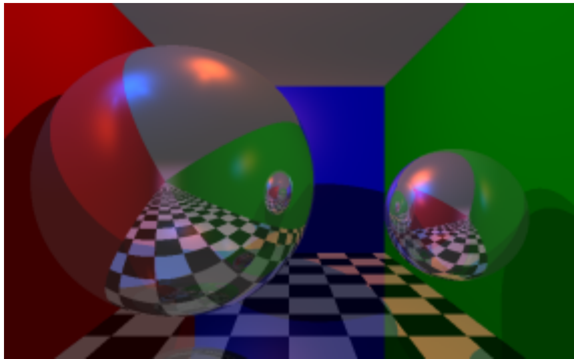
CSI 4105 Computer Graphics

Spring 2017

Lecture 10: Curves

Seon Joo Kim

Yonsei University



Objectives

- Refresher on functions and derivatives
- Discuss parametric curves
  - Motivation and formulation
- Discuss several types of curves
- Give several examples where you' ll see curves

REFRESHER ON FUNCTION

Functions and their derivatives

- Functions

$$x(u) = 5u^3 + 2u^2 + u + 5$$

- Remember, how we evaluate this:

$$x(0.5) = 5(0.5)^3 + 2(0.5)^2 + (0.5) + 5$$

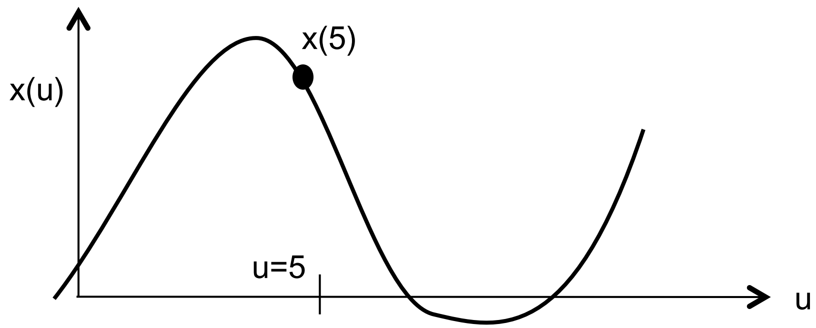
- We also have the derivative (change of the function):

$$x'(u) = (3)5u^2 + 2(2)u + 1$$

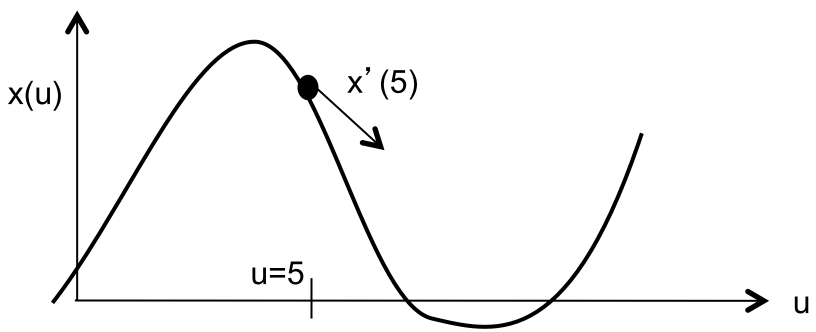
$$x'(0.5) = 15(0.5)^2 + 4(0.5) + 1$$

Function and derivative

- The function tells us how to map our parameter (u) to some output x(u)



- The derivative tells us how x(u) is changing . .



http://en.wikipedia.org/wiki/Derivative

We can take multiple derivatives

- Nth Order derivatives

$$x(u) = 5u^3 + 2u^2 + u + 5$$

$$x'(u) = 15u^2 + 4u^1 + 1 \quad \longleftarrow \text{1st derivative}$$

$$x''(u) = 30u^1 + 4 \quad \longleftarrow \text{2nd derivative}$$

$$x'''(u) = 30 \quad \longleftarrow \text{3rd derivative}$$

- Other notations for derivatives
  - [convention is *f(x)* instead of *x(u)*, or *y=f(x)*]

$$\frac{dy}{dx}, \quad \frac{df}{dx}(x), \quad \text{or} \quad \frac{d}{dx}f(x) \qquad\qquad \frac{d^n y}{dx^n}, \quad \frac{d^n f}{dx^n}(x), \quad \text{or} \quad \frac{d^n}{dx^n}f(x)$$

First Order Derivatives

nth Order

Compact Notion using Summation

- Sometimes, esp in graphics, you'll see a short hand notation used to specify polynomial functions

$$x(u) = 5u^3 + 2u^2 + u + 4$$

is the same as:

$$x(u) = 5u^3 + 2u^2 + u^1 + 4u^0$$

Remember $u^0 = 1$

is the same as:

$$\boxed{\begin{aligned} x(u) &= \sum_{k=0}^{3} a_k u^k \\ a_0 &= 4, a_1 = 1, a_2 = 2, a_3 = 5 \end{aligned}}$$

This is the typical short hand notation.

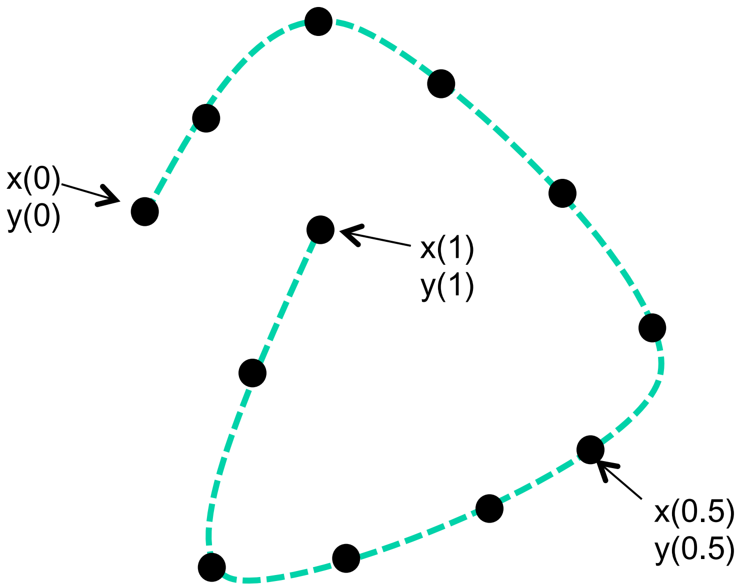
Parametric functions

- One single variable (u), but multiple functions
- We call u the parameter of the function – thus the name "parametric function"

$$x(u) = \sum_{k=0}^{3} a_k u^k$$

$$y(u) = \sum_{k=0}^{3} b_k u^k$$

$$u = [0,1]$$



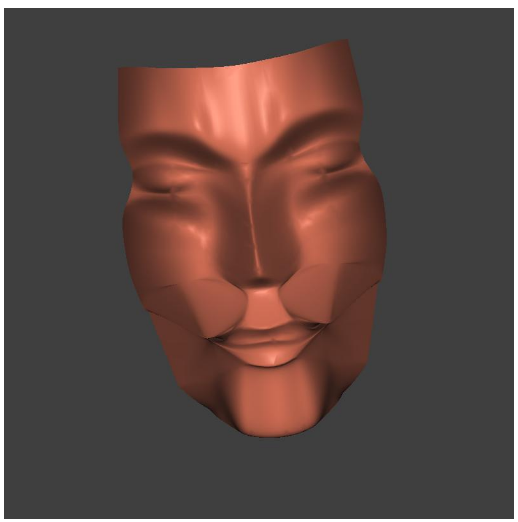
Here, we put in the same variable "u" into two different functions to get back a 2D point (x, y).

