

Lecture 29 of 42

Bezier Curves and Splines

Wednesday, 02 April 2008

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course pages: <http://snipurl.com/1y5gc>

Course web site: <http://www.kddresearch.org/Courses/CIS636>

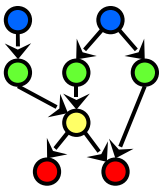
Instructor home page: <http://www.cis.ksu.edu/~bhsu>

Readings:

Sections 11.1 – 11.6, Eberly 2^e – see <http://snurl.com/1ye72>

http://graphics.ucsd.edu/courses/cse167_f06/





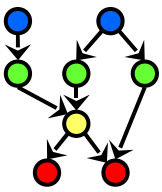
Cubic Curves

CSE167: Computer Graphics
Instructor: Steve Rotenberg
UCSD, Fall 2006

© 2006 Rotenberg, S., University of California San Diego

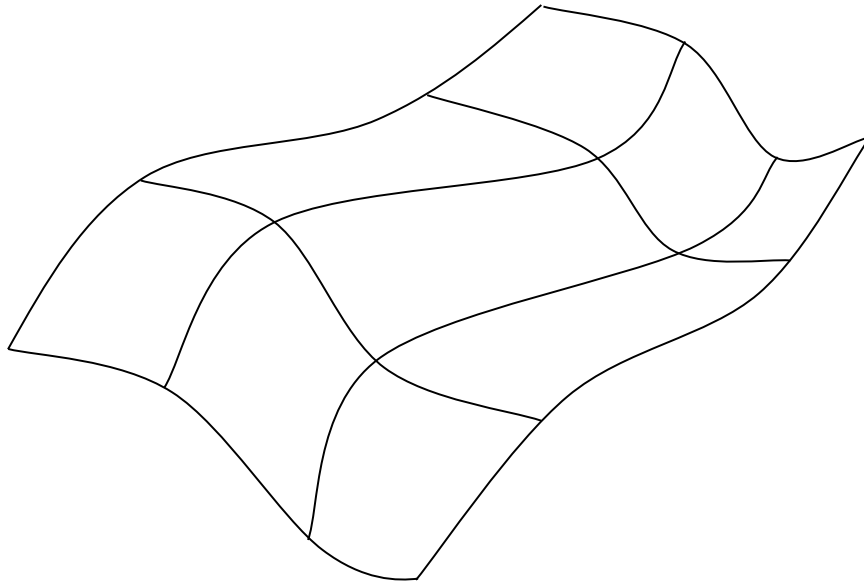
http://graphics.ucsd.edu/courses/cse167_f06/





Understanding and Using OpenGL EvalMesh/EvalCurve

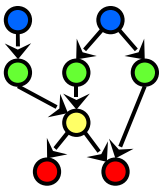
- Render a smooth, cubic surface by tessellating it into a set of triangles
- See: evalMesh*d



© 2006 Rotenberg, S., University of California San Diego

http://graphics.ucsd.edu/courses/cse167_f06/



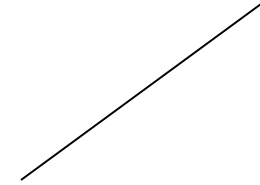


Polynomial Functions

- Linear:

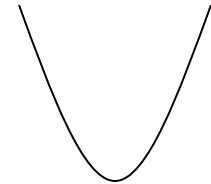
- Quadratic:

$$f(t) = at + b$$

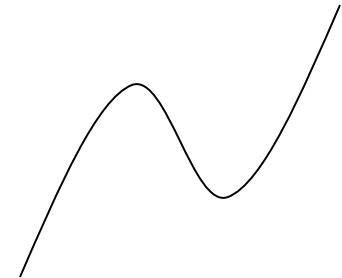


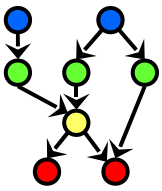
- Cubic:

$$f(t) = at^2 + bt + c$$



$$f(t) = at^3 + bt^2 + ct + d$$

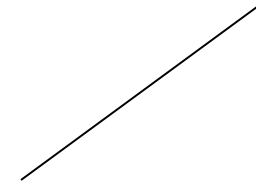




Vector Polynomials (Curves)

- Linear:

- Quadratic: $\mathbf{f}(t) = \mathbf{a}t + \mathbf{b}$



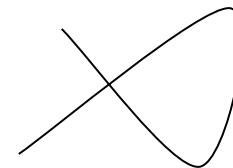
- Cubic:

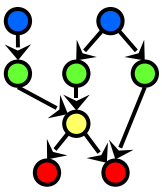
$$\mathbf{f}(t) = \mathbf{a}t^2 + \mathbf{b}t + \mathbf{c}$$



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

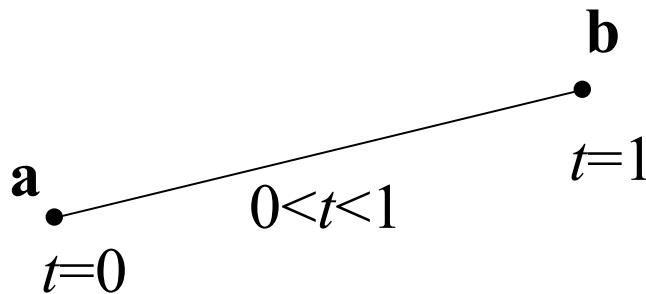
$$\mathbf{f}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$





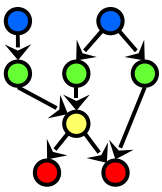
Linear Interpolation

- Linear interpolation (Lerp) is a common technique for generating a new value that is somewhere in between two other values
- A 'value' could be a number, vector, color, or even something more complex like an entire 3D object...
- Consider interpolating between two points **a** and **b** by some parameter t



