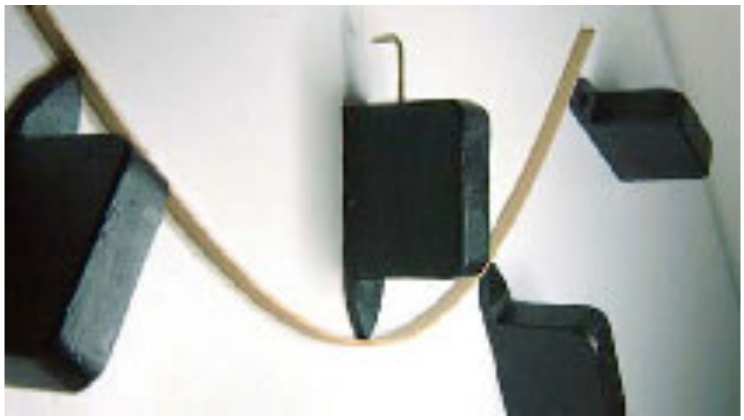
CS148: Introduction to Computer Graphics and Imaging

Splines and Curves



Topics

Splines

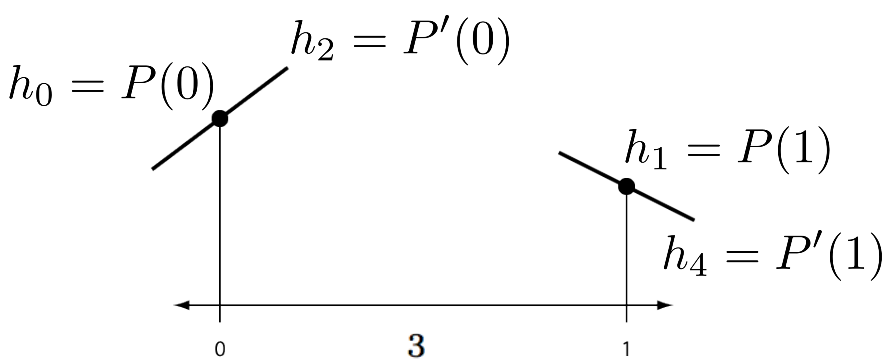
- Cubic Hermite interpolation
- Matrix representation of cubic polynomials
- Catmull-Rom interpolation

Curves

- Bezier curve
- Chaiken's subdivision algorithm
- Properties of Bezier curves

Cubic Hermite Interpolation

Cubic Hermite Interpolation



$$P(t) = \sum_{i=0}^{3} h_i \, H_i(t)$$

Given: values and derivatives at 2 points

Cubic Hermite Interpolation

Assume cubic polynomial

$$P(t) = a\,t^3 + b\,t^2 + c\,t + d$$
$$P'(t) = 3a\,t^2 + 2b\,t + c$$

Solve for coefficients:

$$P(0) = h_0 = d$$
$$P(1) = h_1 = a + b + c + d$$
$$P'(0) = h_2 = c$$
$$P'(1) = h_3 = 3a + 2b + c$$

Matrix Representation

$$h_0 = d$$
$$h_1 = a + b + c + d$$
$$h_2 = c$$
$$h_3 = 3a + 2b + c$$

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Matrix Representation of Polynomials

$$\begin{bmatrix} a & b & c & d \end{bmatrix} \underbrace{\begin{bmatrix} 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}}_{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

CS148 Lecture 8 Pat Hanrahan, Fall 2009

Hermite Basis Matrix

$$\begin{bmatrix} H_0(t) \\ H_1(t) \\ H_2(t) \\ H_3(t) \end{bmatrix} = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