(You have already used parameteric functions, when you drew circles: $x=r\cos(\theta)$, $y=r\sin(\theta)$, in this case, θ, was the parameter)

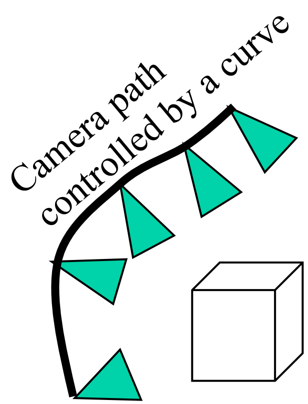
CURVES FOR GRAPHICS

Motivation for curves

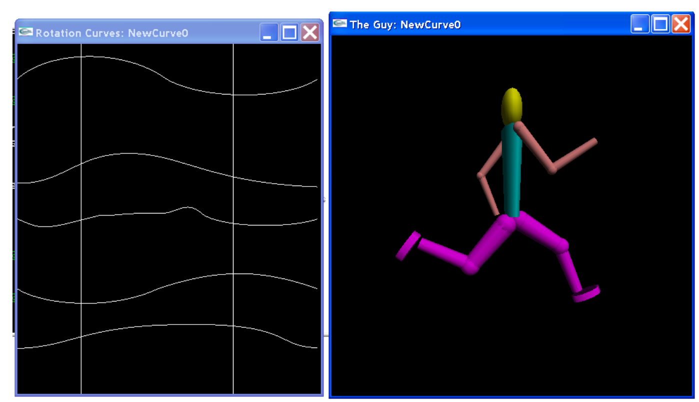
- What do we use curves for?
  - building models -- of course
- But also
  - Controlling movement paths
  - Controlling smooth animation



Surface composed of curves



Curves control the animation

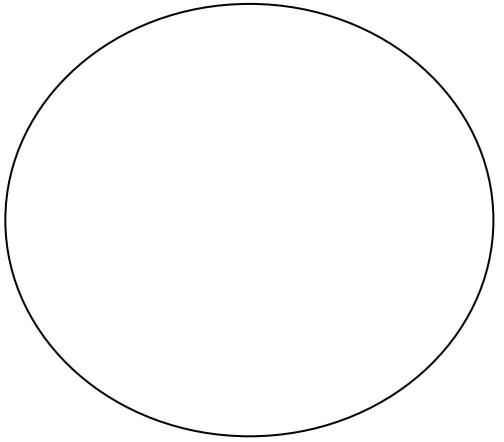


Mathematical Curve Representation

- Explicit $y = f(x)$
  - what if the curve is not a function?
- Implicit $f(x,y,z) = 0$

  $$x^2 + y^2 - R^2 = 0$$

  - hard to work with



- Parametric $(x(u), y(u)) \quad u \in [0,1]$
  - easier to work with

$$x(\boldsymbol{u}) = \cos 2\pi \boldsymbol{u}$$

$$y(\boldsymbol{u}) = \sin 2\pi \boldsymbol{u}$$

Parametric Polynomial Curves

- We' ll use parametric curves -- our functions will be polynomials of the parameter.

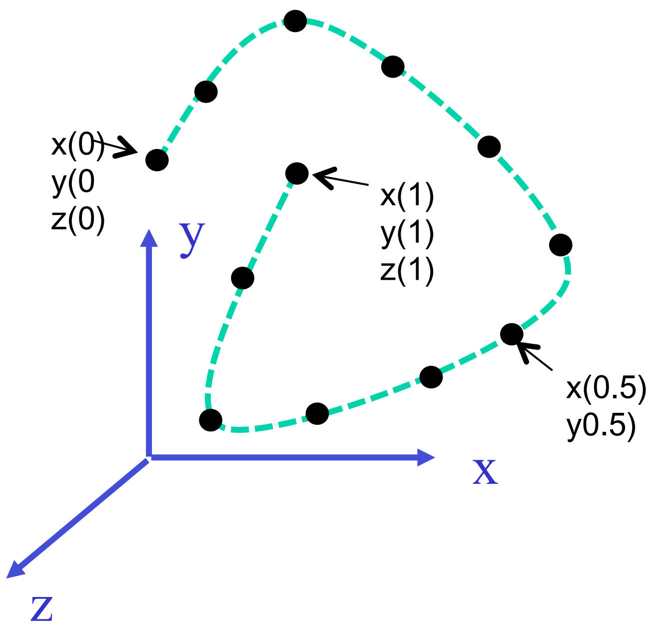
- Advantages
  - easy (and efficient) to compute
  - differentiable

$$x(u) = \sum_{k=0}^{3} a_k u^k$$

$$y(u) = \sum_{k=0}^{3} b_k u^k$$

$$z(u) = \sum_{k=0}^{3} c_k u^k$$

$$u = [0,1]$$



Cubic Curves

- For most graphics purposes, we generally don't need to use polynomials greater than n=3

$$\mathrm{p}(u) = c_3 u^3 + c_2 u^2 + c_1 u + c_0$$

$$\mathrm{p}(u) = \sum_{k=0}^{3} c_k u^k$$

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}$$

We can use a compact form and vector-matrix notation.

$$\mathrm{p}(u) = \mathbf{u}^T \mathbf{c} = \mathbf{c}^T \mathbf{u}$$

Nice thing about the vec-mat notation

- Taking derivative is very easy

$$\mathrm{p}(u) = c_3 u^3 + c_2 u^2 + c_1 u + c_0$$

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}$$

$$p(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\mathrm{p}'(u) = 3c_3 u^2 + 2c_2 u^1 + c_1 + 0c_0$$

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 1 \\ 2u^1 \\ 3u^2 \end{bmatrix}$$

$$p'(u) = \begin{bmatrix} 0 & 1 & 2u^1 & 3u^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Types of Curves

- We will discuss “approximating” curves
  - Discuss two key types

    (1) Hermite Curves

    (2) Bézier Curves

Controlling the Curve - Hermite

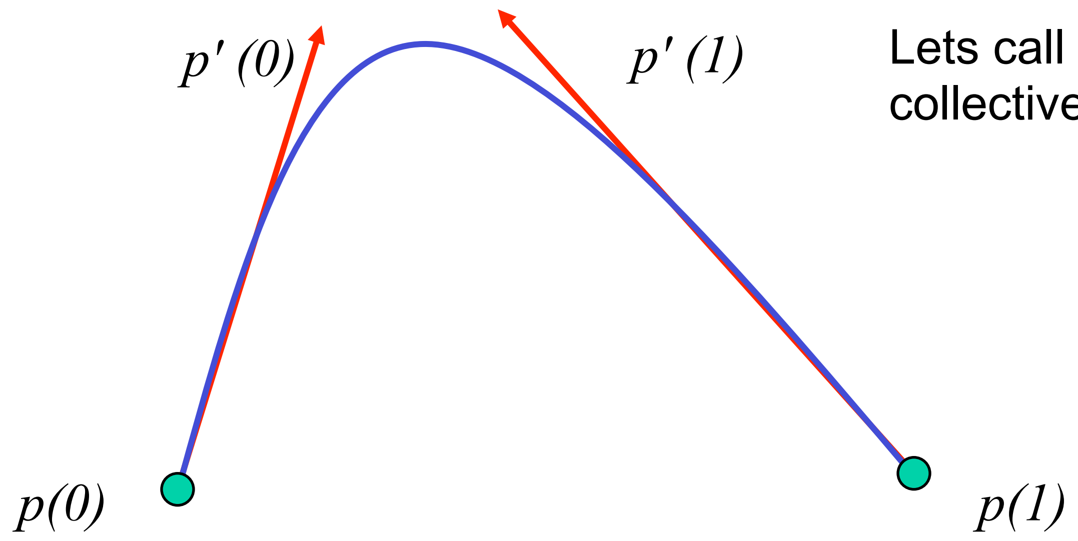
Named for Charles Hermite
Great French Mathematician

https://www.youtube.com/watch?v=bTOWVuwnOHc

- The Hermite curve requires 4 constraints
  - The 2 end points – p(0), p(1)
  - The 2 tangent vectors at the end point, - p'(0), p'(1)
- The geometric constraints are supplied by the user/programmer and control the shape of the curve
- The task now is to compute the correct *c0,c1,c2,c3* given the constraints
  - Remember, we actually have 2 or 3 functions (depending if 2D or 3D curve)

$$
\mathrm{p}(u) = c_3 u^3 + c_2 u^2 + c_1 u + c_0 \quad
\begin{cases}
\mathrm{p_x}(u) = c_{3x} u^3 + c_{2x} u^2 + c_{1x} u + c_{0x} \\
\mathrm{p_y}(u) = c_{3y} u^3 + c_{2y} u^2 + c_{1y} u + c_{0y} \\
\mathrm{p_z}(u) = c_{3z} u^3 + c_{2z} u^2 + c_{1z} u + c_{0z}
\end{cases}
$$

