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

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2012

Announcements

- Homework project #5 due Nov. 9th at 1:30pm
  - To be presented in lab 260
- Veterans Day: reschedule homework introduction
  - Friday at 3:30pm?
  - Tuesday before or after class?
- In Winter: CSE 190: 3D User Interaction
  - 4 Units
  - 2 lectures (Mon/Wed 11am-12:20pm)
  - Programming assignments for 3D input devices

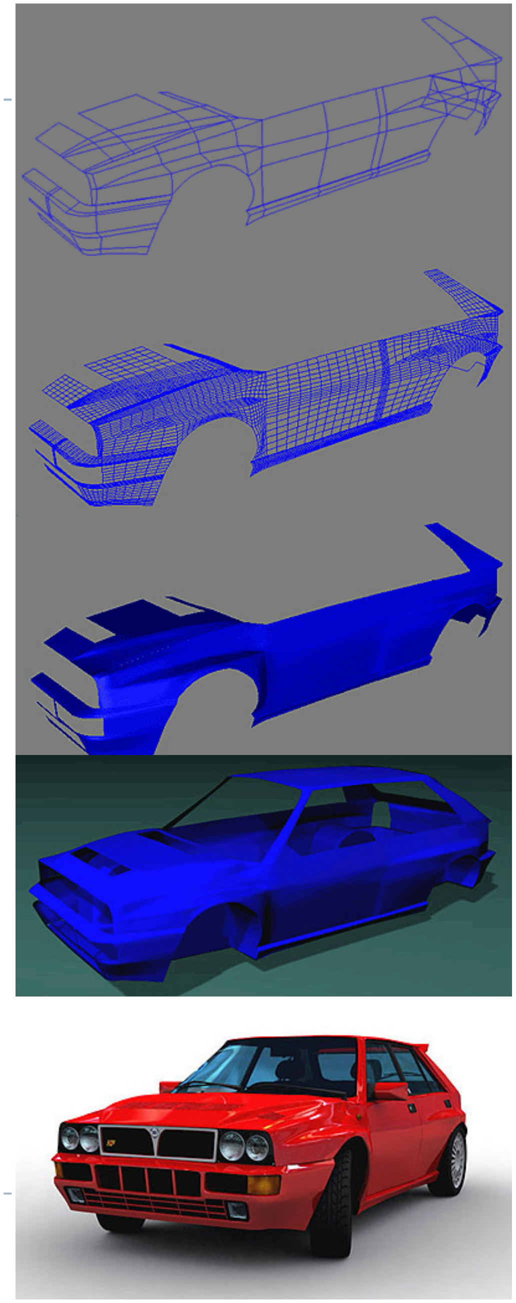
Lecture Overview

- Polynomial Curves
  - Introduction
  - Polynomial functions
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves

Modeling

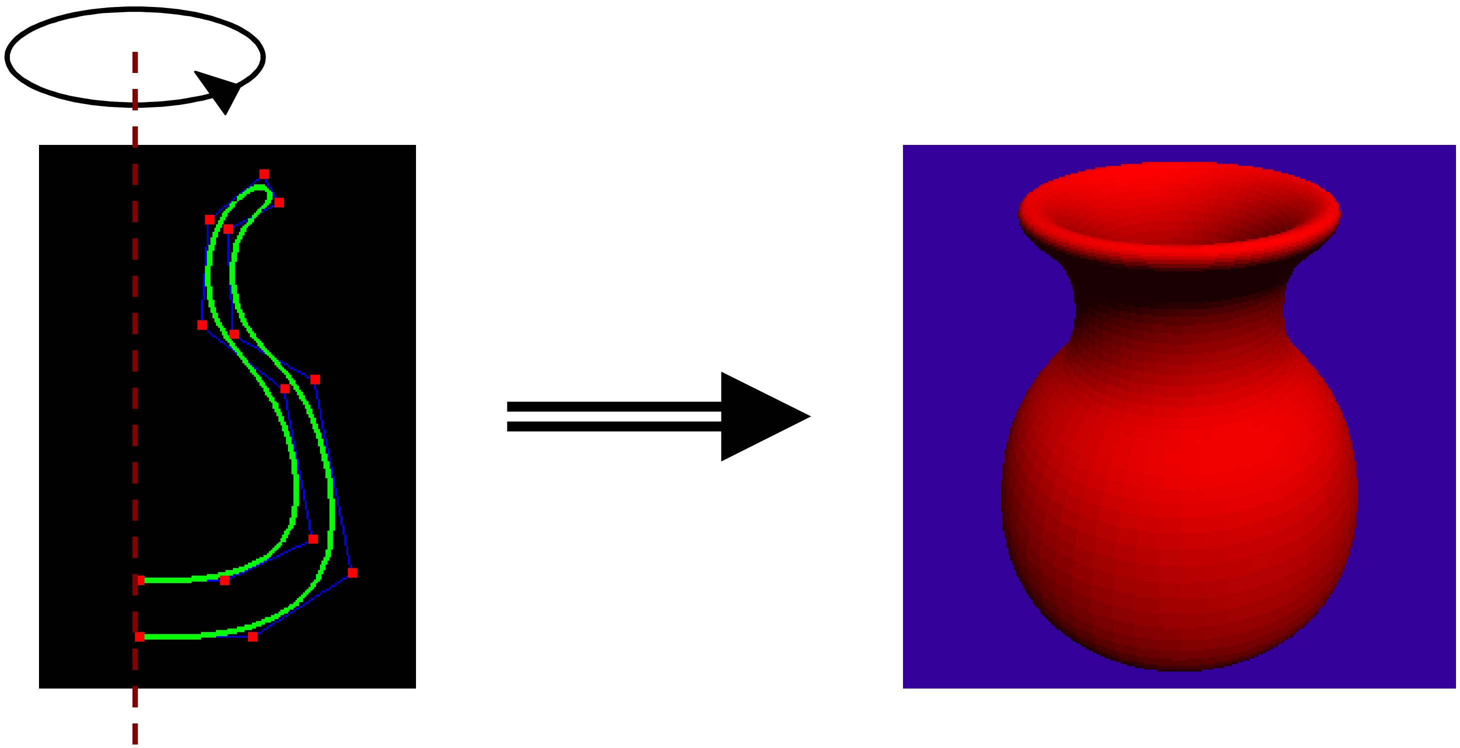
- Creating 3D objects
- How to construct complex surfaces?
- Goal
  - Specify objects with control points
  - Objects should be visually pleasing (smooth)
- Start with curves, then generalize to surfaces

- Next: What can curves be used for?