$$H_0(t) = 2t^3 - 3t^2 + 1$$

$$H_1(t) = -2t^3 + 3t^2$$

$$H_2(t) = t^3 - 2t^2 + t$$

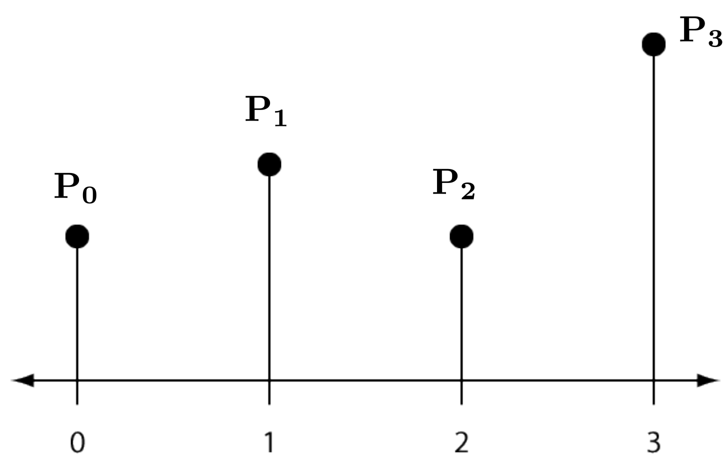
$$H_3(t) = t^3 - t^2$$

$$\mathbf{M_H} = \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

CS148 Lecture 8 Pat Hanrahan, Fall 2009

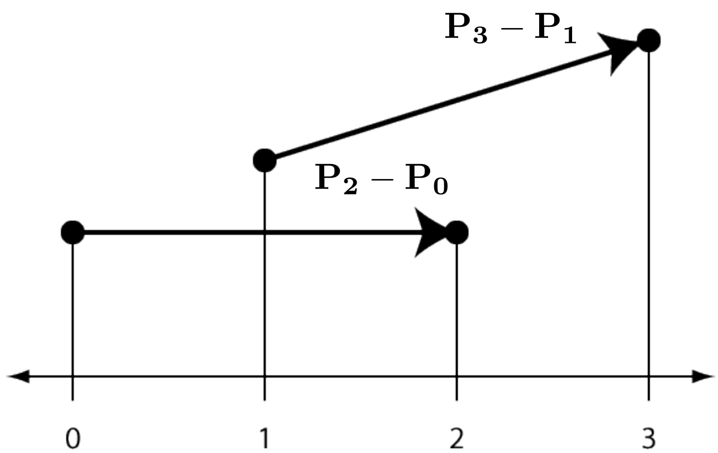
Catmull-Rom Interpolation

Catmull-Rom Interpolation



Interpolate points smoothly
Slopes not given though

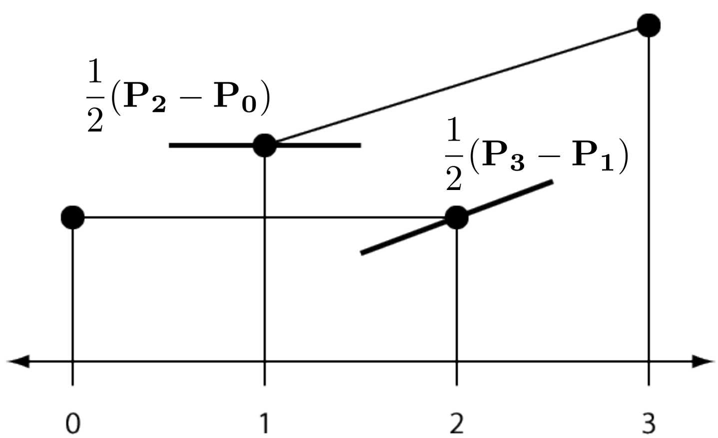
Catmull-Rom Interpolation



CS148 Lecture 8

Pat Hanrahan, Fall 2009

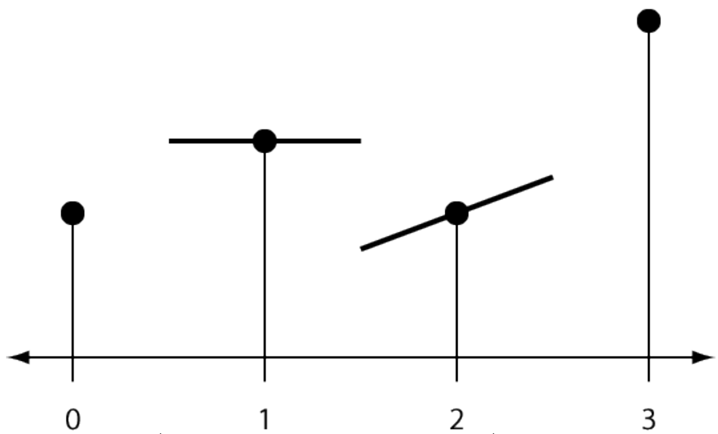
Catmull-Rom Interpolation