User provides 4 constraints



Lets call these constraints collectively: q

$$\mathbf{q} = \begin{bmatrix} \mathrm{p}_0 \\ \mathrm{p}_3 \\ \mathrm{p}'_0 \\ \mathrm{p}'_3 \end{bmatrix} = \begin{bmatrix} \mathrm{p}_{0x} & \mathrm{p}_{0y} & \mathrm{p}_{0z} \\ \mathrm{p}_{3x} & \mathrm{p}_{3y} & \mathrm{p}_{3z} \\ \mathrm{p}'_{x0} & \mathrm{p}'_{y0} & \mathrm{p}'_{z0} \\ \mathrm{p}'_{x3} & \mathrm{p}'_{y3} & \mathrm{p}'_{z3} \end{bmatrix}$$

Here:
p0 = p(0)
p3 = p(1)

We use this notation, because later we will see that Bezier curves are related to Hermite curves.

What do we know?

$$\mathbf{q} = \begin{bmatrix} \mathrm{p}_0 \\ \mathrm{p}_3 \\ \mathrm{p}'_0 \\ \mathrm{p}'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_{ox} & c_{oy} & c_{oz} \\ c_{1x} & c_{1y} & c_{1z} \\ c_{2x} & c_{2y} & c_{2z} \\ c_{3x} & c_{3y} & c_{3z} \end{bmatrix}$$

Why?

$$\mathrm{p}(u) = c_3 u^3 + c_2 u^2 + c_1 u + c_0$$

$$\mathrm{p}'(u) = 3c_3 u^2 + 2c_2 u^1 + c_1$$

$$\mathrm{p}_0 = \mathrm{p}(0) = c_3(0)^3 + c_2(0)^2 + c_1(0) + c_0$$

$$\mathrm{p}_3 = \mathrm{p}(1) = c_3(1)^3 + c_2(1)^2 + c_1(1) + c_0$$

$$\mathrm{p}'_0 = \mathrm{p}'(0) = 3c_3(0)^2 + 2c_2(0)^1 + c_1$$

$$\mathrm{p}'_3 = \mathrm{p}'(1) = 3c_3(1)^2 + 2c_2(1)^1 + c_1$$

Multiple out the matrix and you'll see that:

$$\mathrm{p}_0 = \mathrm{p}(0) = c_0$$

$$\mathrm{p}_3 = \mathrm{p}(1) = c_3 + c_2 + c_1 + c_0$$

$$\mathrm{p}'_0 = \mathrm{p}'(0) = c_1$$

$$\mathrm{p}'_3 = \mathrm{p}'(1) = 3c_3 + 2c_2 + c_1$$

To derive the Hermite form we have:

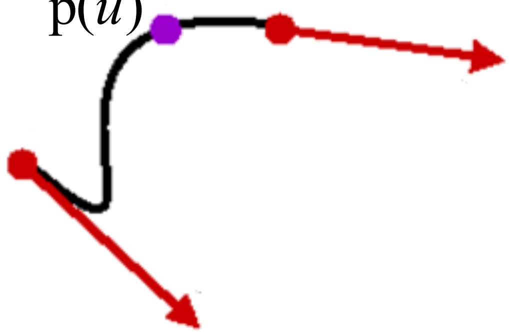
$$\mathbf{q} = \begin{bmatrix} p_0 \\ p_3 \\ p'_0 \\ p'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c}$$

Solving for **c**, we find **q=Mc**, so, $\mathbf{c} = \mathbf{M}^{-1}\mathbf{q}$. We let $\mathbf{M_H}=\mathbf{M}^{-1}$ and call it the Hermite matrix:

$$\mathbf{M}_H = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}}_{\text{We call this the Hermite Basis}} \begin{bmatrix} p_0 \\ p_3 \\ p'_0 \\ p'_3 \end{bmatrix}$$

Putting it all together

- Given two endpoints (P_o,P_3) and two endpoint tangents (p'_o,p'_3): $u \in [0,1]$



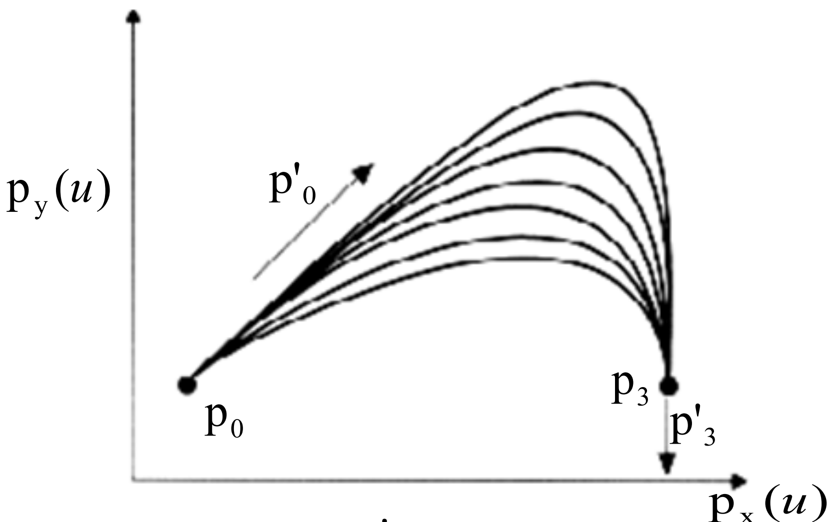
$$p(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_0 \\ p_3 \\ p'_0 \\ p'_3 \end{bmatrix}$$

Compute this vector for each **u**

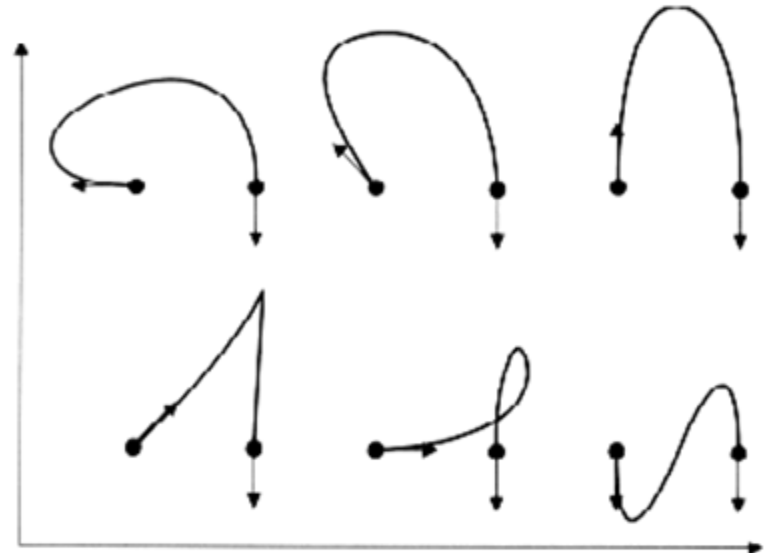
Hermite Basis

Geometric Constraints

Hermite Curve Examples



(only p'_0 is changed)



(p0 and p3 are fixed. p’ 0 and p’ 3 are changed.)