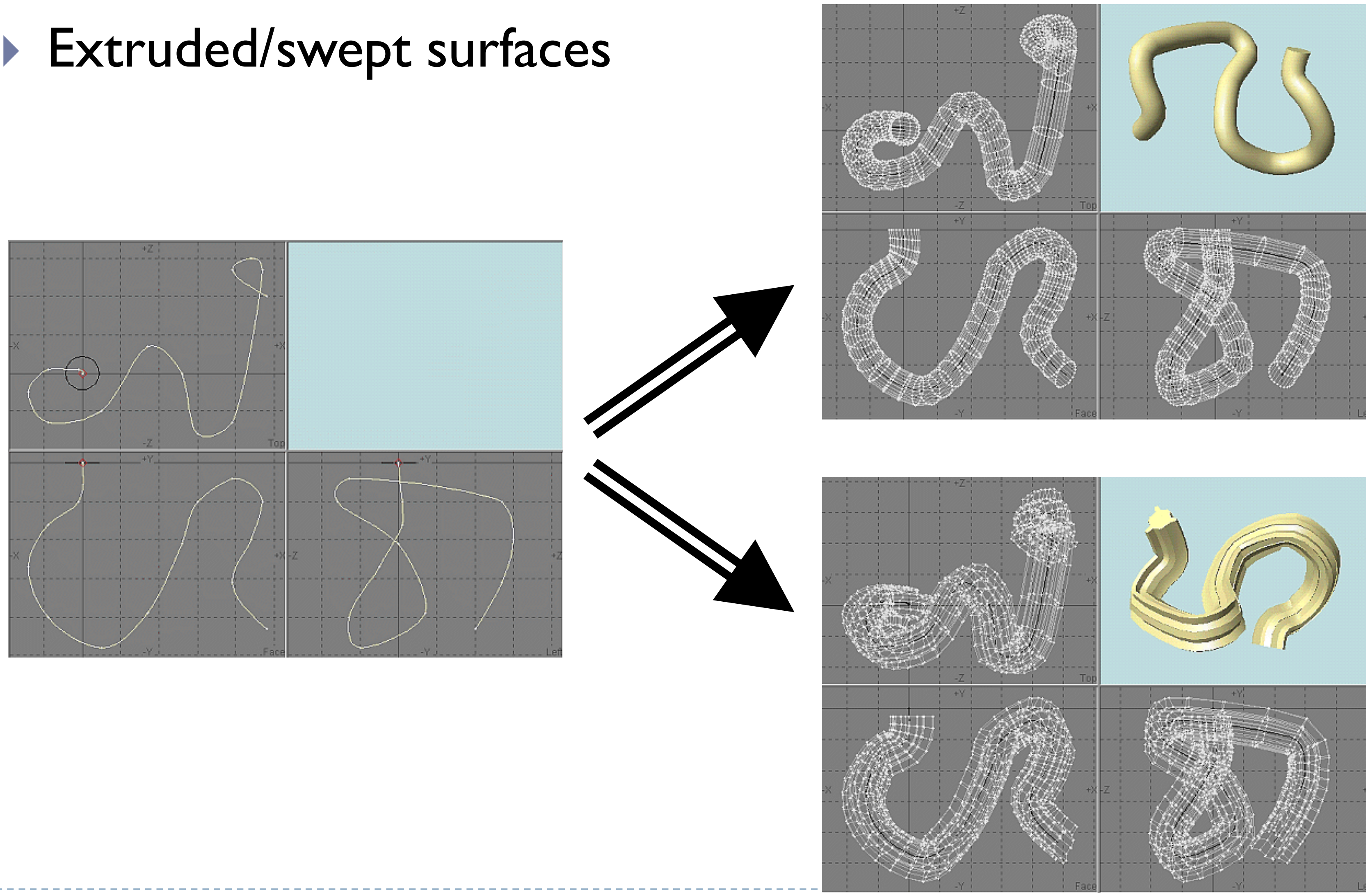
Curves

- Surface of revolution



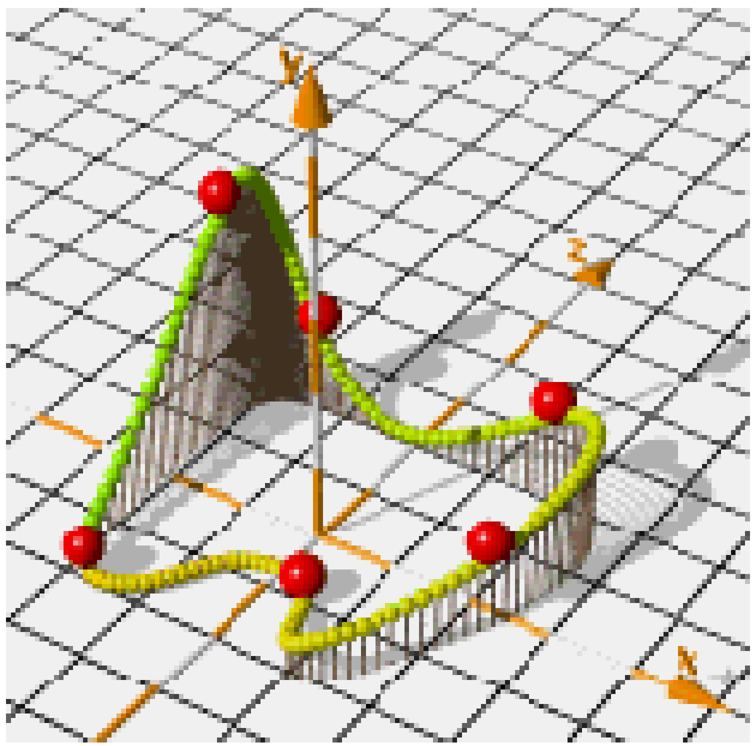
Curves

- Extruded/swept surfaces



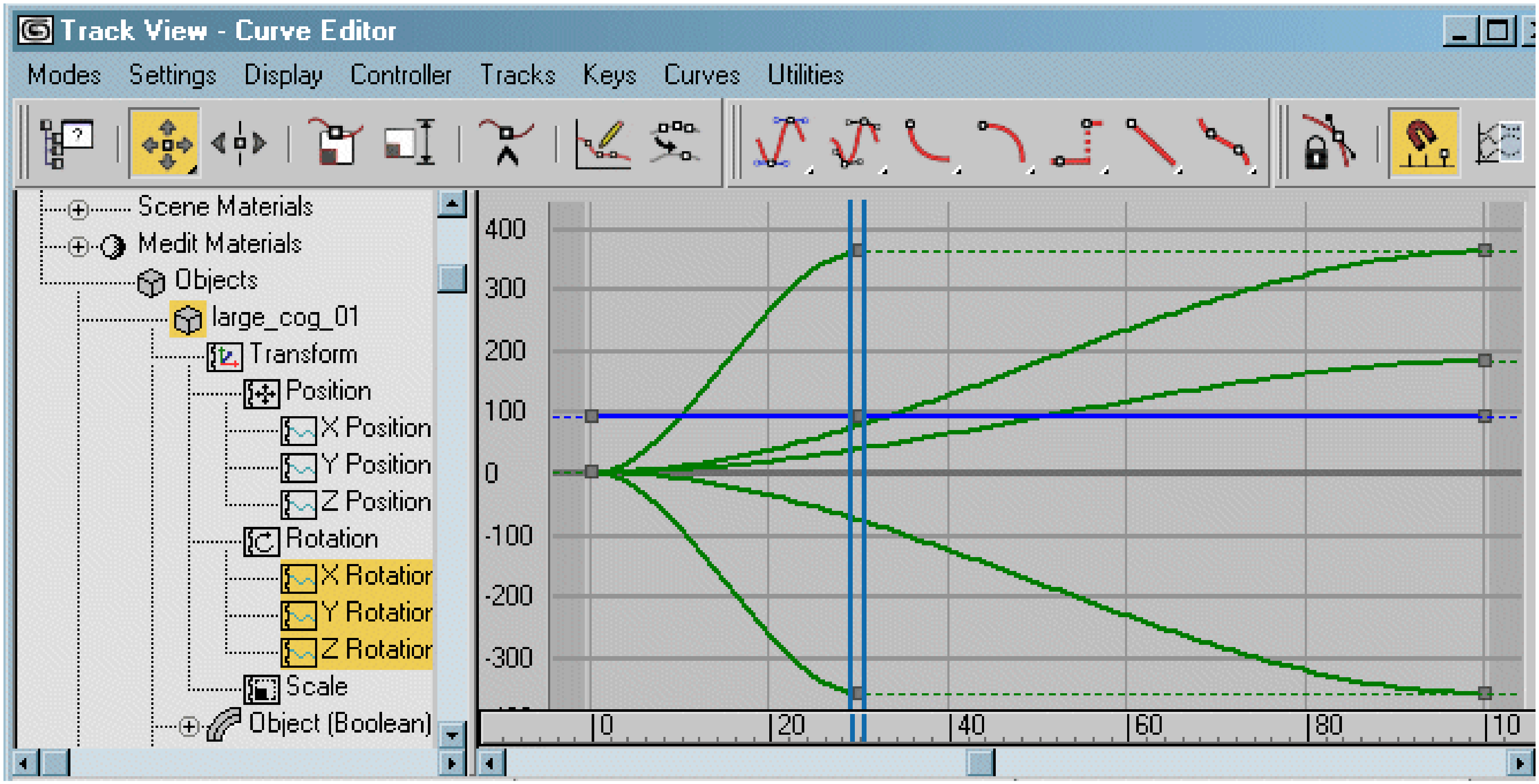
Curves

- Animation
  - Provide a "track" for objects
  - Use as camera path



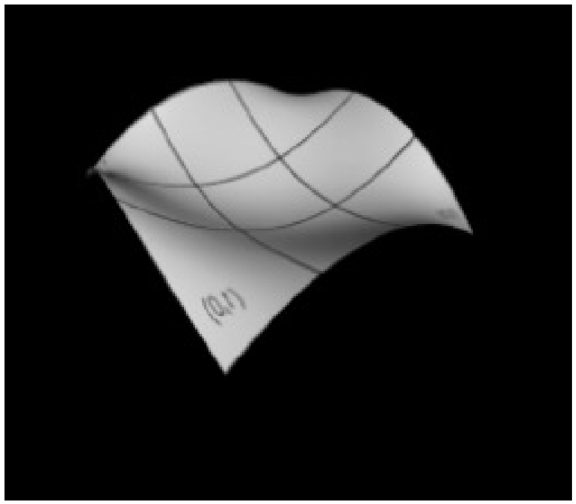
Curves

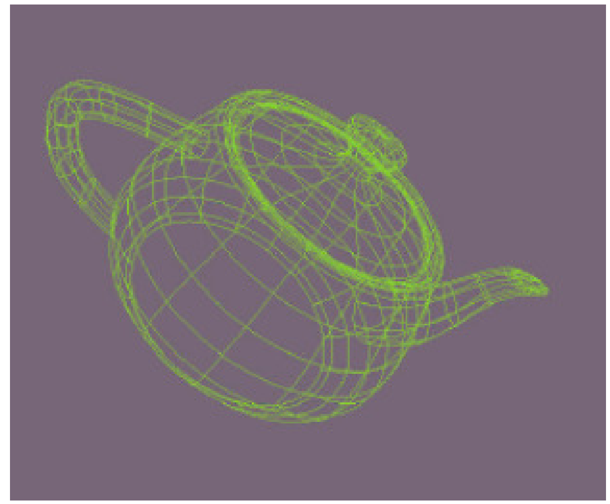
- Specify parameter values over time



Curves

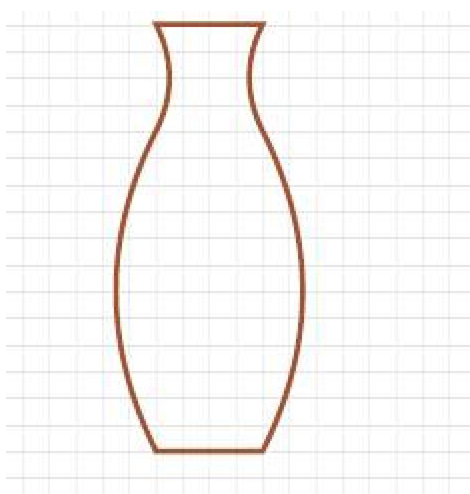
- Can be generalized to surface patches

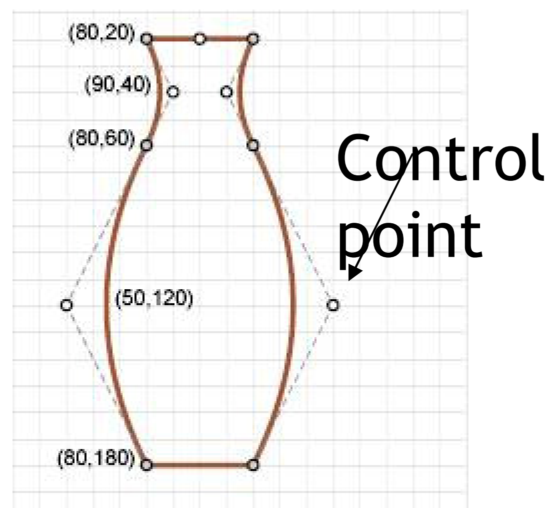




Curve Representation

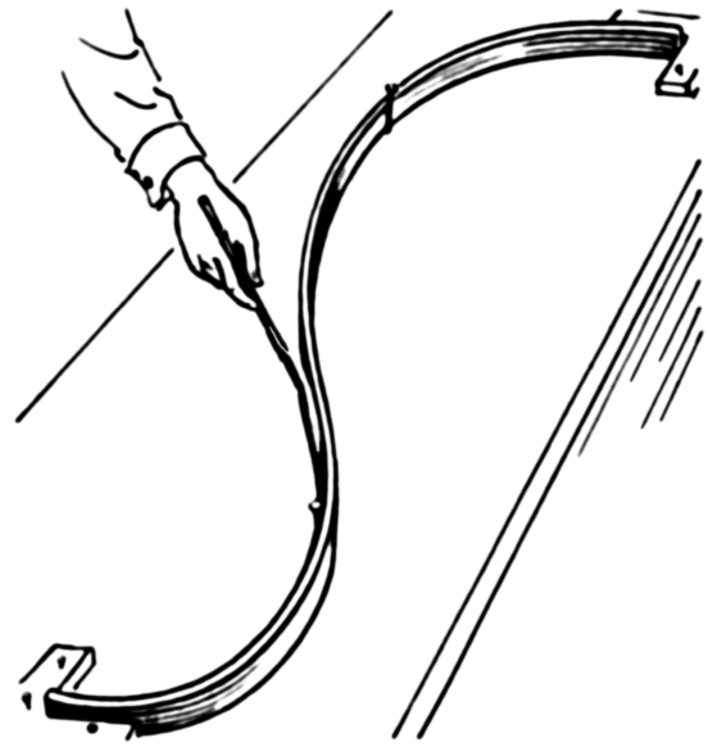
- Specify many points along a curve, connect with lines?
  - Difficult to get precise, smooth results across magnification levels
  - Large storage and CPU requirements
  - How many points are enough?
