Approximating Curves

- Interpolation isn’t the only way to describe a curve
- Give points that "influence" a curve

- Why?
  - Better control of what happens in between points

- 2 important cases for computer graphics
  - Bezier
  - B-Spline

Bezier Segments

- Curve is made of many segments
  - Nomenclature issue
- Each segment is a polynomial
  - Of any degree
  - 3 is most common in computer graphics

Can we do Beziers this way?

- Yes – set up constraints and solve
  - \( f(0) = p_0 \)
  - \( f(1) = p_n \)
  - \( f(0) = 3 (p_1 - p_0) \)
  - \( f(1) = 3 (p_n - p_{n-1}) \)
- Doesn’t generalize well (OK for 2,3,4)
- General form for Bezier blending functions
  - Bernstein Basis Polynomials

Bezier Segments (2)

- The nth derivative depends on the first (or last) n points
  - Cubics are similar to Hermite's
    - All points in space (not derivative amounts)
    - Scaling factors
  - Pieces connected by placing points correctly
    - \( C(0) \) by matching endpoints
    - \( C(1) \) by aligning end vectors
    - \( G(1) \) by end-vectors being co-linear
Properties of Bezier Curves

- Simple mathematical form for basis functions
- Good algorithms for computation
  - Subdivision procedure
  - De Casteljau algorithm
  - Divide and conquer because...
- Convex Hull Properties
- Variation Diminishing
- Symmetric
- Affine invariant
  - NOT perspective invariant

De Casteljau Algorithm

- Evaluate curve at \( u \)
  - Divide line segments
  - \( U \) of the way
- Can use to subdivide curves
- Repeated linear interpolation for ANY degree!

Decasteljau to Bernstein

- Apply geometric construction to derive equations
- Different groups came at this differently
- Algebraic vs. Subdivision

\[
\begin{align*}
\pi_{17} &= (1-u)\, P_1 + u\, P_2 \\
\pi_{123} &= (1-u)\, P_{12} + u\, P_{23} \\
F(u) &= \left( (1-u)(1-u) \, P_1 + (1-u)\, P_2 + u\, P_3 \right) \\
&+ \left( u(1-u) \, P_{12} + u\, P_{23} \right) \\
&+ \left( (1-u)\, P_{123} \right)
\end{align*}
\]

How do we make Smooth Curves?

- Will be approximating
- Want flexibility
  - Any number of points
  - Any degree of continuity
- Want good properties
  - Locality
  - Convex Hull
  - Variation Diminishing
  - Shift Invariant (sum of blending functions = 1)

An example of blending functions

- Lines
- Blending functions are “bumps”
- Each piece is a spline
  - Two polynomials, degree 1
- Locality
  - Non-Zero over small range
  - Between 3 knots (d+2)
- Smoothness
  - D-1 continuity
- Shift invariant
- Convex Hull Property
- Shiftable (periodic)

Consider B-Splines

- \( N \) points
  - General – any \( N \)
  - \( P_0, \ldots, P_{n-1} \)
- WARNING: not same notation as book
- Consider linear interpolation
  - \( F(t) = \sum b_i(t) \, P_i(t) \)

**B-Splines**

- General scheme for generating blending functions
  - Any number of points (need more points than degree)
  - Any degree of polynomial (higher degree = smoother)
  - Any knot vector
- Blending function of degree D are B-Splines
  - Made of D+1 segments (span D+2 knots)
  - Each segment is a degree D polynomial
  - Only D+1 of them are non-zero at any time
  - Sum to one
  - Zero outside of the range
  - D-1 continuous
- Note: usually talk about “order” (degree+1)

**Linear B-Spline**

- Each blending function is a bump
- All the same (different ones are shifts)
- Active from i-1 to i+1
  - Over 2 spans, 3 integers
- In between 2 pts are active
  - One in each “phase”
- “before” t=0 and “after” n-1
  - Not enough points

**Creating B-Splines**

- Cox-de Boor recurrence
- Convolutions of the unit box
- When d>1, the functions do not interpolate
  - They never reach the value of 1

**Cubic Blending Functions**

- Active over 4 regions (d=3, k=d+1=4)
- At any time, one point in each phase
- Example t=4.5
  - Eval point 3 @ 1.5
  - Eval point 4 @ .5
  - Eval point 5 @ -.5
  - Eval point 6 @ -1.5
  - Each in a different part

**Quadratic, Uniform B-Splines**

- \( B(t) = \begin{cases} \frac{1}{3}u^2 - u - \frac{1}{2} & \text{if } i \leq t < i + 1 \\ \frac{1}{3} & \text{if } i + 1 \leq t < i + 2 \\ 0 & \text{otherwise} \end{cases} \)
  
**Even More General?**

- B-Splines cannot represent conic sections
  - Can’t make an exact circle
- Express curves / surfaces as the RATIO
- Non-Uniform Rational B-Spline Surfaces
  - (NURBS)
- Extensions to surfaces later in the class
Knot vectors

- Allow us to assign parameter values to points
- Makes it possible to alter the set of points but keep parameter values fixed
- Allows us to alter the spacing
- Allows us to create discontinuities
- (picture with lines)
- Uniform vs. Non-Uniform

Using B-Splines

- Figure out closed form basis functions
  - Rather than using Cox-de Boor
- Can encode into a Basis matrix
  - But cannot derive the same way
- Periodic basis functions are nice
  - Implement once
- Gives a nice way to get very smooth curves
  - Cubics (usually) in graphics to provide C(2)