CS-C3100 Computer Graphics

Bézier Curves and Splines

Majority of slides from Frédo Durand

Before We Begin

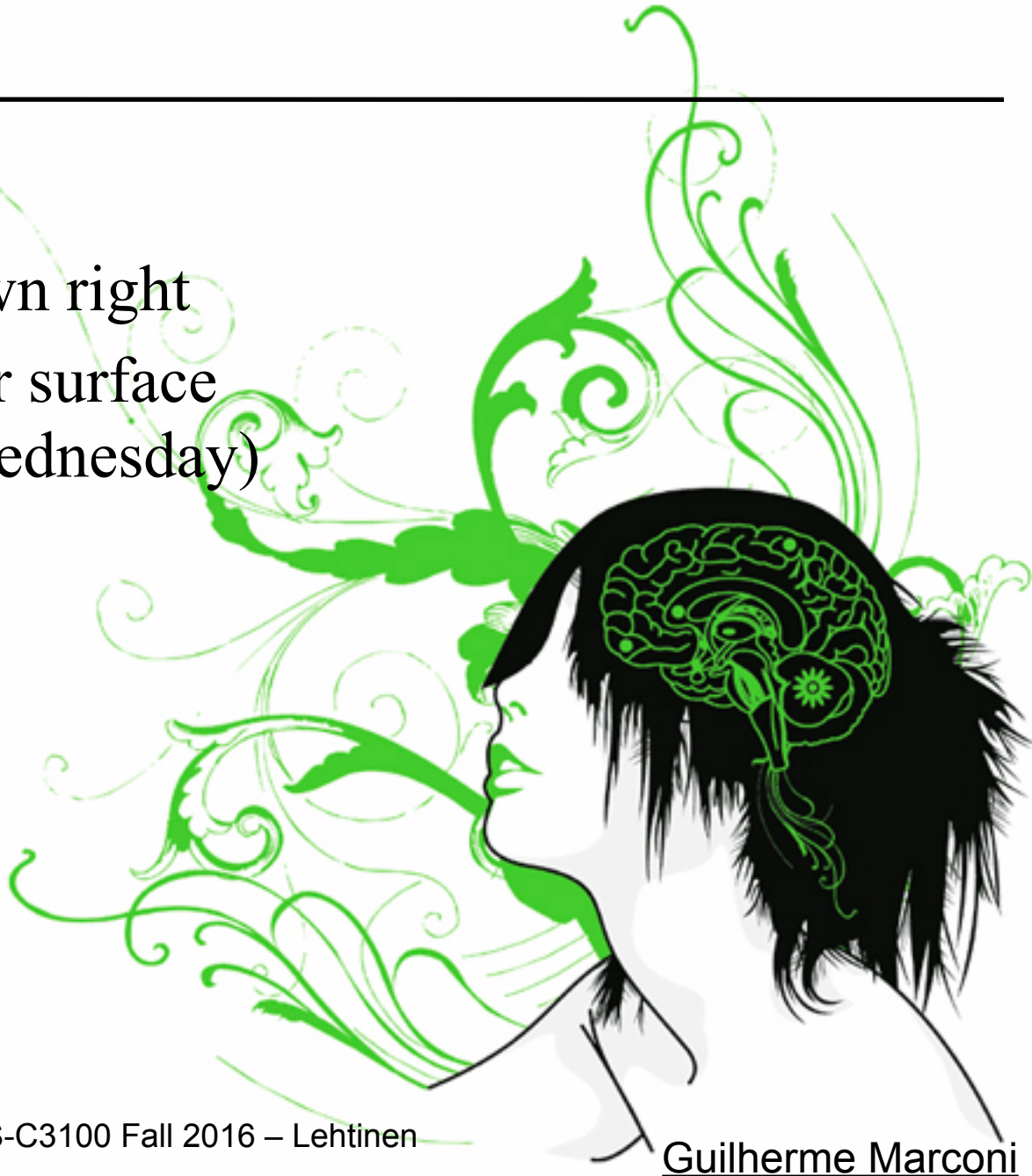
- Anything on your mind concerning Assignment 1?

- Linux makefiles released on MyCourses
  - Unsupported! For you to play with

- Assignment 2 (Curves & Surfaces, stuff covered today!) will be posted before Sunday's deadline

Today

- Curves in 2D
  - Useful in their own right
  - Provides basis for surface modeling (this Wednesday)



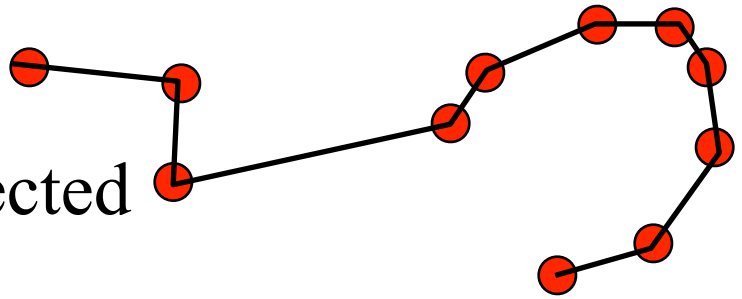
Guilherme Marconi

Modeling Curves in 2D

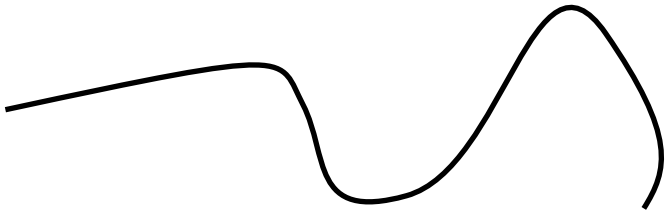
- How?

Modeling Curves in 2D

- Polylines
  - Sequence of vertices connected by straight line segments
  - Useful, but not for smooth curves
  - Very easy!
  - This is the representation that usually gets drawn in the end (smooth curves converted into these)

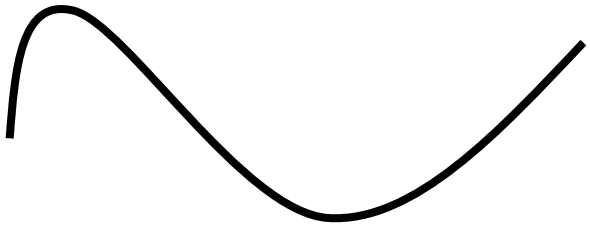


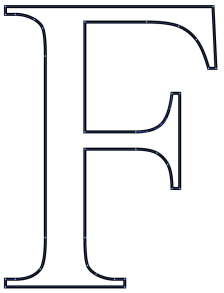
- **Smooth curves**
  - How do we specify them?
  - A little harder (but not too much)

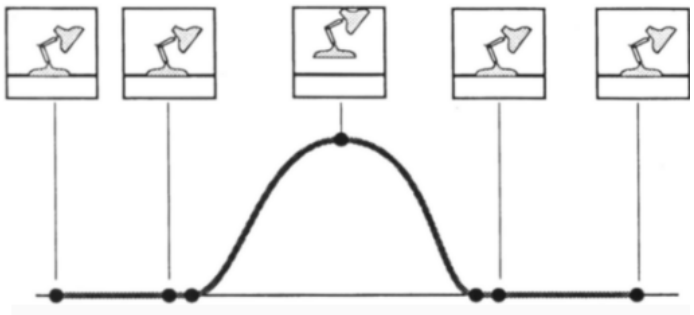


Splines

- A type of smooth curve in the plane or in 3D
- Many uses
  - 2D Illustration (e.g. Adobe Illustrator)
  - Fonts
  - 3D Modeling
  - Animation: Trajectories
- In general: Interpolation and approximation

