CSE 167: Introduction to Computer Graphics Lecture #11: Curves

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2010

Announcements

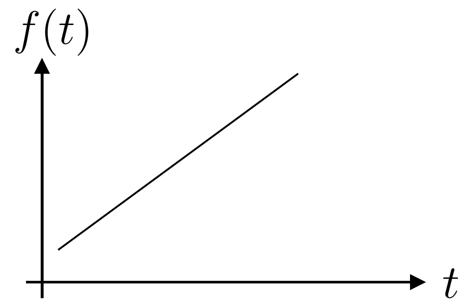
- Homework assignment #5 due Friday, Nov 5
- Phi is not having an office hour this Thursday
- Instead, Phi's office hour will be this Friday, 1-2pm
- Midterm grading to be completed by Thursday

Lecture Overview

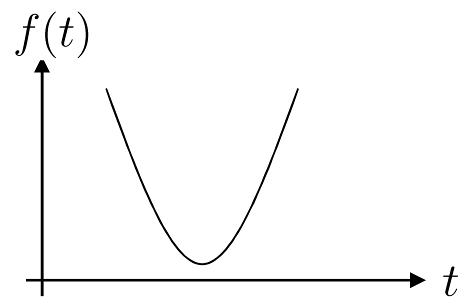
- Bézier curves
- Drawing Bézier curves
- Piecewise Bézier curves

Polynomial Functions

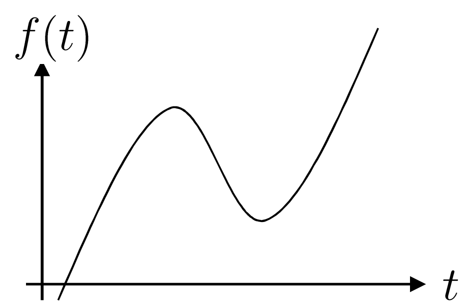
- **Linear:** (1st order) $$f(t) = at + b$$



- **Quadratic:** (2nd order) $$f(t) = at^2 + bt + c$$



- **Cubic:** (3rd order) $$f(t) = at^3 + bt^2 + ct + d$$



Linear Interpolation

- Three equivalent ways to write the equation
- Each emphasizes different properties

1. Weighted sum of the control points

$$\mathbf{x}(t) = \mathbf{p}_0(1-t) + \mathbf{p}_1 t$$

2. Polynomial in t

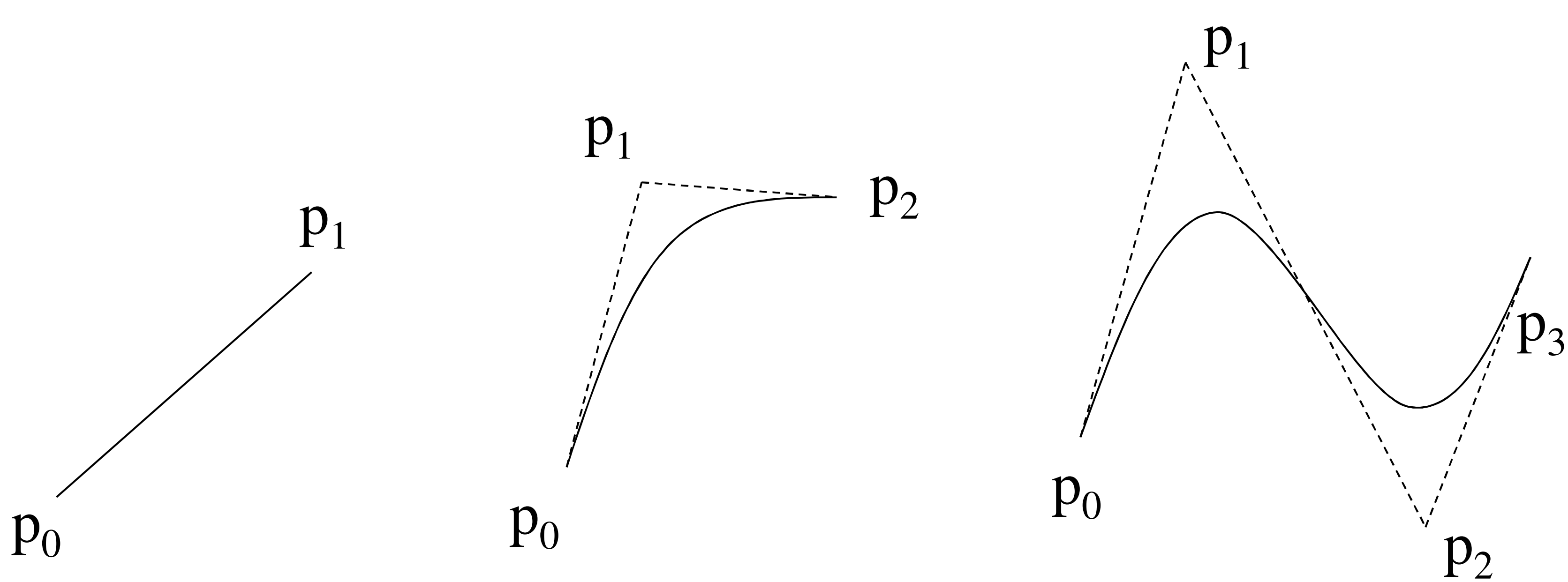
$$\mathbf{x}(t) = (\mathbf{p}_1 - \mathbf{p}_0)t + \mathbf{p}_0$$

3. Matrix form

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

Bézier Curves

- Are a higher order extension of linear interpolation



Linear Quadratic Cubic

Bézier Curves

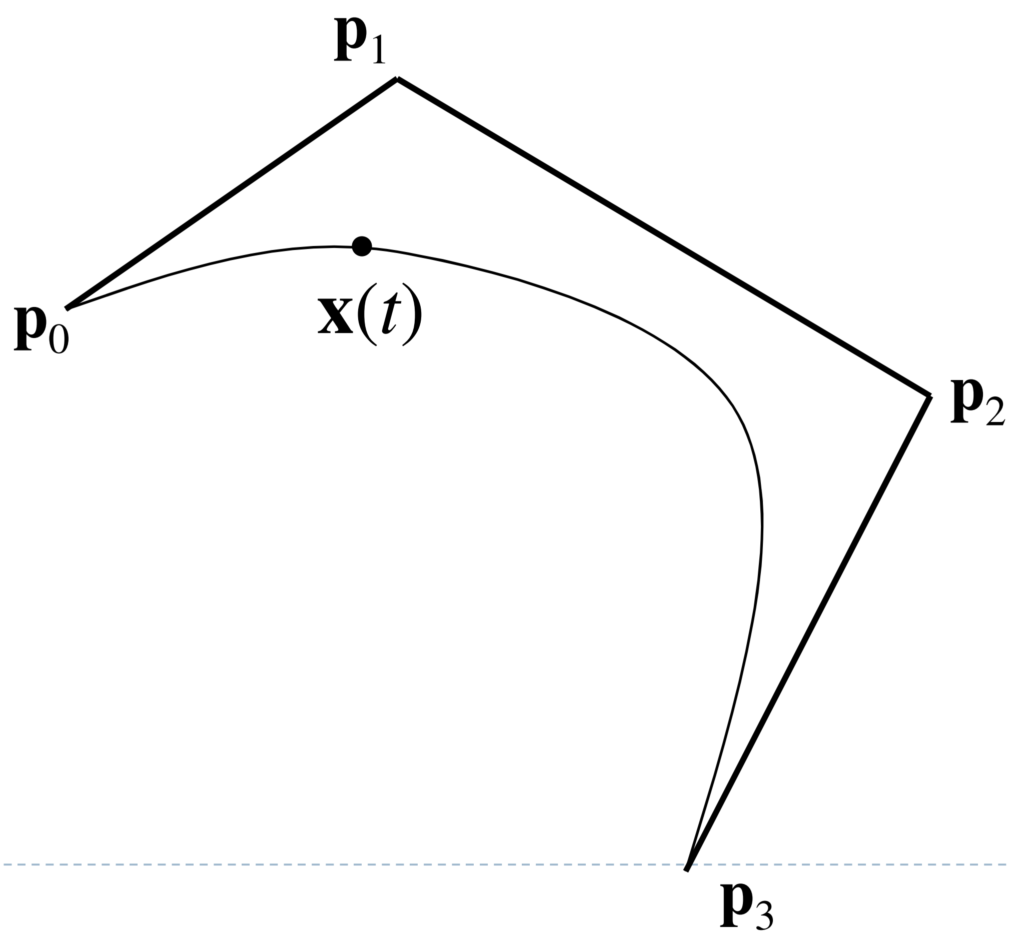
- Give intuitive control over curve with control points
  - Endpoints are interpolated, intermediate points are approximated
  - Convex Hull property
  - Variation-Diminishing property
- Many demo applets online
  Examples:
  - Demo: http://www.cs.princeton.edu/~min/cs426/jar/bezier.html
  - http://www.theparticle.com/applets/nyu/BezierApplet/
  - http://www.sunsite.ubc.ca/LivingMathematics/V001N01/UBCExamples/Bezier/bezier.html