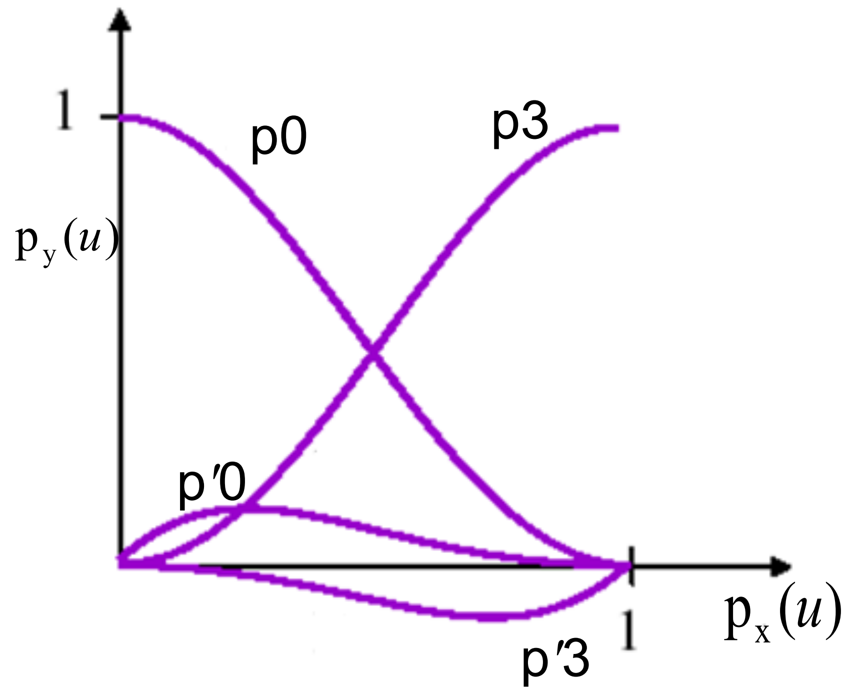
http://staff.www.ltu.se/~peppar/presentations/bibdc961114/misc_applets/ParamCurve//

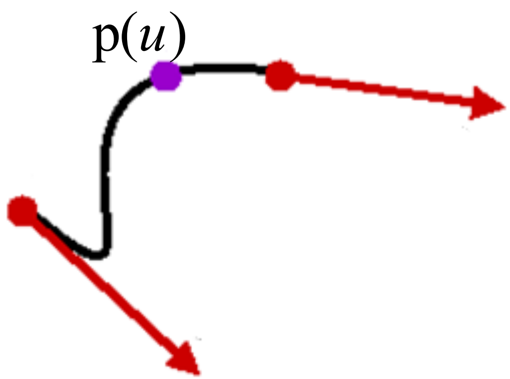
Blending Functions

- Another way to conceptualize the curve ***p(u)***
  - each point on the curve ***p(u)*** is a weighted combination of all the geometric constraints – i.e. it is a blend of the constraints
- The basis matrix (M) is defining this blend
  - M is <u>constant</u> for all Hermite curves
  - Only the geometric constraints $(p0, p3, p'0, p'3)$ change
- ***p(u)*** then is polynomial weighting of each element of the geometric constraints
- We call these polynomial weights the "blending" functions

Blending Functions

$$\underbrace{\begin{bmatrix}1 & u & u^2 & u^3\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ -3 & 3 & -2 & -1\\ 2 & -2 & 1 & 1\end{bmatrix}}_{B_H(u)}\begin{bmatrix}p_0\\ p_3\\ p'_0\\ p'_3\end{bmatrix}$$

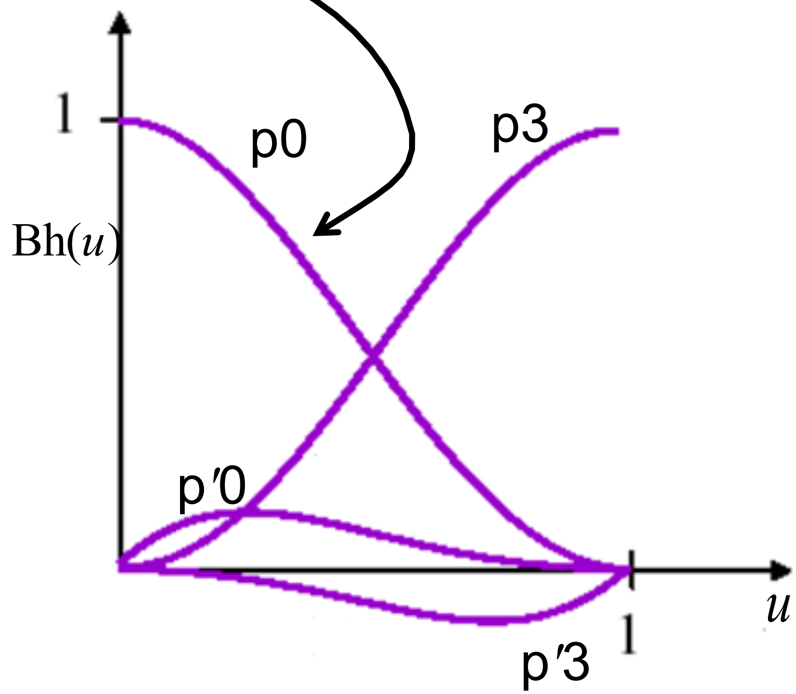


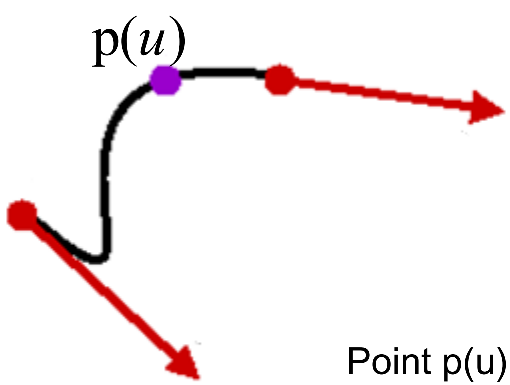


Point ***p(u)*** position is a blending of the 4 geometric constraints, based on the Hermite blending functions, $\boldsymbol{B_H(u)}$

Blending Functions

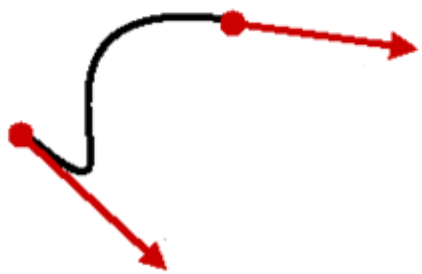
$$\underbrace{\boxed{\begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}}_{B_H(u)} \begin{bmatrix} p_0 \\ p_3 \\ p'_0 \\ p'_3 \end{bmatrix}$$



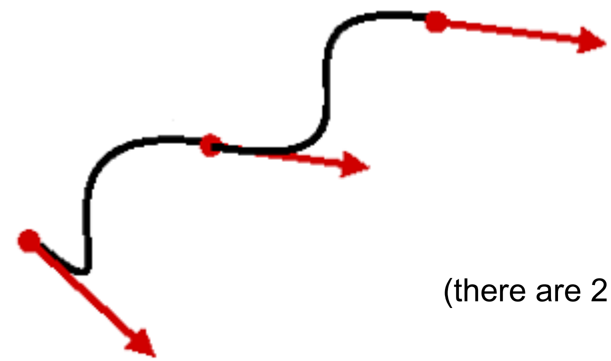


Point p(u) position is a blending of the 4 geometric constraints, based on the Hermite blending functions, $\boldsymbol{B_H(u)}$

Piecewise Curves



One curve is boring.