CS148 Lecture 8

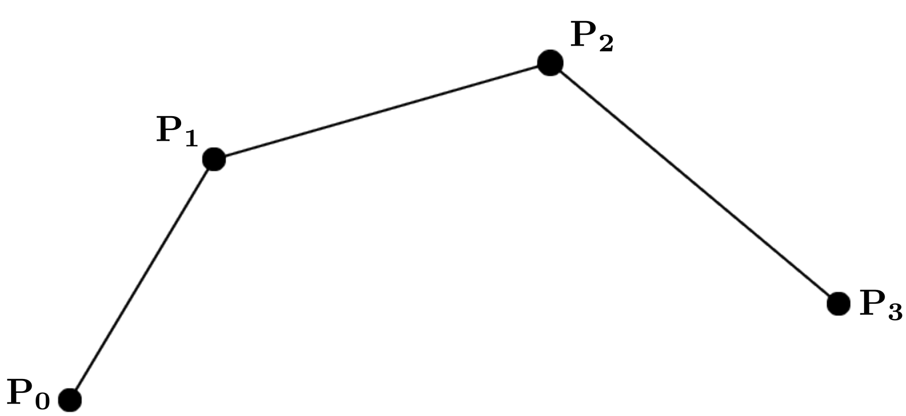
Pat Hanrahan, Fall 2009

Catmull-Rom Interpolation

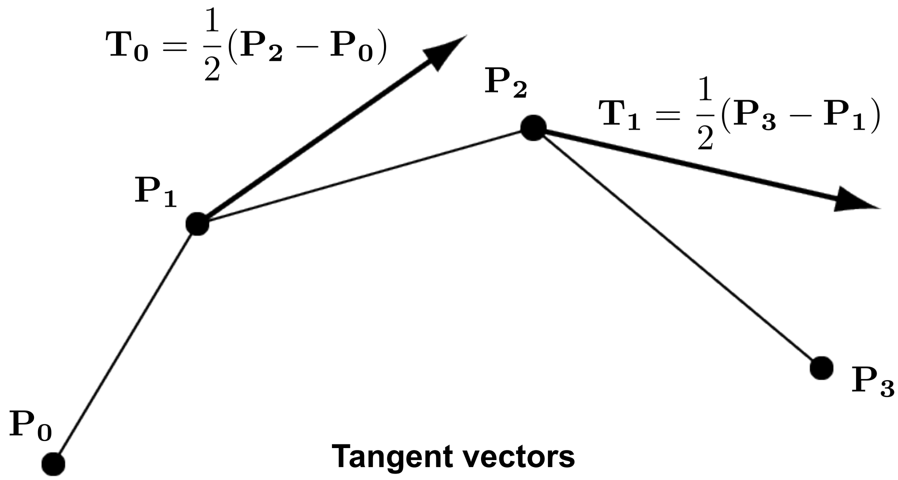


$$\mathbf{T_0} = \frac{1}{2}(\mathbf{P_2} - \mathbf{P_0}) \quad \mathbf{T_1} = \frac{1}{2}(\mathbf{P_3} - \mathbf{P_1})$$

Catmull-Rom Interpolation



Catmull-Rom Interpolation



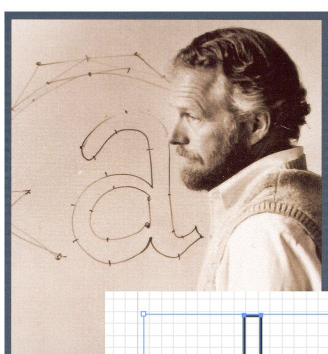
Catmull-Rom To Hermite Interpolation

$$\mathbf{P_0} = \mathbf{P_1}$$
$$\mathbf{P_1} = \mathbf{P_2}$$
$$\mathbf{T_0} = \frac{1}{2}(\mathbf{P_2} - \mathbf{P_0})$$
$$\mathbf{T_1} = \frac{1}{2}(\mathbf{P_3} - \mathbf{P_1})$$