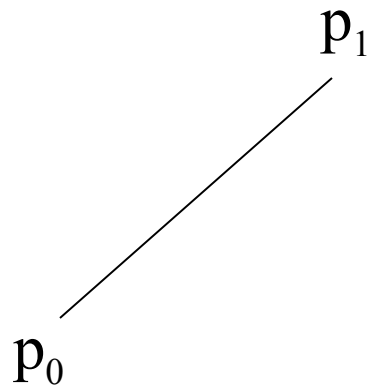
$$\text{Lerp}(t, \mathbf{a}, \mathbf{b}) = (1 - t)\mathbf{a} + t\mathbf{b}$$



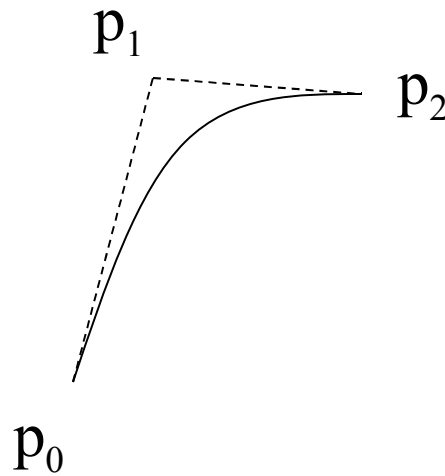


Bezier Curves

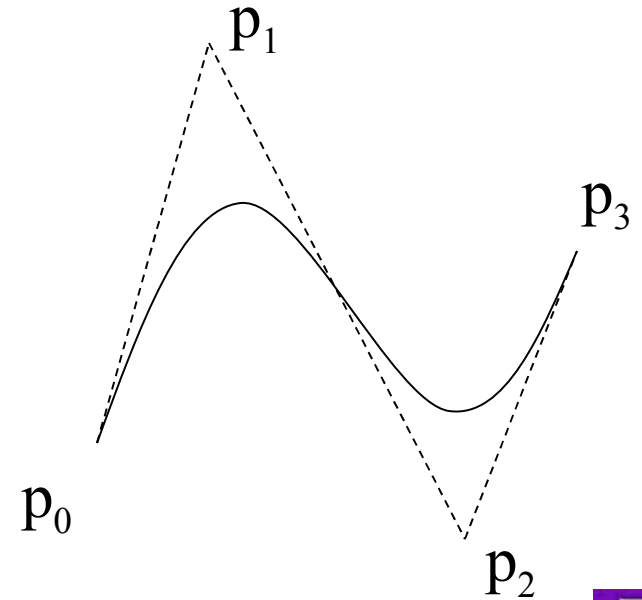
- Bezier curves can be thought of as a higher order extension of linear interpolation



Linear

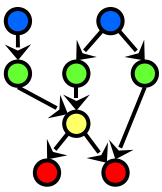


Quadratic



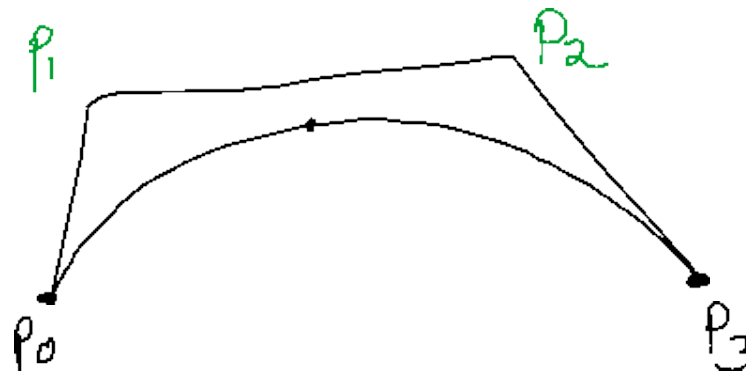
Cubic

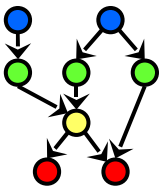




Bezier Curve Formulation

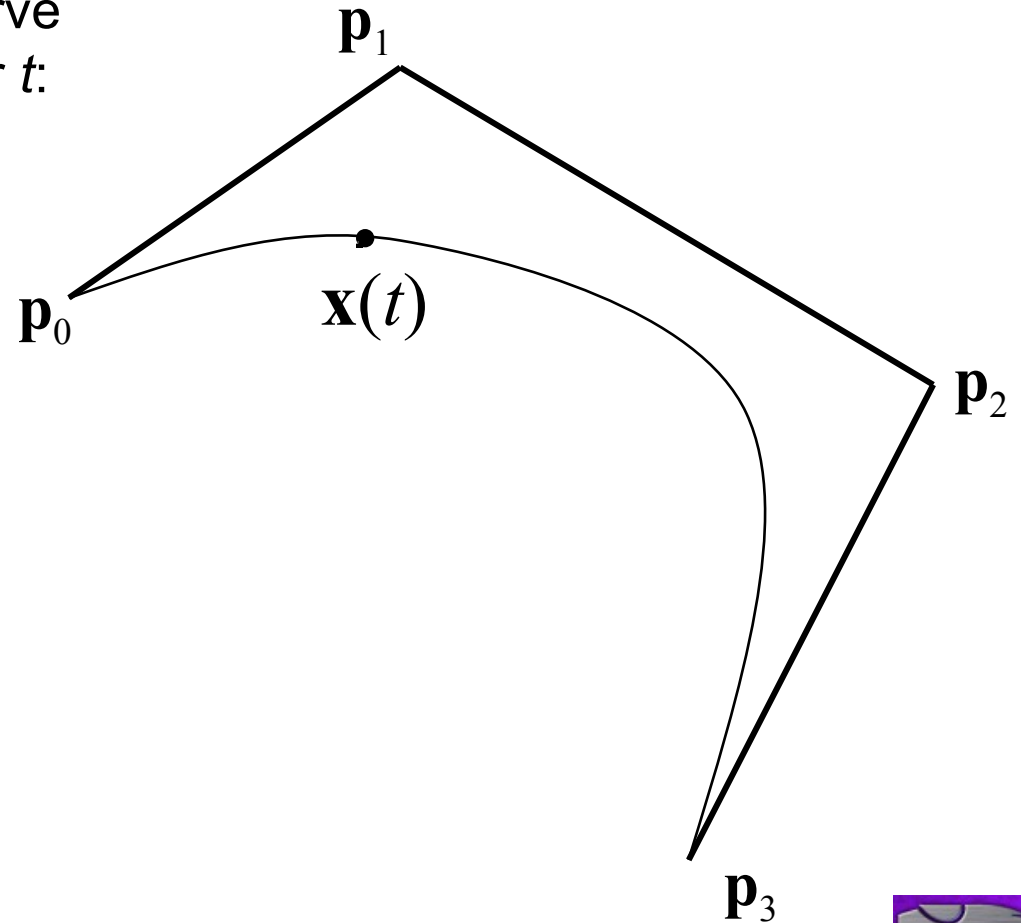
- There are lots of ways to formulate Bezier curves mathematically. Some of these include:
 - * de Casteljau (recursive linear interpolations)
 - * Bernstein polynomials (functions that define the influence of each control point as a function of t)
 - * Cubic equations (general cubic equation of t)
 - * Matrix form
- We will briefly examine each of these