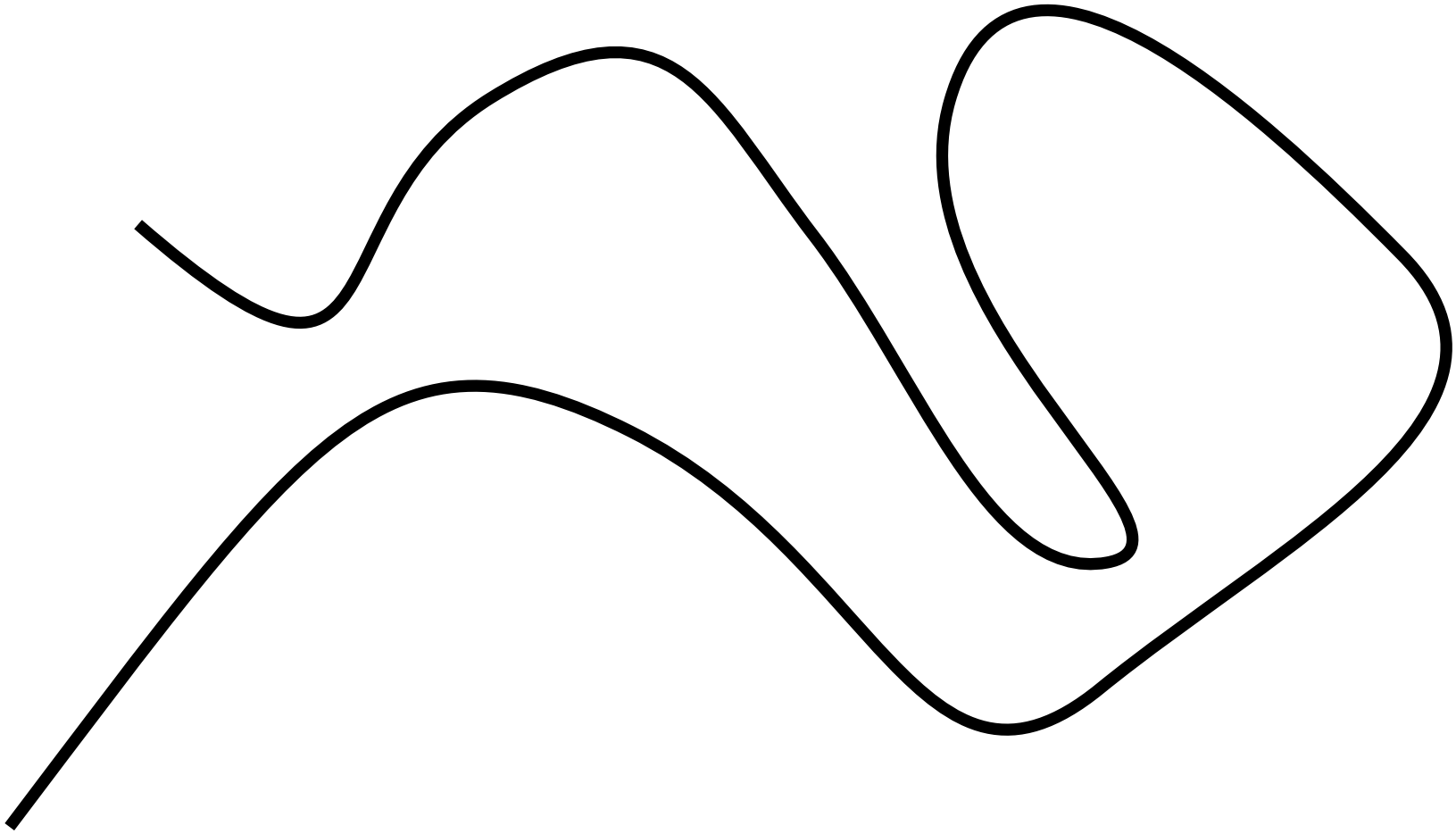




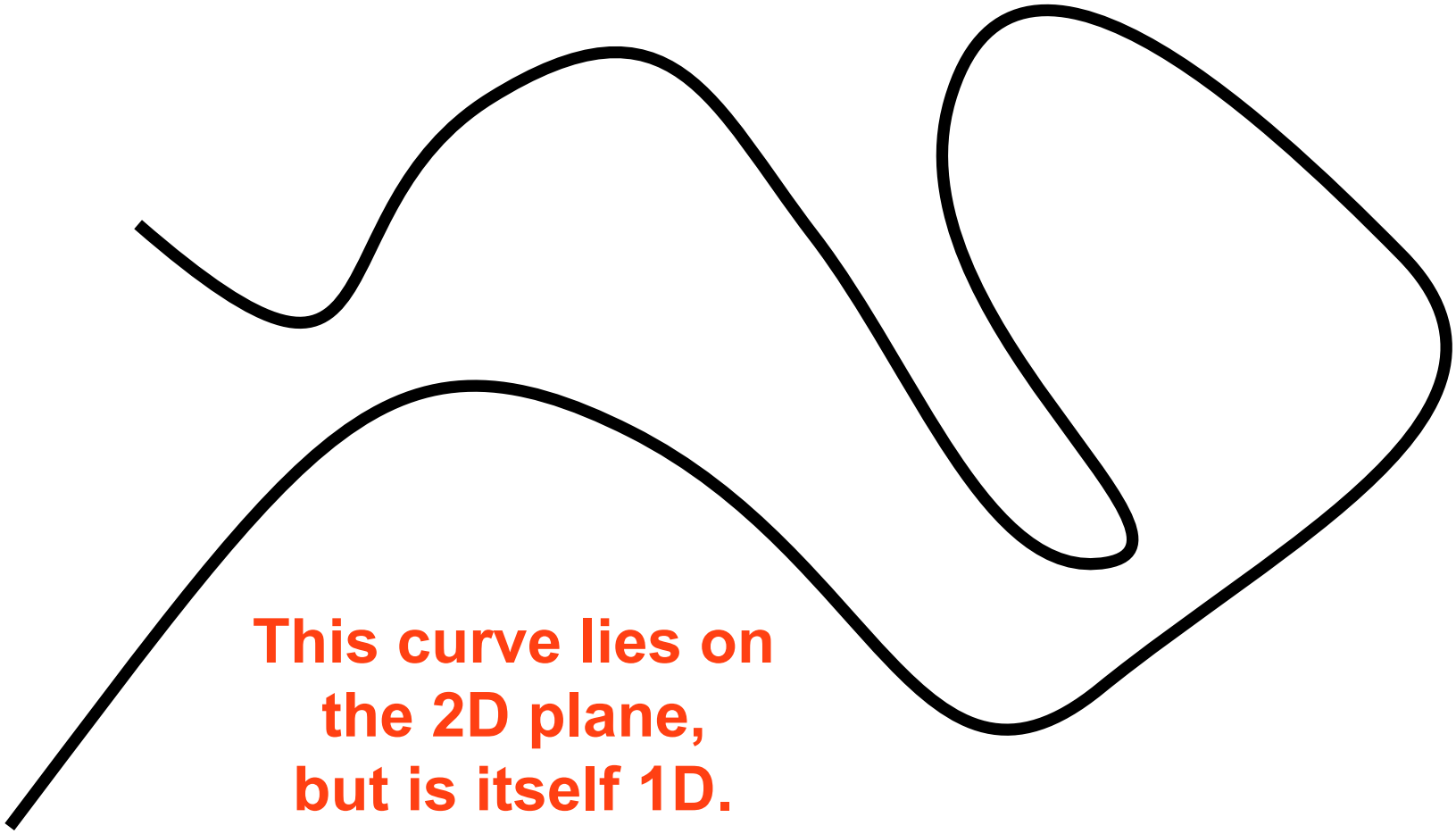


ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

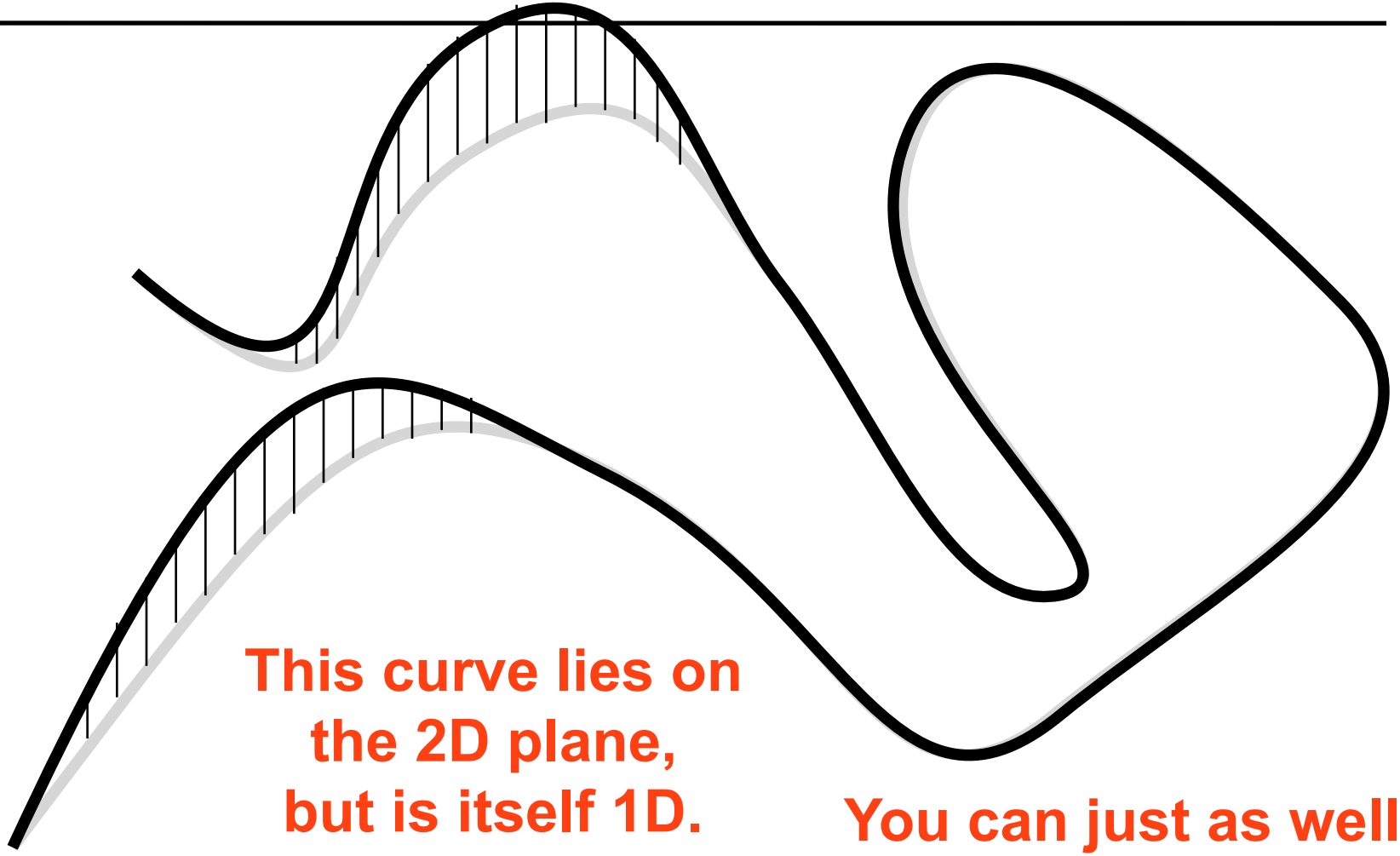
How Many Dimensions?



How Many Dimensions?



How Many Dimensions?



This curve lies on the 2D plane, but is itself 1D.

You can just as well define 1D curves in 3D space.

Two Definitions of a Curve

1. A continuous 1D point set on the plane or space
2. A mapping from an interval S onto the plane
   - That is, P(t) is the point of the curve at parameter t

$$P : \mathbb{R} \ni S \mapsto \mathbb{R}^2, \quad P(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

- Big differences
  - It's easy to generate points on the curve from the 2nd
  - The second definition can describe trajectories, the speed at which we move on the curve

Example of the First Definition?

- **A continuous 1D point set on the plane or space**
- A mapping from an interval S onto the plane