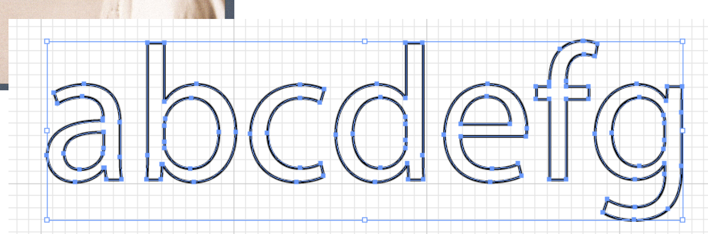
CS148 Lecture 8 Pat Hanrahan, Fall 2009

Bezier Curves

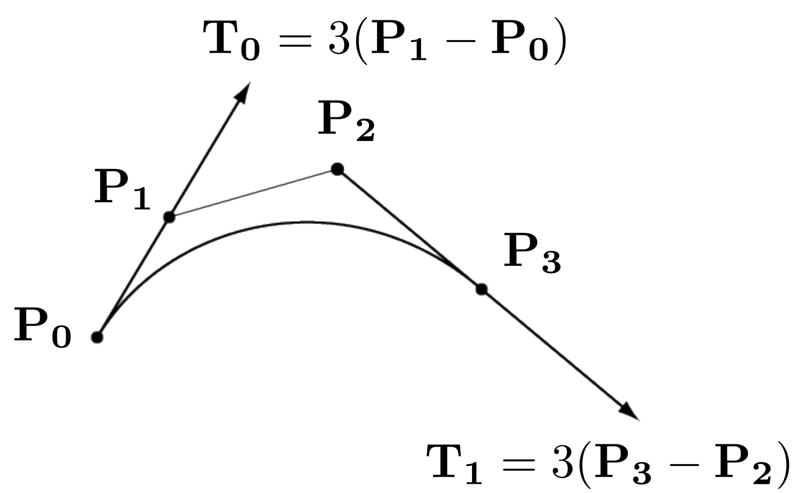
Paths



Capabilities
1.Smooth curves
2.Line and curve segments
3.Sharp corners



Cubic Bezier Curve



Bezier To Hermite Interpolation

$$\mathbf{P_0} = \mathbf{P_0}$$

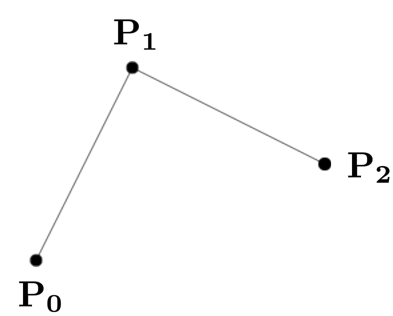
$$\mathbf{P_1} = \mathbf{P_3}$$

$$\mathbf{T_0} = 3(\mathbf{P_1} - \mathbf{P_0})$$

$$\mathbf{T_1} = 3(\mathbf{P_3} - \mathbf{P_2})$$

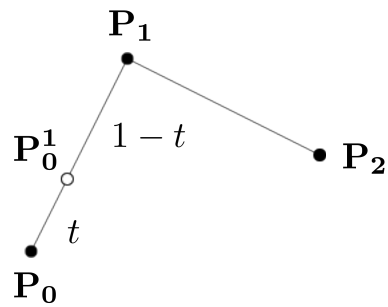
Subdivision

Quadratic Bezier Curve (3 Pts)



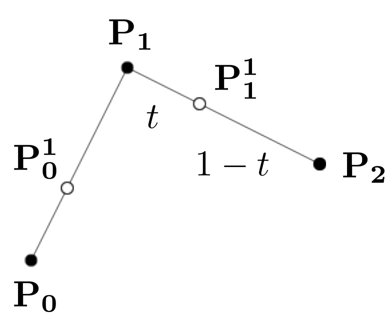
Chaiken's Algorithm

$$\mathbf{P_0^1} = (1-t)\mathbf{P_0} + t\mathbf{P_1}$$



Chaiken's Algorithm

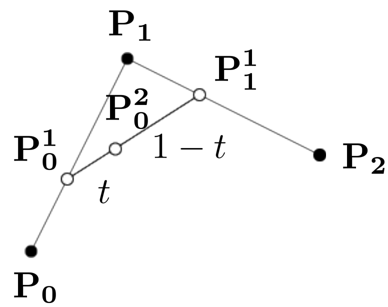
$$\mathbf{P_0^1} = (1-t)\mathbf{P_0} + t\mathbf{P_1}$$
$$\mathbf{P_1^1} = (1-t)\mathbf{P_1} + t\mathbf{P_2}$$



Chaiken's Algorithm

$$\mathbf{P_0^1} = (1-t)\mathbf{P_0} + t\mathbf{P_1}$$
$$\mathbf{P_1^1} = (1-t)\mathbf{P_1} + t\mathbf{P_2}$$

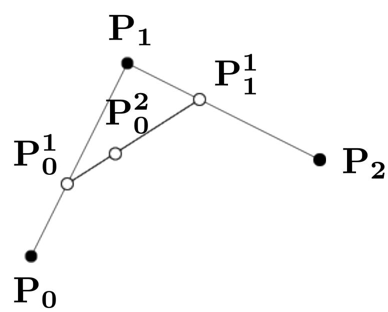
$$\mathbf{P_0^2} = (1-t)\mathbf{P_0^1} + t\mathbf{P_1^1}$$