- Specify a curve using a small number of "control points"
  - Known as a *spline curve* or just *spline*





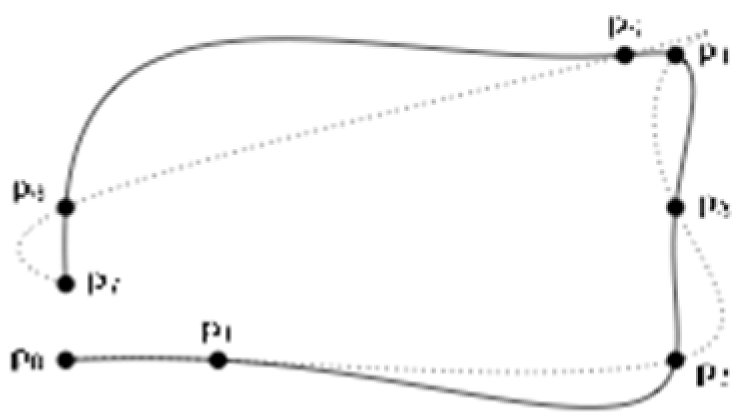
Spline: Definition

- Wikipedia:
  - Term comes from flexible spline devices used by shipbuilders and draftsmen to draw smooth shapes.
  - Spline consists of a long strip fixed in position at a number of points that relaxes to form a smooth curve passing through those points.



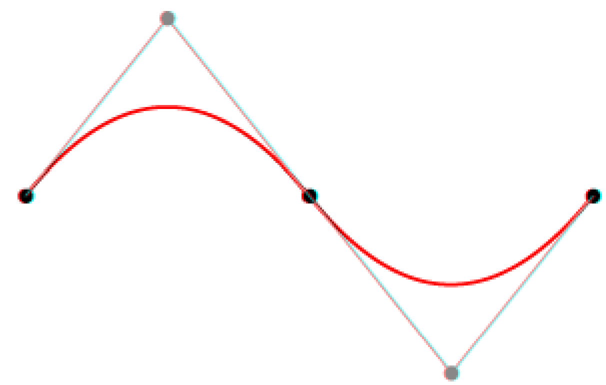
Interpolating Control Points

- "Interpolating" means that curve goes through all control points
- Seems most intuitive
- Surprisingly, not usually the best choice
  - Hard to predict behavior
  - Hard to get aesthetically pleasing curves



Approximating Control Points

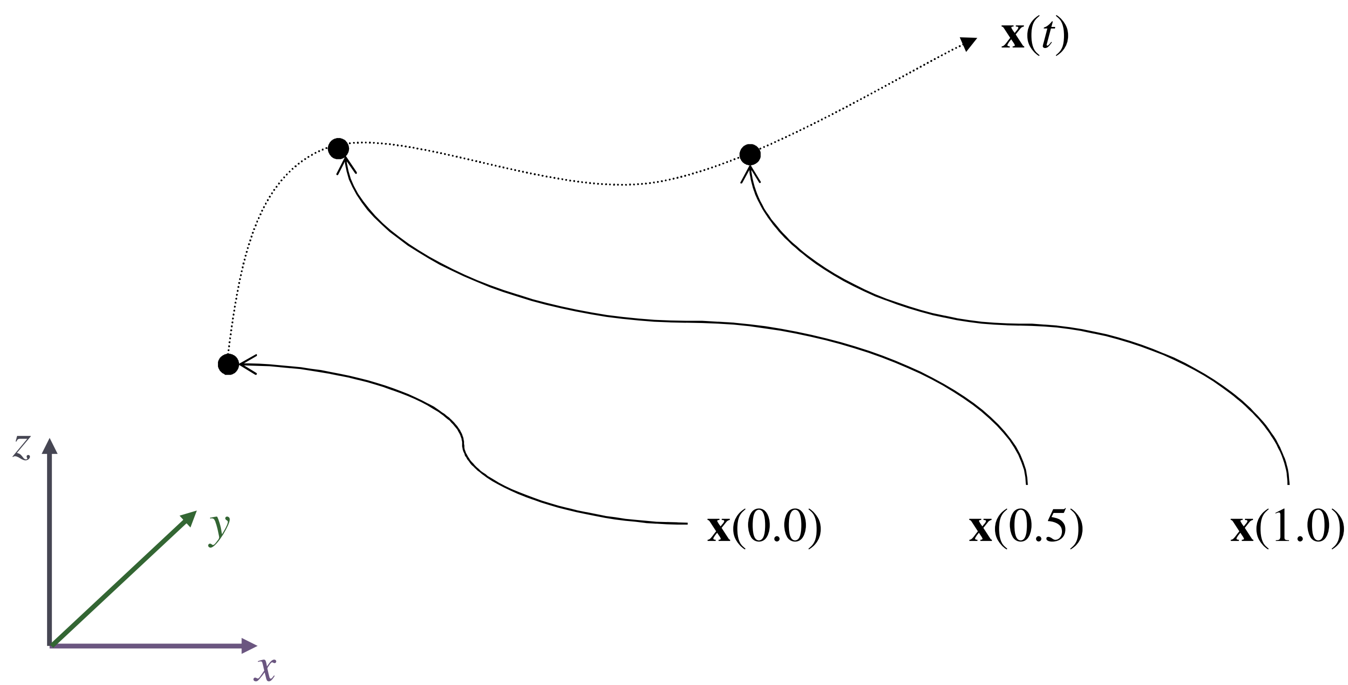
- Curve is "influenced" by control points