Cubic Bézier Curve

- Most common case
- Defined by four control points:
  - Two interpolated endpoints (points are on the curve)
  - Two points control the tangents at the endpoints

Cubic Bézier Curve

- Define point $\mathbf{x}$ on the curve as a function of parameter t



Bézier Curve Formulation

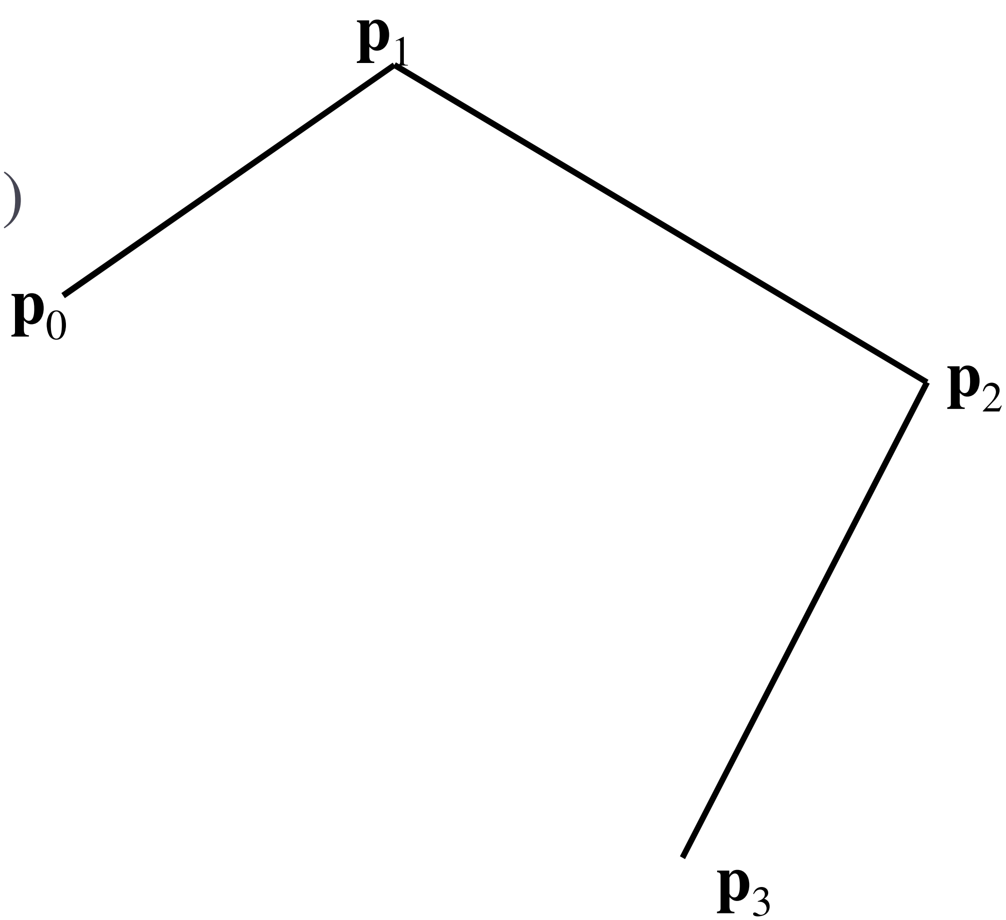
- Three alternatives, analogous to linear case
  1. Weighted average of control points
  2. Cubic polynomial function of t
  3. Matrix form
- Algorithmic construction
  - *De Casteljau* algorithm, developed at Citroen in 1959
  - Developed independently from Bézier's work:
    Bézier created the formulation using blending functions,
    Casteljau devised the recursive interpolation algorithm

De Casteljau Algorithm

- A recursive series of linear interpolations
  - Works for any order, not only cubic
- Not very efficient to evaluate
  - Other forms more commonly used
- Why study it?
  - Intuition about the geometry
  - Useful for subdivision (later today)

De Casteljau Algorithm

- Given:
  - Four control points
  - A value of t (here $t \approx 0.25$)

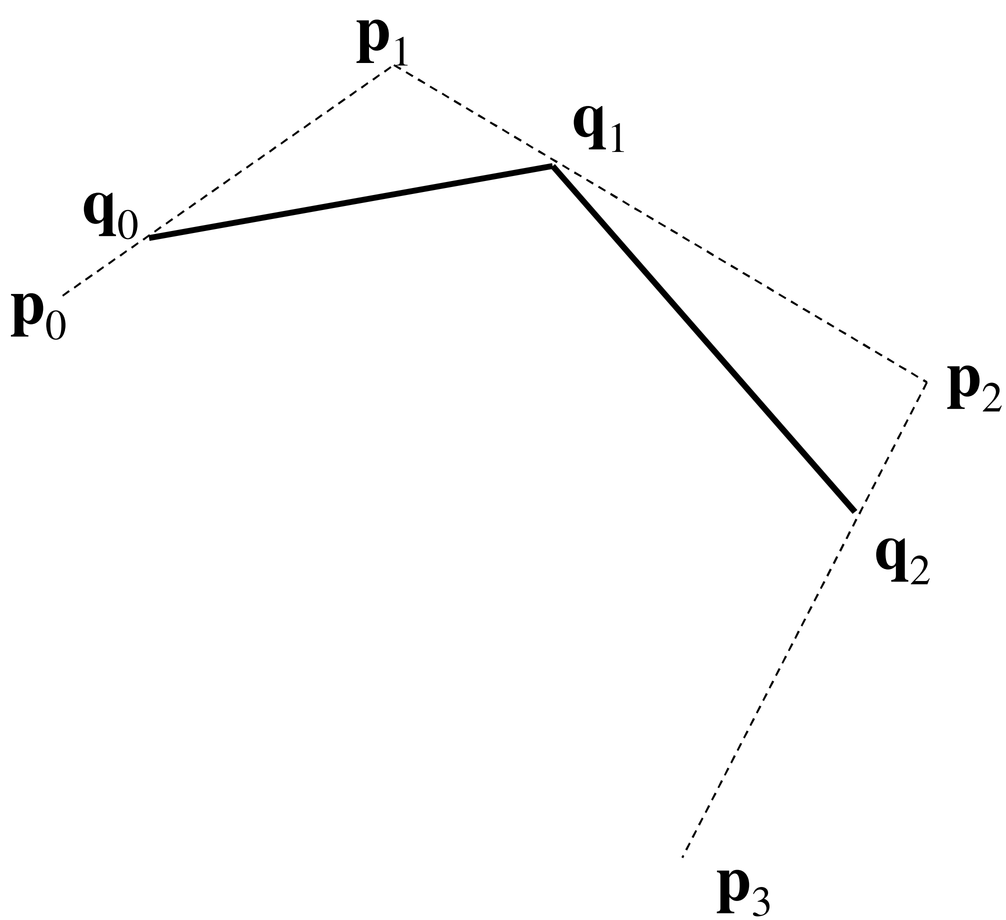


De Casteljau Algorithm

$$\mathbf{q}_0(t) = Lerp\left(t, \mathbf{p}_0, \mathbf{p}_1\right)$$

$$\mathbf{q}_1(t) = Lerp\left(t, \mathbf{p}_1, \mathbf{p}_2\right)$$

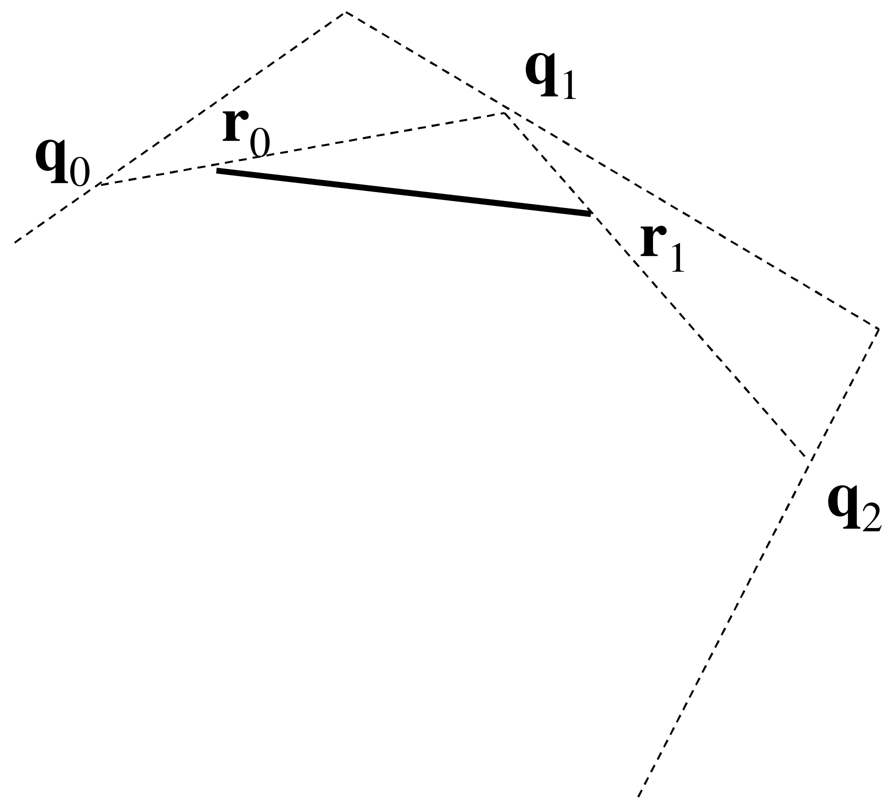
$$\mathbf{q}_2(t) = Lerp\left(t, \mathbf{p}_2, \mathbf{p}_3\right)$$