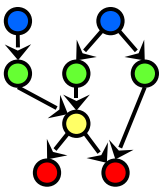




Bezier Curve

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

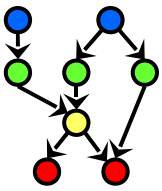




de Casteljau Algorithm

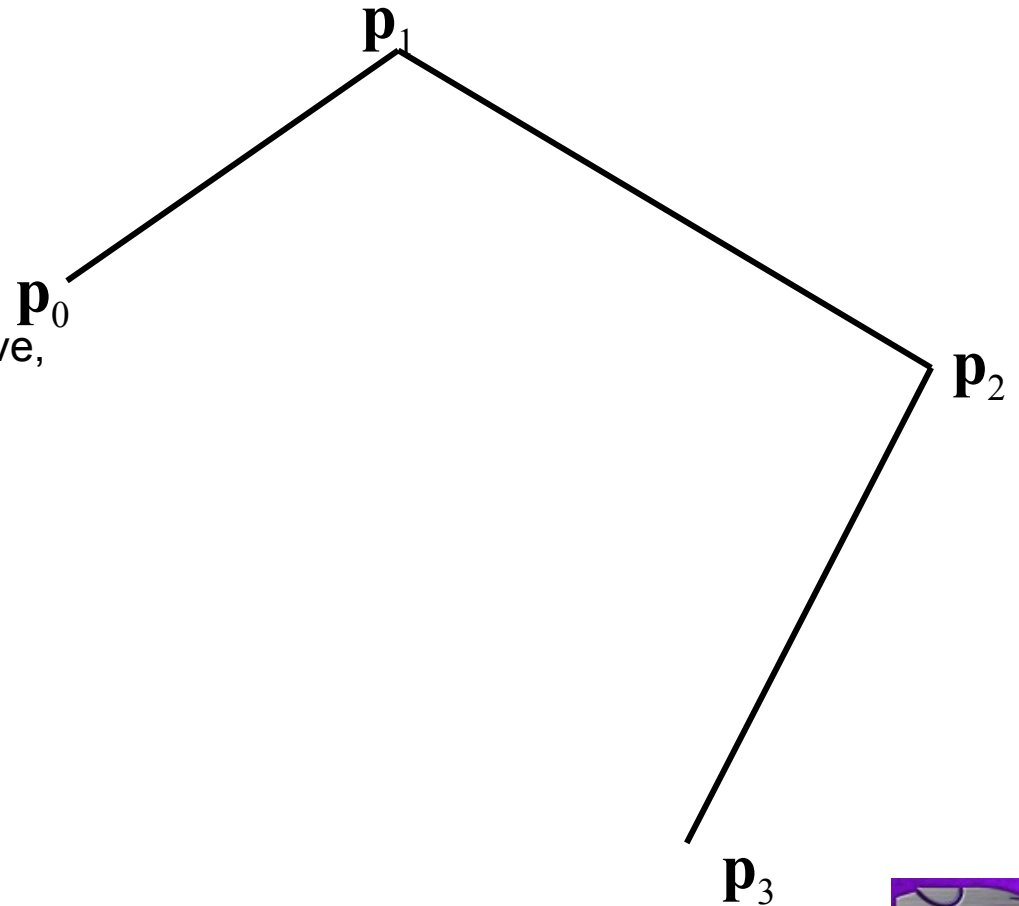
- The de Casteljau algorithm describes the curve as a recursive series of linear interpolations
- This form is useful for providing an intuitive understanding of the geometry involved, but it is not the most efficient form

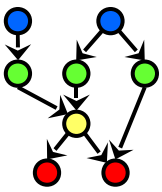




de Casteljau Algorithm

- We start with our original set of points
- In the case of a cubic Bezier curve, we start with four points