- Various types
- Most common: polynomial functions
  - Bézier spline (our main focus)
  - B-spline (generalization of Bézier spline)
  - NURBS (Non Uniform Rational Basis Spline): used in CAD tools

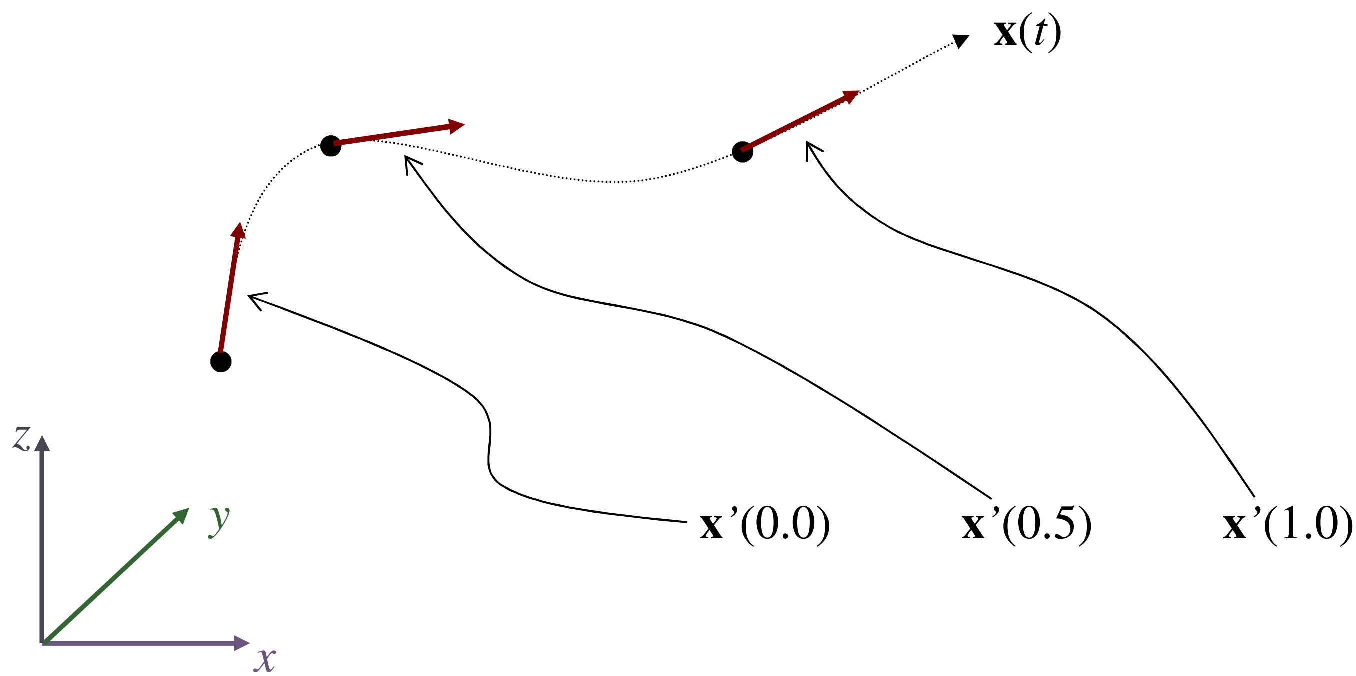
Mathematical Definition

- A vector valued function of one variable $\mathbf{x}(t)$
  - Given t, compute a 3D point $\mathbf{x}=(x,y,z)$
  - Could be interpreted as three functions: $x(t)$, $y(t)$, $z(t)$
  - Parameter t "moves a point along the curve"



Tangent Vector

- Derivative $\mathbf{x}'(t) = \dfrac{d\mathbf{x}}{dt} = (x'(t), y'(t), z'(t))$
- Vector x' points in direction of movement
- Length corresponds to speed

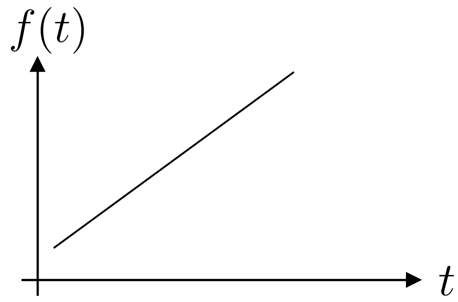


Lecture Overview

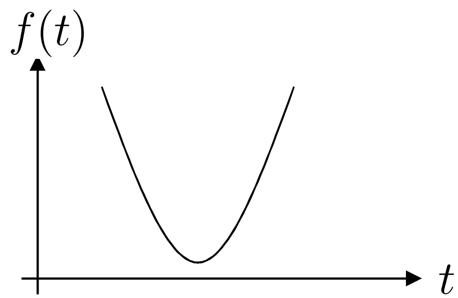
- Polynomial Curves
  - Introduction
  - Polynomial functions
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves

Polynomial Functions

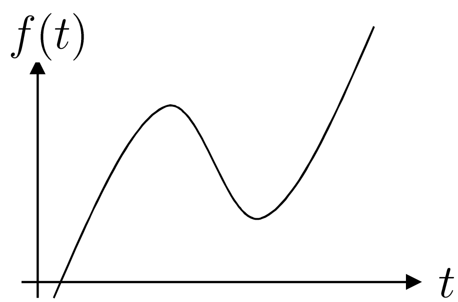
- **Linear:** $f(t) = at + b$
  (1st order)



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



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



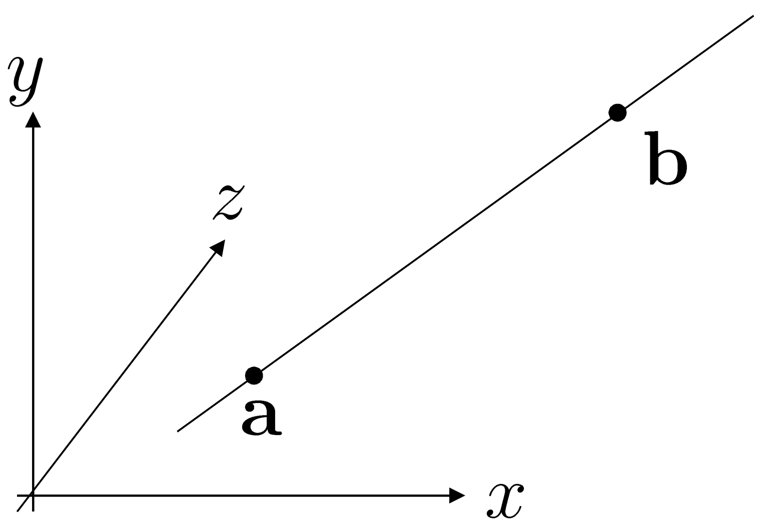
Polynomial Curves

- Linear $\mathbf{x}(t) = \mathbf{a}t + \mathbf{b}$

$$\mathbf{x} = (x, y, z), \mathbf{a} = (a_x, a_y, a_z), \mathbf{b} = (b_x, b_y, b_z)$$

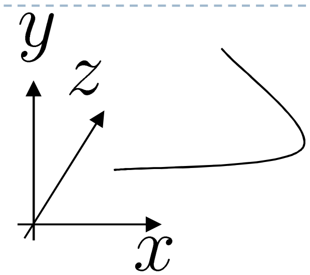
- Evaluated as:

$$x(t) = a_x t + b_x$$
$$y(t) = a_y t + b_y$$
$$z(t) = a_z t + b_z$$

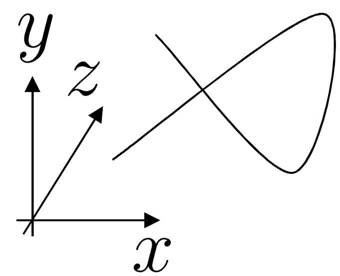


Polynomial Curves

- Quadratic: $\mathbf{x}(t) = \mathbf{a}t^2 + \mathbf{b}t + \mathbf{c}$
  (2nd order)



- Cubic: $\mathbf{x}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$
  (3rd order)