Chaiken's Algorithm

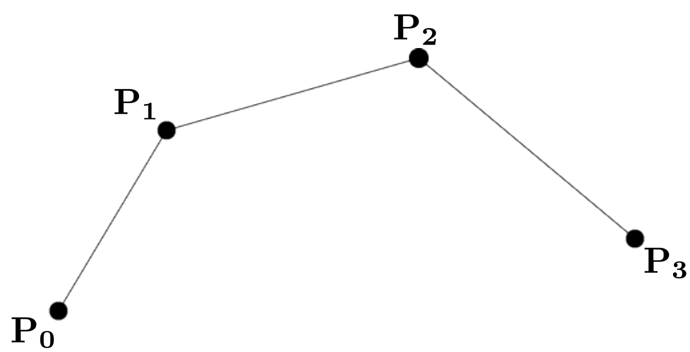
$$\mathbf{P_0^1} = (1-t)\mathbf{P_0} + t\mathbf{P_1}$$
$$\mathbf{P_1^1} = (1-t)\mathbf{P_1} + t\mathbf{P_2}$$

$$\mathbf{P_0^2} = (1-t)\mathbf{P_0^1} + t\mathbf{P_1^1}$$

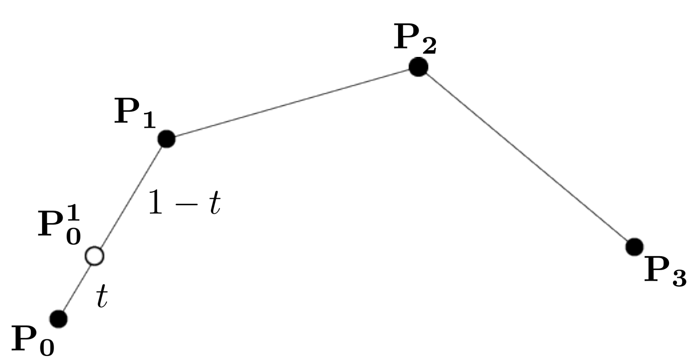


$$\mathbf{P}(t) = \mathbf{P_0^2}$$

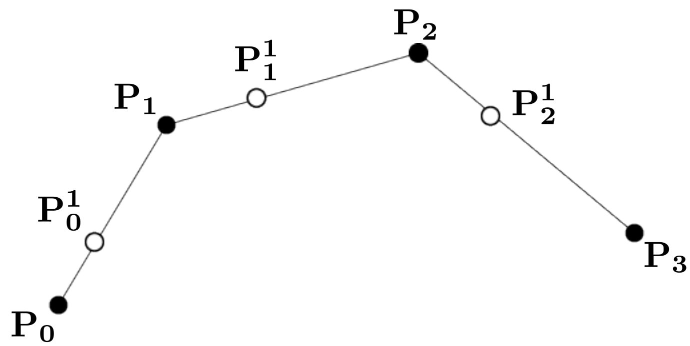
Cubic Bezier Curve (4 Pts)



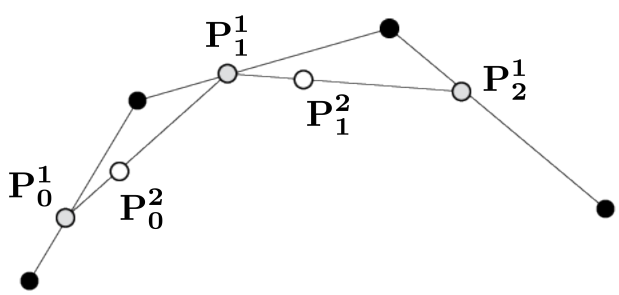
Bezier Curve



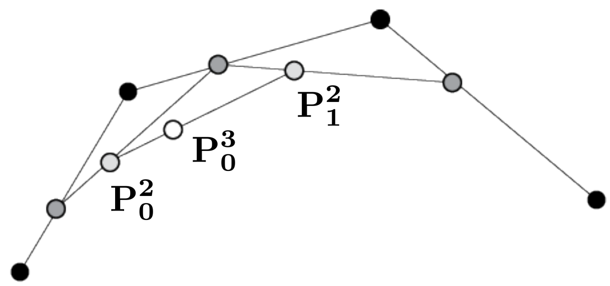
Bezier Curve



Bezier Curve



Bezier Curve



Properties

Property 1: Interpolate end points

$$\mathbf{P}(0) = \mathbf{P_0}$$

$$\mathbf{P}(1) = \mathbf{P_3}$$

Property 2: Tangents

$$\mathbf{P}'(0) = 3(\mathbf{P_1} - \mathbf{P_0})$$

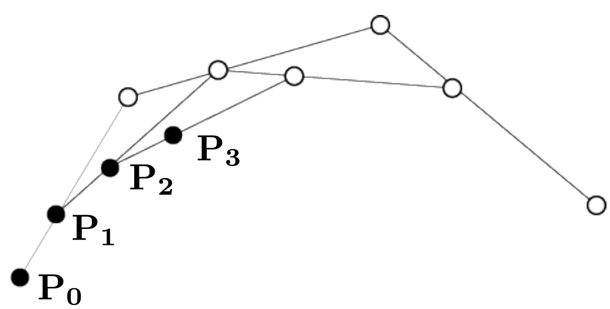
$$\mathbf{P}'(1) = 3(\mathbf{P_3} - \mathbf{P_2})$$