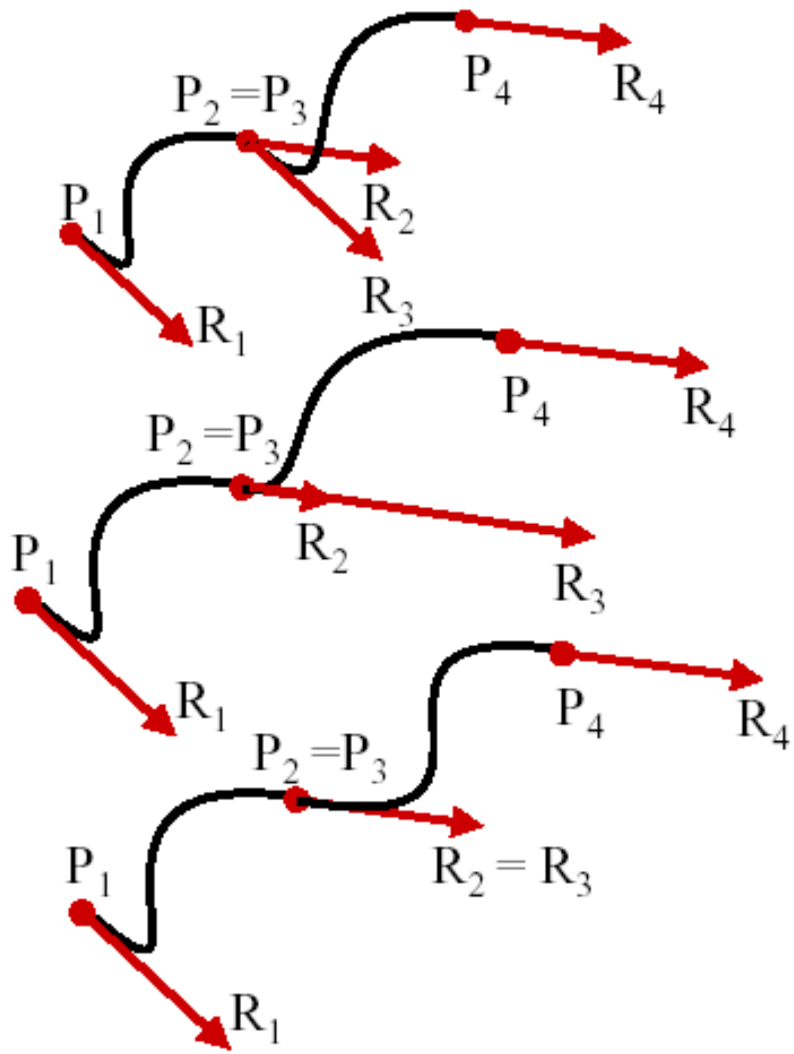
(there are 2 curves here)

We can define multiple curve segments.

How we place the geometric constraints on curve pairs controls the continuity or “smoothness”

Continuity

Here, let’s change the notation:
P1 = first point, P2=end point of a curve piece
R1 = 1st tangent, R2=2nd tangent of a curve piece



Geometric Continuity

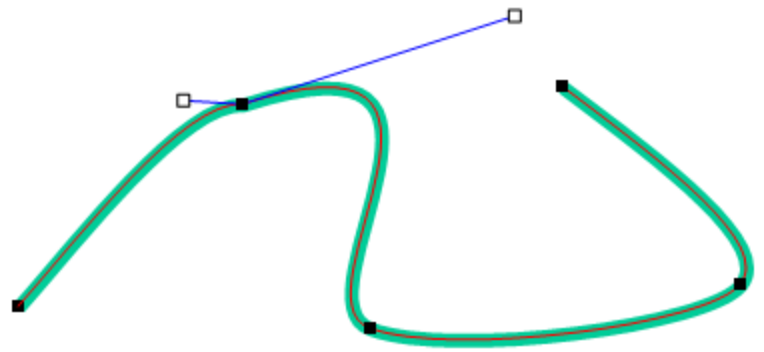
- Geometric continuity is expressed by G^n
- G^0 means points coincide, but velocity directions are different
- G^1: points coincide, velocities have the same direction.

Parametric Continuity

- Parametric continuity is expressed by C^n
- C^0: points coincide, velocity doesn’t
- C^1: points and velocities coincide
- C^2 : points,velocities, and acceleration coincide

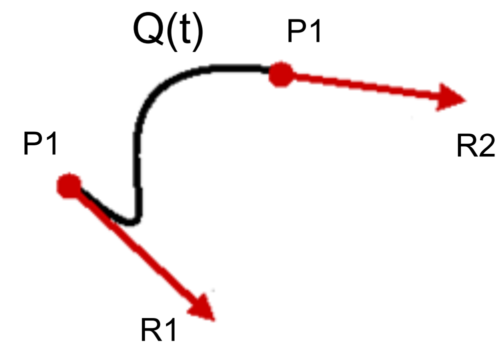
Where can you see Hermite Curves?

- Try PowerPoint 2007
  - Draw a curve, and edit the points
  - It will let you modify the Rs (tangent) vectors



Next Bézier Curves

- Going to change notions:
  - Points are now numbered starting with 1, so P1, P2, P3, P4
  - Velocity are noted by R
  - Parameter is changed to *t*



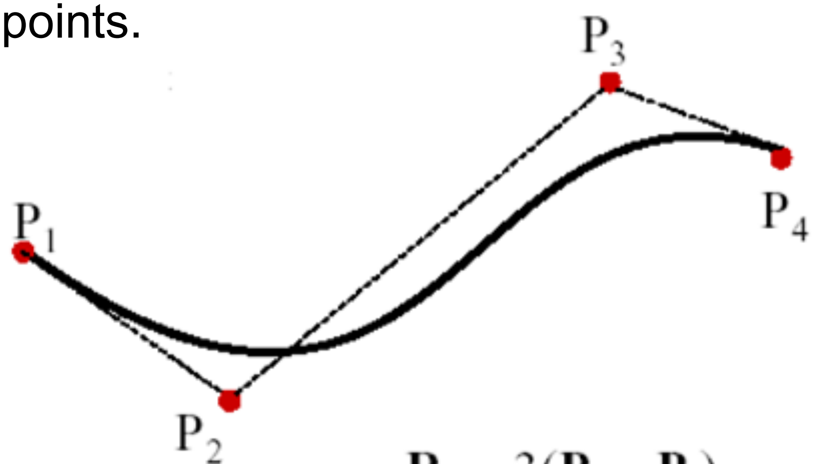
Bézier Curves

https://www.youtube.com/watch?v=6mjzYuyrbdc

Pierre Bezier

Paul de Casteljau

- Patented in 1962 by Pierre Bézier for French car manufacturer Renault
  - Developed by Paul De Casteljau in 1959
- Tries to avoid the need for specifying tangent vectors
- Instead, the approach indirectly provides the tangent vectors by specifying two intermediate points.



$$\mathbf{R}_1 = 3(\mathbf{P}_2 - \mathbf{P}_1)$$
$$\mathbf{R}_4 = 3(\mathbf{P}_4 - \mathbf{P}_3)$$

So, if we think in terms of Hermite basis, then our R1 and R4 (tangents) are now defined by the points P1, P2, P3, and P4

$$\begin{bmatrix} P_{1x} & P_{1y} & P_{1z} \\ P_{2x} & P_{2y} & P_{2z} \\ P_{3x} & P_{3y} & P_{3z} \\ P_{4x} & P_{4y} & P_{4z} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix}$$

The derivation of this relationship is a little tricky. For more details is: **http://ript.net/~spec/curves/**

Bézier-to-Hermite Matrix

- This matrix relates the Hermite and the Bézier constraints – look at it carefully:

$$\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_4 \\ \mathbf{R}_1 \\ \mathbf{R}_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix}}\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix}$$

$$\mathbf{R}_1 = 3(\mathbf{P}_2 - \mathbf{P}_1)$$
$$\mathbf{R}_4 = 3(\mathbf{P}_4 - \mathbf{P}_3)$$

Matrix converts Bézier Geometric Constraints
into Hermite Geometric Constraints. Let's call this matrix $\mathbf{M_{bh}}$