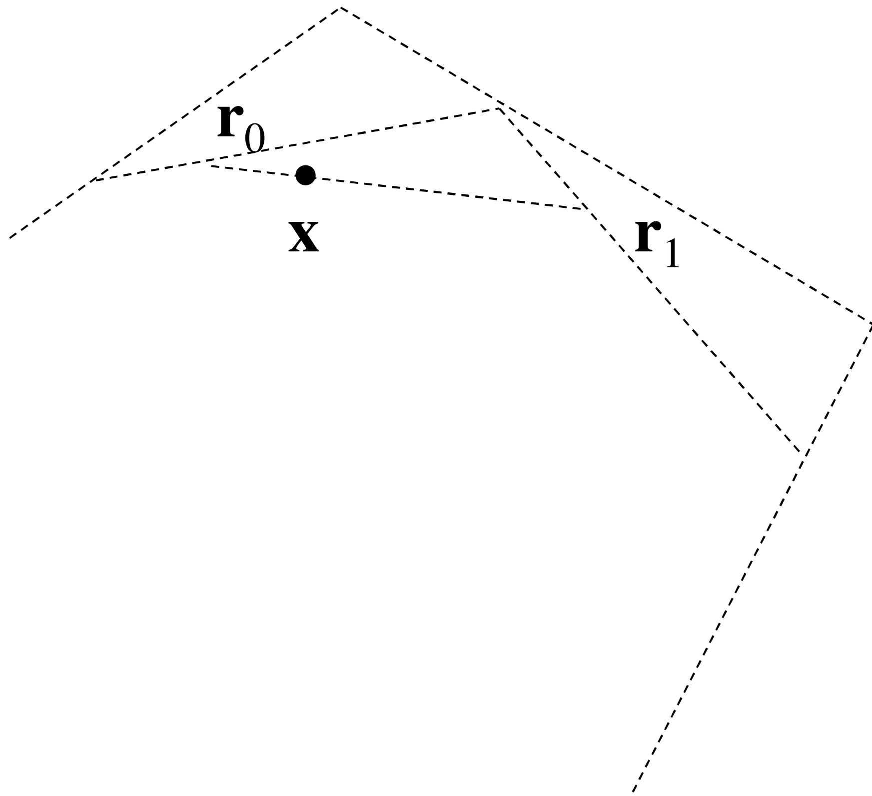
De Casteljau Algorithm

$$\mathbf{r}_0(t) = Lerp\big(t, \mathbf{q}_0(t), \mathbf{q}_1(t)\big)$$

$$\mathbf{r}_1(t) = Lerp\big(t, \mathbf{q}_1(t), \mathbf{q}_2(t)\big)$$

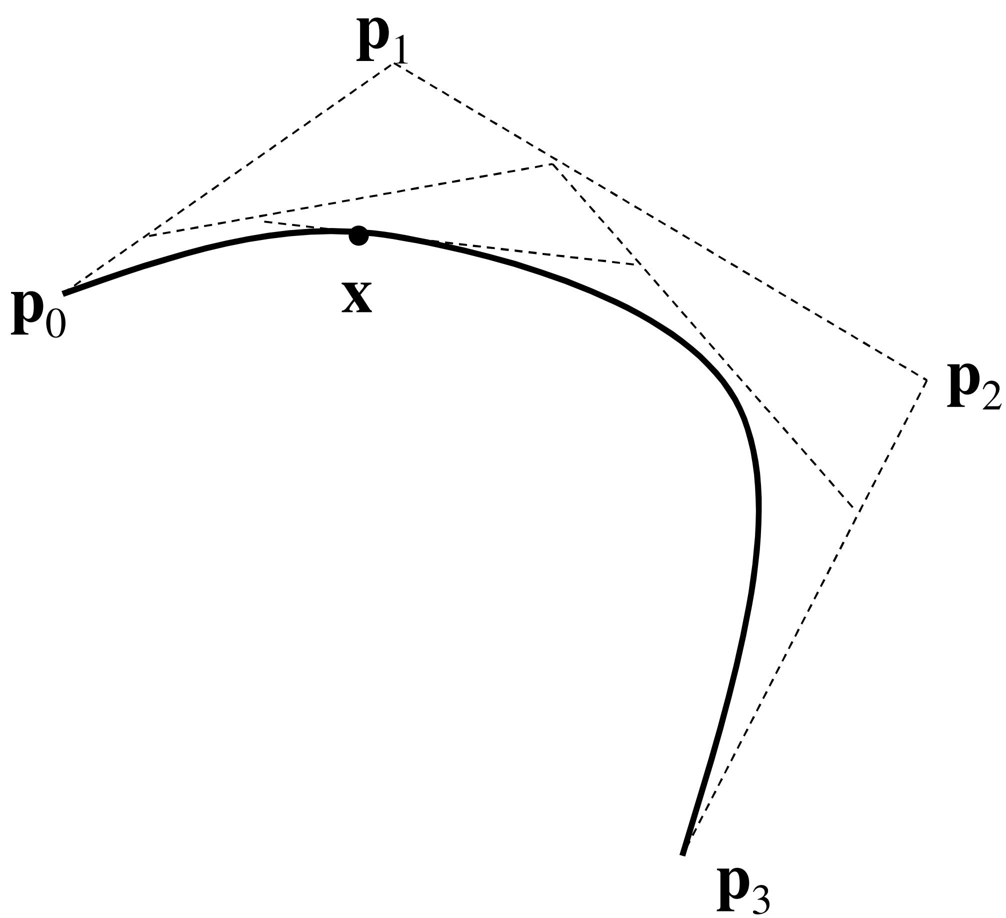


De Casteljau Algorithm



$$\mathbf{x}(t) = Lerp\left(t, \mathbf{r}_0(t), \mathbf{r}_1(t)\right)$$

De Casteljau Algorithm



- Applets
  - Demo: http://www2.mat.dtu.dk/people/J.Gravesen/cagd/decast.html
  - http://www.caffeineowl.com/graphics/2d/vectorial/bezierintro.html

Recursive Linear Interpolation

$$\mathbf{x} = Lerp(t, \mathbf{r}_0, \mathbf{r}_1)$$

$$\mathbf{r}_0 = Lerp(t, \mathbf{q}_0, \mathbf{q}_1)$$

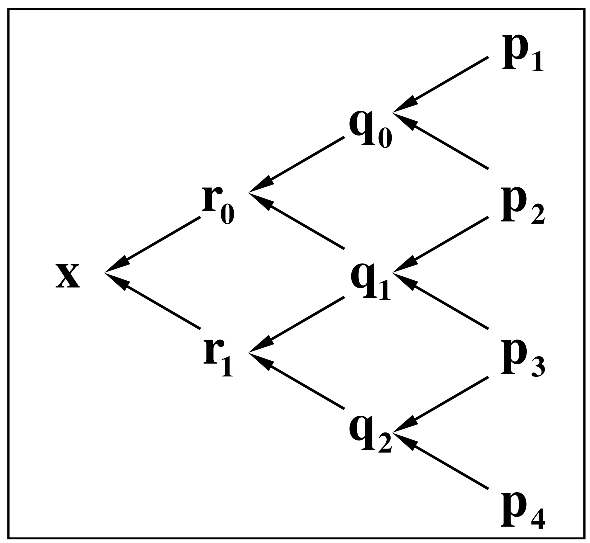
$$\mathbf{r}_1 = Lerp(t, \mathbf{q}_1, \mathbf{q}_2)$$