Property 3: Convex hull property

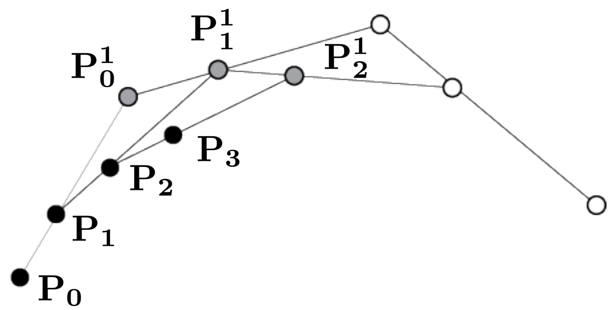
$$\mathbf{P}(t) \text{ inside chull}(\mathbf{P_0}, \mathbf{P_1}, \mathbf{P_2}, \mathbf{P_2})$$

Extrapolation and Subdivision

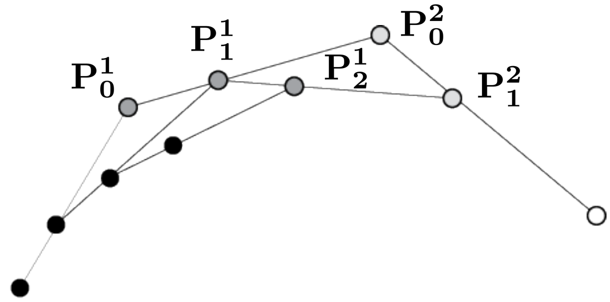
Bezier Curve - Extrapolation



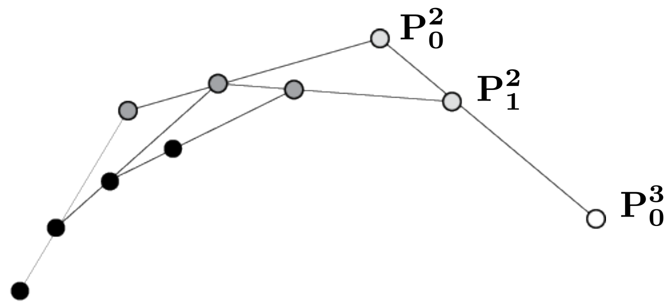
Bezier Curve - Extrapolation



Bezier Curve - Extrapolation

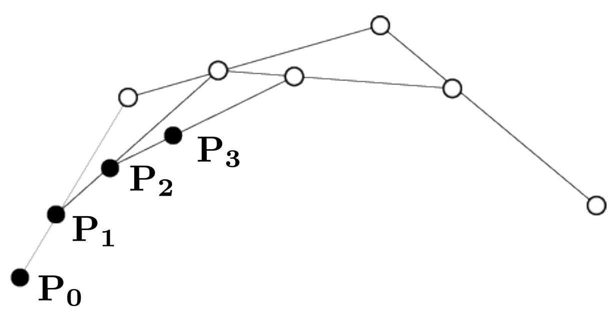


Bezier Curve - Extrapolation



CS148 Lecture 8 Pat Hanrahan, Fall 2009

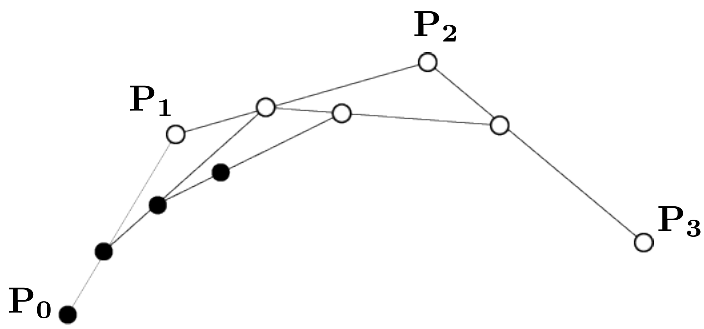
Bezier Curve



Original Bezier Curve

CS148 Lecture 8 Pat Hanrahan, Fall 2009

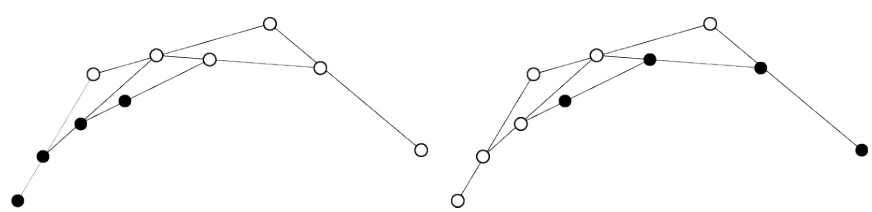
Bezier Curve



New Bezier Curve
Can run the algorithm in reverse to get the original control points. Can prove that the original curve is a piece of the new curve

Bezier Curve

Evaluate the algorithm at t=1/2
This subdivides the curve into two pieces



Left Bezier Curve **Right Bezier Curve**

Applications of Subdivision

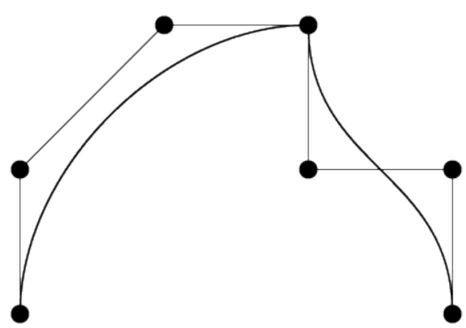
Drawing Bezier curve

??

Intersect two Bezier curves

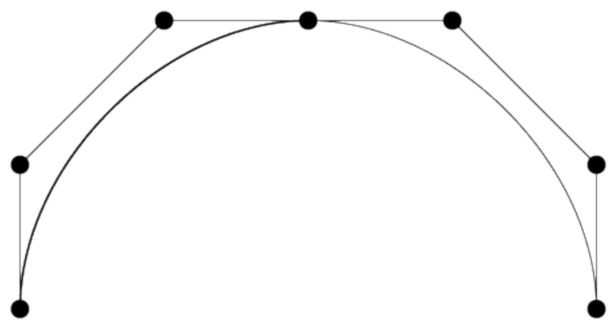
??

Continuity between 2 Bezier Curves