  - That is, P(t) is the point of the curve at parameter t

$$P : \mathbb{R} \ni S \mapsto \mathbb{R}^2, \quad P(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

Example of the First Definition?

- **A continuous 1D point set on the plane or space**
- A mapping from an interval S onto the plane
  - That is, P(t) is the point of the curve at parameter t

$$P : \mathbb{R} \ni S \mapsto \mathbb{R}^2, \quad P(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

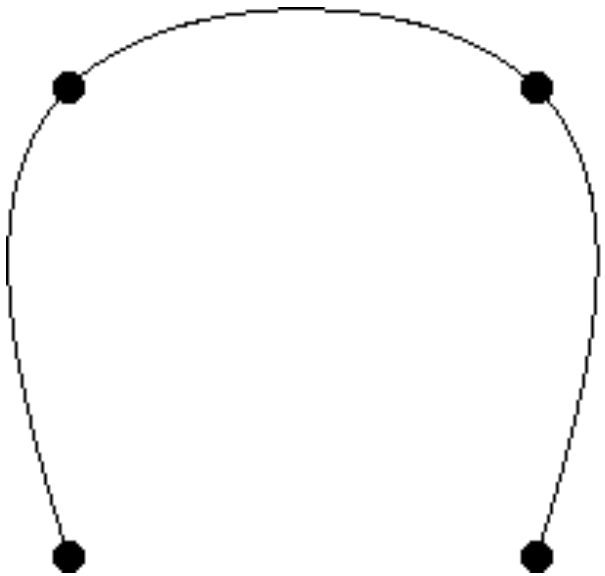
- An "algebraic curve", e.g. intersection of

$$x^2 + y^2 + z^2 = 1 \quad \text{and} \quad z = 0$$

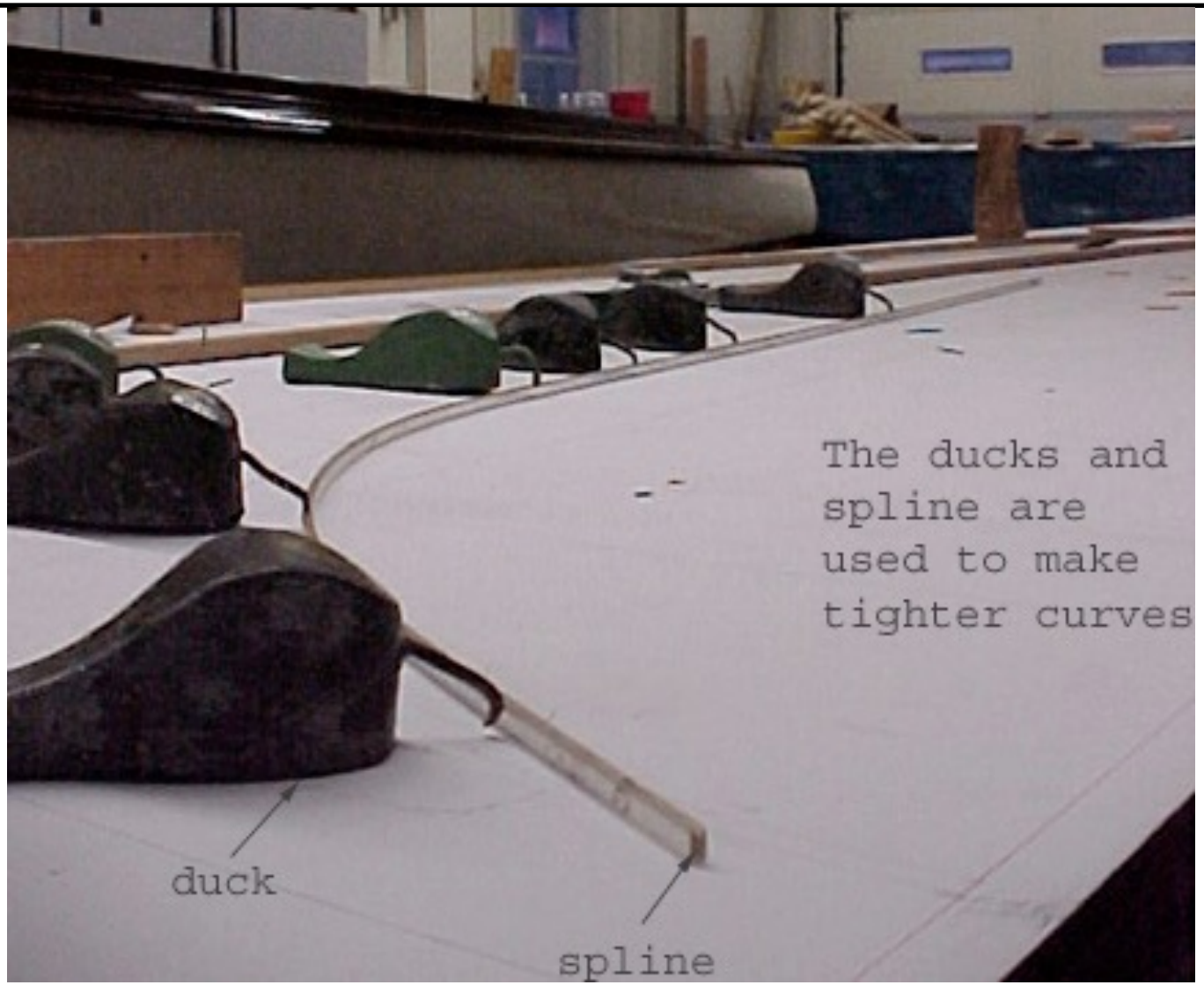
It's a circle on the xy plane :)

General Principle for Modeling

- User specifies "**control points**"
- We'll interpolate the control points by a smooth curve
  - The curve is completely determined by the control points.
  - I.e., we need an unambiguous rule that gives us the curve points given the control points.