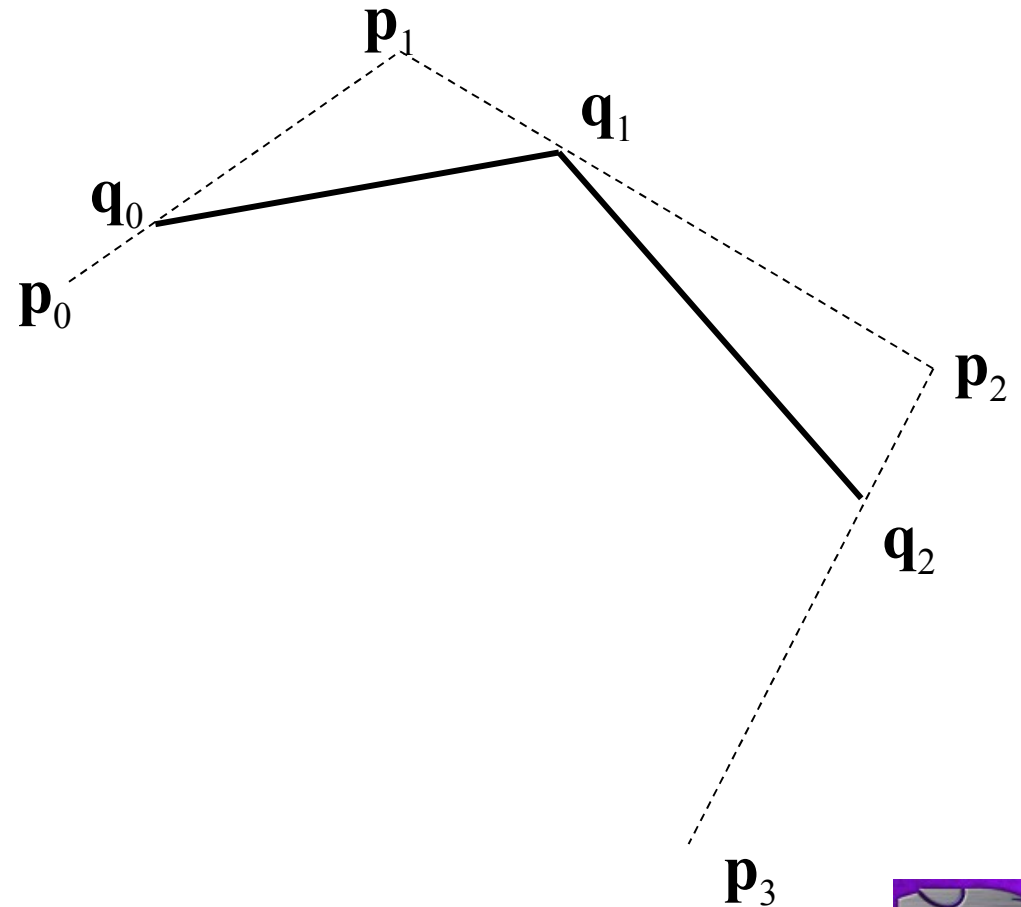


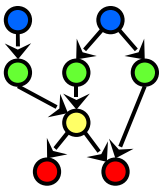
de Casteljau Algorithm

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

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

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

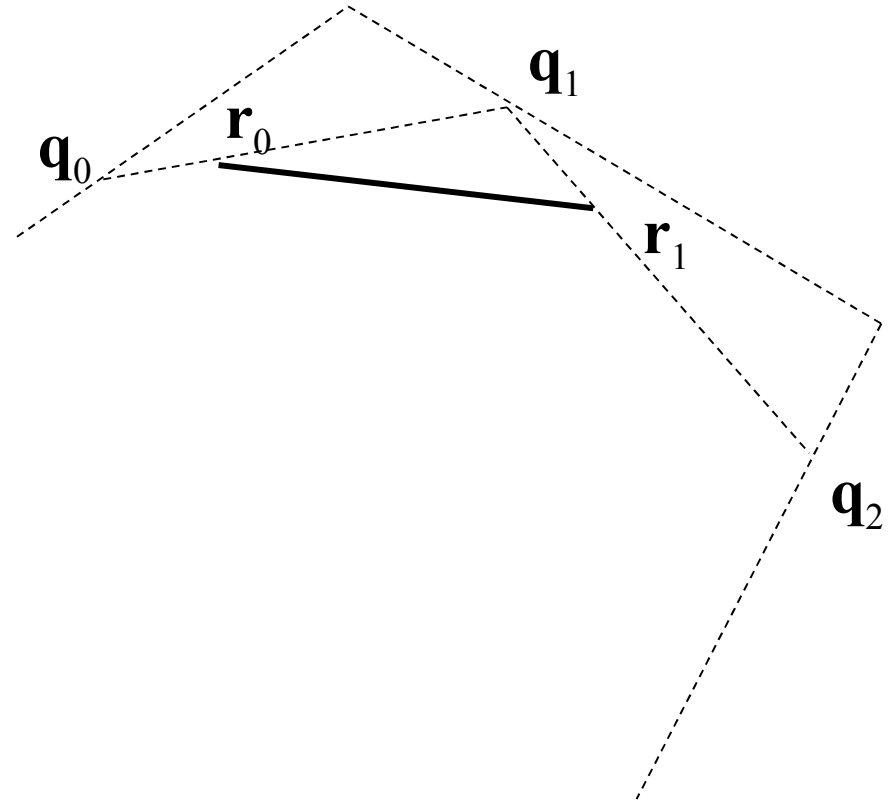


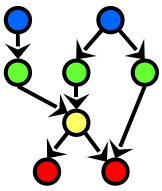


de Casteljau Algorithm

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

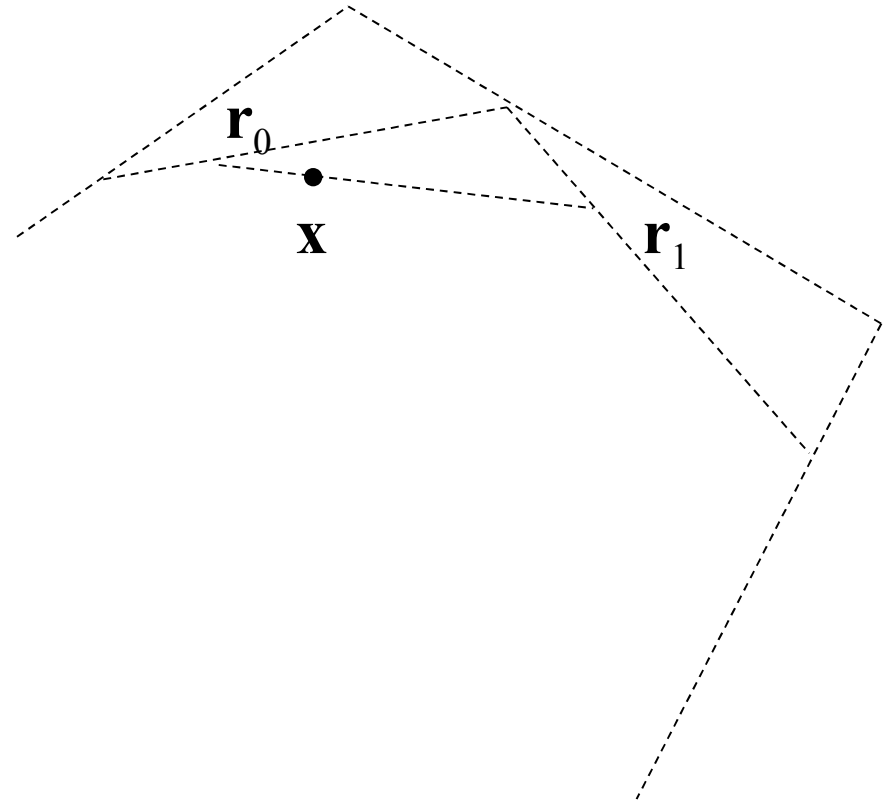
$$\mathbf{r}_1 = \text{Lerp}(t, \mathbf{q}_1, \mathbf{q}_2)$$