3rd point of the 1st curve is the same as the 1st point of the 2nd curve

Continuity between 2 Bezier Curves

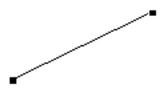


Tangent of the 1st curve is equal to the tangent of the 2nd curve

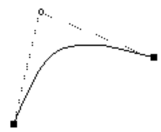
Demo of Bezier Curves

TrueType Quadratic Bezier Splines

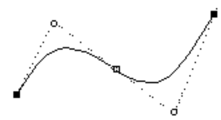
Successive control points are marked as either on or off the curve



two successive *on* points form a line segment.



a *conic off* point between two *on* points forms a conic bezier arc



two successive *conic off* points define a *virtual on* point at their exact middle. It is used to join two conic arcs

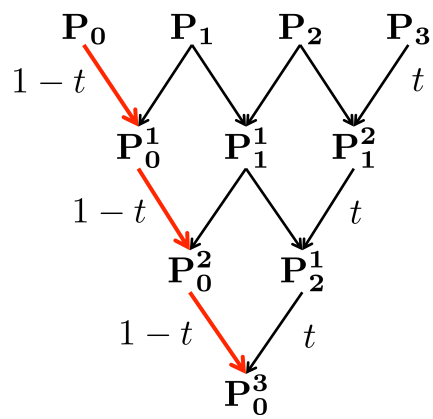
http://freetype.sourceforge.net/freetype2/docs/glyphs/glyphs-6.html

Pyramid Algorithm

$$
\begin{tikzcd}[column sep=small]
\mathbf{P_0} \arrow[dr, "1-t"'] & & \mathbf{P_1} \arrow[dl] \arrow[dr] & & \mathbf{P_2} \arrow[dl] \arrow[dr] & & \mathbf{P_3} \arrow[dl, "t"] \\
& \mathbf{P_0^1} \arrow[dr, "1-t"'] & & \mathbf{P_1^1} \arrow[dl] \arrow[dr] & & \mathbf{P_1^2} \arrow[dl, "t"] & \\
& & \mathbf{P_0^2} \arrow[dr, "1-t"'] & & \mathbf{P_2^1} \arrow[dl, "t"] & & \\
& & & \mathbf{P_0^3} & & &
\end{tikzcd}
$$