Physical Splines

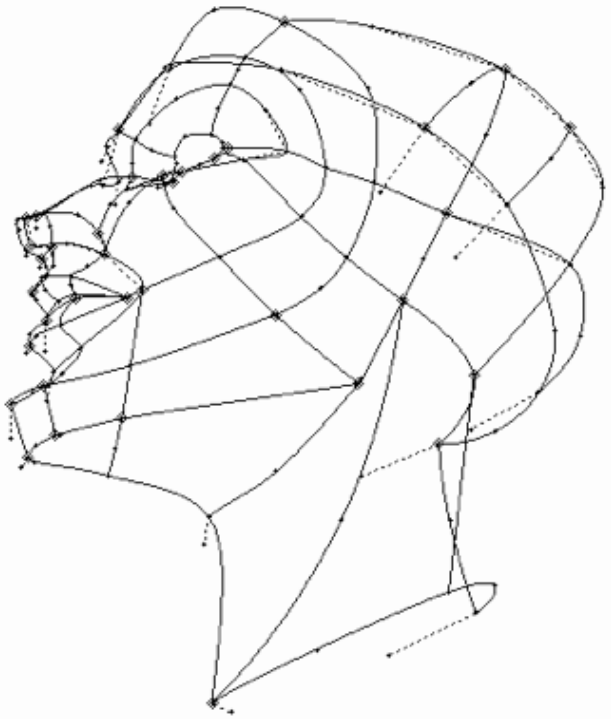


www.abm.org

See http://en.wikipedia.org/wiki/Flat_spline

Two Points of View

- Approximation/interpolation
  - We have “data points”, how can we interpolate?
  - Important in lots of applications, both graphics and non-graphics
- User interface/modeling
  - What is an easy way to specify a smooth curve?
  - Our main perspective today.



