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

Survey Feedback

- 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 \( (u_1, u_2) \)

\[
\begin{align*}
&f(0,0) = F(0) \quad f(1,1) = F(1) \\
&f(0,1) = f(1,0)
\end{align*}
\]

Geometric interpretation: Quadratic
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: hw3
  - Recursively draw each piece
  - At some tolerance, draw control polygon
  - Trivial for Bezier curves (from deCasteljau algorithm): hence widely used for drawing
- Why specific labels/ control points on left/right?
  - How do they follow from deCasteljau?
Summary for HW 3 (with demo)

- **Bezier** (Bezier discussed last time)
  - Given arbitrary degree Bezier curve, recursively subdivide for some levels, then draw control polygon
  - 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

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

DeCasteljau: Recursive Subdivision

Input: Control points \( C_i \) with \( 0 \leq i \leq n \) where \( n \) is the degree.
Output: \( L_0, R_0 \) for left and right control points in recursion.

1. for \( \text{level} = n \) ; \( \text{level} > 0 \) ; \( \text{level} \rightarrow 0 \) \{  
2. if \( \text{level} == n \) \{ \rightarrow \text{initial control points} \}
3. \( \forall i : 0 \leq i \leq n : p^{\text{level}} = C_i ; \text{ continue} ; \}
4. for \( i = 0 ; i \leq \text{level} ; i + + \) \{  
5. \( p^{\text{level}} = p^{\text{level}+1} - p^{\text{level}+1} \)
6. \}
7. \( \forall i : 0 \leq i \leq n : L_1 = p^0_i ; R_0 = p^0_i ; \)

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

Beziers: Disadvantages

- Single piece, no local control (move a control point, whole curve changes) [Demo of HW 3]
- 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 of HW 3
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)

\[
F(u) = [u^3 \ u^2 \ u \ 1] \begin{bmatrix}
P_0 \\
P_1 \\
P_2 \\
P_3
\end{bmatrix} = \frac{1}{6} \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 0 & 3 & 0 \\
1 & 4 & 1 & 0
\end{bmatrix}
\]

Summary of HW 3

- BSpline Demo hw3
- 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?