$$P(t) = \sum_i^n P_i \, B_i^n(t)$$

Pyramid Algorithm

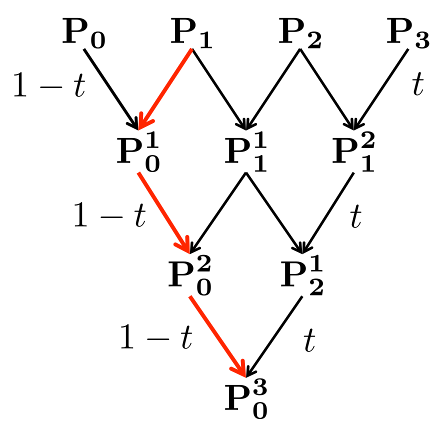


$$P(t) = \sum_i^n P_i\, B_i^n(t)$$

$$(1-t)^3\mathbf{P_0}$$

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

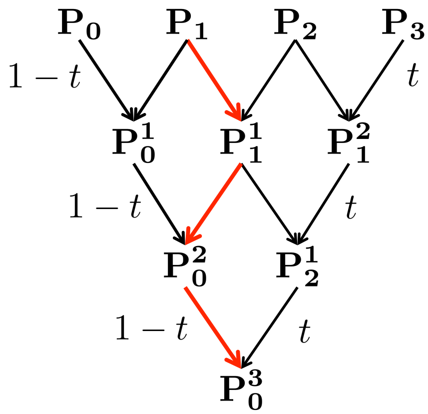
Pyramid Algorithm



$$P(t) = \sum_i^n P_i\, B_i^n(t)$$

$$t(1-t)^2\mathbf{P_1}$$

Pyramid Algorithm

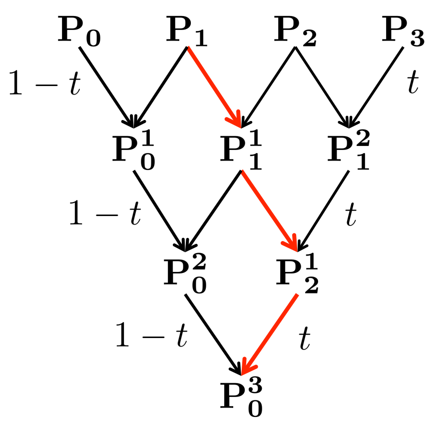


$$P(t) = \sum_i^n P_i \, B_i^n(t)$$

$$t(1-t)^2 \mathbf{P_1}$$

CS148 Lecture 8 Pat Hanrahan, Fall 2009

Pyramid Algorithm



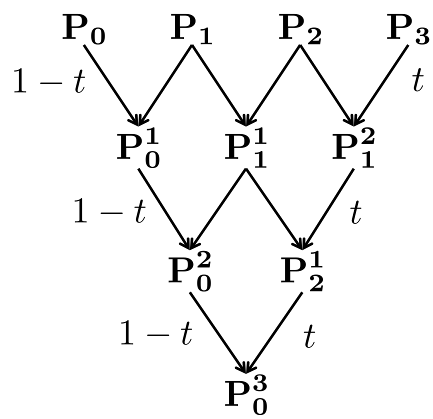
$$P(t) = \sum_i^n P_i \, B_i^n(t)$$

$$t(1-t)^2 \mathbf{P_1}$$

$$B_1^3(t) = 3t(1-t)^2$$

CS148 Lecture 8 Pat Hanrahan, Fall 2009

Bernstein Polynomials



$$P(t) = \sum_i^n P_i \, B_i^n(t)$$

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

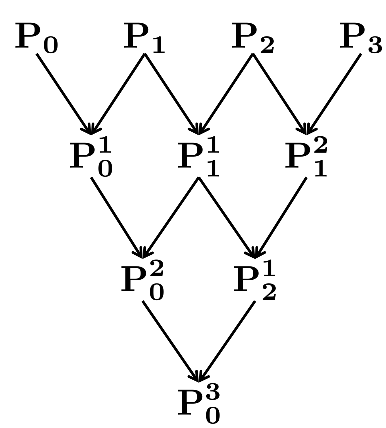
$$B_1^3(t) = 3t(1-t)^2$$

$$B_2^3(t) = 3t^2(1-t)$$

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

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

Pyramid Algorithm



$$P(t) = \sum_i^n P_i \, B_i^n(t)$$

Things to Remember

Splines

- Cubic Hermite interpolation
- Matrix representation of cubic polynomials
- Catmull-Rom (CR) splines
- How to think of CR in terms of Hermite spline

Curves

- Bezier curve (BC)
- How to think of BC in terms of Hermite spline
- Chaiken’s algorithm
- Subdivision algorithm including applications
- Properties of Bezier curves