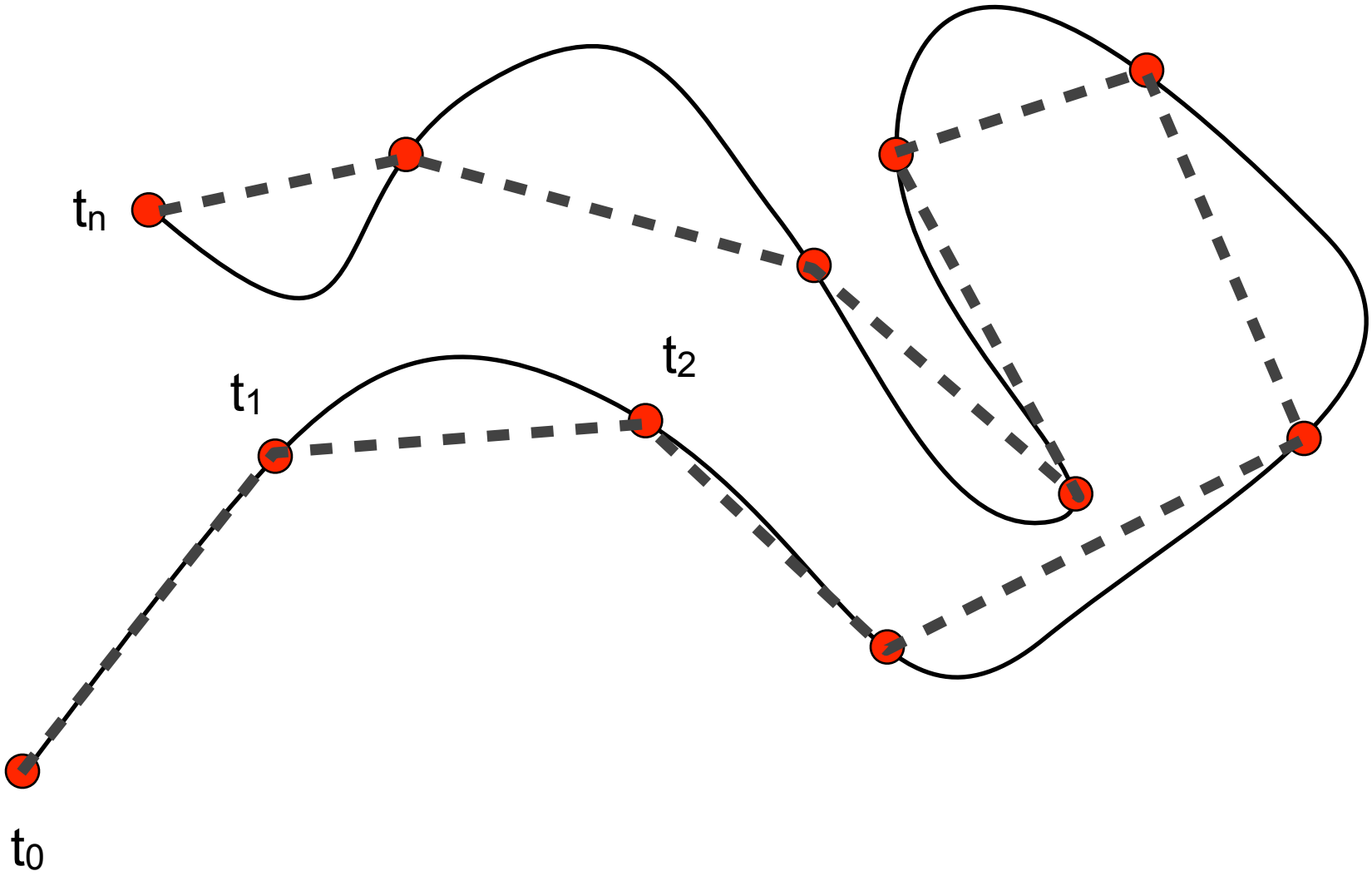
celshader.com

Splines

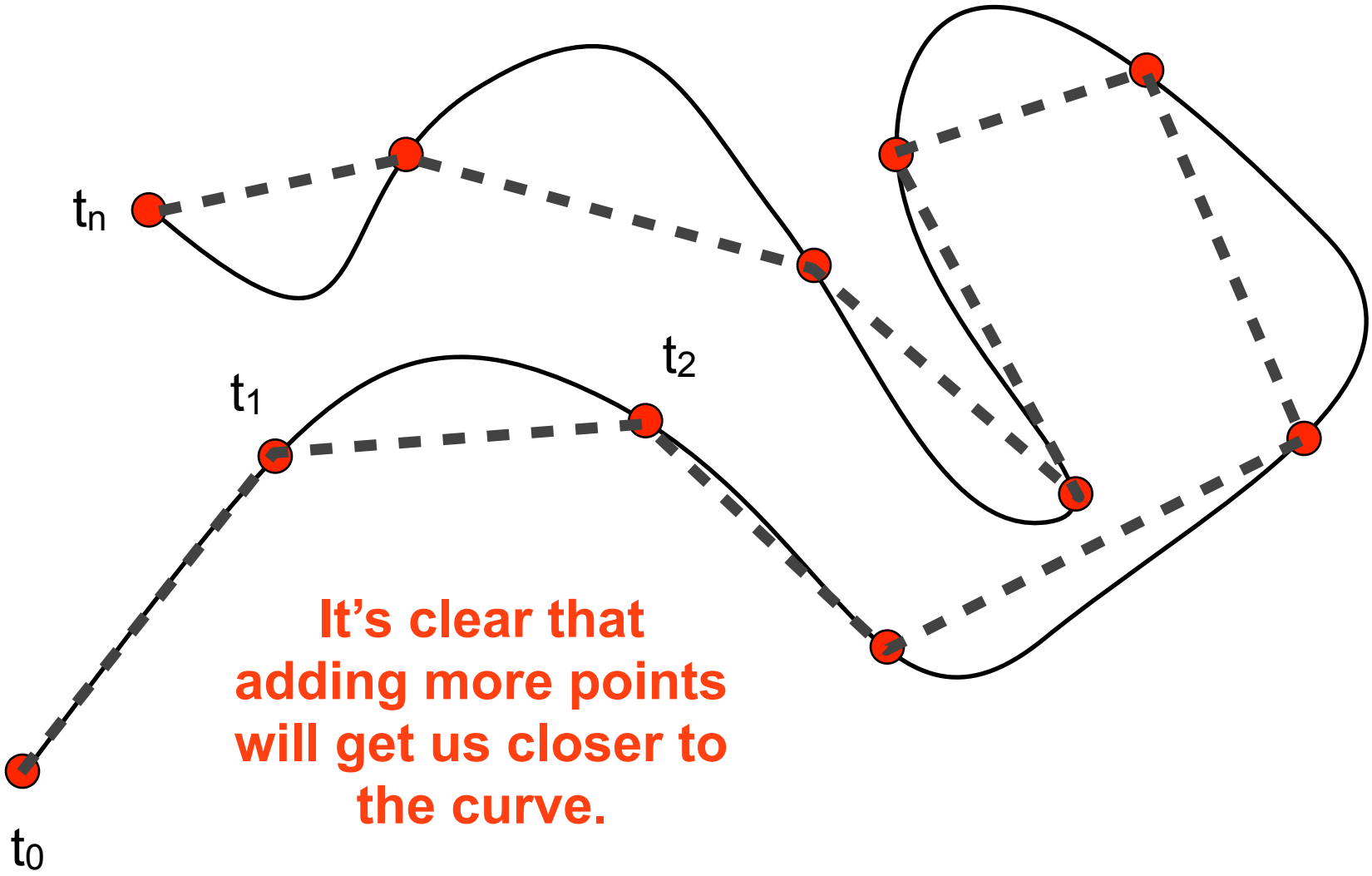
- Specified by a few control points
  - Good for UI
  - Good for storage

- Results in a smooth parametric curve P(t)
  - Just means that we specify $x(t)$ and $y(t)$
  - In practice **low-order polynomials chained together**
  - Convenient for animation, where t is time
  - Convenient for *tessellation* because **we can discretize t and approximate the curve with a polyline**

Tessellation



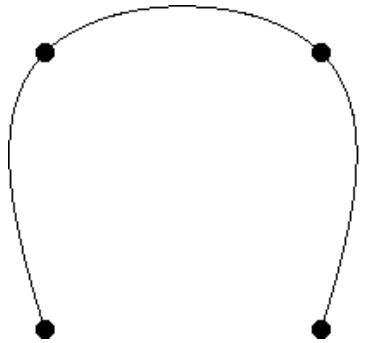
Tessellation



It's clear that adding more points will get us closer to the curve.

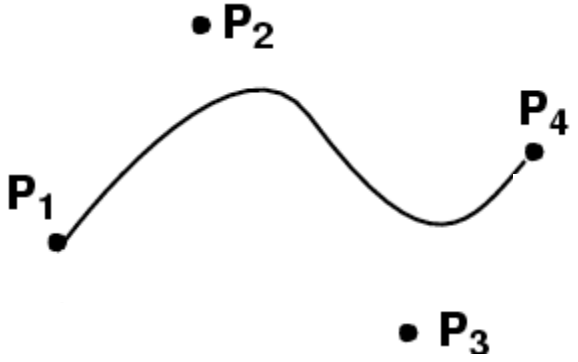
Interpolation vs. Approximation

- Interpolation
  - Goes through all specified points
  - Sounds more logical



Interpolation

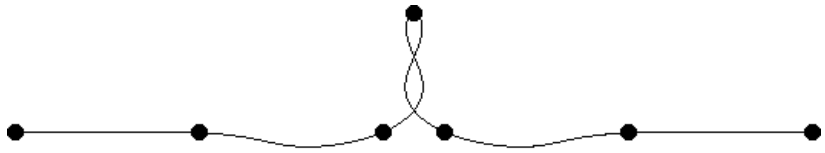
- Approximation
  - Does not go through all points



Approximation

Interpolation vs. Approximation

- Interpolation
  - Goes through all specified points
  - Sounds more logical
  - But can be more unstable, “ringing”



Interpolation

- Approximation
  - Does not go through all points
  - Turns out to be convenient

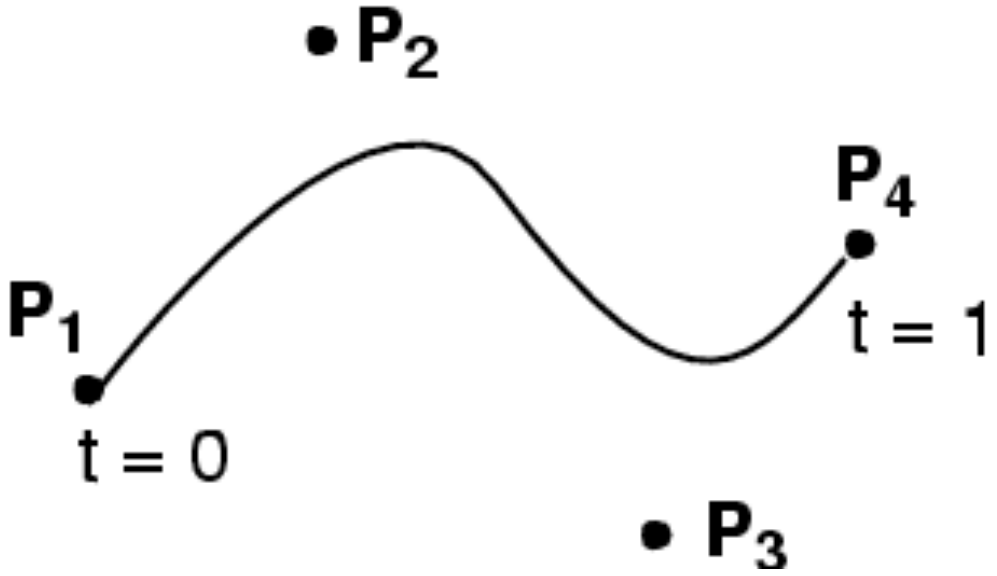
Approximation

- In practice, we’ll do something in between.

Questions?

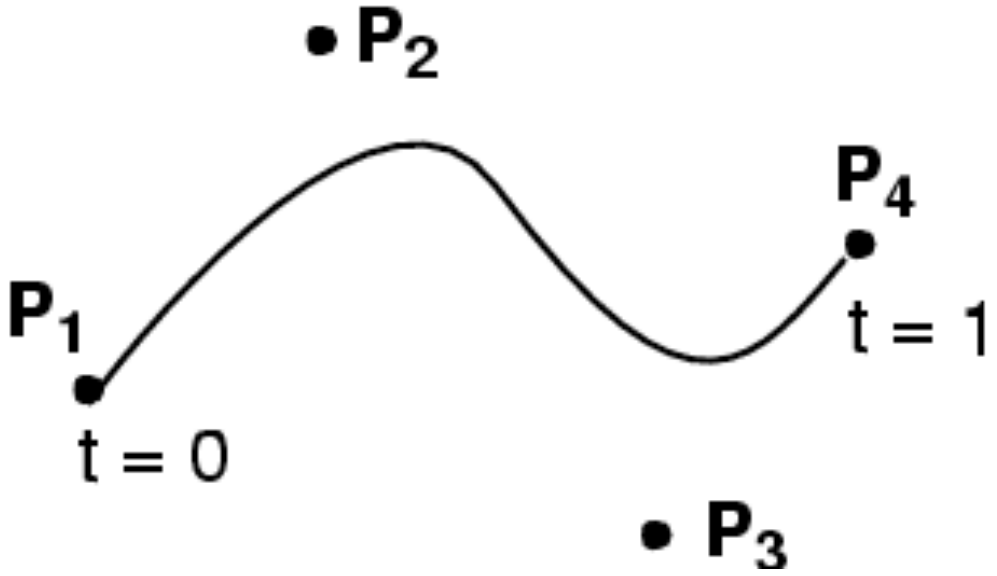
Cubic Bézier Curve

- User specifies 4 control points $\mathbf{P}_1$ ... $\mathbf{P}_4$
- Curve goes through (interpolates) the ends $\mathbf{P}_1$, $\mathbf{P}_4$
- Approximates the two other ones
- Cubic polynomial



