CSE-170 Computer Graphics

Lecture 17

Bézier Curves

Dr. Renato Farias
rfarias2@ucmerced.edu

Bézier Curves

Pierre Étienne Bézier
Renault Engineer during 1933-1975

De Casteljau Algorithm

Bézier: De Casteljau Algorithm

- De Casteljau is valid for any degree:

  - Based on sequence of linear interpolations

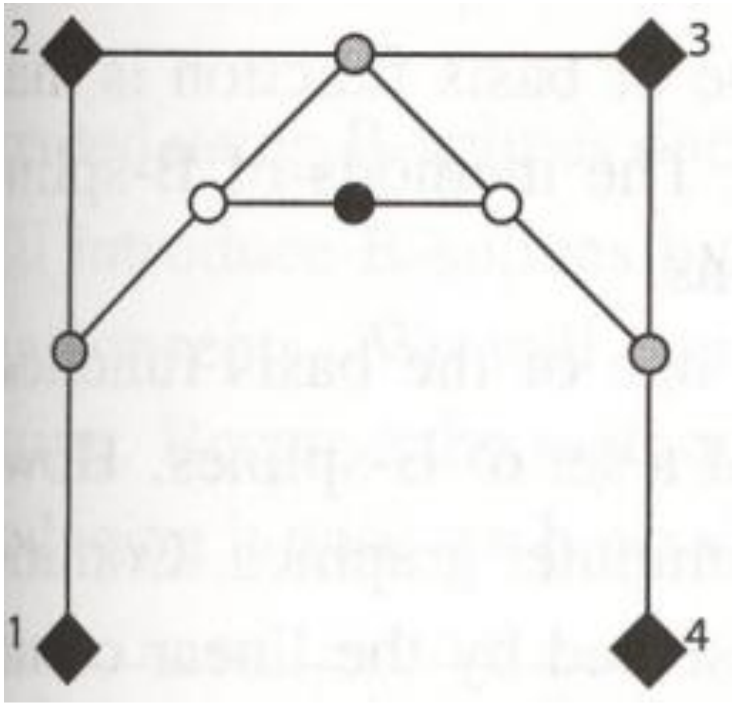
  - Given t in [0,1], and n control points $\mathbf{p}_i$:

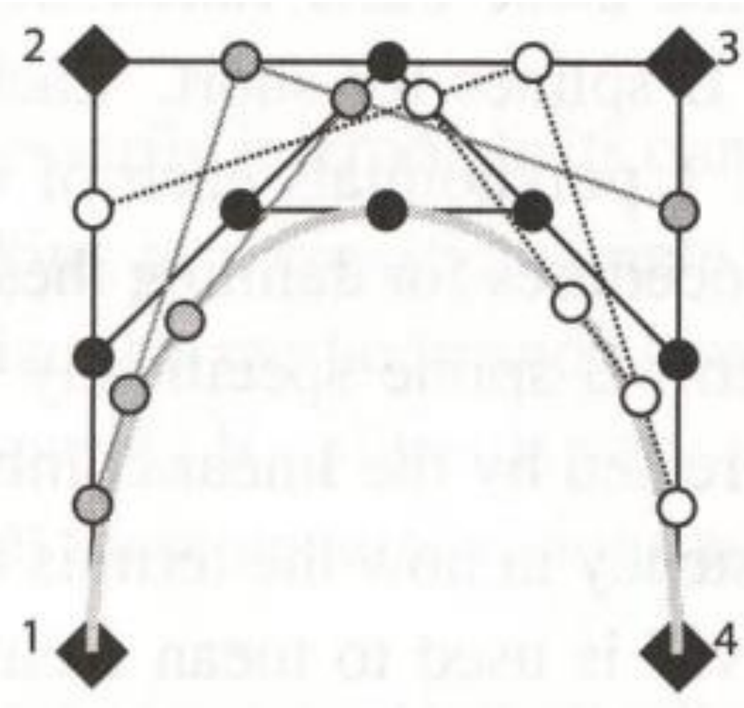
    1. Apply linear interpolation with parameter t for every adjacent pair of control points, determining new (n-1) control points.