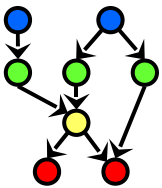




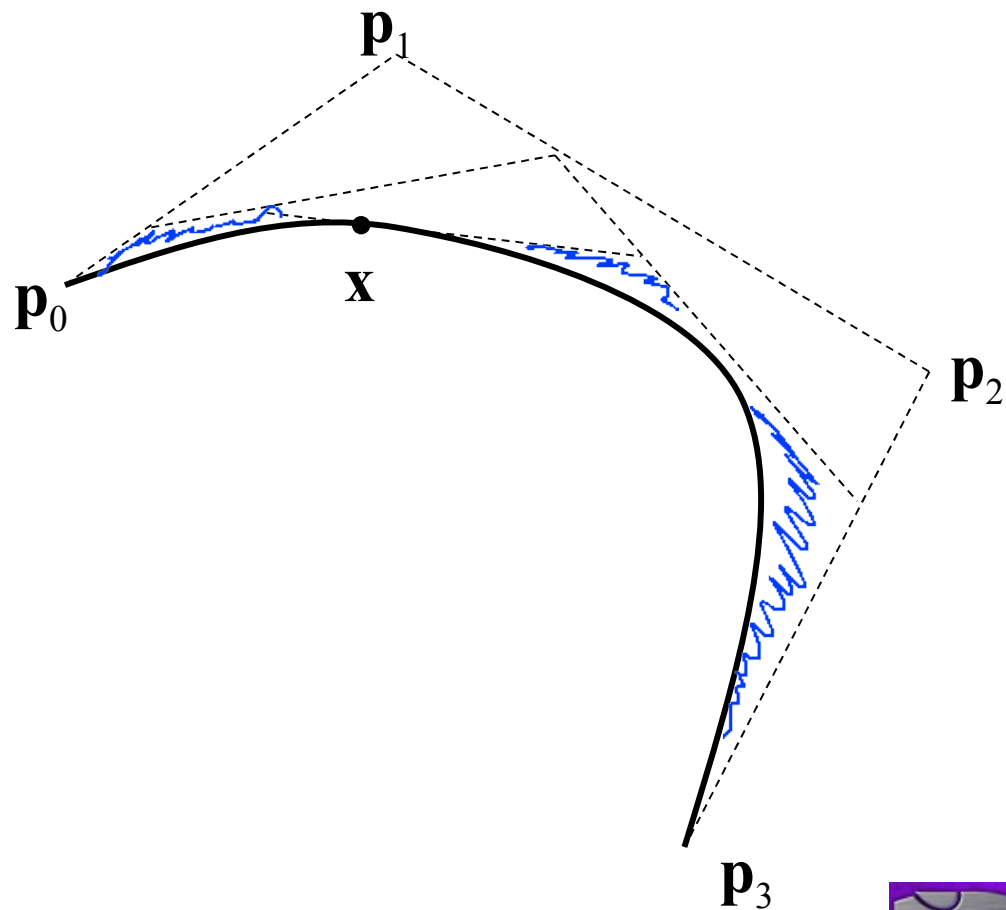
de Casteljau Algorithm

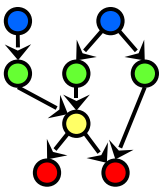
$$\mathbf{x} = \text{Lerp}(t, \mathbf{r}_0, \mathbf{r}_1)$$





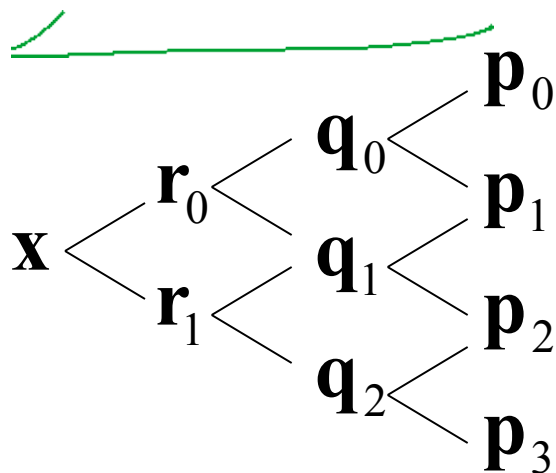
Bezier Curve

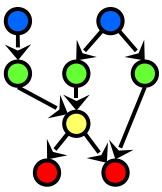




Recursive Linear Interpolation

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





Expanding the Lerps

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

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

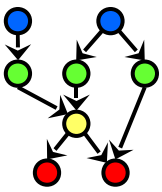
$$\mathbf{q}_2 = \text{Lerp}(t, \mathbf{p}_2, \mathbf{p}_3) = (1-t)\mathbf{p}_2 + t\mathbf{p}_3$$

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

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

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





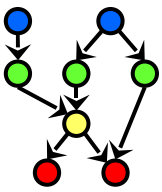
Bernstein Polynomial Form

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

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

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





Cubic Bernstein Polynomials

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

$$\mathbf{x} = B_0^3(t)\mathbf{p}_0 + B_1^3(t)\mathbf{p}_1 + B_2^3(t)\mathbf{p}_2 + B_3^3(t)\mathbf{p}_3$$