Cubic Bézier Curve

- $P(t) = (1-t)^3 \mathbf{P}_1 + 3t(1-t)^2 \mathbf{P}_2 + 3t^2(1-t) \mathbf{P}_3 + t^3 \mathbf{P}_4$



That is,

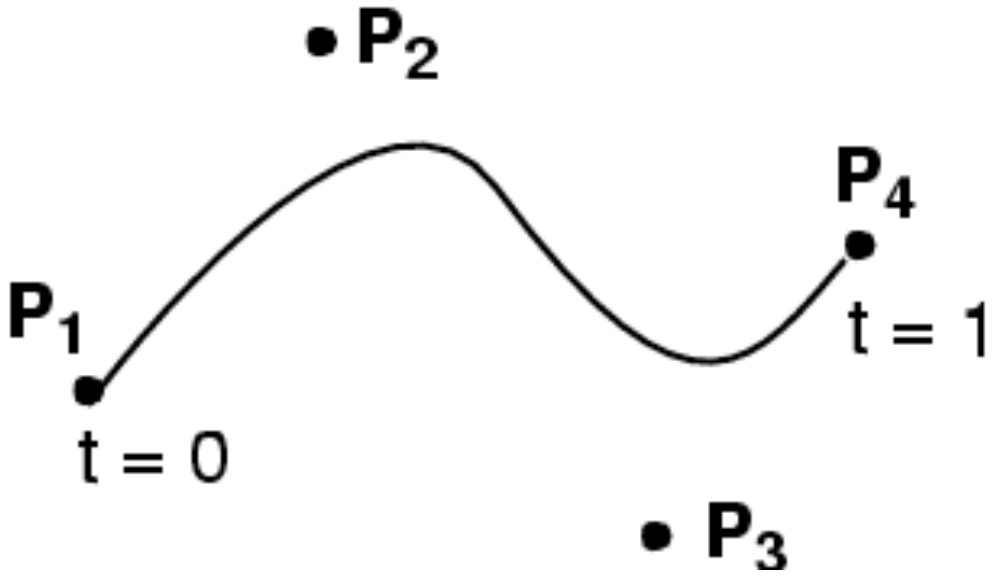
$$x(t) = (1-t)^3 x_1 + 3t(1-t)^2 x_2 + 3t^2(1-t) x_3 + t^3 x_4$$

$$y(t) = (1-t)^3 y_1 + 3t(1-t)^2 y_2 + 3t^2(1-t) y_3 + t^3 y_4$$

Cubic Bézier Curve

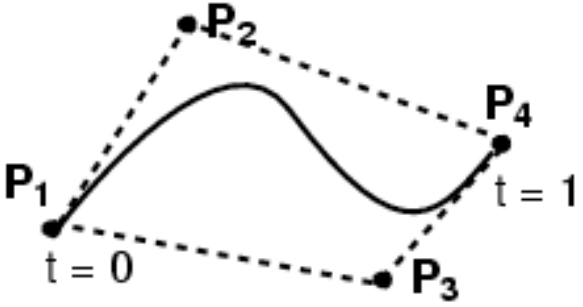
- $P(t) = (1-t)^3 \mathbf{P}_1 + 3t(1-t)^2 \mathbf{P}_2 + 3t^2(1-t) \mathbf{P}_3 + t^3 \mathbf{P}_4$

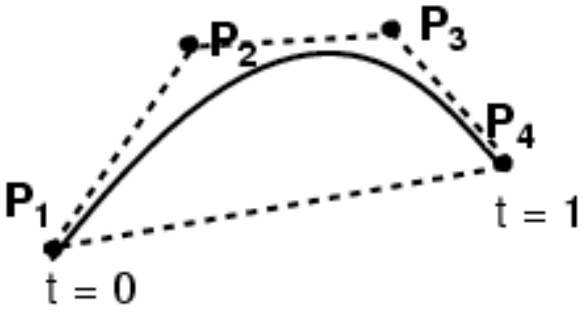
Verify what happens for t=0 and t=1

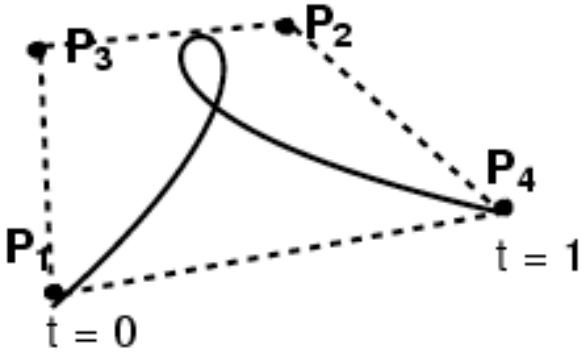


Cubic Bézier Curve

- 4 control points
- Curve passes through first & last control point
- Curve is tangent at $\mathbf{P}_1$ to $(\mathbf{P}_1\text{-}\mathbf{P}_2)$ and at $\mathbf{P}_4$ to $(\mathbf{P}_4\text{-}\mathbf{P}_3)$



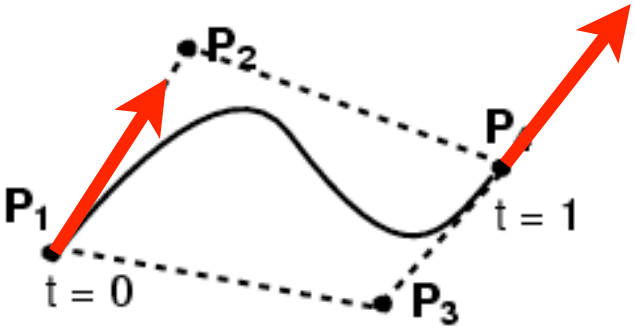


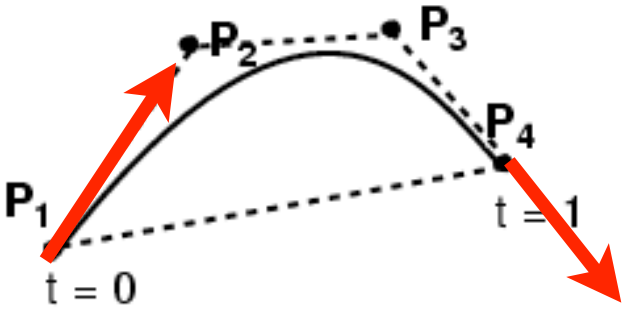


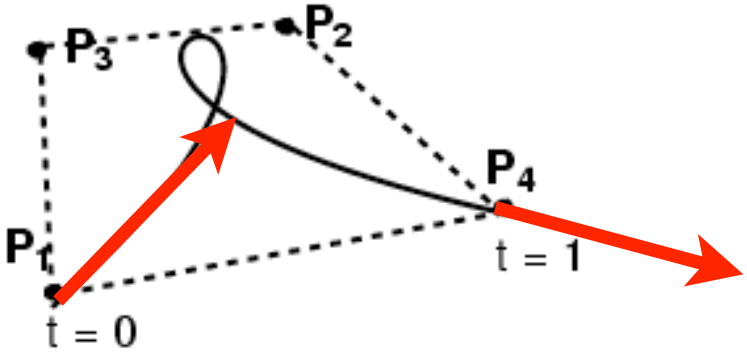
A Bézier curve is bounded by the **convex hull** of its control points.

Cubic Bézier Curve

- 4 control points
- Curve passes through first & last control point
- Curve is tangent at $\mathbf{P}_1$ to $(\mathbf{P}_1\text{-}\mathbf{P}_2)$ and at $\mathbf{P}_4$ to $(\mathbf{P}_4\text{-}\mathbf{P}_3)$







A Bézier curve is bounded by the **convex hull** of its control points.

