Computer Graphics

CSE 167 [Win 19], Lecture 10: Curves 2

Ravi Ramamoorthi

http://viscomp.ucsd.edu/classes/cse167/wi19

Outline of Unit

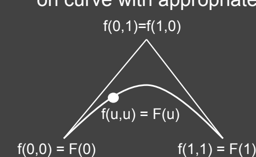
- Bezier curves (last time)
- deCasteljau algorithm, explicit, matrix (last time)
- *Polar form labeling (blossoms)*
- B-spline curves

- 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

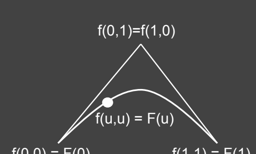
Idea of Blossoms/Polar Forms

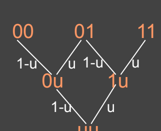
- (Optional) Labeling trick for control points and intermediate deCasteljau points that makes thing intuitive
- E.g. quadratic Bezier curve F(u)
  - Define auxiliary function $f(u_1,u_2)$ [number of args = degree]
  - Points on curve simply have $u_1=u_2$ so that F(u) = f(u,u)
  - And we can label control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



Idea of Blossoms/Polar Forms

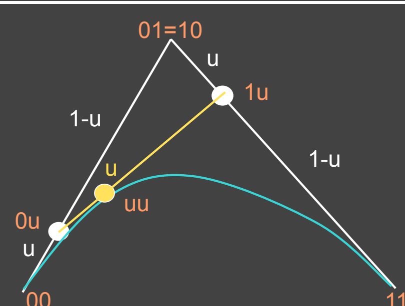
- Points on curve simply have $u_1=u_2$ so that F(u) = f(u,u)
- f is symmetric f(0,1) = f(1,0)
- Only interpolate linearly between points with one arg different
  - f(0,u) = (1-u) f(0,0) + u f(0,1) Here, interpolate f(0,0) and f(0,1)=f(1,0)



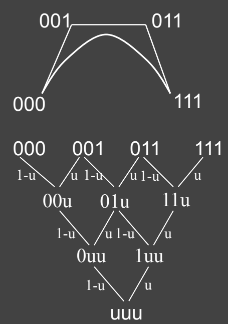


$F(u) = f(uu) = (1-u)^2 P0 + 2u(1-u) P1 + u^2 P2$

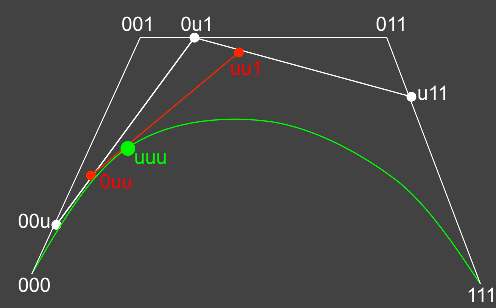
Geometric interpretation: Quadratic



Polar Forms: Cubic Bezier Curve



Geometric Interpretation: Cubic



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

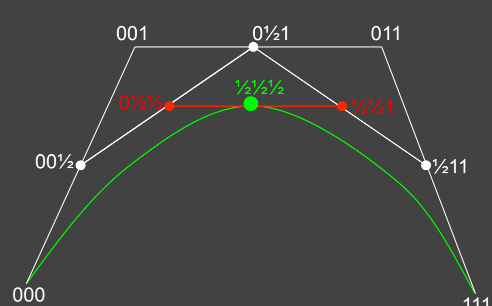
000 001 011 111

000 00u 0uu uuu ← → uuu uu1 u11 111

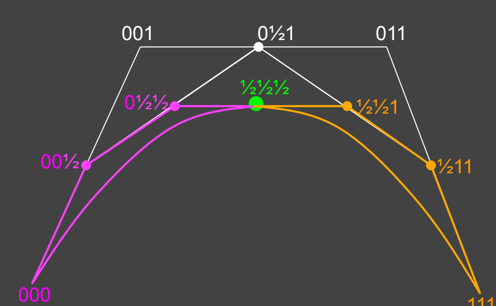
Why specific labels/ control points on left/right?

- How do they follow from deCasteljau?