$A\vec{x} = \vec{b}$

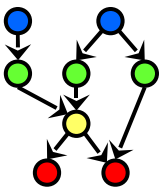
$$B_0^3(t) = -t^3 + 3t^2 - 3t + 1$$

$$B_1^3(t) = 3t^3 - 6t^2 + 3t$$

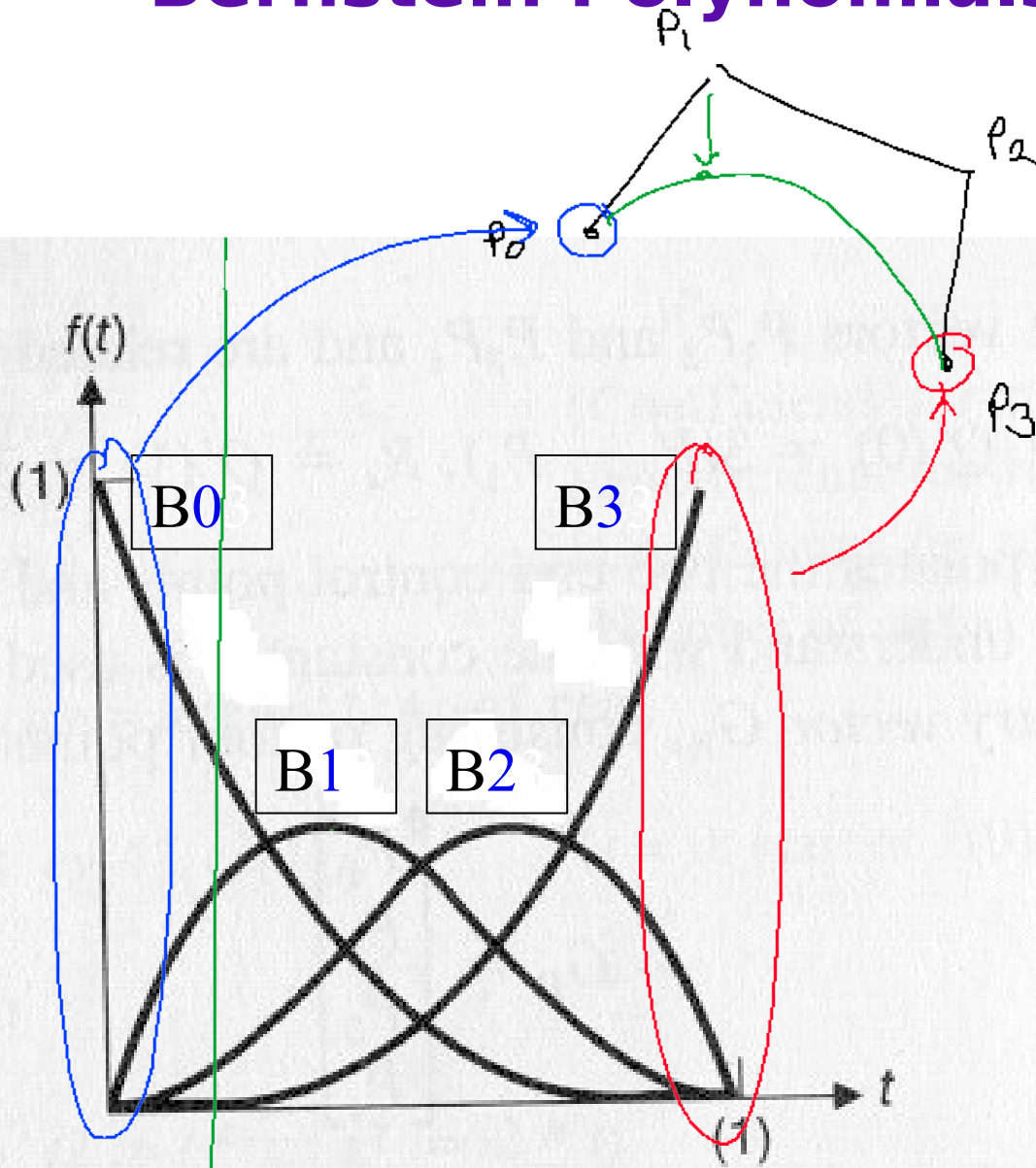
$$B_2^3(t) = -3t^3 + 3t^2$$

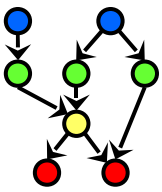
$$B_3^3(t) = t^3$$





Bernstein Polynomials





Bernstein Polynomials

$$B_0^3(t) = -t^3 + 3t^2 - 3t + 1$$

$$B_1^3(t) = 3t^3 - 6t^2 + 3t$$

$$B_2^3(t) = -3t^3 + 3t^2$$

$$B_3^3(t) = t^3$$

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

$$B_1^2(t) = -2t^2 + 2t$$

$$B_2^2(t) = t^2$$

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

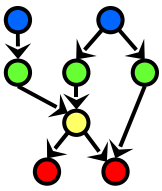
$$B_1^1(t) = t$$

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

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

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





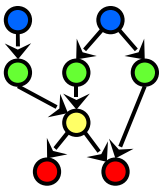
Bernstein Polynomials

- Bernstein polynomial form of a Bezier curve:

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

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

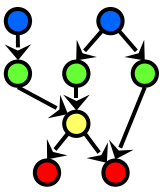




Bernstein Polynomials

- We start with the de Casteljau algorithm, expand out the math, and group it into polynomial functions of t multiplied by points in the control mesh
- The generalization of this gives us the Bernstein form of the Bezier curve
- This gives us further understanding of what is happening in the curve:
 - ★ We can see the influence of each point in the control mesh as a function of t
 - ★ We see that the basis functions add up to 1, indicating that the Bezier curve is a convex average of the control points