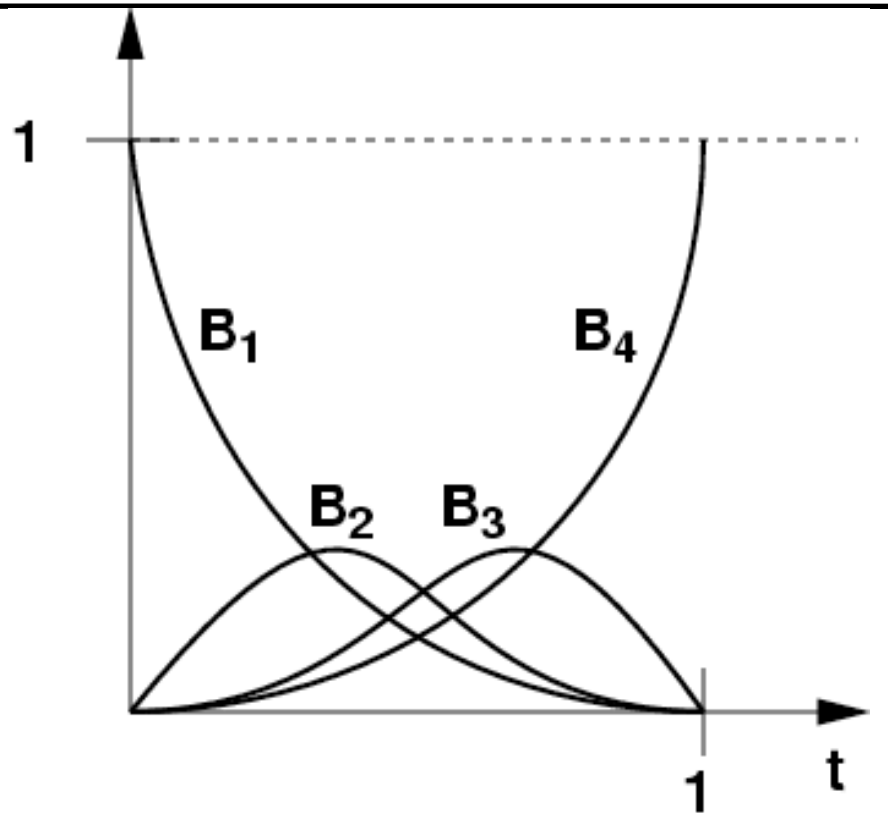
“Bernstein Polynomials”

For cubic:

- $B_1(t)=(1-t)^3$
- $B_2(t)=3t(1-t)^2$
- $B_3(t)=3t^2(1-t)$
- $B_4(t)=t^3$

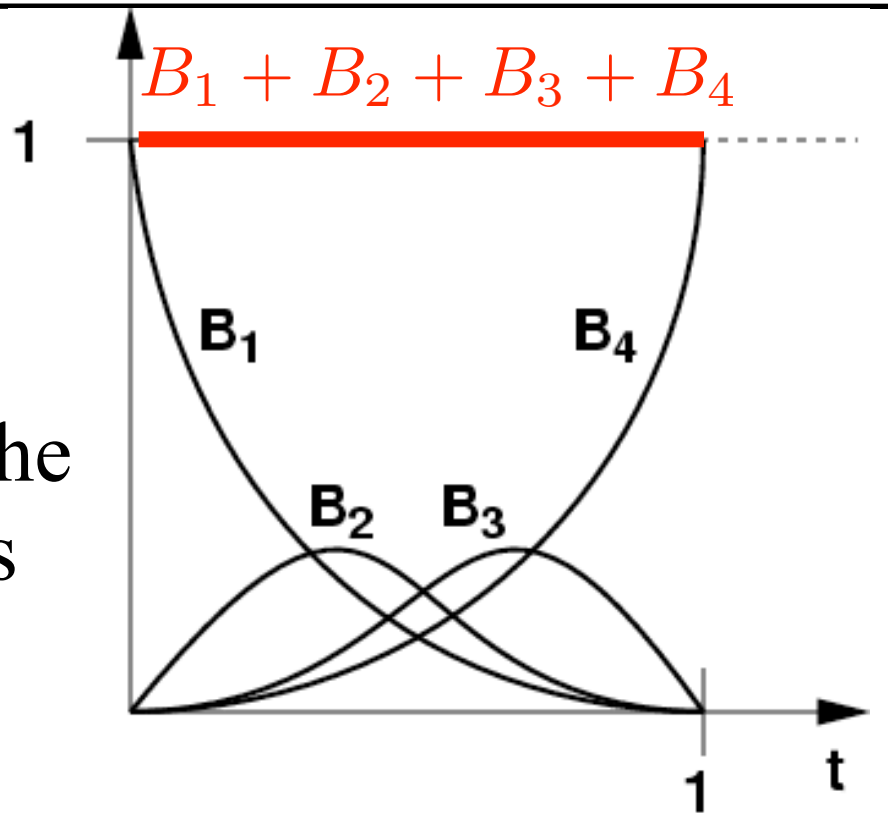
  – (careful with indices, many authors start at 0)

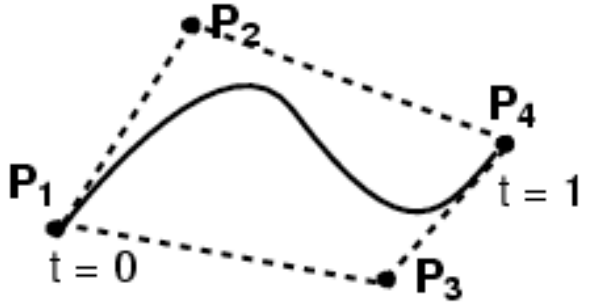
- But defined for any degree



Properties of Bernstein polynomials

- ≥ 0 for all $0 \leq t \leq 1$
- Sum to 1 for every t
  - called *partition of unity*
- (These two together are the reason why Bézier curves lie within convex hull)
- Only B_1 is non-zero at 0
  - Bezier interpolates P_1
  - Same for B_4 and P_4 for t=1



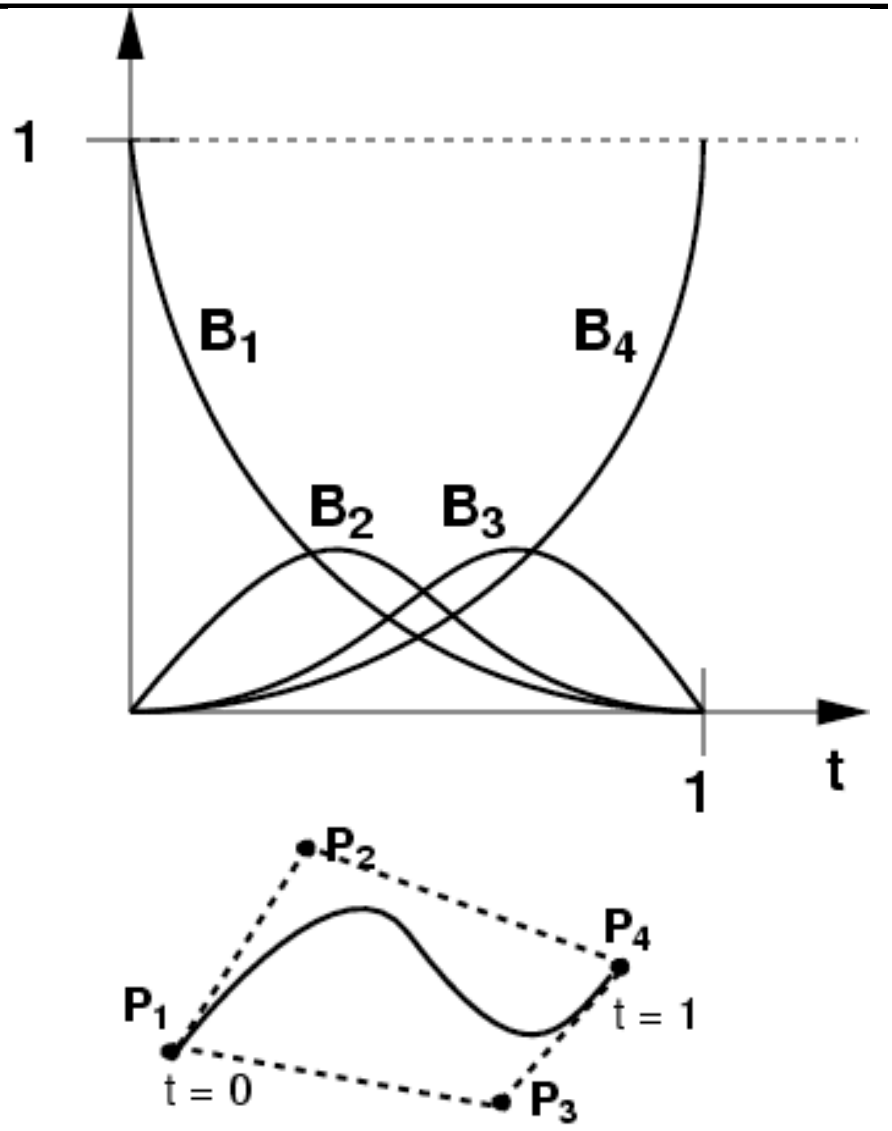


Bézier Curves

- $\mathbf{P}(t) = \mathbf{P}_1 B_1(t) + \mathbf{P}_2 B_2(t) + \mathbf{P}_3 B_3(t) + \mathbf{P}_4 B_4(t)$
  - $\mathbf{P}_i$ are 2D control points (x_i, y_i)
  - For each t, the point $\mathbf{P}(t)$ on a Bézier curve is a linear combination of the control points with weights given by the Bernstein polynomials at t

Interpretation as "Influence"

- Each B_i specifies the influence of P_i
- First, P_1 is the most influential point, then P_2, P_3, and P_4
- P_2 and P_3 never have full influence
  - Not interpolated!