    2. Repeat the process until achieving only one point, which is the point on the Bézier curve at position t.

Bézier: De Casteljau Algorithm

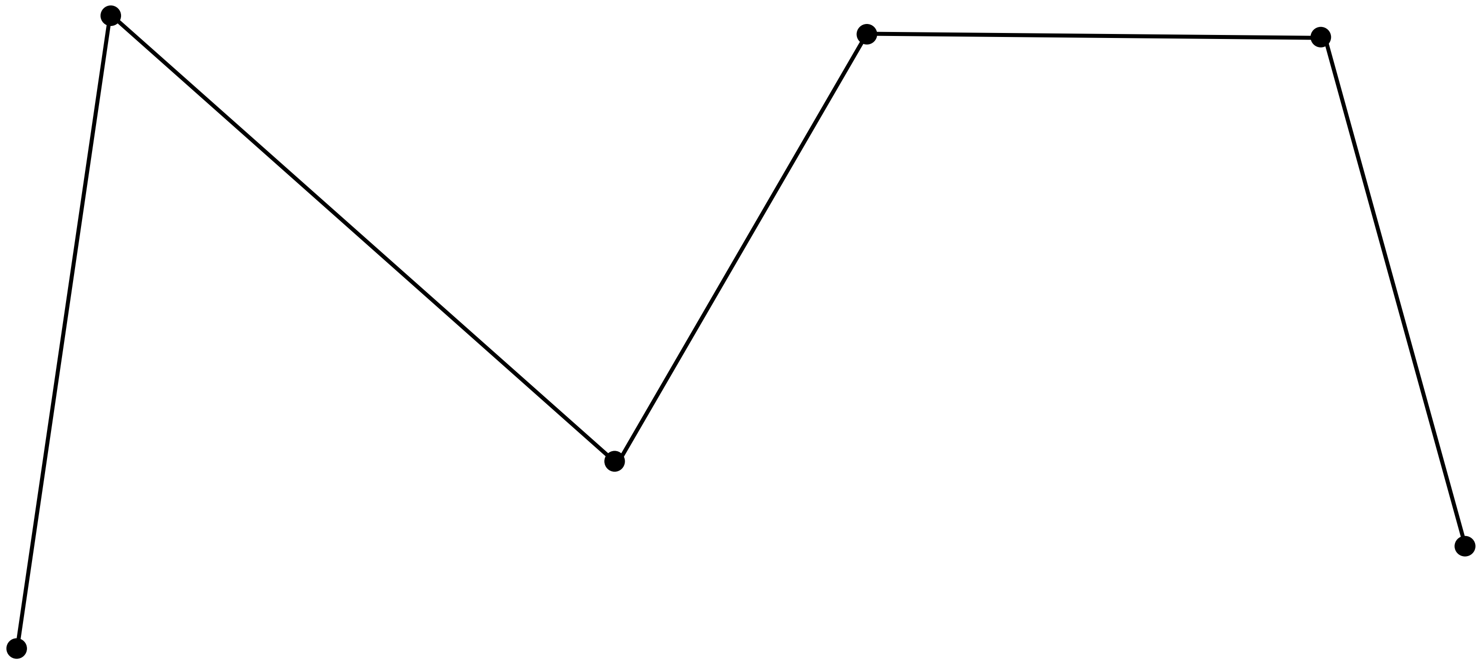
- Example:





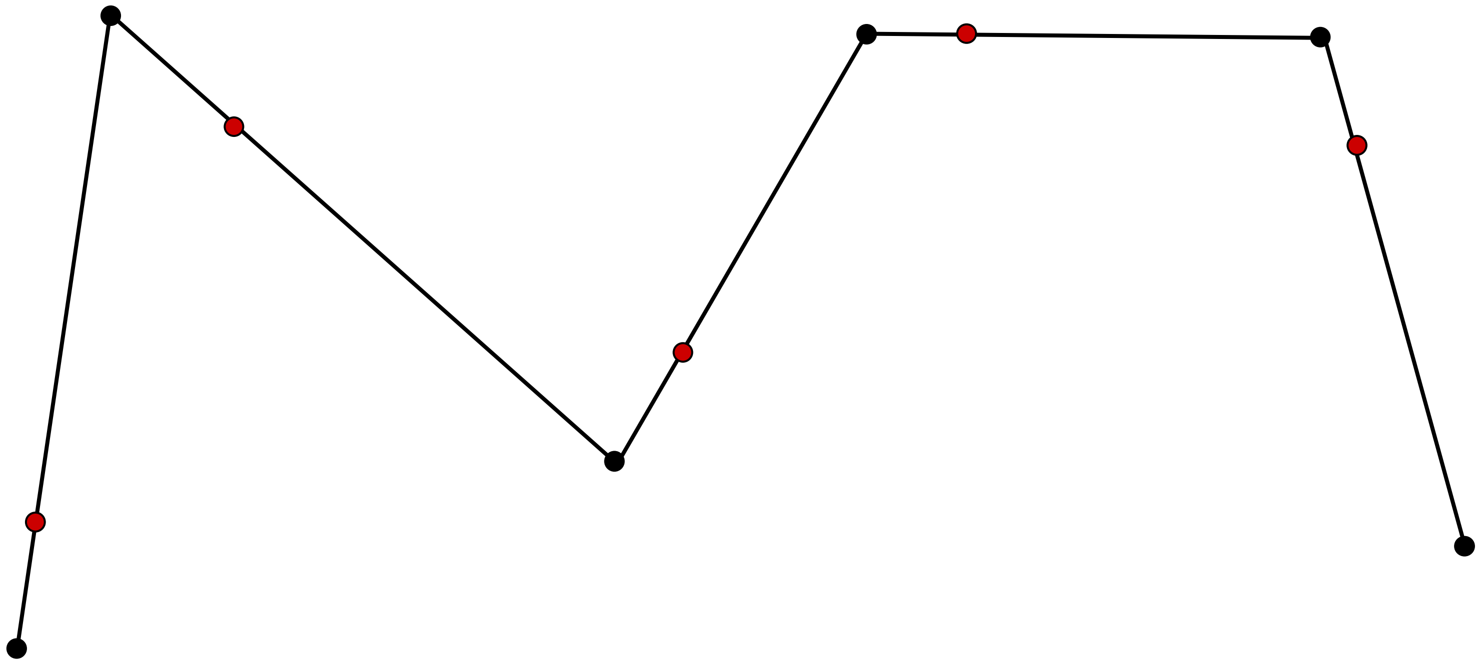
Bézier: De Casteljau Algorithm

- Step-by-step example
  - Compute point at $t = 0.25$ in the Bezier curve defined by the black control polygon given below:



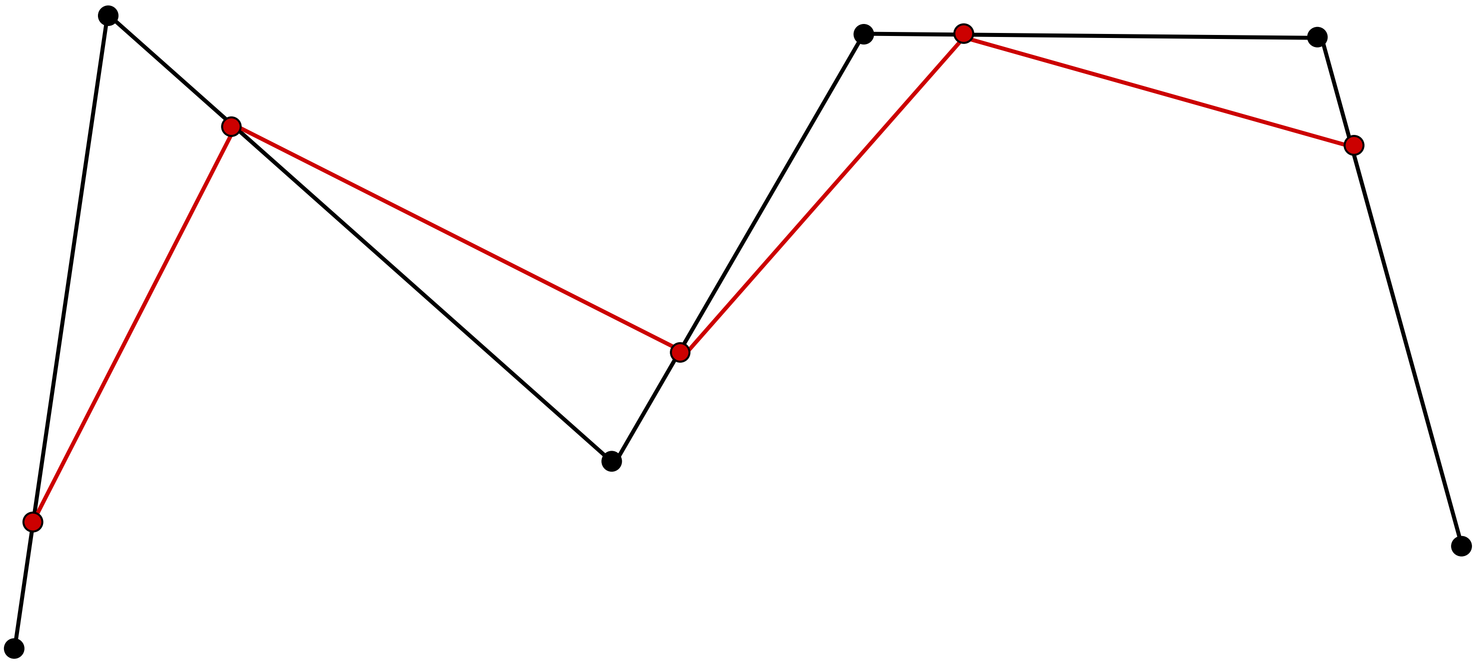
Bézier: De Casteljau Algorithm

- Step-by-step example
  1) Interpolate endpoints of each control segment at .25
     - Red points below are obtained:



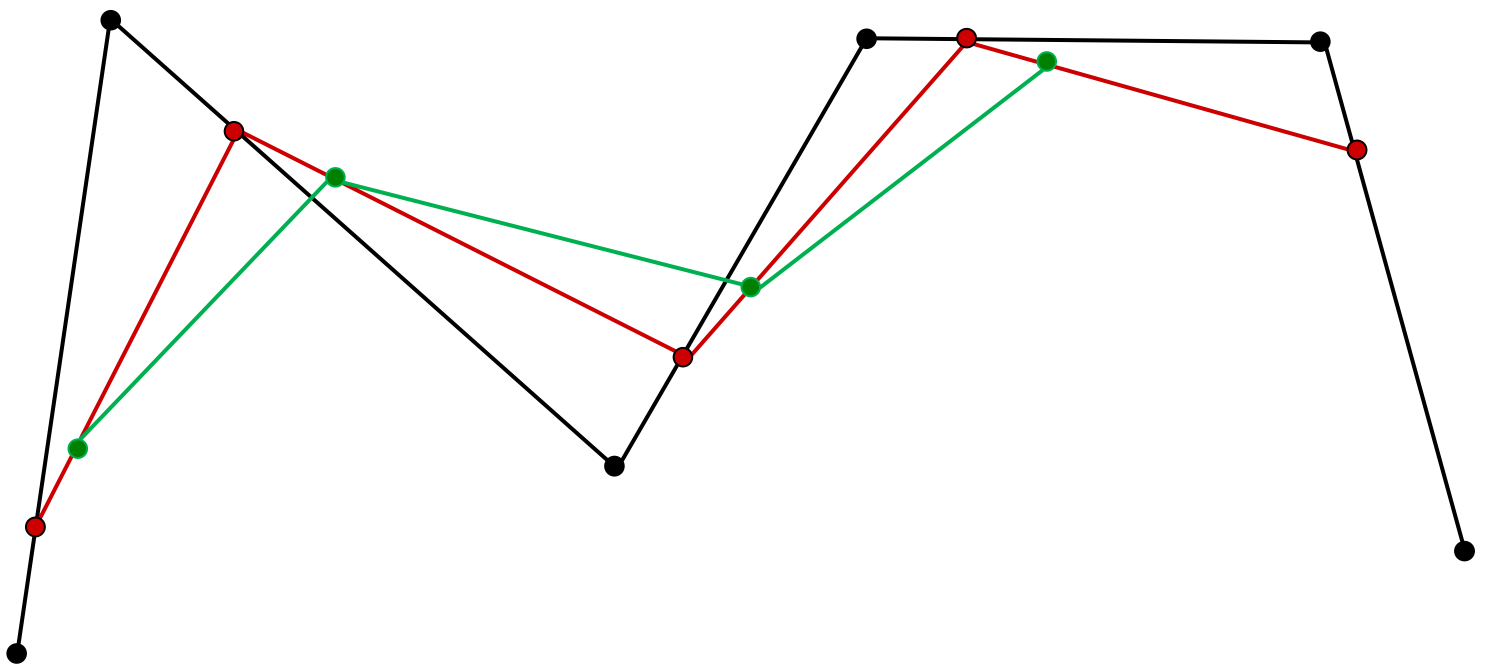
Bézier: De Casteljau Algorithm

- Step-by-step example
  1) Interpolate endpoints of each control segment at .25
     - A new control polygon is obtained, the one in red below:



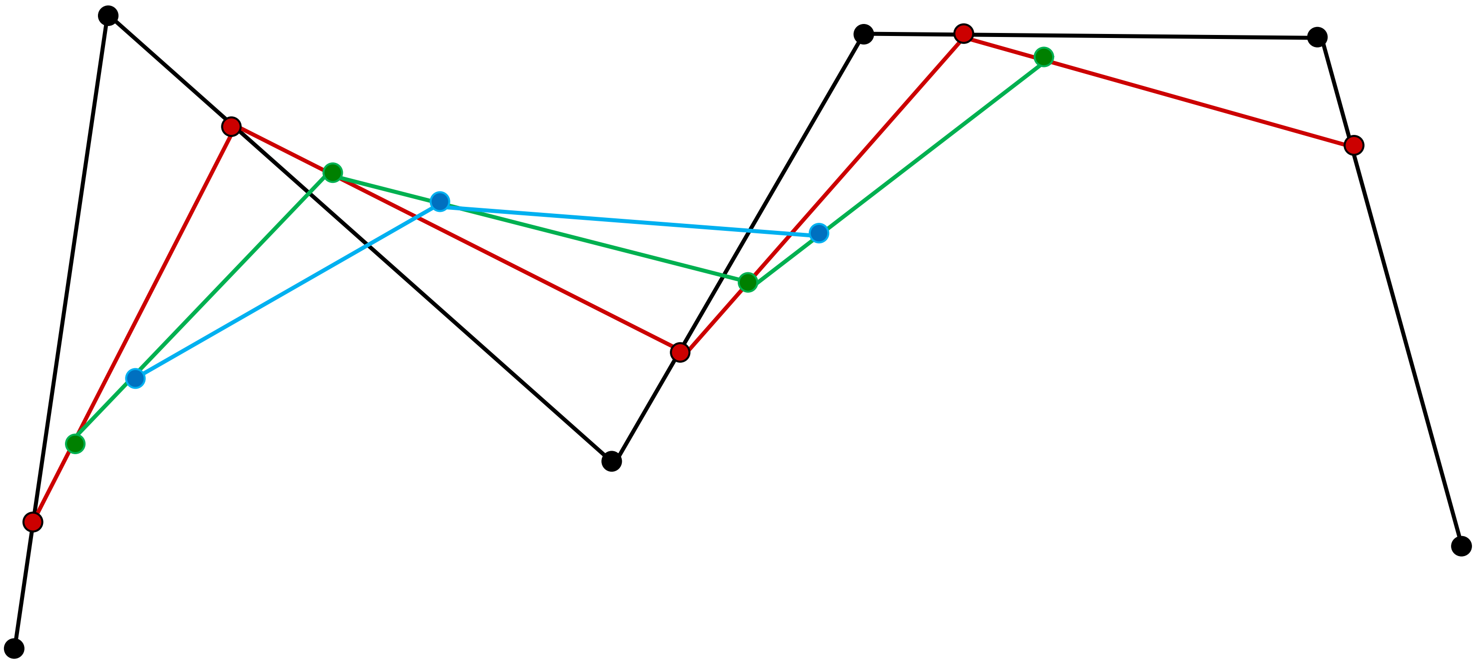
Bézier: De Casteljau Algorithm

- Step-by-step example
  1) Interpolate endpoints of each control segment at .25
  2) Now interpolate endpoints (at .25) of red control polygon
     - New green control polygon is obtained:



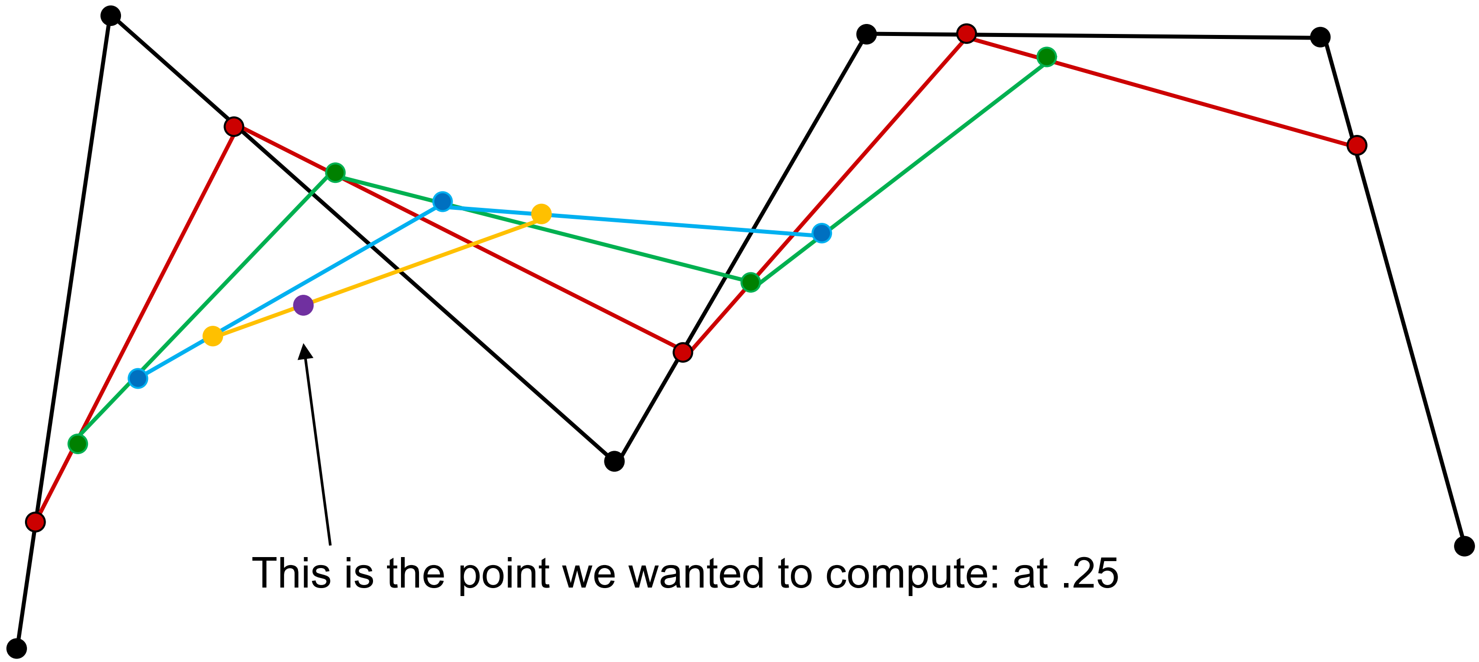
Bézier: De Casteljau Algorithm

- Step-by-step example
  1) Interpolate endpoints of each control segment at .25
  2) Now interpolate endpoints (at .25) of red control polygon
  3) Interpolate again in the new green polygon
     - Obtain blue one