- We usually define the curve for $0 \leq t \leq 1$

Control Points

- Polynomial coefficients **a, b, c, d** can be interpreted as *control points*
  - Remember: **a, b, c, d** have x,y,z components each
- Unfortunately, they do not intuitively describe the shape of the curve
- Goal: intuitive control points

Control Points

- How many control points?
  - Two points define a line (1st order)
  - Three points define a quadratic curve (2nd order)
  - Four points define a cubic curve (3rd order)
  - $k+1$ points define a k-order curve
- Let's start with a line…

First Order Curve

- Based on linear interpolation (LERP)
  - Weighted average between two values
  - "Value" could be a number, vector, color, …
- Interpolate between points $\mathbf{p_0}$ and $\mathbf{p_1}$ with parameter t
  - Defines a "curve" that is straight (first-order spline)
  - $t=0$ corresponds to $\mathbf{p_0}$
  - $t=1$ corresponds to $\mathbf{p_1}$
  - $t=0.5$ corresponds to midpoint

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

Linear Interpolation

- Three equivalent ways to write it
  - Expose different properties

1. Regroup for points **p**

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

2. Regroup for t

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

3. Matrix form

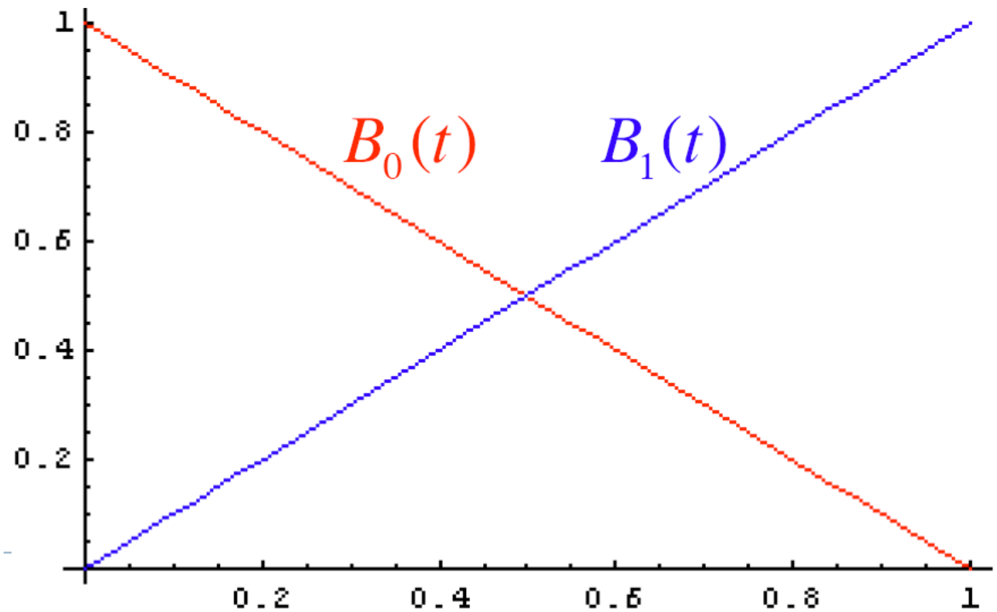
$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

Weighted Average

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

$$= B_0(t)\,\mathbf{p}_0 + B_1(t)\mathbf{p}_1, \text{ where } B_0(t) = 1-t \text{ and } B_1(t) = t$$

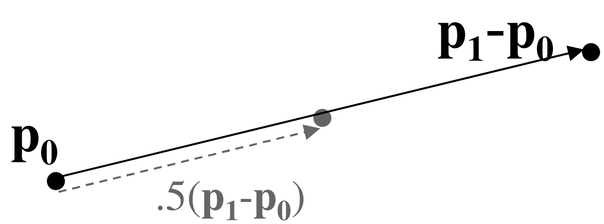
- Weights are a function of t
  - Sum is always 1, for any value of t
  - Also known as *blending functions*



Linear Polynomial

$$\mathbf{x}(t) = \underbrace{(\mathbf{p}_1 - \mathbf{p}_0)}_{\substack{\text{vector} \\ \mathbf{a}}} \, t + \underbrace{\mathbf{p}_0}_{\substack{\text{point} \\ \mathbf{b}}}$$

- Curve is based at point $\mathbf{p_0}$
- Add the vector, scaled by t



Matrix Form

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

- Geometry matrix $$\mathbf{G} = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 \end{bmatrix}$$
- Geometric basis $$\mathbf{B} = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}$$
- Polynomial basis $$T = \begin{bmatrix} t \\ 1 \end{bmatrix}$$
- In components $$\mathbf{x}(t) = \begin{bmatrix} p_{0x} & p_{1x} \\ p_{0y} & p_{1y} \\ p_{0z} & p_{1z} \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}$$

Tangent

- For a straight line, the tangent is constant

$$\mathbf{x}'(t) = \mathbf{p}_1 - \mathbf{p}_0$$

- Weighted average $\mathbf{x}'(t) = (-1)\mathbf{p}_0 + (+1)\mathbf{p}_1$

- Polynomial $\mathbf{x}'(t) = 0t + (\mathbf{p}_1 - \mathbf{p}_0)$

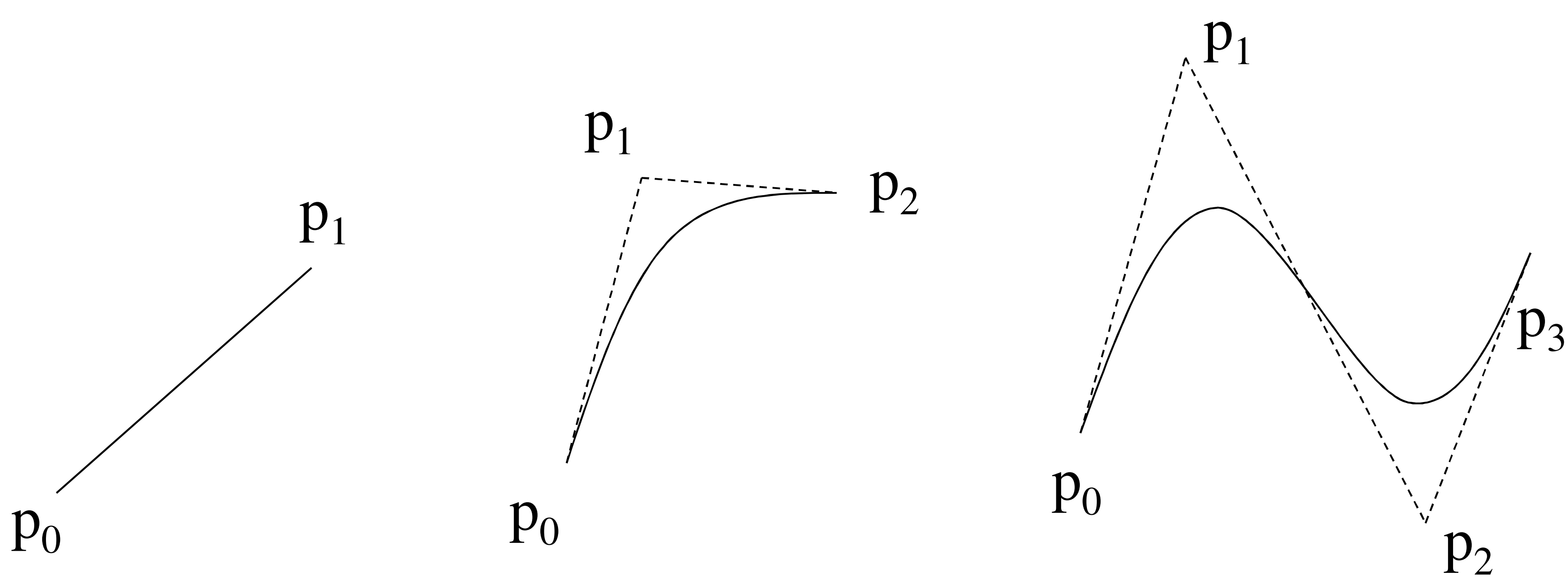
- Matrix form $\mathbf{x}'(t) = \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Lecture Overview

- Polynomial Curves
  - Introduction
  - Polynomial functions
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves

Bézier Curves

- Are a higher order extension of linear interpolation



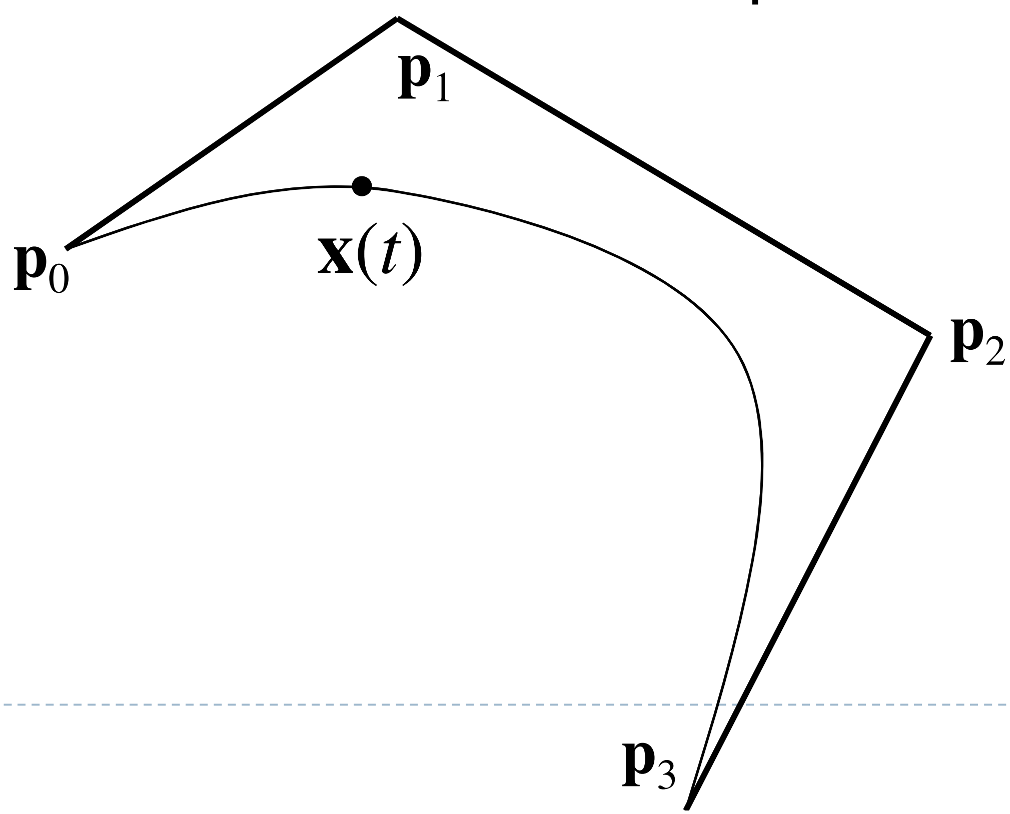
Linear Quadratic Cubic

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, for example:
  - 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 commonly used case
- Defined by four control points:
  - Two interpolated endpoints (points are on the curve)
  - Two points control the tangents at the endpoints
- Points $\mathbf{x}$ on curve defined as function of parameter t



Algorithmic Construction

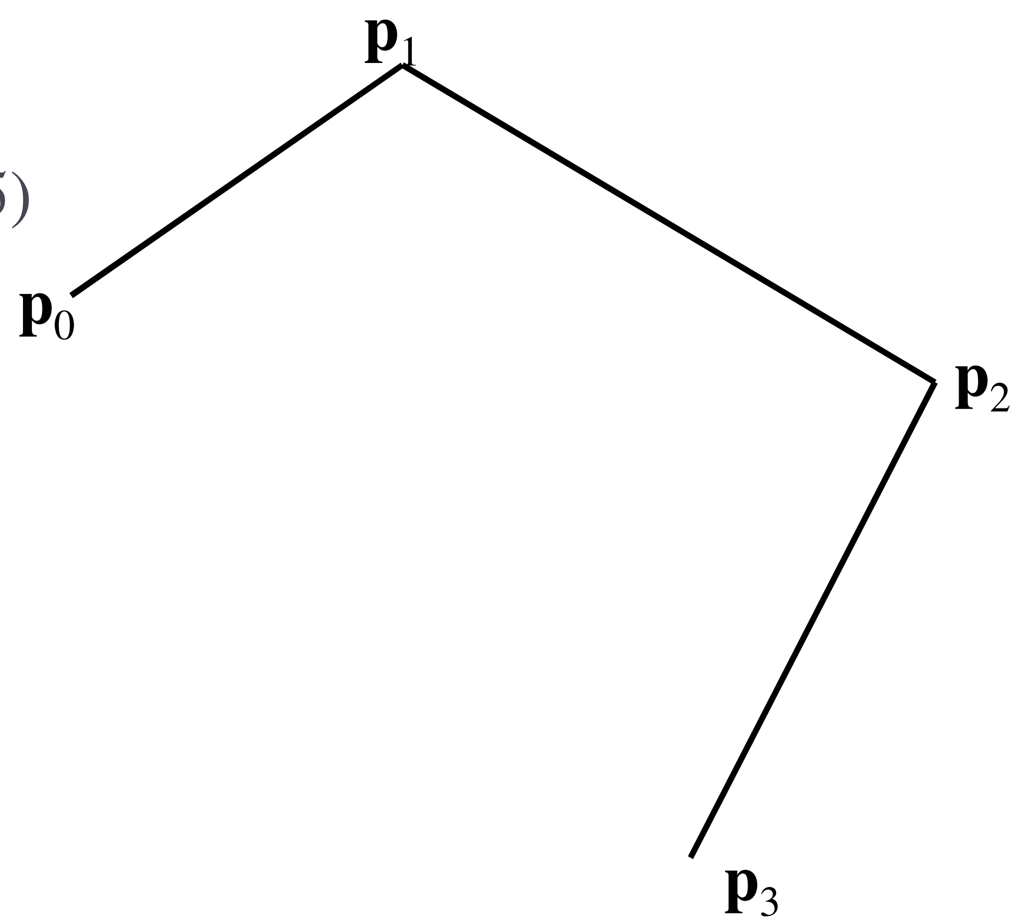
- Algorithmic construction
  - *De Casteljau* algorithm, developed at Citroen in 1959, named after its inventor Paul de Casteljau (pronounced "Cast-all-'Joe")
  - Developed independently from Bézier's work:
    Bézier created the formulation using blending functions, Casteljau devised the recursive interpolation algorithm

De Casteljau Algorithm

- A recursive series of linear interpolations
  - Works for any order Bezier function, not only cubic
- Not very efficient to evaluate
  - Other forms more commonly used
- But:
  - Gives intuition about the geometry
  - Useful for subdivision