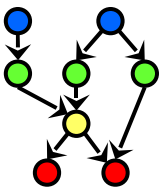


Cubic Equation Form

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

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





Cubic Equation Form

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

\vec{b} $A\vec{x}$

$$\mathbf{x} = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

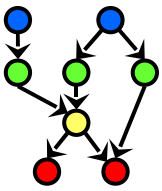
$$\mathbf{a} = (-\mathbf{p}_0 + 3\mathbf{p}_1 - 3\mathbf{p}_2 + \mathbf{p}_3)$$

$$\mathbf{b} = (3\mathbf{p}_0 - 6\mathbf{p}_1 + 3\mathbf{p}_2)$$

$$\mathbf{c} = (-3\mathbf{p}_0 + 3\mathbf{p}_1)$$

$$\mathbf{d} = (\mathbf{p}_0)$$

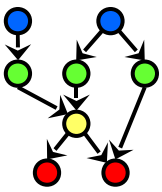




Cubic Equation Form

- If we regroup the equation by terms of exponents of t , we get it in the standard cubic form
- This form is very good for fast evaluation, as all of the constant terms (**a,b,c,d**) can be precomputed
- The cubic equation form obscures the input geometry, but there is a one-to-one mapping between the two and so the geometry can always be extracted out of the cubic coefficients





Cubic Matrix Form

$$\mathbf{x} = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

$$\mathbf{a} = (-\mathbf{p}_0 + 3\mathbf{p}_1 - 3\mathbf{p}_2 + \mathbf{p}_3)$$

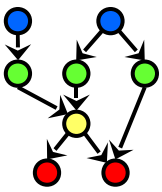
$$\mathbf{b} = (3\mathbf{p}_0 - 6\mathbf{p}_1 + 3\mathbf{p}_2)$$

$$\mathbf{c} = (-3\mathbf{p}_0 + 3\mathbf{p}_1)$$

$$\mathbf{d} = (\mathbf{p}_0)$$

$$\mathbf{x} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$



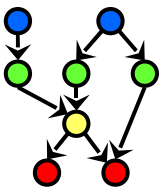


Cubic Matrix Form

$$\mathbf{x} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

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





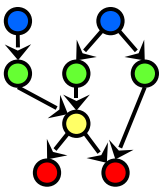
Matrix Form

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

$$\mathbf{x} = \mathbf{t} \cdot \mathbf{B}_{Bez} \cdot \mathbf{G}_{Bez}$$

$$\mathbf{x} = \mathbf{t} \cdot \mathbf{C}$$

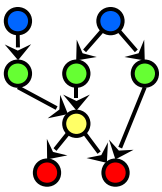




Matrix Form

- We can rewrite the equations in matrix form
- This gives us a compact notation and shows how different forms of cubic curves can be related
- It also is a very efficient form as it can take advantage of existing 4x4 matrix hardware support...

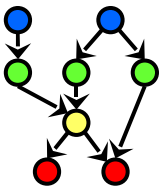




Bezier Curves & Cubic Curves

- By adjusting the 4 control points of a cubic Bezier curve, we can represent any cubic curve
- Likewise, any cubic curve can be represented uniquely by a cubic Bezier curve
- There is a one-to-one mapping between the 4 Bezier control points $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ and the pure cubic coefficients $(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})$
- The Bezier basis matrix \mathbf{B}_{Bez} (and its inverse) perform this mapping
- There are other common forms of cubic curves that also retain this property (Hermite, Catmull-Rom, B-Spline)





Derivatives

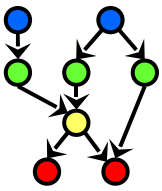
- Finding the derivative (tangent) of a curve is easy:

$$\mathbf{x} = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

$$\frac{d\mathbf{x}}{dt} = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}$$

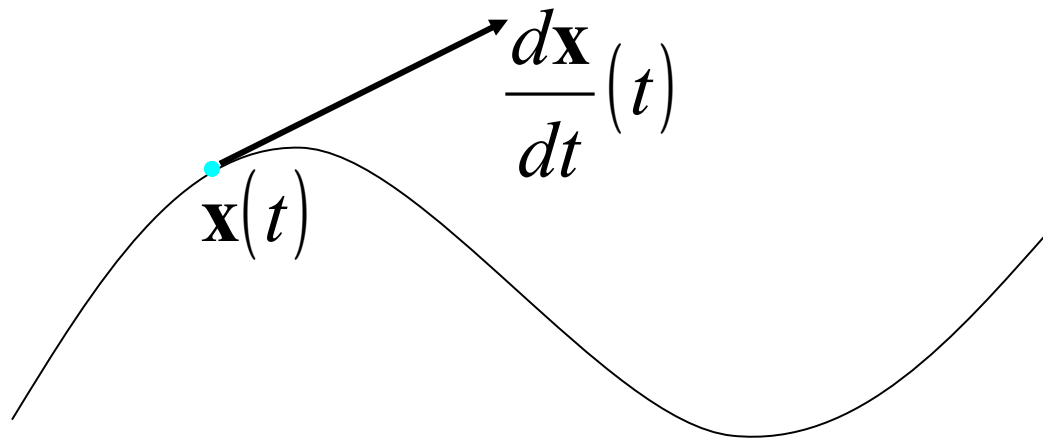
$$\mathbf{x} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

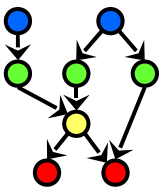
$$\frac{d\mathbf{x}}{dt} = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$



Tangents

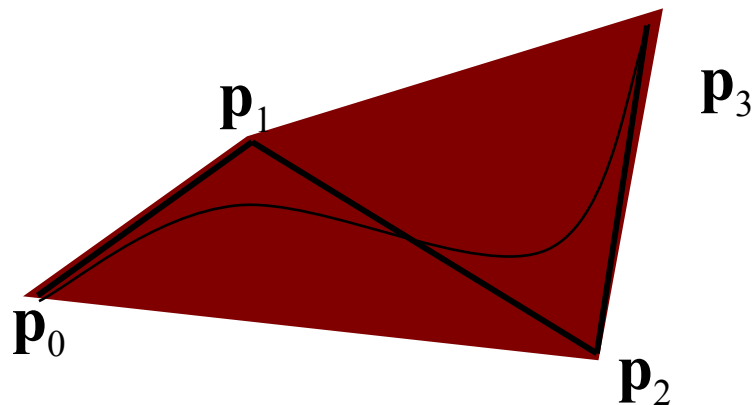
- The derivative of a curve represents the tangent vector to the curve at some point

