CS171 HW 5 - NURBS Editor

Julian Panetta
11/6/09
Based on Slides by Tom Raney (11/9/07)
NURBS

• Non-Uniform Rational B-Splines
• Representation of smooth curves ($C^2$!)
• Intuitive, flexible, precise
• Generalization of Bezier paths (uniform, non-rational b-splines)
General Idea

- Control points, \( p_i, i \in \{0, ..., n\} \)
- Parameterized by \( u \in [0, 1] \)
- Curve \( Q(u): u \rightarrow (x, y) \)
  - Blending of control points
How are they blended?

- Cox de Boor’s Algorithm

\[ Q(u) = \sum_{i=0}^{n} p_i N_{i,k}(u) \]

- Control points, \( p_i, i \in \{0, ..., n\} \)

- \((x, y)\) coordinates controlling curve shape

- Parameter \( u \in [0, 1] \)

- \( k = \text{curve order} = \text{degree} + 1 \)

- For cubics, \( k = 3 + 1 = 4 \)
Knots

• Nondecreasing sequence $t_i \in \{0, \ldots, n+k\}$
• All $t_i$ are in the parameter space $([0, 1])$
• Intuitively: specify what portions of the curve the control points affect. For instance:

$$u \notin [t_i, t_{i+k}] \Rightarrow N_{i,k} = 0$$

• (Control $p_i$ only affects curve segment over $[t_i, t_{i+k}]$)
Knots

• Invisible to user of your program!
• Integral part of $N_{i,k}$ computation
• Note, there are $n + k + 1$ knots, or:
  • $\# \text{Knots} = \# \text{Control Points} + \text{Order}$
What’s this magic $N_{i,k}$?

• Recursively defined:

$$N_{i,1}(u) = \begin{cases} 1 & t_i \leq u \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k}(u) = \frac{(u - t_i) N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u) N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}}$$

• To handle repeated knots ($t_i = t_{i+1}$) we must define $0/0 = 0$ or we get an undefined result

• You will need to do this!
Knot Multiplicity

• To get the curve to end at the last control point, you must have endpoints of multiplicity \( k \)

• For cubic splines \((k = 4)\), this means:
  
  • knots = \([0, 0, 0, 0, t_k, \ldots, t_n, 1, 1, 1, 1]\)
Simple Example

- 4 control points
- \( n + k + 1 = 8 \) knots
- knots = [0, 0, 0, 0, 1, 1, 1, 1] for curve to meet endpoints
- This is your initial editor configuration!
I’m not lying about end knot multiplicity

- What happens when we don’t make the end knots of multiplicity \( k \)?
- Take \([0, 0, 0, .25, 1, 1, 1, 1]\) as the knot vector
- Uh-oh!
Drawing Curves

• Keep track of control points and knots

• Choose a step size, $\Delta u$, small enough to get a smooth curve

• Adaptive $\Delta u$ is extra credit!

• Draw short line segments from $Q(u)$ to $Q(u + \Delta u)$
Inserting Control Points

• We don’t want to alter the curve!

• Old control points will have to move...

• Another control point means another knot. Insert a knot with the u-value of the mouse click in the correct position of the knot array!

• Then create the new control point and compute positions:

\[
p'_{n+1} = p_n \quad p'_0 = p_0 \quad a_i = \begin{cases} 
1 & i = 1, \ldots, j - k \\
\frac{t' - t_i}{t_{i+k} - t_i} & i = j - k + 1, \ldots, j \\
0 & i = j + 1, \ldots, n
\end{cases}
\]

\[
p'_i = (1 - a_i)p_{i-1} + a_ip_i
\]

Note: Assignment page had the order of this procedure incorrect until this morning.
Questions?

• Implementation is straightforward.
• I’ve tried to fix all of the old, confusing errors on the assignment page. Please let me know if you find any more, though!
• This was a very practical overview--read through some of the linked references for a deeper discussion of the math!