Foundations of Computer Graphics
(Spring 2010)
COMS 4160, Lecture 7: Curves 2
http://inst.eecs.berkeley.edu/~cs184

To Do
- Submit HW 1 (by tomorrow)!
- Start on HW 2 (cannot be done at last moment)
  - This (and previous) lecture should have all information need
- Start thinking about partners for HW 3 and HW 4
  - Remember though, that HW1, HW2 are done individually
  - Your submission of HW 2 must include partner for HW 3

Outline of Unit
- Bezier curves (last time)
- deCasteljau algorithm, explicit, matrix (last time)
- Polar form labeling (blossoms)
- B-spline curves

Idea of Blossoms/Polar Forms
- (Optional) Labeling trick for control points and intermediate
deCasteljau points that makes thing intuitive
  - E.g. quadratic Bezier curve $F(u)$
  - Define auxiliary function $f(u_1,u_2)$ [number of args = degree]
  - Points on curve simply have $u_1=u_2$ so that $F(u) = f(u,u)$
  - And we can label control points and deCasteljau points not
    on curve with appropriate values of $(u_1,u_2)$

Geometric interpretation: Quadratic
- Points on curve simply have $u_1=u_2$ so that $F(u) = f(u,u)$
- $f$ is symmetric $f(0,1) = f(1,0)$
- Only interpolate linearly between points with one arg different
  - $f(0,u) = (1-u) f(0,0) + u f(0,1)$ Here, interpolate $f(0,0)$ and $f(0,1)=f(1,0)$

Idea of Blossoms/Polar Forms
- Points on curve simply have $u_1=u_2$ so that $F(u) = f(u,u)$
- $f$ is symmetric $f(0,1) = f(1,0)$
- Only interpolate linearly between points with one arg different
  - $f(0,u) = (1-u) f(0,0) + u f(0,1)$ Here, interpolate $f(0,0)$ and $f(0,1)=f(1,0)$
Polar Forms: Cubic Bezier Curve

Why Polar Forms?

- Simple mnemonic: which points to interpolate and how in deCasteljau algorithm
- Easy to see how to subdivide Bezier curve (next) which is useful for drawing recursively
- Generalizes to arbitrary spline curves (just label control points correctly instead of 00 01 11 for Bezier)
- Easy for many analyses (beyond scope of course)

Subdividing Bezier Curves

Drawing: Subdivide into halves (u = ½) Demo: hw2.exe
- Recursively draw each piece
- At some tolerance, draw control polygon
- Trivial for Bezier curves (from deCasteljau algorithm): hence widely used for drawing

Geometric Interpretation: Cubic

Geometrically
Subdivision in deCasteljau diagram

Left part of Bezier curve (000, 00u, 0uu, uuu)
Always left edge of deCasteljau pyramid

Right part of Bezier curve (uuu, 1uu, 11u, 111)
Always right edge of deCasteljau pyramid

Summary for HW 2

- Bezier2 (Bezier discussed last time)
- Given arbitrary degree Bezier curve, recursively subdivide for some levels, then draw control polygon hw2.exe
- Generate deCasteljau diagram; recursively call a routine with left edge and right edge of this diagram
- You are given some code structure; you essentially just need to compute appropriate control points for left, right

DeCasteljau: Recursive Subdivision

Input: Control points $C_i$ with $0 \leq i \leq n$ where $n$ is the degree
Output: $L_i$ for left and right control points in recursion.

1. for ($level = n$ ; $level \geq 0$ ; $level -= 1$) {
2.   if ($level = n$) { /* Initial control points */
3.     $for (i = 0 : i < n : i += 1)$:
4.       $p_{i+1 i+2} = \frac{1}{2} \times (p_{i+1} + 2p_{i+2} + p_{i})$;
5.     }
6.   }$}
7. $for (i = 0 : i < n : i += 1)$:
8.   $L_i = p_i$;
9.   $R_i = p_{i+1}$;

- DeCasteljau (from last lecture) for midpoint
- Followed by recursive calls using left, right parts

Outline of Unit

- Bezier curves (last time)
- deCasteljau algorithm, explicit, matrix (last time)
- Polar form labeling (blossoms)
- B-spline curves

- Not well covered in textbooks (especially as taught here). Main reference will be lecture notes. If you do want a printed ref, handouts from CAGD, Seidel

Bezier: Disadvantages

- Single piece, no local control (move a control point, whole curve changes) hw2.exe
- Complex shapes: can be very high degree, difficult
- In practice, combine many Bezier curve segments
  - But only position continuous at join since Bezier curves interpolate end-points (which match at segment boundaries)
  - Unpleasant derivative (slope) discontinuities at end-points
  - Can you see why this is an issue?

B-Splines

- Cubic B-splines have $C^2$ continuity, local control
- 4 segments / control point, 4 control points/segment
- Knots where two segments join: Knotvector
- Knotvector uniform/non-uniform (we only consider uniform cubic B-splines, not general NURBS)

Knot: $C^2$ continuity Demo: hw2.exe
Polar Forms: Cubic Bspline Curve

- Labeling little different from in Bezier curve
- No interpolation of end-points like in Bezier
- Advantage of polar forms: easy to generalize

Uniform knot vector: \(-2, -1, 0, 1, 2, 3\)
Labels correspond to this

deCasteljau: Cubic B-Splines

- Easy to generalize using polar-form labels
- Impossible remember without

Explicit Formula (derive as exercise)

\[ F(u) = [u^3, u^2, u, 1] M \]

Summary of HW 2

- BSpline Demo `hw2.exe`
- Arbitrary number of control points / segments
  - Do nothing till 4 control points (see demo)
  - Number of segments = # cpts – 3
- Segment A will have control pts A,A+1,A+2,A+3
- Evaluate Bspline for each segment using 4 control points (at some number of locations, connect lines)
- Use either deCasteljau algorithm (like Bezier) or explicit form [matrix formula on previous slide]
- Questions?