De Casteljau Algorithm

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

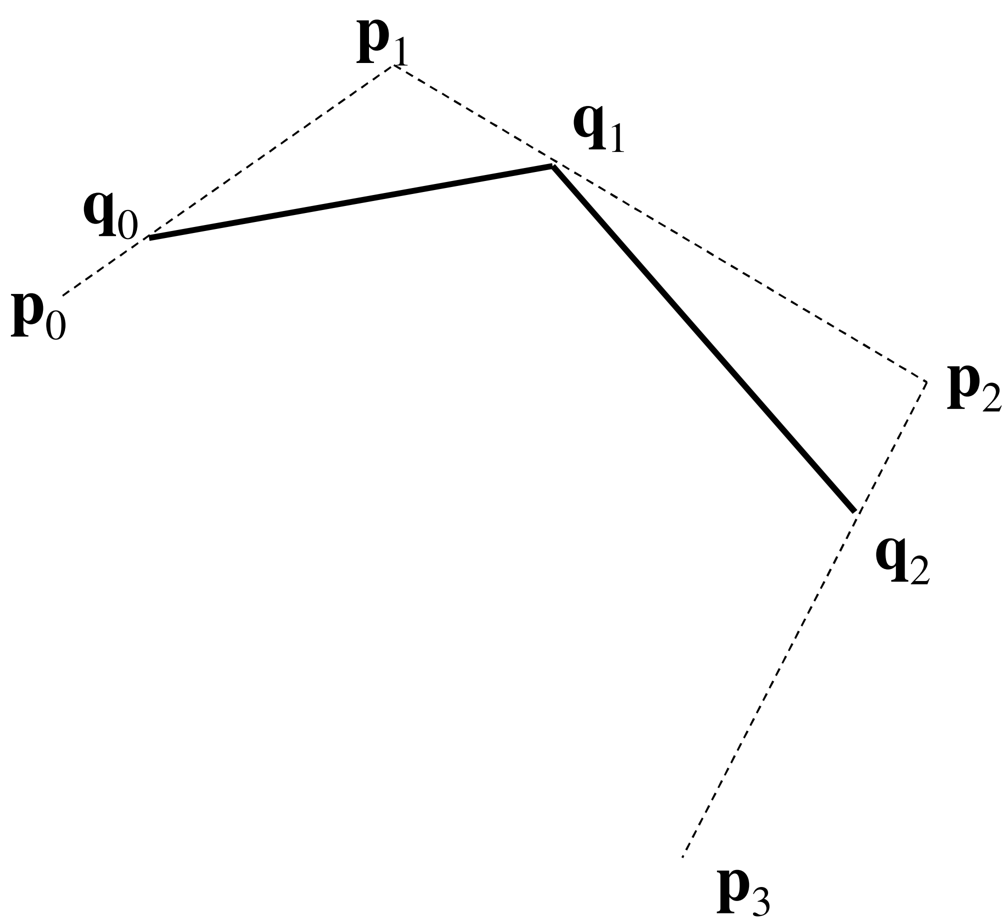


De Casteljau Algorithm

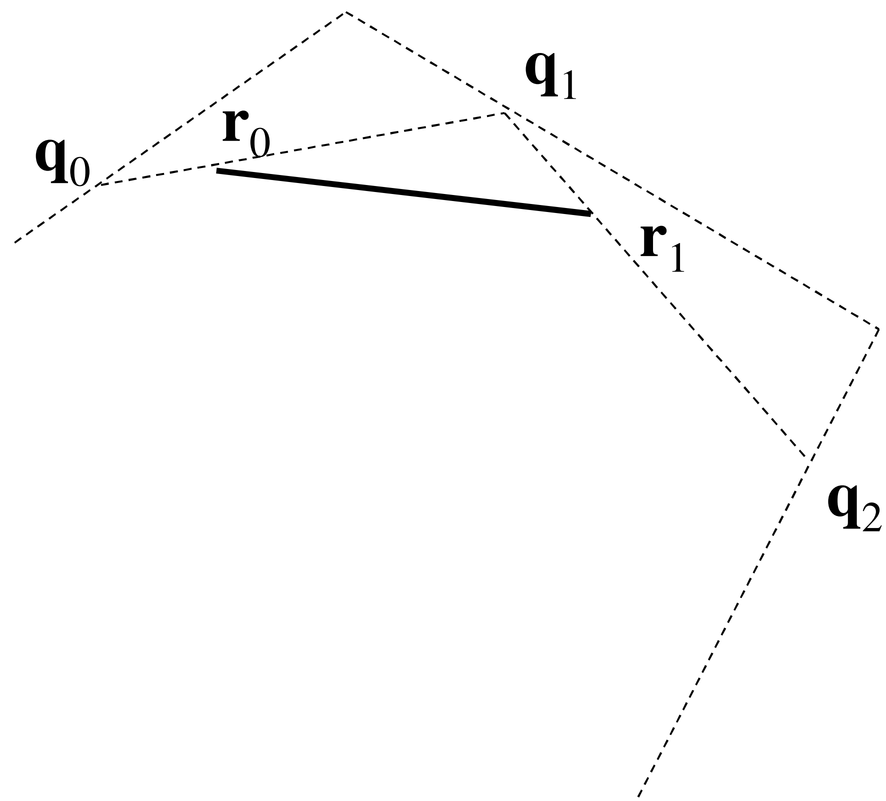
$$\mathbf{q}_0(t) = Lerp\left(t, \mathbf{p}_0, \mathbf{p}_1\right)$$

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

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



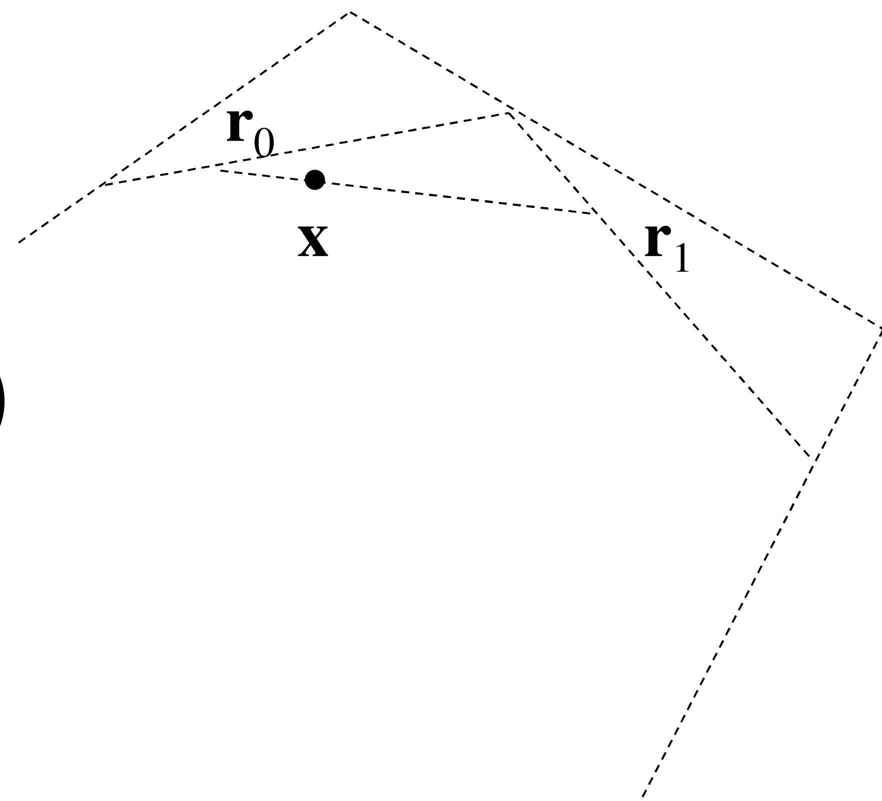
De Casteljau Algorithm



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

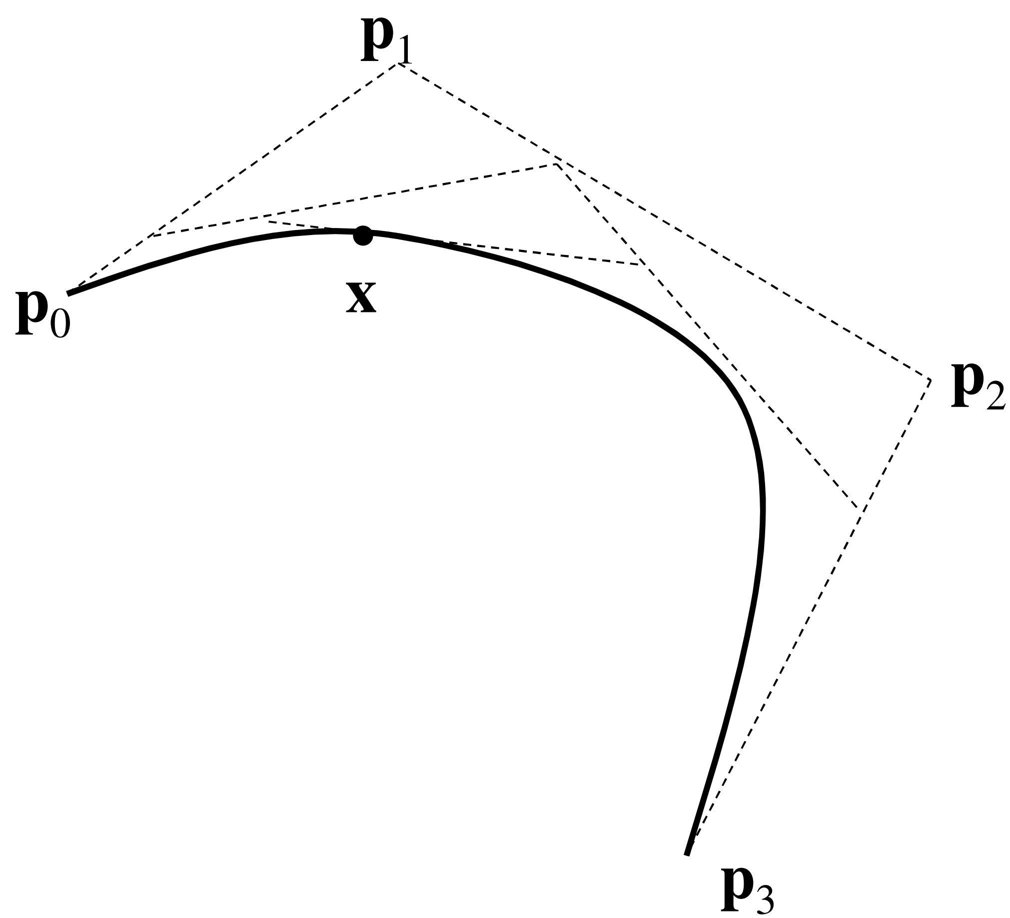
$$\mathbf{r}_1(t) = Lerp\left(t, \mathbf{q}_1(t), \mathbf{q}_2(t)\right)$$