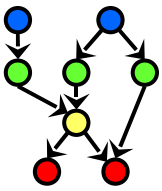




Convex Hull Property

- If we take all of the control points for a Bezier curve and construct a convex polygon around them, we have the *convex hull* of the curve
- An important property of Bezier curves is that every point on the curve itself will be somewhere within the convex hull of the control points

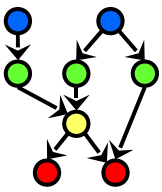




Continuity

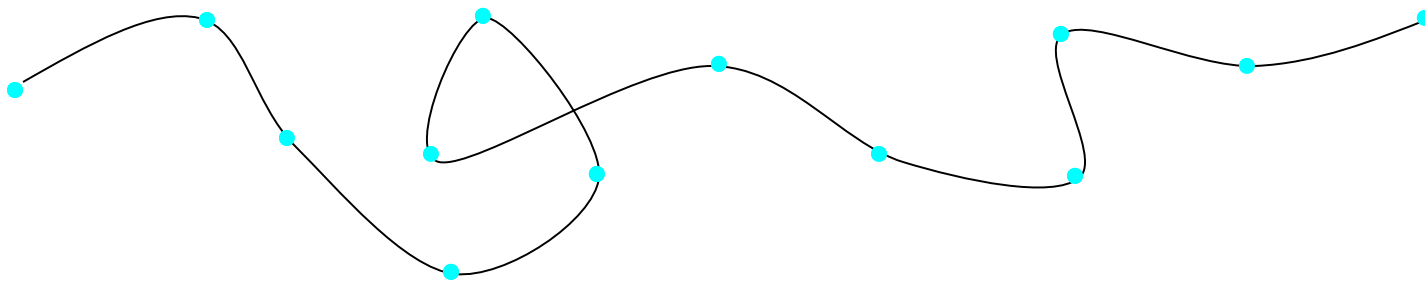
- A cubic curve defined for t ranging from 0 to 1 will form a single continuous curve and not have any gaps
- We say that it has *geometric continuity*, or C^0 continuity
- We can easily see that the first derivative will be a continuous quadratic function and the second derivative will be a continuous linear function
- The third derivative (and all others) are continuous as well, but in a trivial way (constant), so we generally just say that a cubic curve has *second derivative continuity* or C^2 continuity
- In general, the higher the continuity value, the ‘smoother’ the curve will be, although it’s actually a little more complicated than that...

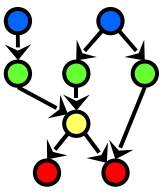




Interpolation / Approximation

- We say that cubic Bezier curves *interpolate* the two endpoints (\mathbf{p}_0 & \mathbf{p}_3), but only *approximate* the interior points (\mathbf{p}_1 & \mathbf{p}_2)
- In geometric design applications, it is often desirable to be able to make a single curve that interpolates through several points



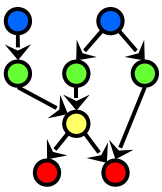


Piecewise Curves

- Rather than use a very high degree curve to interpolate a large number of points, it is more common to break the curve up into several simple curves
- For example, a large complex curve could be broken into cubic curves, and would therefore be a *piecewise cubic curve*
- For the entire curve to look smooth and continuous, it is necessary to maintain C^1 continuity across segments, meaning that the position and tangents must match at the endpoints
- For smoother looking curves, it is best to maintain the C^2 continuity as well

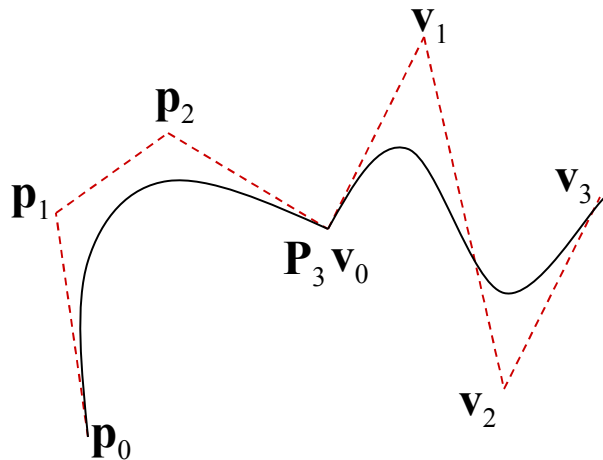
knot \equiv join point



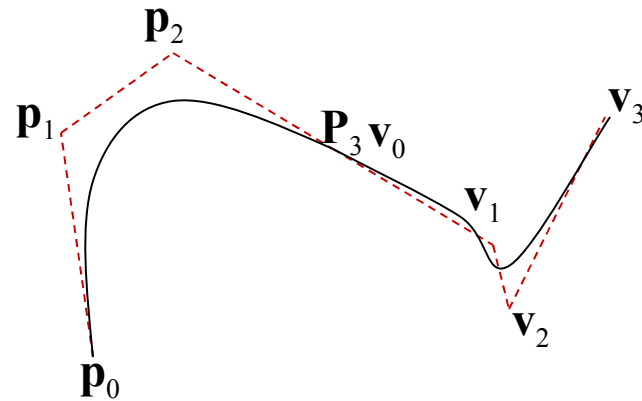


Connecting Bezier Curves

- A simple way to make larger curves is to connect up Bezier curves
- Consider two Bezier curves defined by $\mathbf{p}_0 \dots \mathbf{p}_3$ and $\mathbf{v}_0 \dots \mathbf{v}_3$
- If $\mathbf{p}_3 = \mathbf{v}_0$, then they will have C^0 continuity
- If $(\mathbf{p}_3 - \mathbf{p}_2) = (\mathbf{v}_1 - \mathbf{v}_0)$, then they will have C^1 continuity
- C^2 continuity is more difficult...



C^0 continuity



C^1 continuity