Bézier Basis Matrix

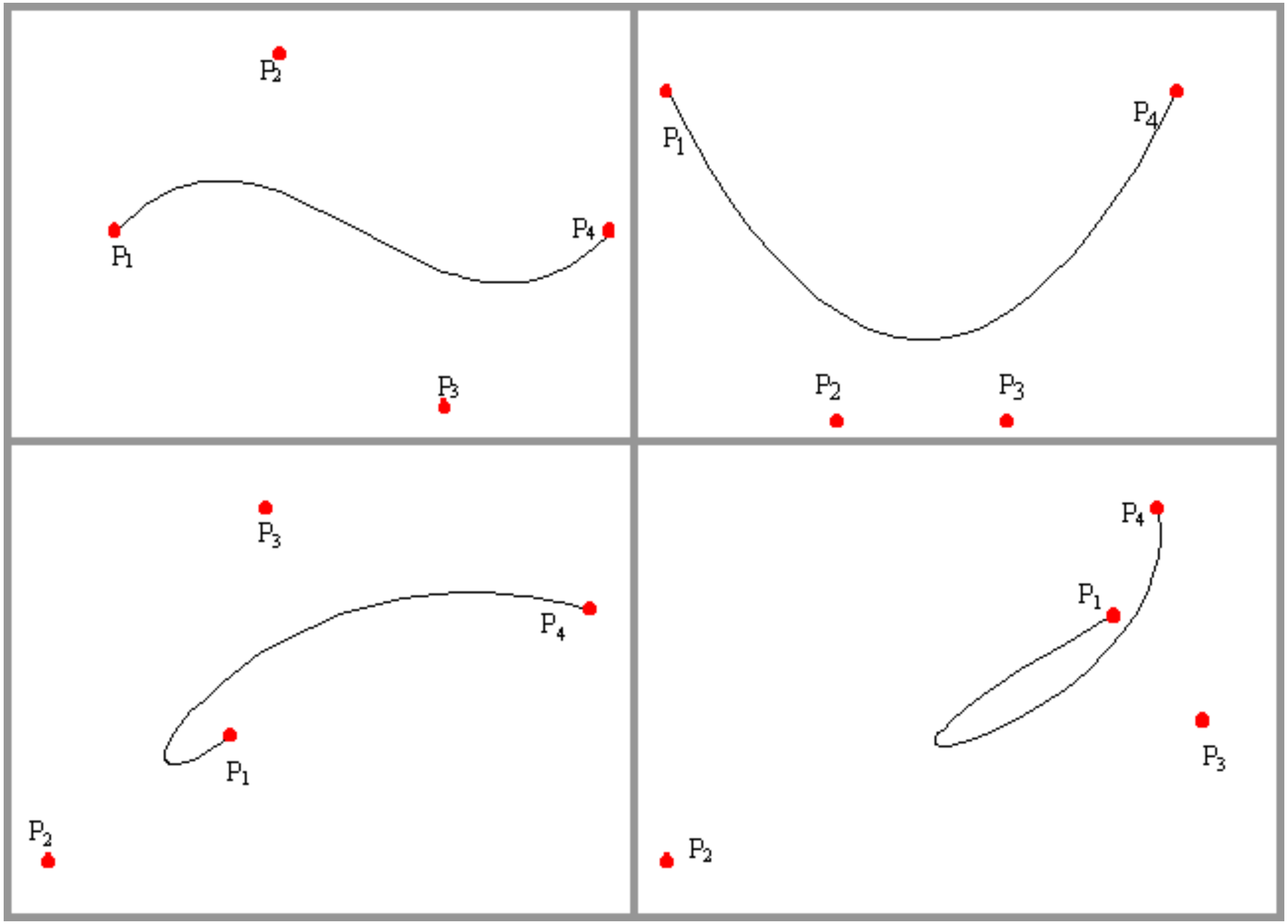
We can derive the Bézier basis matrix, $\mathbf{M_B}$, by pre-multiplying the Hermite basic M matrix by the Bézier-Hermite conversion matrix shown on the previous slide

$$Q(t) = [1 \quad t \quad t^2 \quad t^3] \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix}} \begin{bmatrix} \mathrm{p}_1 \\ \mathrm{p}_2 \\ \mathrm{p}_3 \\ \mathrm{p}_4 \end{bmatrix}$$

$$Q(t) = [1 \quad t \quad t^2 \quad t^3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{p}_1 \\ \mathrm{p}_2 \\ \mathrm{p}_3 \\ \mathrm{p}_4 \end{bmatrix}$$

Bezier Basis Matrix, call it $\mathbf{M_B}$

Bézier Examples

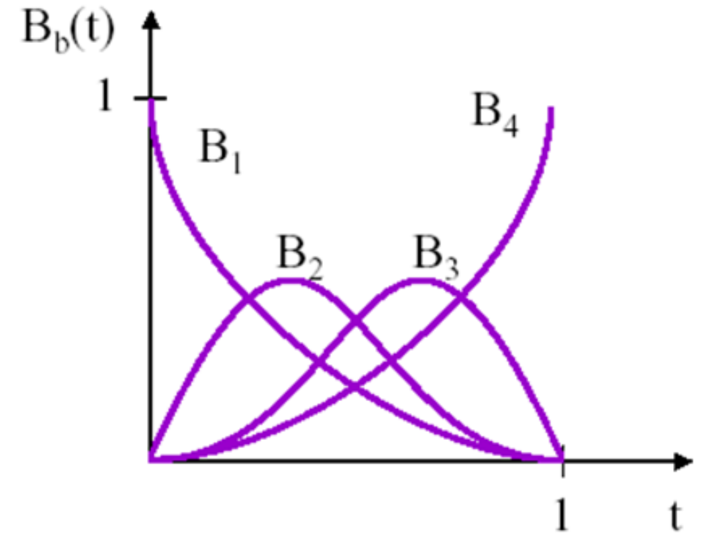


http://staff.www.ltu.se/~peppar/presentations/bibdc961114/misc_applets/ParamCurve//

Bézier Blending Functions

Just like Hermite curves, we can think of this as blending functions.
Note the Bezier blending functions don't change, only the
user supplied control points p1, p2, p3, p4.

$$Q(t) = [1 \quad t \quad t^2 \quad t^3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$