$$\mathbf{q}_0 = Lerp(t, \mathbf{p}_0, \mathbf{p}_1)$$

$$\mathbf{q}_1 = Lerp(t, \mathbf{p}_1, \mathbf{p}_2)$$

$$\mathbf{q}_2 = Lerp(t, \mathbf{p}_2, \mathbf{p}_3)$$

$$\mathbf{p}_0 \quad \mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3$$



Expand the LERPs

$$\mathbf{q}_0(t) = Lerp\left(t, \mathbf{p}_0, \mathbf{p}_1\right) = (1-t)\mathbf{p}_0 + t\mathbf{p}_1$$

$$\mathbf{q}_1(t) = Lerp\left(t, \mathbf{p}_1, \mathbf{p}_2\right) = (1-t)\mathbf{p}_1 + t\mathbf{p}_2$$

$$\mathbf{q}_2(t) = Lerp\left(t, \mathbf{p}_2, \mathbf{p}_3\right) = (1-t)\mathbf{p}_2 + t\mathbf{p}_3$$

$$\mathbf{r}_0(t) = Lerp\left(t, \mathbf{q}_0(t), \mathbf{q}_1(t)\right) = (1-t)\left((1-t)\mathbf{p}_0 + t\mathbf{p}_1\right) + t\left((1-t)\mathbf{p}_1 + t\mathbf{p}_2\right)$$

$$\mathbf{r}_1(t) = Lerp\left(t, \mathbf{q}_1(t), \mathbf{q}_2(t)\right) = (1-t)\left((1-t)\mathbf{p}_1 + t\mathbf{p}_2\right) + t\left((1-t)\mathbf{p}_2 + t\mathbf{p}_3\right)$$

$$\begin{aligned} \mathbf{x}(t) &= Lerp\left(t, \mathbf{r}_0(t), \mathbf{r}_1(t)\right) \\ &= (1-t)\left((1-t)\left((1-t)\mathbf{p}_0 + t\mathbf{p}_1\right) + t\left((1-t)\mathbf{p}_1 + t\mathbf{p}_2\right)\right) \\ &\qquad + t\left((1-t)\left((1-t)\mathbf{p}_1 + t\mathbf{p}_2\right) + t\left((1-t)\mathbf{p}_2 + t\mathbf{p}_3\right)\right) \end{aligned}$$

Weighted Average of Control Points

- Regroup

$$\mathbf{x}(t) = (1-t)\Big((1-t)\big((1-t)\mathbf{p}_0 + t\mathbf{p}_1\big) + t\big((1-t)\mathbf{p}_1 + t\mathbf{p}_2\big)\Big)$$
$$+t\Big((1-t)\big((1-t)\mathbf{p}_1 + t\mathbf{p}_2\big) + t\big((1-t)\mathbf{p}_2 + t\mathbf{p}_3\big)\Big)$$

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

$$\mathbf{x}(t) = \overbrace{\left(-t^3 + 3t^2 - 3t + 1\right)}^{B_0(t)}\mathbf{p}_0 + \overbrace{\left(3t^3 - 6t^2 + 3t\right)}^{B_1(t)}\mathbf{p}_1$$
$$+\underbrace{\left(-3t^3 + 3t^2\right)}_{B_2(t)}\mathbf{p}_2 + \underbrace{\left(t^3\right)}_{B_3(t)}\mathbf{p}_3$$

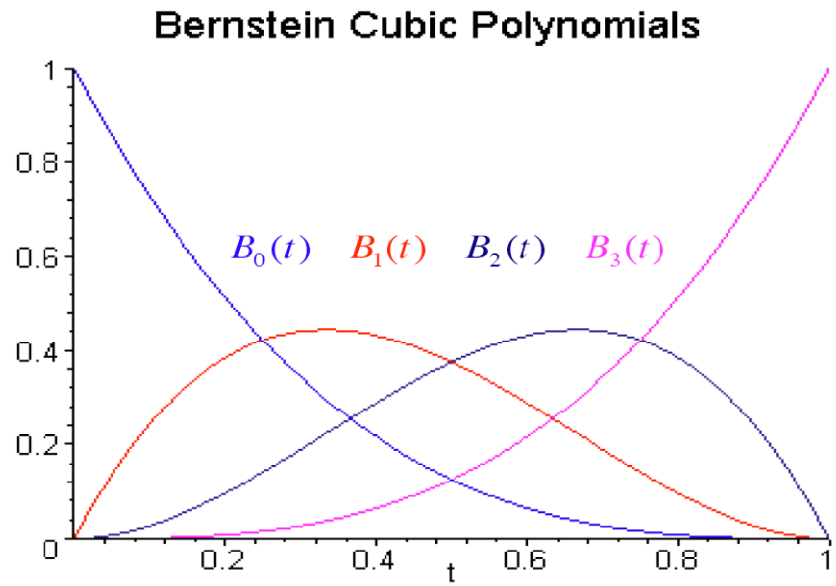
Cubic Bernstein Polynomials

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

The cubic *Bernstein polynomials :*

$$B_0(t) = -t^3 + 3t^2 - 3t + 1$$
$$B_1(t) = 3t^3 - 6t^2 + 3t$$
$$B_2(t) = -3t^3 + 3t^2$$
$$B_3(t) = t^3$$

$$\sum B_i(t) = 1$$



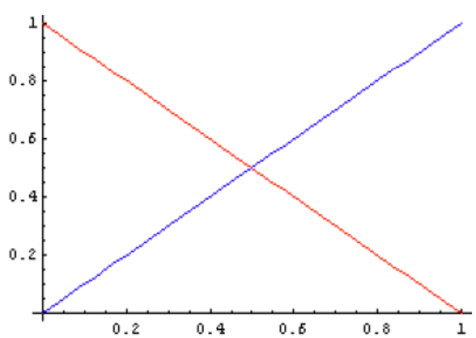
- Partition of unity, weights always add up to 1
- Endpoint interpolation, B_0 and B_3 go to 1

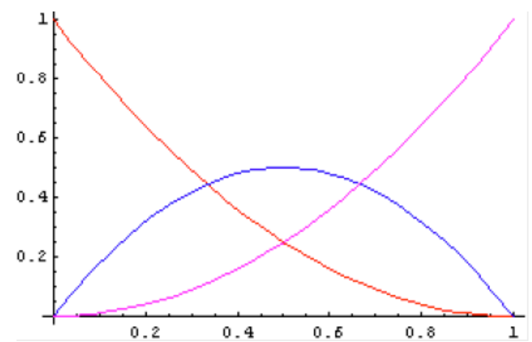
General Bernstein Polynomials

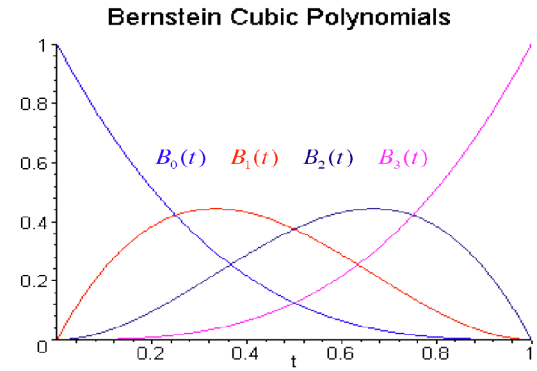
$$B_0^1(t) = -t + 1$$
$$B_1^1(t) = t$$

$$B_0^2(t) = t^2 - 2t + 1$$
$$B_1^2(t) = -2t^2 + 2t$$
$$B_2^2(t) = t^2$$

$$B_0^3(t) = -t^3 + 3t^2 - 3t + 1$$
$$B_1^3(t) = 3t^3 - 6t^2 + 3t$$
$$B_2^3(t) = -3t^3 + 3t^2$$
$$B_3^3(t) = t^3$$







$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}(t)^i \qquad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$

$$\sum B_i^n(t) = 1$$

n! = factorial of n
(n+1)! = n! x (n+1)

General Bézier Curves

- nth-order Bernstein polynomials form nth-order Bézier curves

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}(t)^i$$

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

Bézier Curve Properties

Overview:

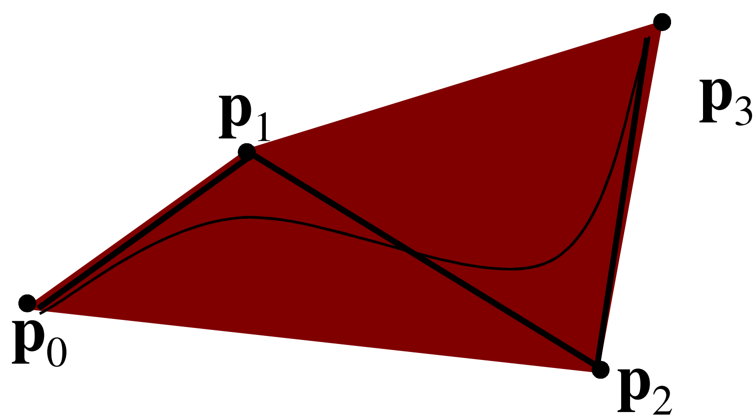
- Convex Hull property
- Variation Diminishing property
- Affine Invariance

