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: 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°

Knot Insertion

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

Knot Insertion Algorithm

- Create new control point
  \[ Q_j = (1 - \alpha)P_{j-1} + \alpha P_j \]
- Where \( \alpha \) is defined as
  \[ \alpha_j = \frac{t - u_j}{u_{j+d} - u_j} \]

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 exactly one control point \( p_i \)
- In fact \( p(u) = p_i \)
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
De Boor’s Algorithm

If \( u \) lies in \([u_0, u_{d+1})\) and \( u = u_k \), let \( h = d \)
If \( u = u_k \) and \( u_{k+1} \) is a knot of multiplicity \( s \), let \( h = d - s \)
Copy the affected control points \( p_{u_0}, p_{u_1}, \ldots, p_{u_{h-1}}, p_{u_{d}} \)
to a new array and rename them as \( p_{u_0}, p_{u_1}, \ldots, p_{u_{d}} \)
for \( i := k + r \) to \( k + s \) do
{ 
  \( \alpha_i = (u - u_i) / (u_{i+s+1} - u_i) \)
  \( \beta_i = (1 - \alpha_i) p_{i+1} + \alpha_i p_i \)
}\( p_{u_{k+r,0}} \) is the point \( p(u) \).

Degree 3 B-spline curve (i.e., \( d = 3 \))

Defined by seven control points \( p_0, \ldots, p_6 \)

And knot vector \( u \) at 0, 4

Example of de Boor’s Algorithm

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

Oslo Algorithm

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

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

Properties of Barycentric Coordinates

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

Calculating the Weights

• Given vertices A, B, C and Centroid K
• What are the weights, \( w_A, w_B, w_C? \)

\[
\begin{align*}
    x_K &= w_A x_A + w_B x_B + w_C x_C \\
    y_K &= w_A y_A + w_B y_B + w_C y_C \\
    x_K &= w_A x_A + w_B x_B + (1 - w_A - w_B) x_C \\
    y_K &= w_A y_A + w_B y_B + (1 - w_A - w_B) y_C
\end{align*}
\]

• Substitute \( w_C = 1 - w_A - w_B \)

Calculating Weights (cont.)

• Solve for \( w_A \) and \( w_B \)

\[
\begin{align*}
    w_A &= \frac{(x_B - x_C)(y_C - y_K) - (x_C - x_K)(y_B - y_C)}{(x_A - x_C)(y_B - y_C) - (x_B - x_C)(y_A - y_C)} \\
    w_B &= \frac{(x_A - x_C)(y_C - y_K) - (x_C - x_K)(y_A - y_C)}{(x_B - x_C)(y_A - y_C) - (x_A - x_C)(y_B - y_C)} \\
    w_C &= 1 - w_A - w_B
\end{align*}
\]
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