$$B_1 = 1 - 3t + 3t^2 - t^3$$

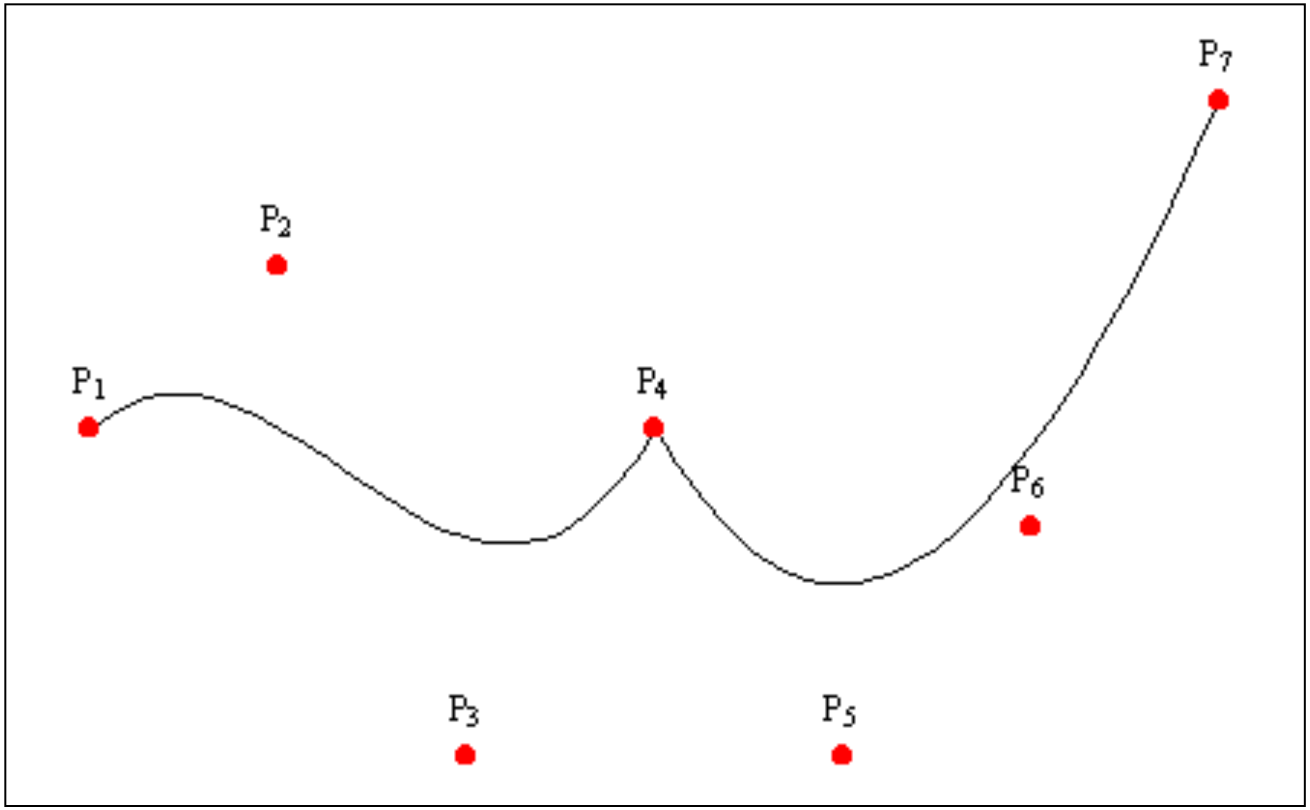
$$B_2 = 3t - 6t^2 + 3t^3$$

$$B_3 = 3t^2 - 3t^3$$

$$B_4 = t^3$$

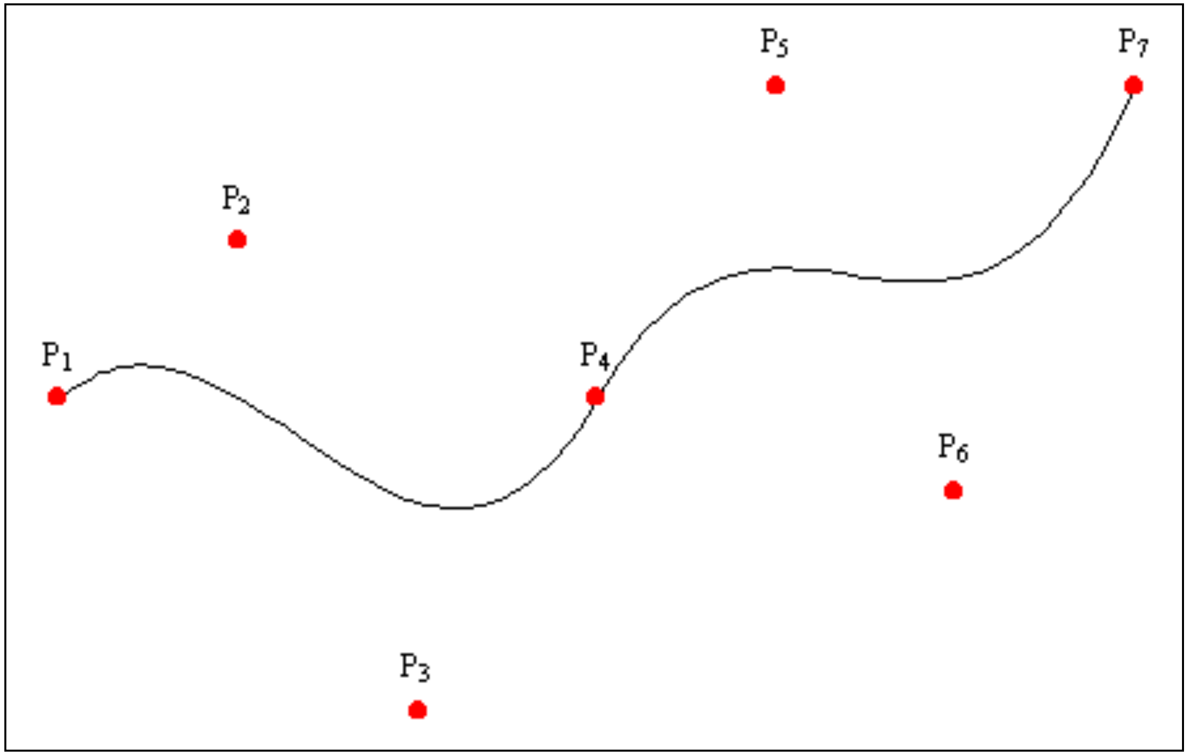
There are also known as the Bernstein polynomial

Piecewise Bézier



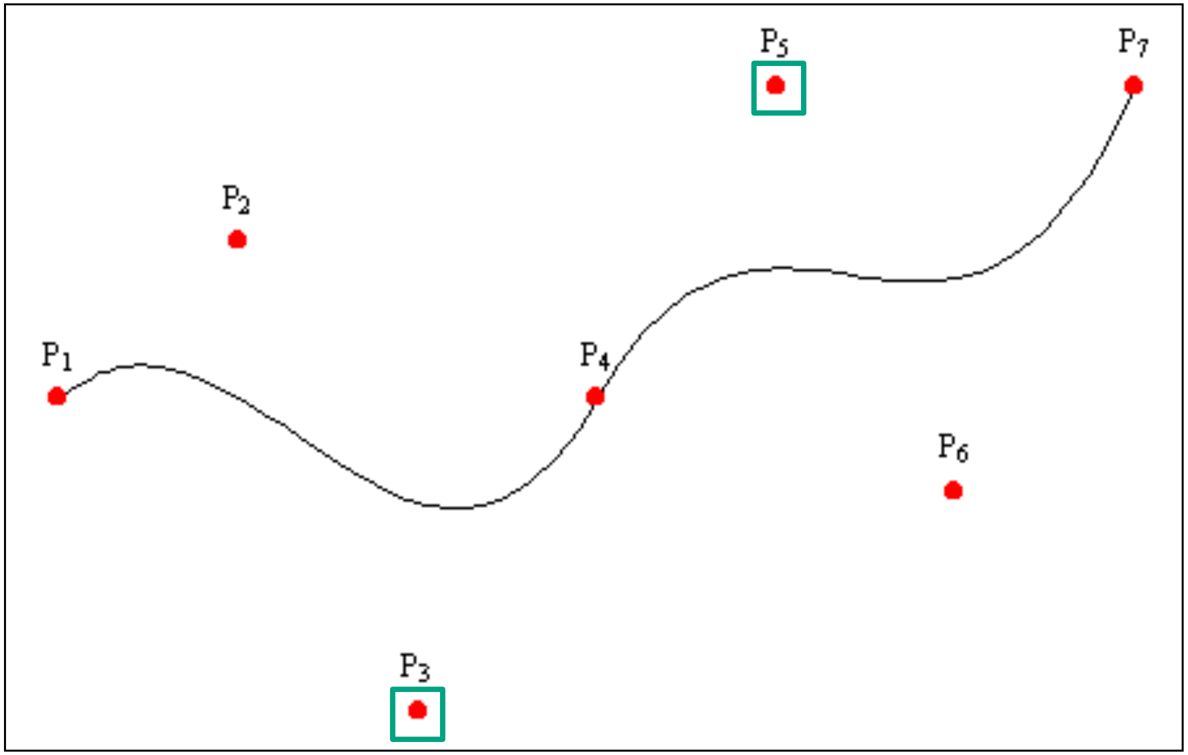
C0 continuity if adjacent control points are not co-linear

Piecewise Bézier



C^1 continuity if adjacent control points are co-linear and have the same distance to the the end-point

More than 4 control points



If we forced P3 and P5 to be “mirrored” reflection of each other about P4, we could guarantee a smooth connected curve as more points were added.

Bézier Curves Properties

- A Bézier curve Q(t) is always bounded by the convex hull of the control points

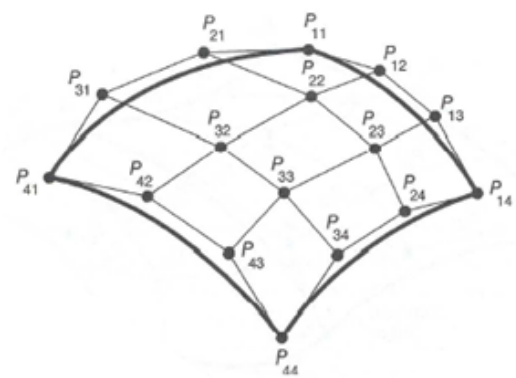
- Adjusting the shape of the curve behaves in a "predictable" manner. The curve "follows" the control point.