Definitions

- Convex hull of a set of points:
  - Polyhedral volume created such that all lines connecting any two points lie completely inside it (or on its boundary)
- Convex combination of a set of points:
  - Weighted average of the points, where all weights between 0 and 1, sum up to 1
- Any convex combination always of a set of points lies within the convex hull

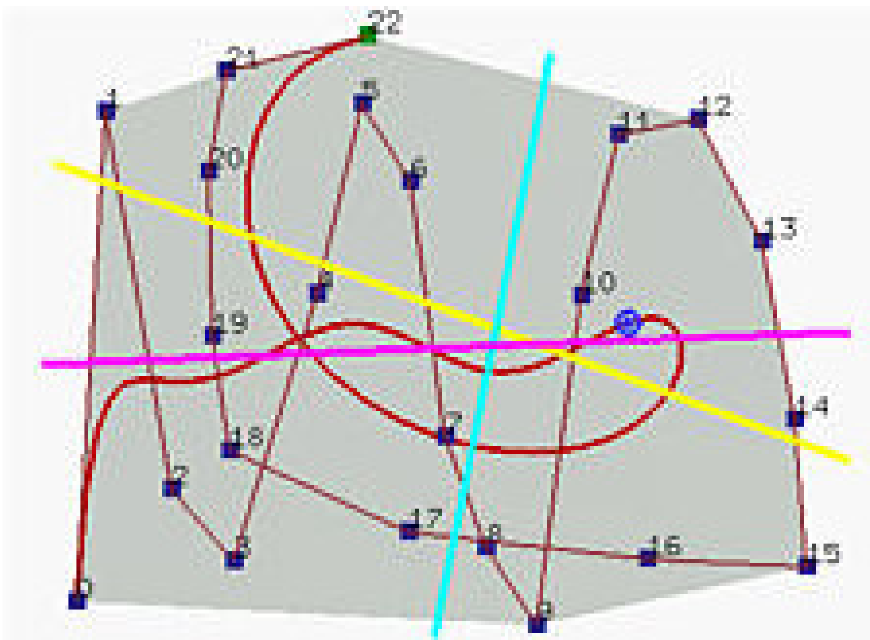
Convex Hull Property

- A Bézier curve is a convex combination of the control points (by definition, see Bernstein polynomials)
- Bézier curve is always inside the convex hull
  - Makes curve predictable
  - Allows culling, intersection testing, adaptive tessellation



Variation Diminishing Property

- If the curve is in a plane, this means no straight line intersects a Bézier curve more times than it intersects the curve's control polyline
- “Curve is not more wiggly than control polyline”



Affine Invariance

Transforming Bézier curves

- Two ways to transform:
  - Transform the control points, then compute resulting spline points
  - Compute spline points then transform them
- Either way, we get the same points
  - Curve is defined via affine combination of points
  - Invariant under affine transformations
  - Convex hull property remains true

Cubic Polynomial Form

Start with Bernstein form:

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

Regroup into coefficients of t :

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

$$\boxed{\mathbf{x}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d} \qquad \begin{aligned} \mathbf{a} &= \left(-\mathbf{p}_0 + 3\mathbf{p}_1 - 3\mathbf{p}_2 + \mathbf{p}_3\right) \\ \mathbf{b} &= \left(3\mathbf{p}_0 - 6\mathbf{p}_1 + 3\mathbf{p}_2\right) \\ \mathbf{c} &= \left(-3\mathbf{p}_0 + 3\mathbf{p}_1\right) \\ \mathbf{d} &= \left(\mathbf{p}_0\right) \end{aligned}}$$

- Good for fast evaluation
  - Precompute constant coefficients $(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d})$
- Not much geometric intuition

Cubic Matrix Form

$$\mathbf{x}(t) = \begin{bmatrix} \vec{\mathbf{a}} & \vec{\mathbf{b}} & \vec{\mathbf{c}} & \mathbf{d} \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$\vec{\mathbf{a}} = \left(-\mathbf{p}_0 + 3\mathbf{p}_1 - 3\mathbf{p}_2 + \mathbf{p}_3\right)$$
$$\vec{\mathbf{b}} = \left(3\mathbf{p}_0 - 6\mathbf{p}_1 + 3\mathbf{p}_2\right)$$
$$\vec{\mathbf{c}} = \left(-3\mathbf{p}_0 + 3\mathbf{p}_1\right)$$
$$\mathbf{d} = \left(\mathbf{p}_0\right)$$

$$\mathbf{x}(t) = \underbrace{\begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 \end{bmatrix}}_{\mathbf{G}_{Bez}} \underbrace{\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{B}_{Bez}} \underbrace{\begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}}_{\mathbf{T}}$$

- Other cubic splines use different basis matrix $\mathbf{B}$
  - Hermite, Catmull-Rom, B-Spline, …

Cubic Matrix Form

- In 3D: 3 parallel equations for x, y and z:

$$\mathbf{x}_x(t) = \begin{bmatrix} p_{0x} & p_{1x} & p_{2x} & p_{3x} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$\mathbf{x}_y(t) = \begin{bmatrix} p_{0y} & p_{1y} & p_{2y} & p_{3y} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$\mathbf{x}_z(t) = \begin{bmatrix} p_{0z} & p_{1z} & p_{2z} & p_{3z} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Matrix Form

- Bundle into a single matrix

$$\mathbf{x}(t) = \begin{bmatrix} p_{0x} & p_{1x} & p_{2x} & p_{3x} \\ p_{0y} & p_{1y} & p_{2y} & p_{3y} \\ p_{0z} & p_{1z} & p_{2z} & p_{3z} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$\boxed{\begin{aligned} \mathbf{x}(t) &= \mathbf{G}_{Bez}\mathbf{B}_{Bez}\mathbf{T} \\ \mathbf{x}(t) &= \mathbf{C}\ \mathbf{T} \end{aligned}}$$

- Efficient evaluation
  - Precompute $\mathbf{C}$
  - Take advantage of existing 4x4 matrix hardware support

Lecture Overview

- Bézier curves
- Drawing Bézier curves
- Piecewise Bézier curves

Drawing Bézier Curves

- Draw *line segments* or individual pixels
- Approximate the curve as a series of line segments (*tessellation*)
  - Uniform sampling
  - Adaptive sampling
  - Recursive subdivision

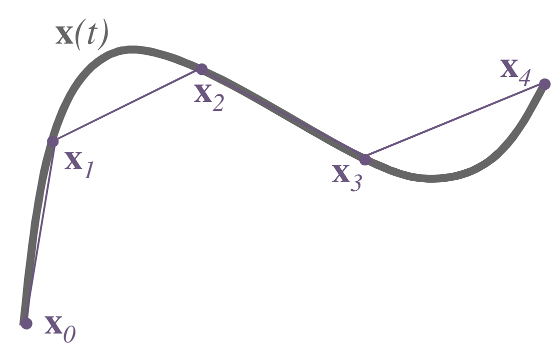
Uniform Sampling

- Approximate curve with N straight segments
  - N chosen in advance
  - Evaluate

$$\mathbf{x}_i = \mathbf{x}(t_i) \text{ where } t_i = \frac{i}{N} \text{ for } i = 0, 1, \ldots, N$$

$$\mathbf{x}_i = \vec{\mathbf{a}}\frac{i^3}{N^3} + \vec{\mathbf{b}}\frac{i^2}{N^2} + \vec{\mathbf{c}}\frac{i}{N} + \mathbf{d}$$

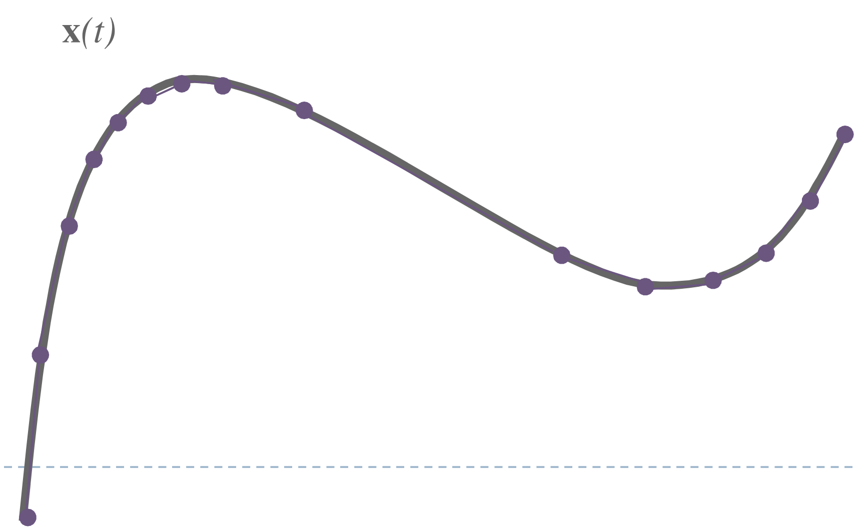
  - Connect the points with lines



- Too few points?
  - Poor approximation
  - "Curve" is faceted
- Too many points?
  - Slow to draw too many line segments
  - Segments may draw on top of each other