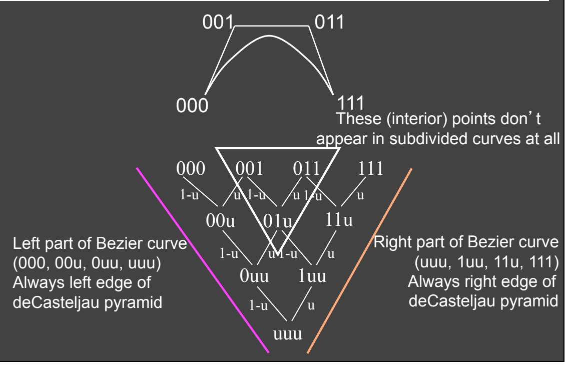
Geometrically



Geometrically



Subdivision in deCasteljau diagram



Summary for HW 3 (with demo)

- Bezier2 (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

DeCasteljau: Recursive Subdivision

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

```
for (level = n ; level >= 0 ; level--) {
  if (level == n) { // Initial control points
    ∀i : 0 <= i <= n : p_i^level = C_i ; continue ; }
  for (i = 0 ; i <= level ; i++)
    p_i^level = 1/2 * (p_i^(level+1) + p_(i+1)^(level+1)) ;
}
∀i : 0 <= i <= n : L_i = p_0^i ;  R_i = p_i^i ;
```

- 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*

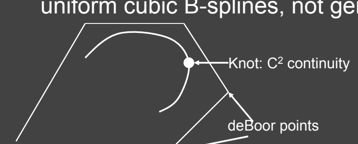
- Not well covered in textbooks (especially as taught here). Main reference will be lecture notes. If you do want a printed ref, handouts from CAGD, Seidel

Bezier: Disadvantages

- Single piece, no local control (move a control point, whole curve changes) [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

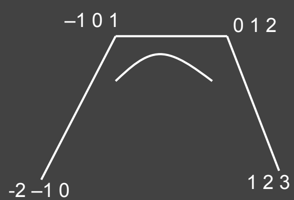
- Cubic B-splines have C^2 continuity, local control
- 4 segments / control point, 4 control points/ segment
- Knots where two segments join: Knotvector
- Knotvector uniform/non-uniform (we only consider uniform cubic B-splines, not general NURBS)



Demo of HW 3

Polar Forms: Cubic Bspline Curve

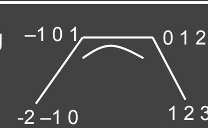
- Labeling little different from in Bezier curve
- No interpolation of end-points like in Bezier
- Advantage of polar forms: easy to generalize

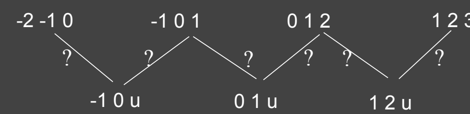


Uniform knot vector:
-2, -1, 0, 1, 2 ,3
Labels correspond to this

deCasteljau: Cubic B-Splines

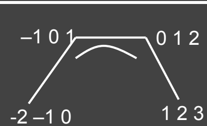
- Easy to generalize using polar-form labels
- Impossible remember without

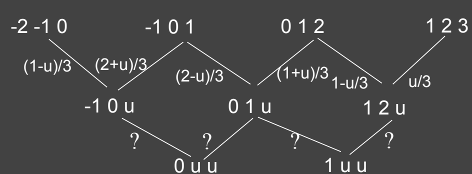




deCasteljau: Cubic B-Splines

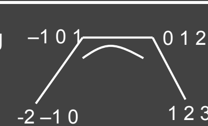
- Easy to generalize using polar-form labels
- Impossible remember without

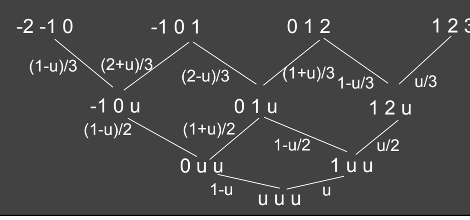




deCasteljau: Cubic B-Splines

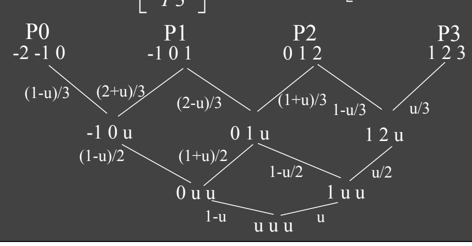
- Easy to generalize using polar-form labels
- Impossible remember without





Explicit Formula (derive as exercise)

$$F(u) = [u^3\ u^2\ u\ 1]\, M \begin{bmatrix} P0 \\ P1 \\ P2 \\ P3 \end{bmatrix} \qquad M = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$



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?