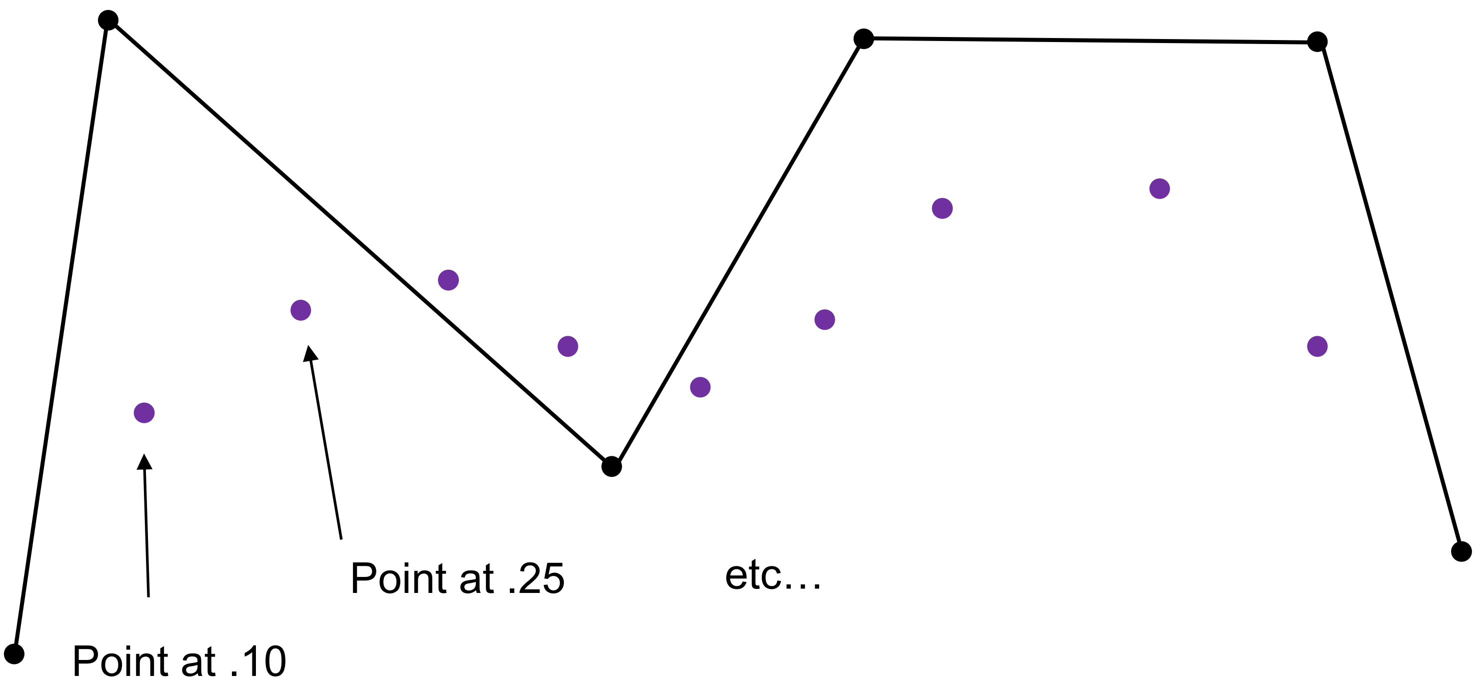
Bézier: De Casteljau Algorithm

- Step-by-step example
  1) Interpolate endpoints of each control segment at .25
  2) Now interpolate endpoints (at .25) of red control polygon
  3) Interpolate again in the new green polygon
  4) Repeat for blue polygon until single point is reached:



Bézier: De Casteljau Algorithm

- Drawing the whole curve
  - To draw the whole curve using this method just compute several points and connect them



(note: points above are just for illustration, they were not computed one by one)

Bézier Curves
Bernstein Polynomials

Bézier

- Control polygon idea:
  – Consider that, similarly to Hermite, there are derivative constraints but which are defined from the control polygon directly:

    $f'(0) = 3(p_1-p_0)$, $f'(1) = 3(p_3-p_2)$

$$\mathbf{f}(t) = \mathbf{a} + \mathbf{b}t + \mathbf{c}t^2 + \mathbf{d}t^3$$

$$\mathbf{f}'(t) = \mathbf{b} + 2\mathbf{c}t + 3\mathbf{d}t^2$$

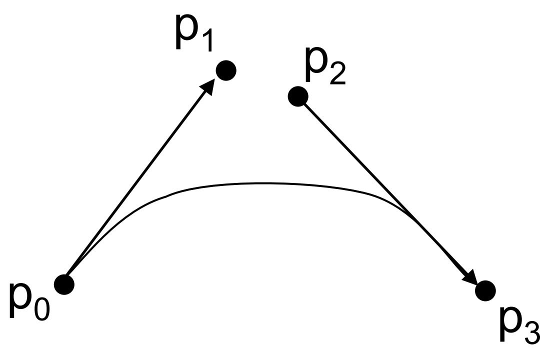
- Cubic case:
  – Bézier basis functions are similar to the Hermite basis:

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}$$

$$\mathbf{p}_3 = \mathbf{f}(1) = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$

$$3(\mathbf{p}_1 - \mathbf{p}_0) = \mathbf{f}'(0) = \mathbf{b}$$

$$3(\mathbf{p}_3 - \mathbf{p}_2) = \mathbf{f}'(1) = \mathbf{b} + 2\mathbf{c} + 3\mathbf{d}$$



Bézier

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}$$

$$\mathbf{p}_3 = \mathbf{f}(1) = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$

$$3(\mathbf{p}_1 - \mathbf{p}_0) = \mathbf{f}'(0) = \mathbf{b}$$

$$3(\mathbf{p}_3 - \mathbf{p}_2) = \mathbf{f}'(1) = \mathbf{b} + 2\mathbf{c} + 3\mathbf{d}$$

How can we obtain a formulation based on blending functions?

$$\mathbf{f}(t) = \boxed{B_0}\mathbf{p}_0 + \boxed{B_1}\mathbf{p}_1 + \boxed{B_2}\mathbf{p}_2 + \boxed{B_3}\mathbf{p}_3$$

$$\mathbf{f}(t) = \sum_{i=0}^{3} \mathbf{p}_i B_i(t)$$

Bézier

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a} \qquad 3(\mathbf{p}_1 - \mathbf{p}_0) = \mathbf{f}'(0) = \mathbf{b}$$

$$\mathbf{p}_3 = \mathbf{f}(1) = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d} \qquad 3(\mathbf{p}_3 - \mathbf{p}_2) = \mathbf{f}'(1) = \mathbf{b} + 2\mathbf{c} + 3\mathbf{d}$$

Just solve the equations above for a, b, c, d, and then re-write the cubic to get the blending functions (as we did in the Hermite case):

$$\mathbf{f}(t) = \boxed{(1 - 3t + 3t^2 - t^3)}\mathbf{p}_0 + \boxed{(3t - 6t^2 + 3t^3)}\mathbf{p}_1 + \boxed{(3t^2 - 3t^3)}\mathbf{p}_2 + \boxed{(t^3)}\mathbf{p}_3$$