Adaptive Sampling

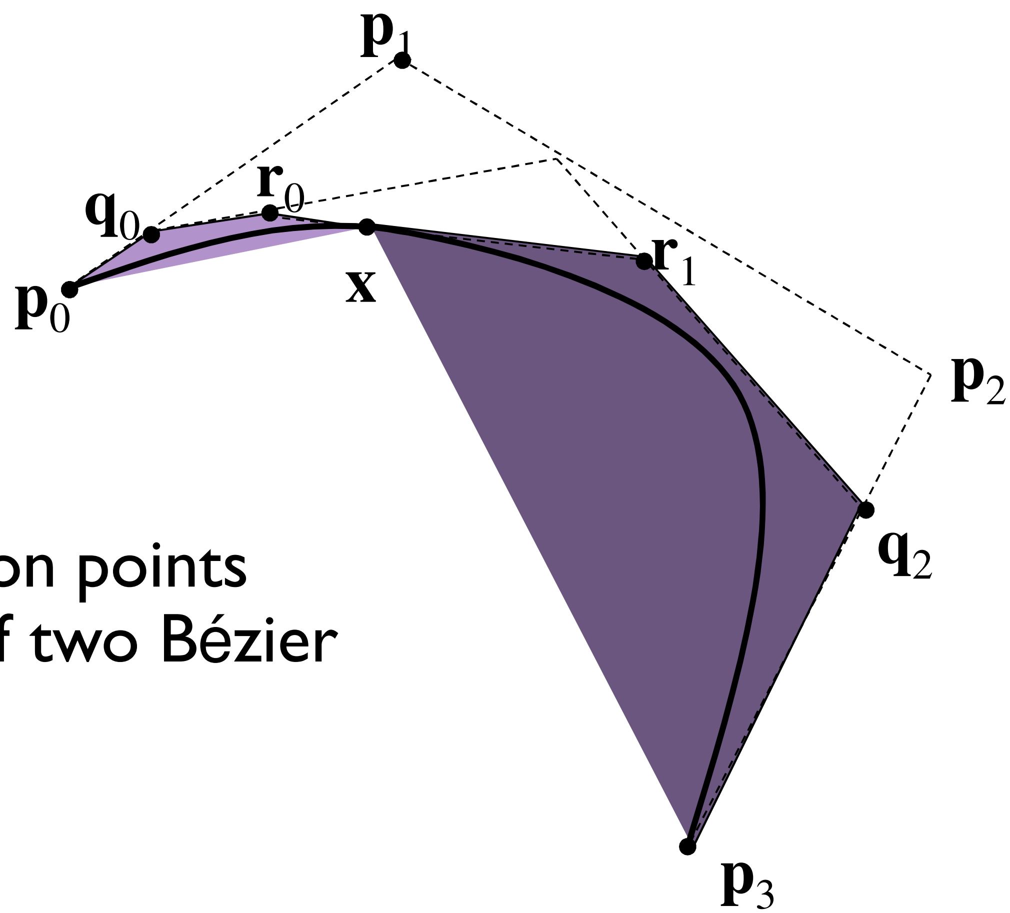
- Use only as many line segments as you need
  - Fewer segments where curve is mostly flat
  - More segments where curve bends
  - Segments never smaller than a pixel
- Various schemes for sampling, checking results, deciding whether to sample more



Recursive Subdivision

- Any cubic curve segment can be expressed as a Bézier curve
- Any piece of a cubic curve is itself a cubic curve
- Therefore:
  - Any Bézier curve can be broken up into smaller Bézier curves

De Casteljau Subdivision



- De Casteljau construction points are the control points of two Bézier sub-segments

Adaptive Subdivision Algorithm

- Use de Casteljau construction to split Bézier segment
- For each half
  - If flat enough: draw line segment
  - Else: recurse
- Curve is flat enough if hull is flat enough
- Test how far the handles are from a straight segment
  - If it is about the distance of a pixel, the hull is flat enough

Drawing Bézier Curves With OpenGL

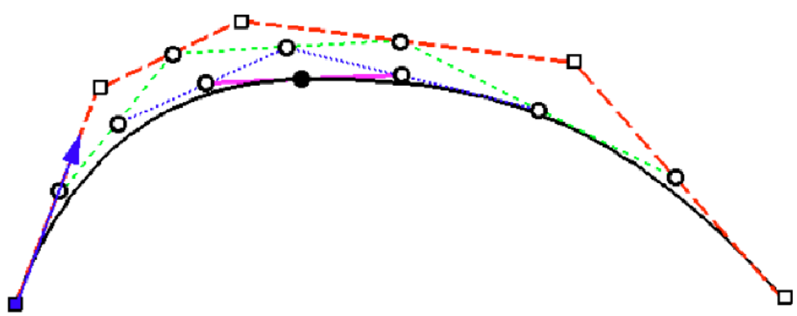
- Indirect OpenGL support for drawing curves:
  - Define evaluator map (`glMap`)
  - Draw line strip by evaluating map (`glEvalCoord`)
  - Optimize by pre-computing coordinate grid (`glMapGrid` and `glEvalMesh`)
- More details about OpenGL implementation:
  - http://www.cs.duke.edu/courses/fall09/cps124/notes/12_curves/opengl_nurbs.pdf

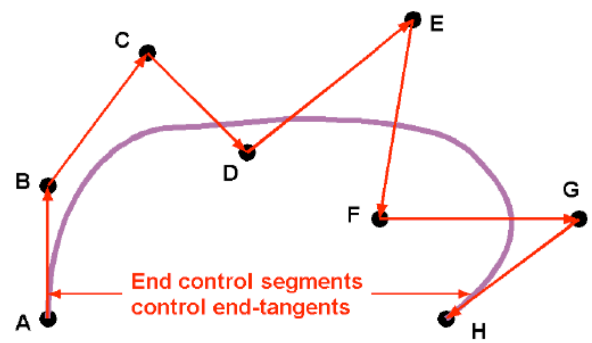
Lecture Overview

- Bézier curves
- Drawing Bézier curves
- Piecewise Bézier curves

More Control Points

- Cubic Bézier curve limited to 4 control points
  - Cubic curve can only have one inflection (point where curve changes direction of bending)
  - Need more control points for more complex curves
- k-1 order Bézier curve with k control points

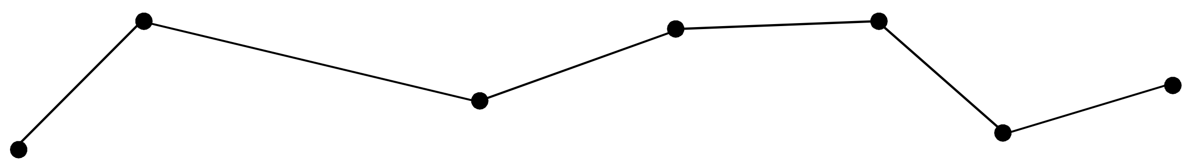




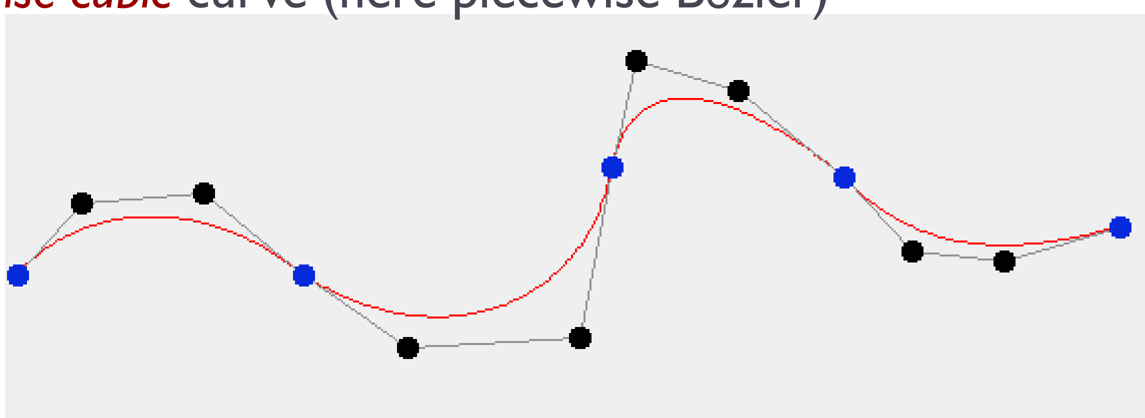
- Hard to control and hard to work with
  - Intermediate points don't have obvious effect on shape
  - Changing any control point changes the whole curve
  - Want *local support*: each control point only influences nearby portion of curve

