})p_{3,1} + a_{4,2}p_{4,1} = 0.7p_{3,1} + 0.3p_{4,1}\]
\[p_{3,2} = (1 - a_{3,2})p_{2,1} + a_{3,2}p_{3,1} = 0.2p_{2,1} + 0.8p_{3,1}\]

\[a_{4,3} = \frac{(u - u_4)}{(u_{4+3-2} - u_4)} = 0.6\]
\[p_{4,3} = (1 - a_{4,3})p_{3,2} + a_{4,3}p_{4,2} = 0.4p_{3,2} + 0.6p_{4,2}\]
Similar but Different

**De Casteljau's:**
- Dividing points are computed with a pair of numbers \((1 - u)\) and \(u\) that never change
- Can be used for curve subdivision
- Uses *all* control points

**De Boor's**
- These pairs of numbers are different and depend on the column number and control point number
- Intermediate control points are not sufficient
- \(d-1\) affected control points are involved in the computation
De Boor’s: Curves
Oslo Algorithm

- A subdivision algorithm for B-splines, the basic idea:
- Take the curve with \( m+1 \) control points \( P_0 \) to \( P_m \)
- Insert a knot in any point (0.5 maybe?)
- As a result you will have 2 new points \( P_k' \) and \( P_k'' \)
- Take curves with \( m+1 \) control points \( P_0 \ldots P_k', P_k'' \ldots P_{m-1} \) and \( P_1 \ldots P_k', P_k'' \ldots P_m \)
- Apply procedure recursively on each part
Oslo Algorithm

Recorded from: http://heim.ifi.uio.no/~trondbre/OsloAlgApp.html
Barycentric Coordinates

• By Ceva's Theorem:
  – For any point $K$ inside the triangle $ABC$
  – Consider the existence of masses $w_A$, $w_B$, and $w_C$, placed at the vertices of the triangle
  – Their center of gravity (barycenter) will coincide with the point $K$.

• August Ferdinand Moebius (1790-1868) defined (1827) $w_A$, $w_B$, and $w_C$ as the barycentric coordinates of $K$
• $K = w_A A + w_B B + w_C C$

http://www.cut-the-knot.org/triangle/barycenter.shtml
Properties of Barycentric Coordinates

- Not unique
- Can be generalized to negative masses
- Can be made unique by setting
  \[ w_A + w_B + w_C = 1 \]
- \( w_A = 0 \) for points on BC
- \( w_B = 0 \) for points on AC
- \( w_C = 0 \) on AB

http://www.cut-the-knot.org/triangle/barycenter.shtml
Given P, how can we compute α, β, γ?

• Compute the areas of the opposite subtriangle
  – Ratio with complete area
    \[ \alpha = \frac{A_a}{A}, \quad \beta = \frac{A_b}{A} \quad \gamma = \frac{A_c}{A} \]

Use signed areas for points outside the triangle

Area Ta:
\[ \frac{|(b-P) \times (c-P)|}{2} \]
Calculating the Weights

- Given vertices A, B, C and Centroid K
- What are the weights, \( w_A, w_B, w_C \)?
  \[
  x_K = w_A x_A + w_B x_B + w_C x_C
  \]
  \[
  y_K = w_A y_A + w_B y_B + w_C y_C
  \]
- Substitute \( w_C = 1 - w_A - w_B \)
  \[
  x_K = w_A x_A + w_B x_B + (1 - w_A - w_B) x_C
  \]
  \[
  y_K = w_A y_A + w_B y_B + (1 - w_A - w_B) y_C
  \]
Calculating Weights (cont.)

- Solve for $w_A$ and $w_B$

\[
\begin{align*}
  w_A &= \frac{(x_B - x_C)(y_C - y_K) - (x_C - x_K)(y_B - y_C)}{(x_A - x_C)(y_B - y_C) - (x_B - x_C)(y_A - y_C)} \\
  w_B &= \frac{(x_A - x_C)(y_C - y_K) - (x_C - x_K)(y_A - y_C)}{(x_B - x_C)(y_A - y_C) - (x_A - x_C)(y_B - y_C)}
\end{align*}
\]

- $w_C = 1 - w_A - w_B$
Programming Assignment 2

- Process command-line arguments
- Read in 3D input points and tangents
- Compute tangents at interior input points
- Modify tangents with tension parameter
- Compute Bezier control points for curves defined by each two input points
- Use HW1 code to compute points on each Bezier curve
- Each Bezier curve should be a polyline
- Output points by printing them to the console as an IndexedLineSet with multiple polylines, and control points as spheres in Open Inventor format