De Casteljau Algorithm



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

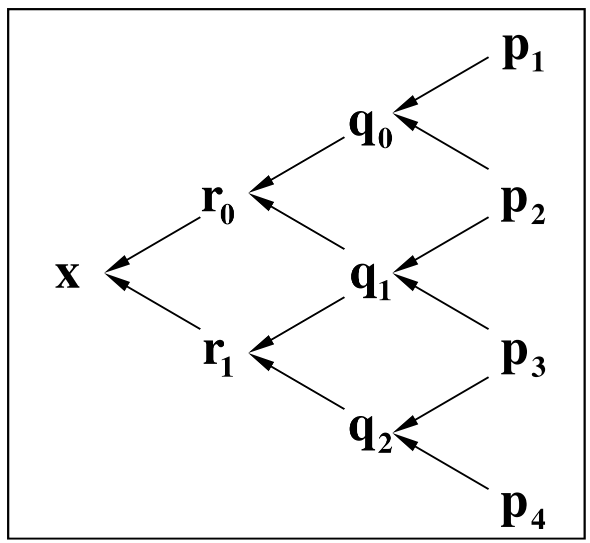
De Casteljau Algorithm



- 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

$$\mathbf{x} = Lerp(t, \mathbf{r}_0, \mathbf{r}_1) \quad \begin{array}{l} \mathbf{r}_0 = Lerp(t, \mathbf{q}_0, \mathbf{q}_1) \\ \mathbf{r}_1 = Lerp(t, \mathbf{q}_1, \mathbf{q}_2) \end{array} \quad \begin{array}{l} \mathbf{q}_0 = Lerp(t, \mathbf{p}_0, \mathbf{p}_1) \\ \mathbf{q}_1 = Lerp(t, \mathbf{p}_1, \mathbf{p}_2) \\ \mathbf{q}_2 = Lerp(t, \mathbf{p}_2, \mathbf{p}_3) \end{array} \quad \begin{array}{l} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{array}$$



Expand the LERPs

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

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

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

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

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

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

Weighted Average of Control Points

- Regroup for p:

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

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

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

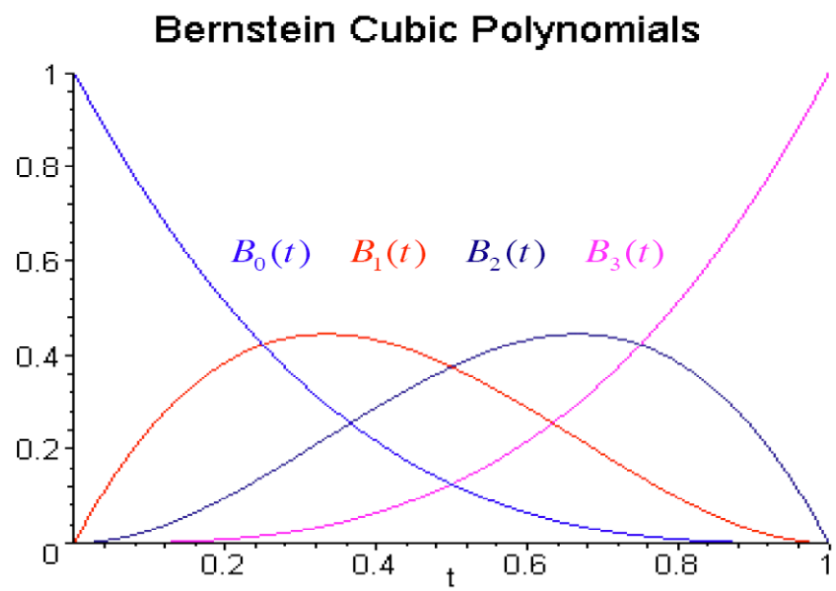
Cubic Bernstein Polynomials

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

The cubic *Bernstein polynomials :*

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

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



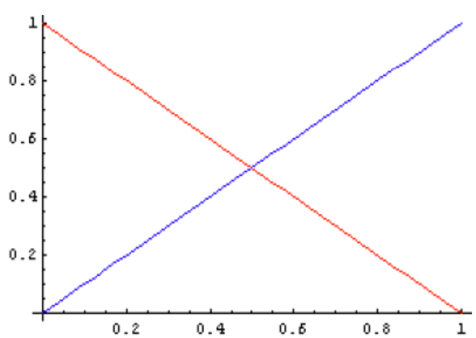
- Weights $B_i(t)$ add up to 1 for any value of t

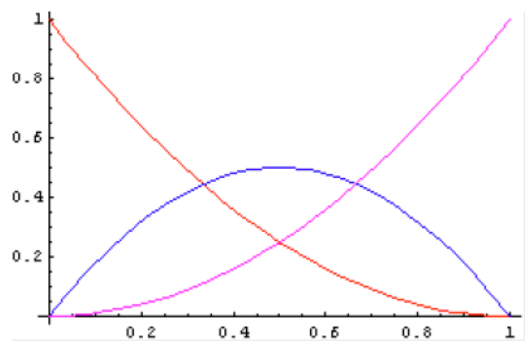
General Bernstein Polynomials

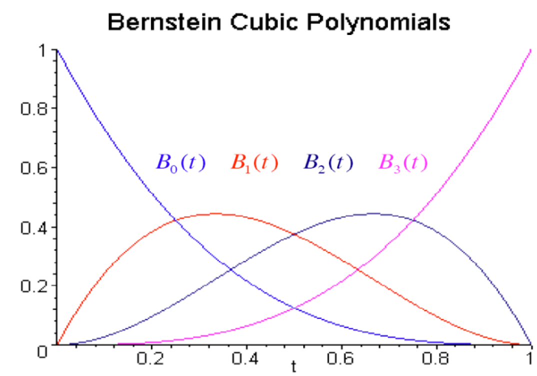
$$B_0^1(t) = -t + 1$$
$$B_1^1(t) = t$$

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

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







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

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

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

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:

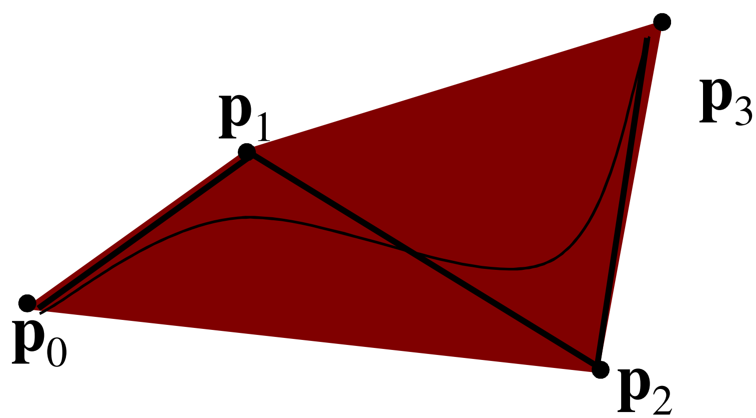
- Convex Hull property
- Variation Diminishing property
- Affine Invariance

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 of a set of points lies within the convex hull

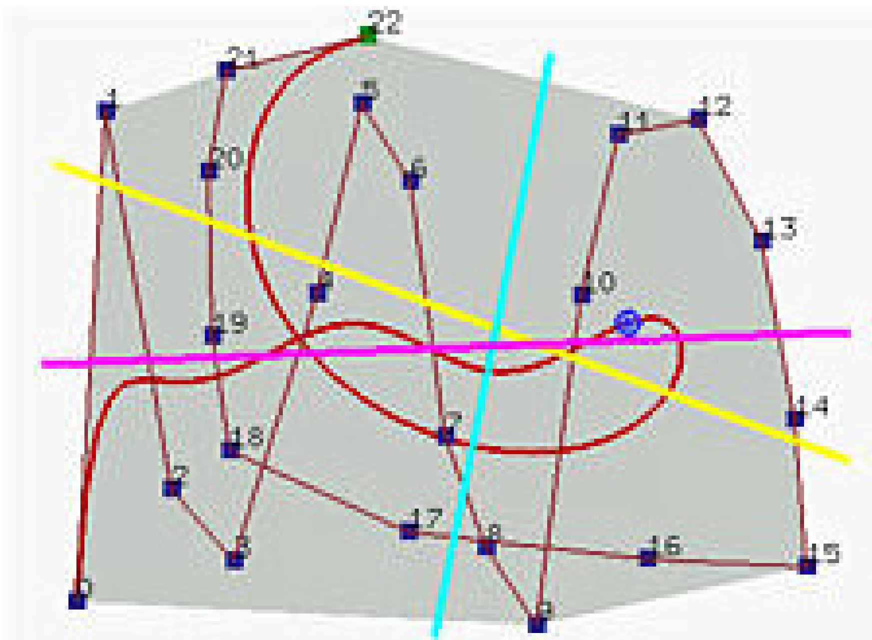
Convex Hull Property

- A Bézier curve is a convex combination of the control points (by definition, see Bernstein polynomials)
- A Bézier curve is always inside the convex hull
  - Makes curve predictable
  - Allows culling, intersection testing, adaptive tessellation
- Demo: http://www.cs.princeton.edu/~min/cs426/jar/bezier.html



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 (i.e., translation, scale, rotation, shear)
  - 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 types of cubic splines use different basis matrices $\mathbf{B}_{\mathrm{Bez}}$

Cubic Matrix Form

- In 3D: 3 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
  - Pre-compute $\mathbf{C}$
  - Take advantage of existing 4x4 matrix hardware support