Piecewise Curves

- Sequence of simple (low-order) curves, end-to-end
  - Known as a *piecewise polynomial curve*
- Sequence of line segments
  - *Piecewise linear* curve

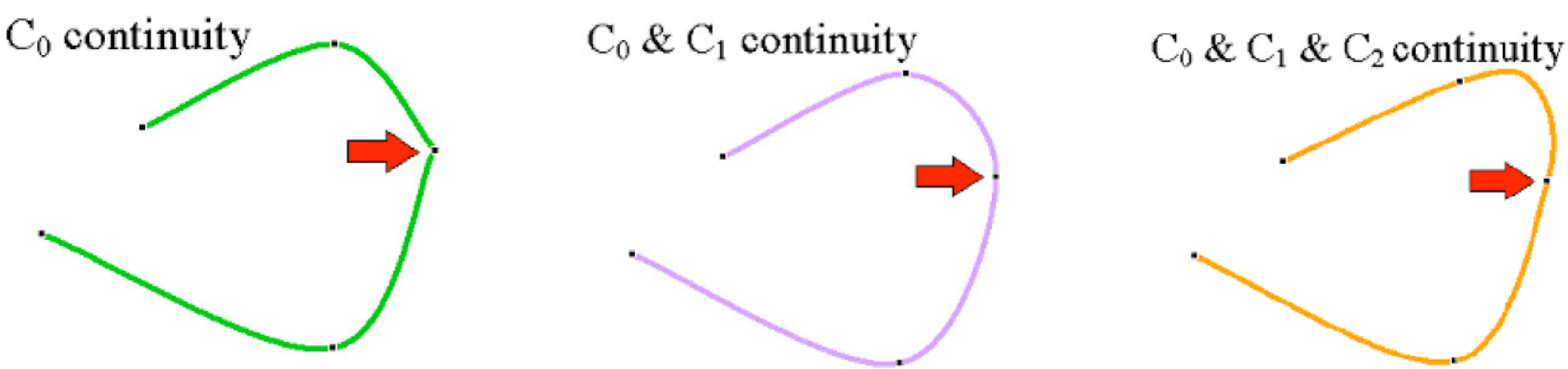


- Sequence of cubic curve segments
  - *Piecewise cubic* curve (here piecewise Bézier)



Continuity

- Goal: smooth curves
- C^0 continuity
  - No gaps
  - Segments meet at the endpoints
- C^1 continuity: first derivative is well defined
  - No corners
  - Tangents/normals are C^0 continuous (no jumps)
- C^2 continuity: second derivative is well defined
  - Tangents/normals are C^1 continuous
  - Important for high quality reflections



Global Parameterization

- Given N curve segments $\mathbf{x}_0(t)$, $\mathbf{x}_1(t)$, …, $\mathbf{x}_{N-1}(t)$
- Each is parameterized for t from 0 to 1
- Define a piecewise curve
  - Global parameter u from 0 to N

$$\mathbf{x}(u) = \begin{cases} \mathbf{x}_0(u), & 0 \le u \le 1 \\ \mathbf{x}_1(u-1), & 1 \le u \le 2 \\ \vdots & \vdots \\ \mathbf{x}_{N-1}(u-(N-1)), & N-1 \le u \le N \end{cases}$$

$$\mathbf{x}(u) = \mathbf{x}_i(u-i), \text{ where } i = \lfloor u \rfloor \quad (\text{and } \mathbf{x}(N) = \mathbf{x}_{N-1}(1))$$

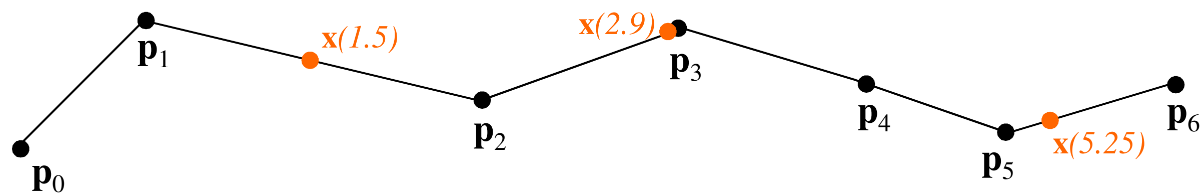
- Alternate: solution u also goes from 0 to 1

$$\mathbf{x}(u) = \mathbf{x}_i(Nu-i), \text{ where } i = \lfloor Nu \rfloor$$

Piecewise-Linear Curve

- Given N+1 points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_N$
- Define curve

$$\mathbf{x}(u) = Lerp(u - i, \mathbf{p}_i, \mathbf{p}_{i+1}), \qquad i \le u \le i+1$$
$$= (1 - u + i)\mathbf{p}_i + (u - i)\mathbf{p}_{\mathbf{i+1}}, \quad i = \lfloor u \rfloor$$

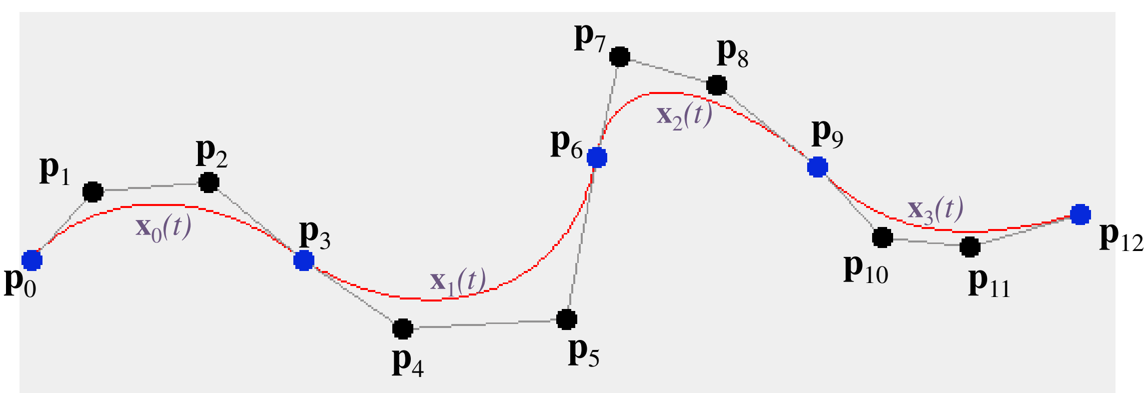


- N+1 points define N linear segments
- $\mathbf{x}(i)=\mathbf{p}_i$
- C^0 continuous by construction
- C^1 at $\mathbf{p}_i$ when $\mathbf{p}_i - \mathbf{p}_{i-1} = \mathbf{p}_{i+1} - \mathbf{p}_i$

Piecewise Bézier curve

- Given $3N+1$ points $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{3N}$
- Define N Bézier segments:

$$
\begin{aligned}
\mathbf{x}_0(t) &= B_0(t)\mathbf{p}_0 + B_1(t)\mathbf{p}_1 + B_2(t)\mathbf{p}_2 + B_3(t)\mathbf{p}_3 \\
\mathbf{x}_1(t) &= B_0(t)\mathbf{p}_3 + B_1(t)\mathbf{p}_4 + B_2(t)\mathbf{p}_5 + B_3(t)\mathbf{p}_6 \\
&\vdots \\
\mathbf{x}_{N-1}(t) &= B_0(t)\mathbf{p}_{3N-3} + B_1(t)\mathbf{p}_{3N-2} + B_2(t)\mathbf{p}_{3N-1} + B_3(t)\mathbf{p}_{3N}
\end{aligned}
$$