Rewrite blending functions to obtain the ***Bernstein*** *polynomials:*

$$\mathbf{f}(t) = \sum_{i=0}^{3} \mathbf{p}_i B_{i,3}(t)$$

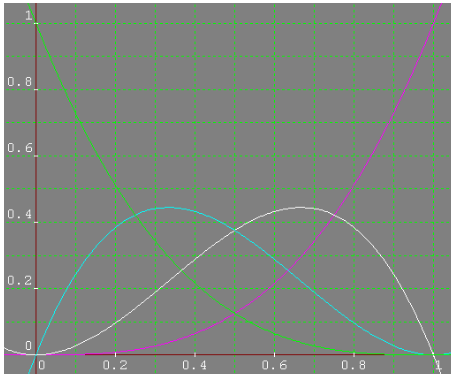
This 3 means this is the basis for a cubic curve (needed for a 4-point control polygon)

$$B_{0,3}(t) = (1-t)^3$$

$$B_{1,3}(t) = 3t(1-t)^2$$

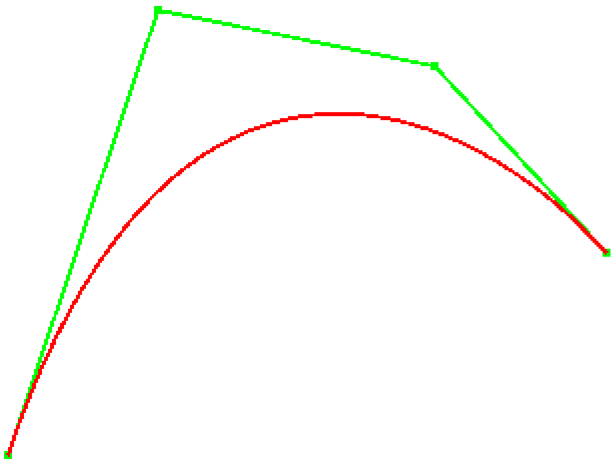
$$B_{2,3}(t) = 3t^2(1-t)$$

$$B_{3,3}(t) = t^3$$

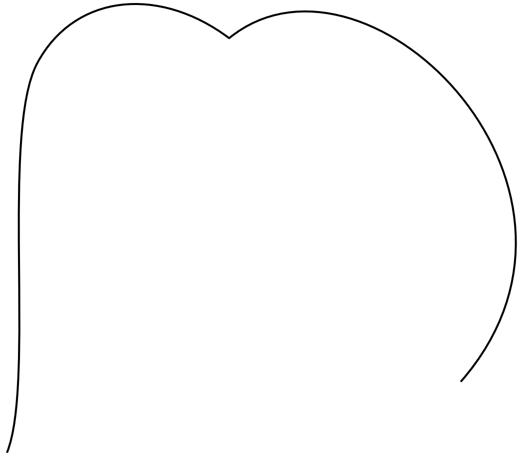


Bézier

- 3^{rd} order examples
  - 3 segments / 4 points => 3^{rd} degree curve (cubic curve)
  - Example of control polygon:



  - Example of two cubic Béziers connected:
    - In the example, the curve is C^0 but not C^1 at the connection (knot) point

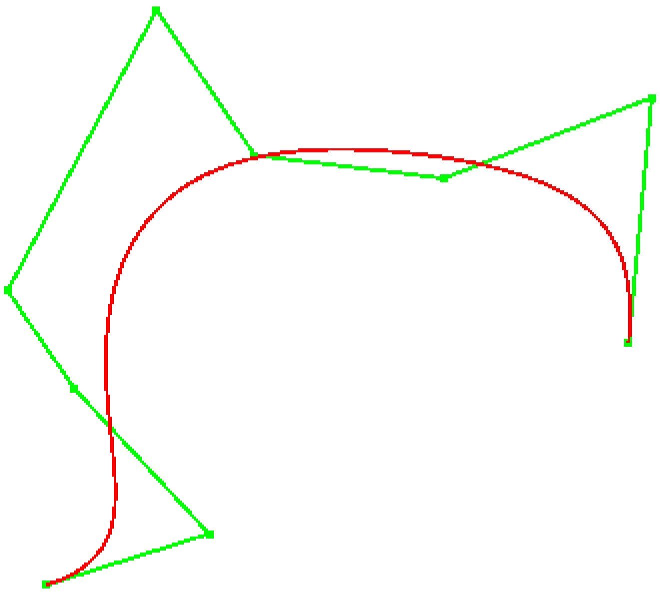


Bézier Curves
Generic Order

Bézier

- Generalization to order n:
  (t in [0,1])

  – Basis functions can be derived from the De Casteljau geometric construction

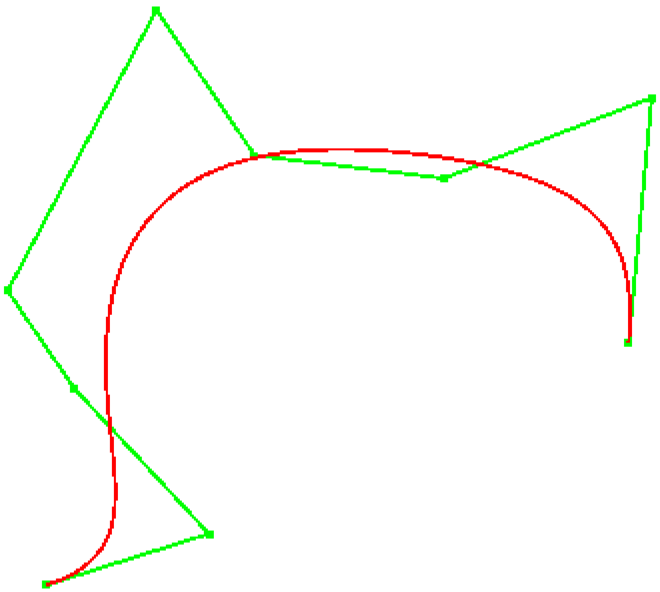


$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(t)$$

Bézier

- Generalization to order n:

  (t in [0,1])



$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(t)$$

Bernstein basis polynomials:

$$\boxed{\begin{aligned} B_{i,n}(t) &= C(n,i)t^i(1-t)^{(n-i)}, \\ C(n,i) &= \frac{n!}{i!(n-i)!} \end{aligned}}$$

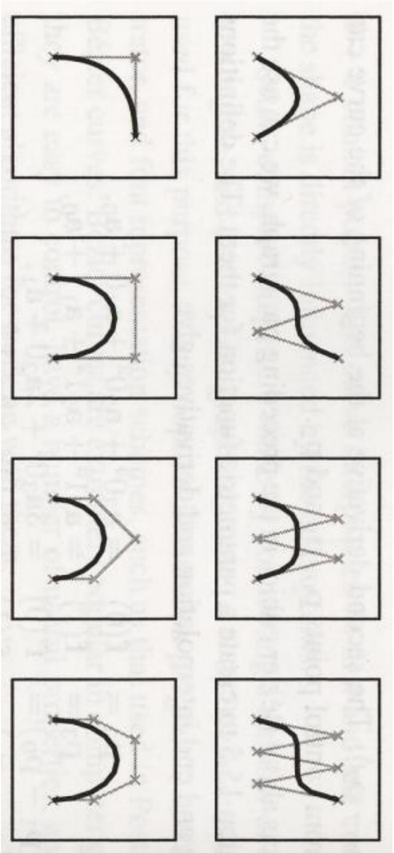
(C(n,i) is the binomial coefficient: ways of picking i unordered outcomes from n)

Ex:

$$B_{0,3}(t) = (1-t)^3$$
$$B_{1,3}(t) = 3t(1-t)^2$$
$$B_{2,3}(t) = 3t^2(1-t)$$
$$B_{3,3}(t) = t^3$$

Bézier

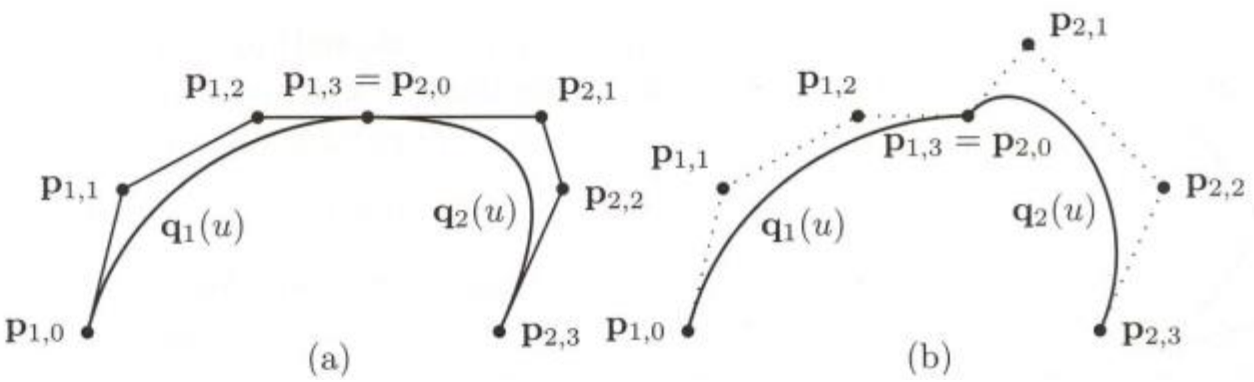
- Curve Properties – Important!
  - Bounded by convex hull of control points
    - The convex hull is the “smallest” convex polygon containing the control points
  - Variation diminishing property
    - The curve does not cross a line more than its control polygon does
  - Symmetric
    - Same curve if control points order is reversed
  - Affine invariant
    - Curve can be transformed by transforming control points with affine transformations
  - There are simple algorithms for evaluation and subdivision



Bézier Curves
Piecewise Formulation

Piecewise Béziers

- We want to, at least, achieve geometric continuity G^1
  - Case (a) is G^1–continuous but not C^1–continuous
  - Case (b) is neither C^1–continuous nor G^1–continuous



Piecewise Béziers

- Desired condition for continuity
  - Based on derivatives:

$$\mathbf{q}'_1(1) = \mathbf{q}'_2(0)$$

  - Based on the control polygon:

$$\boxed{\mathbf{p}_{1,3} - \mathbf{p}_{1,2} = \mathbf{p}_{2,1} - \mathbf{p}_{2,0}}$$

Bézier Surfaces
Cubic Patches

Cubic Bézier Patches

- Are parametric surfaces defined by 16 control points

$$\mathbf{q}(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{p}_{i,j}B_{i,3}(u)B_{j,3}(v)$$

$$B_{0,3}(t)=(1-t)^3$$

$$B_{1,3}(t)=3t(1-t)^2$$

$$B_{2,3}(t)=3t^2(1-t)$$

$$B_{3,3}(t)=t^3$$