Extending to 3D Surfaces

- We can extended the idea to surface
- In this case, we have two parameters (u,v), and 16 control points

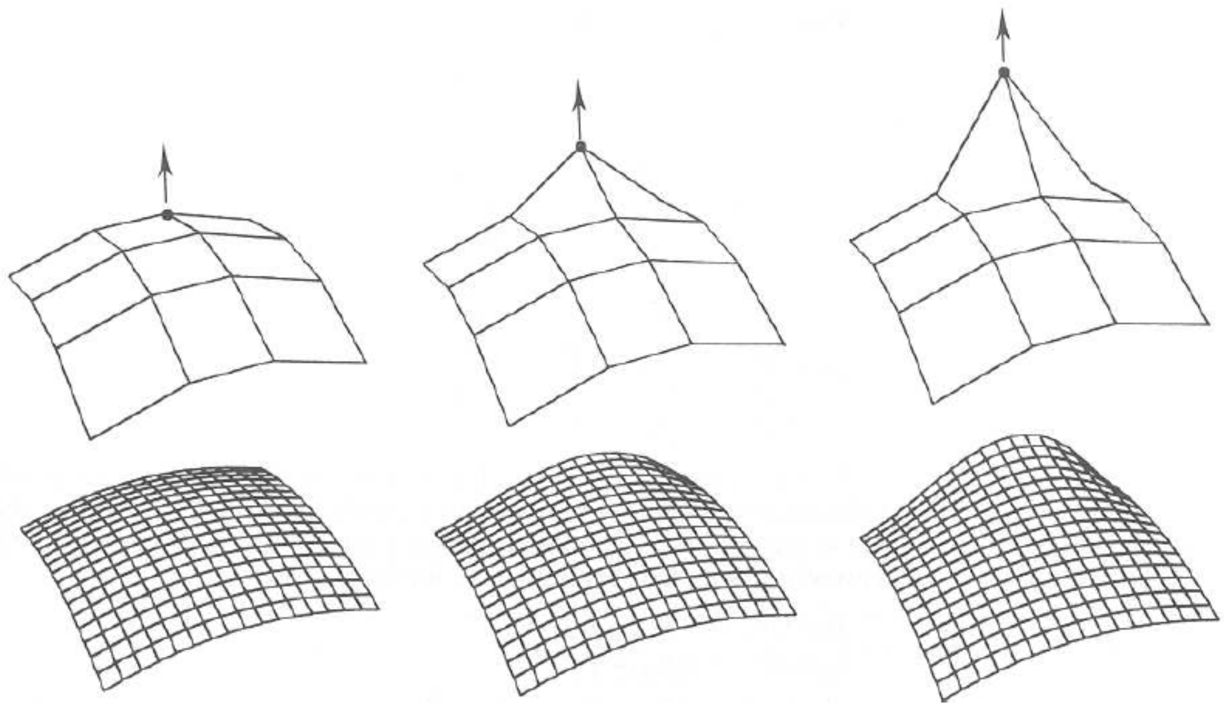
$$\mathrm{p}(u,v) = \mathbf{U}\mathbf{M}_{\mathbf{Bezier}} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \mathbf{M}^{\mathbf{T}}_{\mathbf{Bezier}}\mathbf{V}$$

In this case, its easiest to consider that the point p(u,v) is a combination of all the 16 control points, based on the Bezier blending functions. The



Bezier Patch Properties

- Interpolates four corner points
- The convex hull of the control points bound the surface
- Moving one control point affects only local portion of the surfa

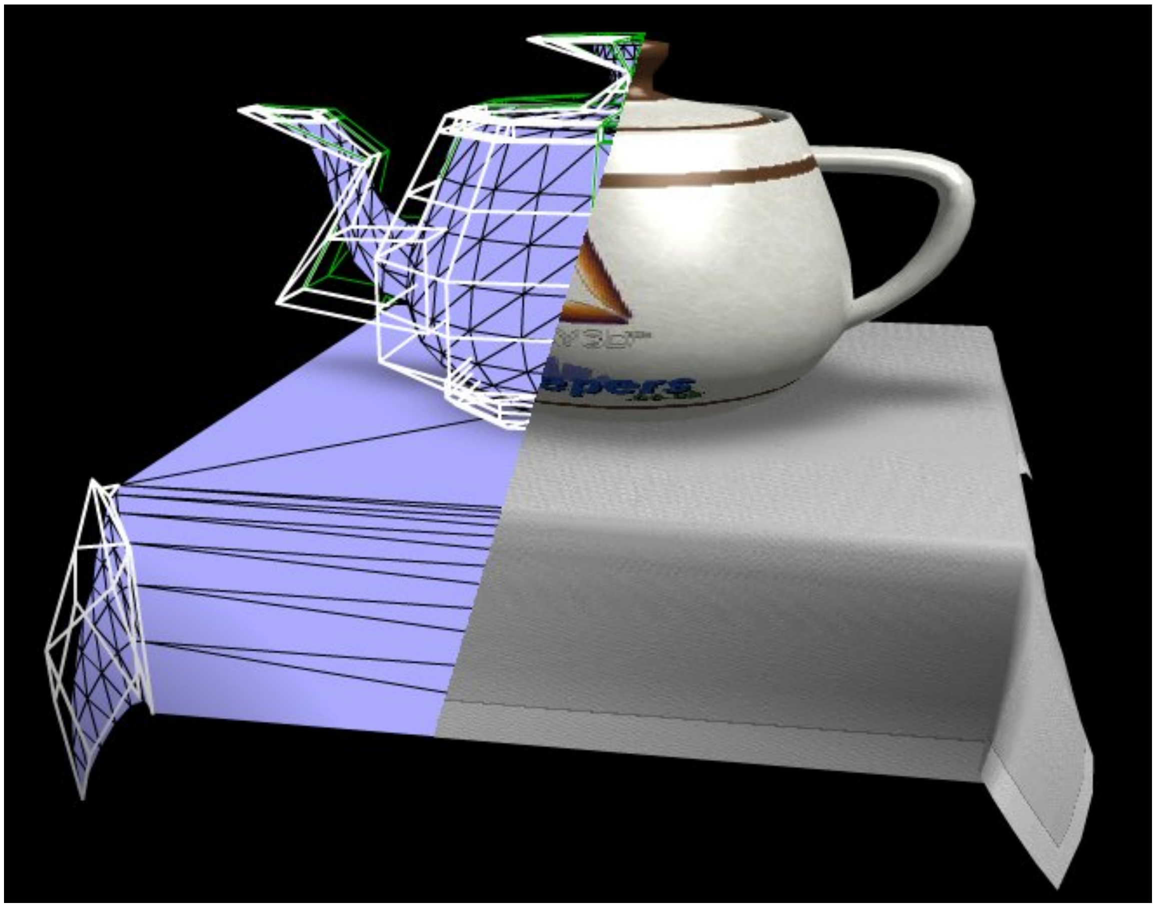


Great applet:

http://www.math.psu.edu/dlittle/java/parametricequations/beziersurfaces/index.html

Bezier Surfaces

The “Utah” teapot is actually modeled by 3D Bezier Patches.



Other Curves

- There many other types of curves
  - Catmull-Rom splines, B-Splines
    - All are some variation on the Bezier curve/Hermite Curves
  - Interpolating splines
    - See Catmul-Rom for graphics
    - Coons Patches
  - And on and on
- What you have learned here should allow you to understand other types of curves
  - Most are designed with various types of continuity and how the control points are specified

Summary

- The world is not flat
- Curves is a nice way to help us
  - We require only a few control points
  - But we get infinite resolution since it is a continuous function
- For OpenGL, the world is flat, and in the end, you'll have to convert the curves into points or polygonal mesh
- Curves can be used in other places
  - Controlling motion of objects, esp for animation