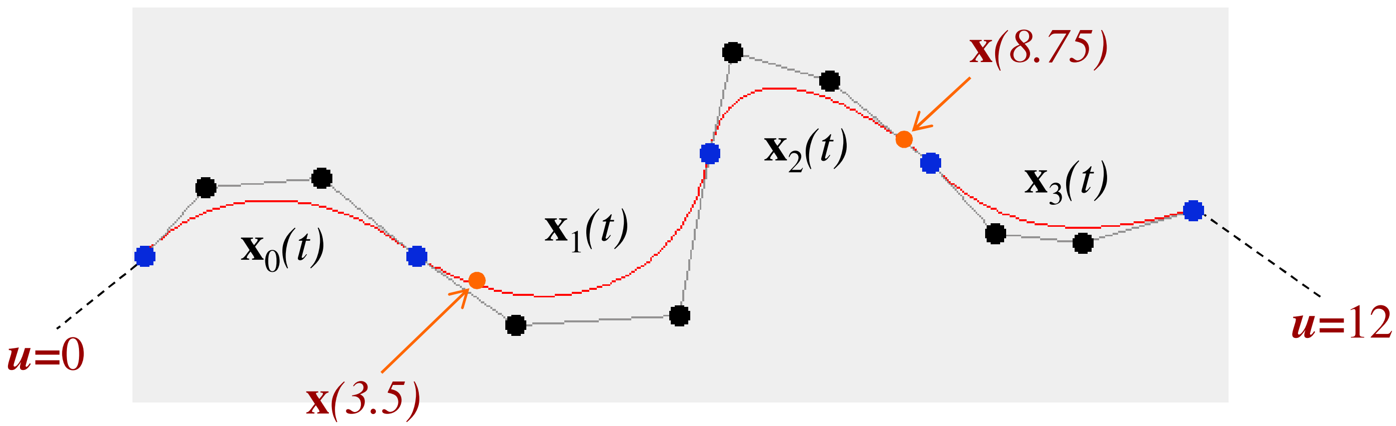


Piecewise Bézier Curve

- Parameter in $0 <= u <= 3N$

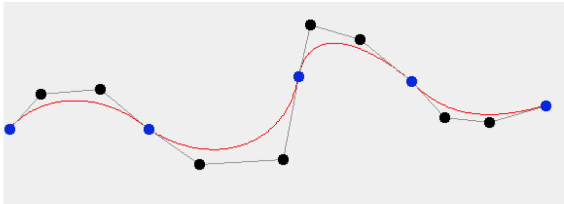
$$\mathbf{x}(u) = \begin{cases} \mathbf{x}_0\left(\tfrac{1}{3}u\right), & 0 \le u \le 3 \\ \mathbf{x}_1\left(\tfrac{1}{3}u - 1\right), & 3 \le u \le 6 \\ \vdots & \vdots \\ \mathbf{x}_{N-1}\left(\tfrac{1}{3}u - (N-1)\right), & 3N-3 \le u \le 3N \end{cases}$$

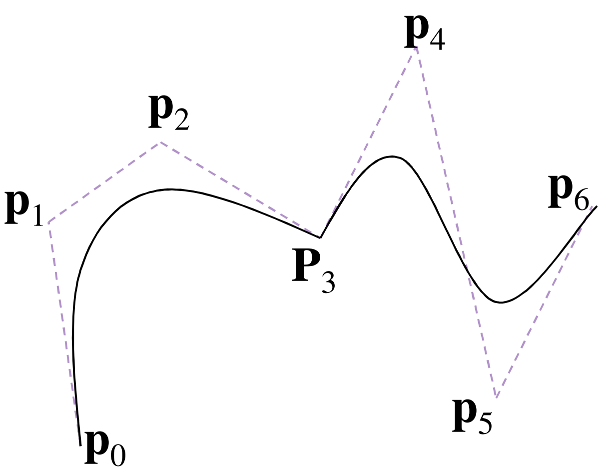
$$\mathbf{x}(u) = \mathbf{x}_i\left(\tfrac{1}{3}u - i\right), \text{ where } i = \left\lfloor \tfrac{1}{3}u \right\rfloor$$



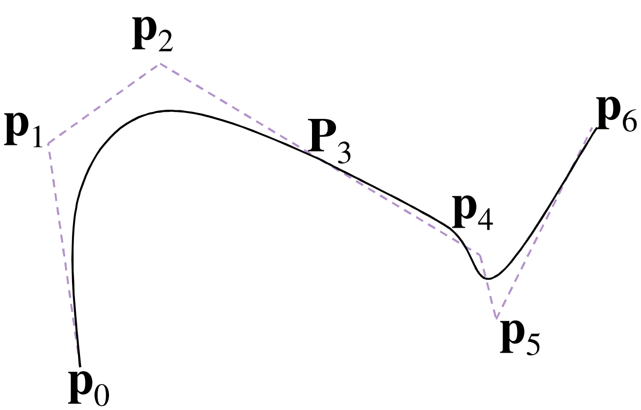
Piecewise Bézier Curve

- $3N+1$ points define N Bézier segments
- $\mathbf{x}(3i)=\mathbf{p}_{3i}$
- C^0 continuous by construction
- C^1 continuous at $\mathbf{p}_{3i}$ when $\mathbf{p}_{3i} - \mathbf{p}_{3i-1} = \mathbf{p}_{3i+1} - \mathbf{p}_{3i}$
- C^2 is harder to achieve





C^1 discontinuous



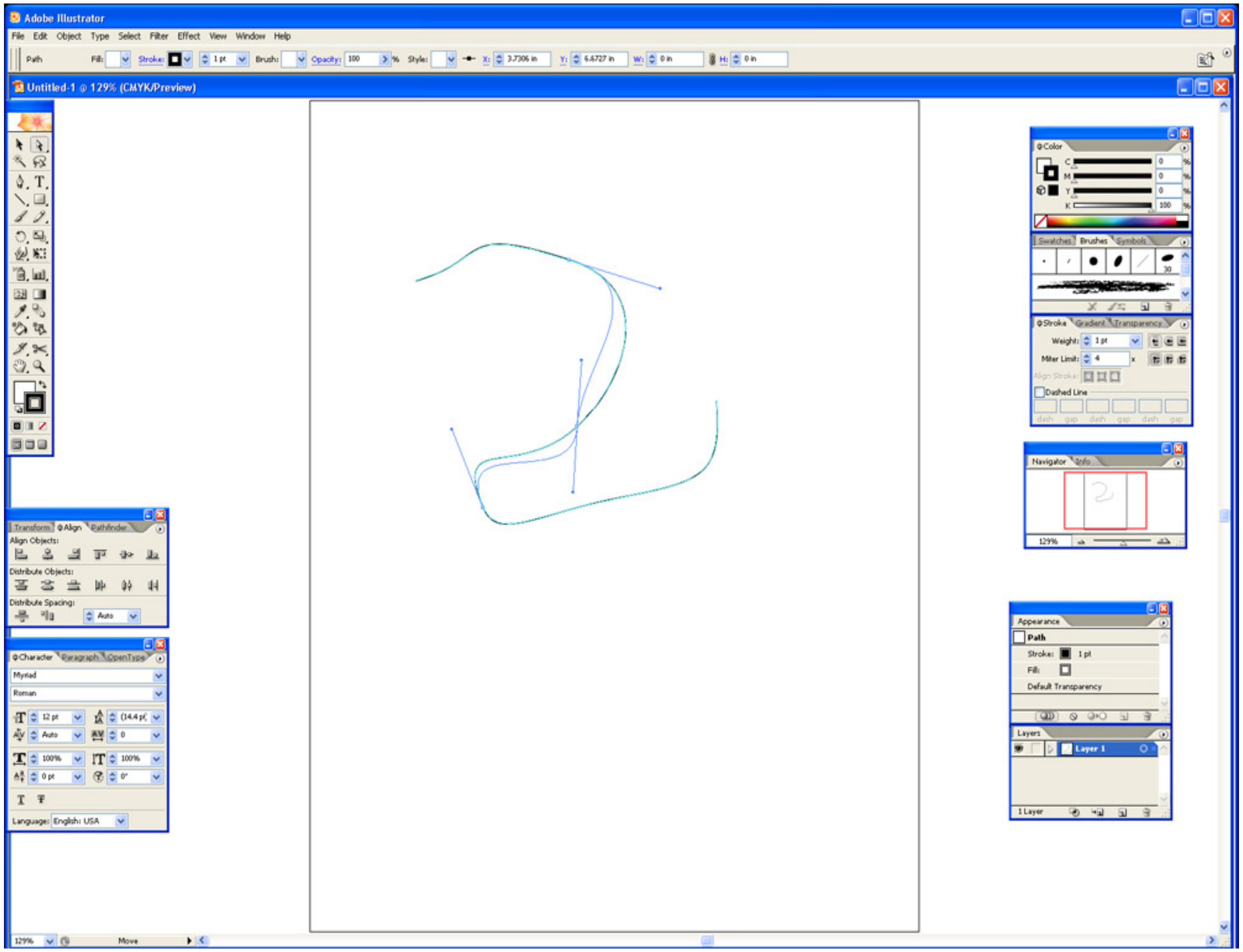
C^1 continuous

Piecewise Bézier Curves

- Used often in 2D drawing programs
- Inconveniences
  - Must have 4 or 7 or 10 or 13 or … (1 plus a multiple of 3) control points
  - Some points interpolate, others approximate
  - Need to impose constraints on control points to obtain C^1 continuity
  - C^2 continuity more difficult
- Solutions
  - User interface using "Bézier handles"
  - Generalization to B-splines or NURBS (details later)

Bézier Handles

- Segment end points (interpolating) presented as curve control points
- Midpoints (approximating points) presented as "handles"
- Can have option to enforce C^1 continuity



Adobe Illustrator

Next Lecture

- Midterm results
- Parametric surfaces