Questions?

What's with the Formula?

- Explanation 1:
  - Magic!

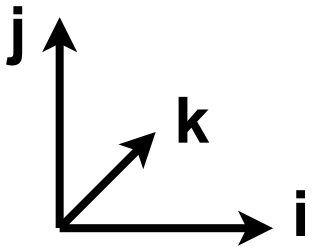
- Explanation 2:
  - It's a linear combination of ***basis polynomials***

  - Let's study this in 1D using curves $y=f(t)$

Usual Vector Spaces

- In 3D, each vector has three components x, y, z
- But geometrically, each vector is actually the sum

$$v = x\,\vec{\boldsymbol{i}} + y\,\vec{\boldsymbol{j}} + z\,\vec{\boldsymbol{k}}$$



- **i**, **j**, **k** are basis vectors

- Vector addition: just add components
- Scalar multiplication: just multiply components

Polynomials as a Vector Space

- Polynomials $y(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n$
- Can be added: just add the coefficients

$$(y + z)(t) = (a_0 + b_0) + (a_1 + b_1)t + (a_2 + b_2)t^2 + \ldots + (a_n + b_n)t^n$$

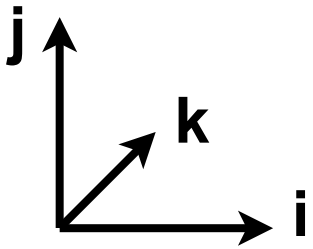
- Can be multiplied by a scalar: multiply the coefficients

$$s \cdot y(t) = (s \cdot a_0) + (s \cdot a_1)t + (s \cdot a_2)t^2 + \ldots + (s \cdot a_n)t^n$$

Polynomials as a Vector Space

- In 3D, each vector has three components x, y, z
- But geometrically, each vector is actually the sum

$$\vec{v} = x\,\vec{i} + y\,\vec{j} + z\,\vec{k}$$



- **i**, **j**, **k** are basis vectors

- In the polynomial vector space, $\{1, t, ..., t^n\}$ are the basis vectors, $a_0, a_1, ..., a_n$ are the components

$$y(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n$$

Polynomials as a Vector Space

- Polynomials are like vectors; you can add them and scale them, and the result is still a polynomial.

- Connections
  - The flavor of math that is concerned with such things is called *functional analysis*.
    - Many familiar concepts, such as orthogonality, carry over from "ordinary" *n*-D vectors to functions!
    - Trivializes many "difficult" things like Fourier series by giving geometric analogue.
    - Many other uses in graphics, e.g. finite element global illumination algorithms such as radiosity.

Questions?

Subset of Polynomials: Cubic

$$y(t) = a_0 + a_1\, t + a_2\, t^2 + a_3\, t^3$$

- Closed under addition & *scalar* multiplication
  - Means the result is still a cubic polynomial (verify!)
  - Not under general multiplication: $t^3 * t^3 = t^6$!
- This means it is also a vector space, a 4D **subspace** of the full space of polynomials
  - How many dimensions does the full space have?
- The *x* and *y* coordinates of cubic Bézier curves belong to this subspace as functions of *t*.

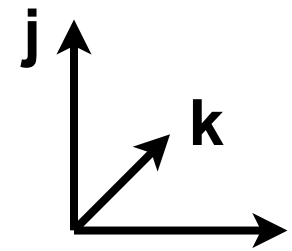
Basis for Cubic Polynomials

More precisely:

What's a basis?

$$\vec{v} = x\,\vec{i} + y\,\vec{j} + z\,\vec{k}$$

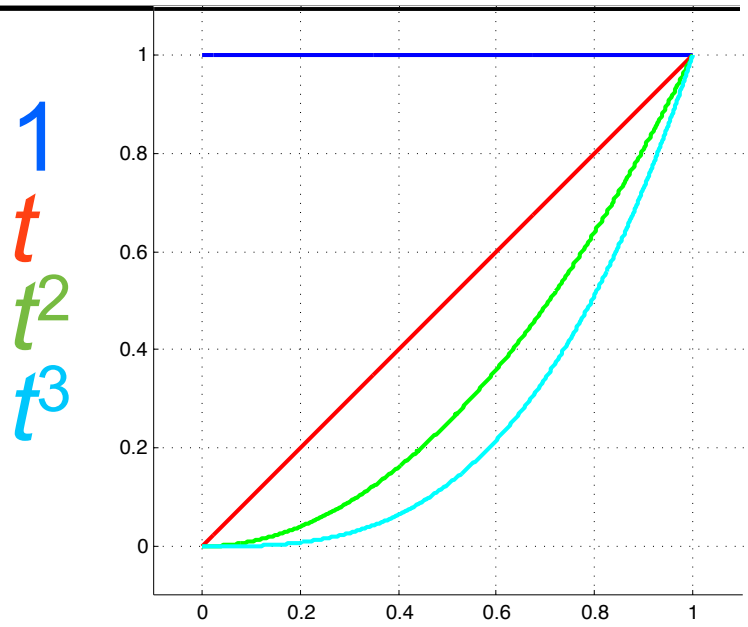
In 3D



- A set of "atomic" vectors
  - Called **basis vectors**
  - Linear combinations of basis vectors span the space
- Linearly independent
  - Means that no basis vector can be obtained from the others by linear combination
    - Example: **i**, **j**, **i**+**j** is not a basis (missing **k** direction!)

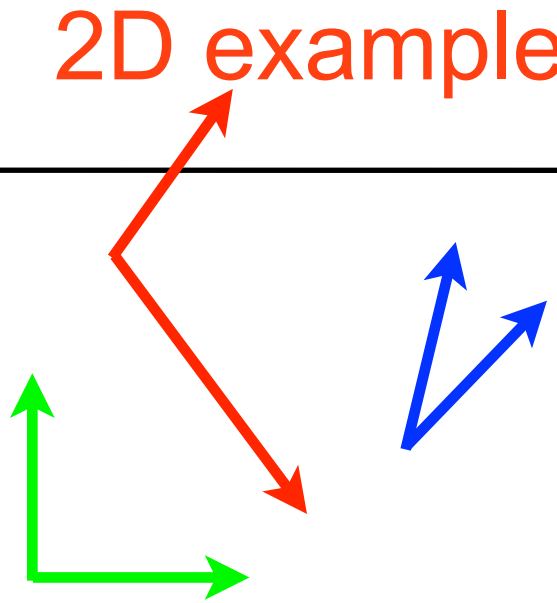
Canonical Basis for Cubics

$$\{1, t, t^2, t^3\}$$



- *Any* cubic polynomial is a linear combination of these:
  - $a_0 + a_1\, t + a_2\, t^2 + a_3\, t^3 = a_0 * 1 + a_1 * t + a_2 * t^2 + a_3 * t^3$
- They are linearly independent
  - Means you can't write any of the four monomials as a a linear combination of the others. (You can try.)

Different basis



- For example:
  - $\{1,\ 1+t,\ 1+t+t^2,\ 1+t-t^2+t^3\}$
  - $\{t^3,\ t^3+t^2,\ t^3+t,\ t^3+1\}$

- These can all be obtained from $1, t, t^2, t^3$ by linear combination
- Infinite number of possibilities, just like you have an infinite number of bases to span R^2

Why we bother:

Matrix-Vector Notation For Polynomials

- For example:
  - $1,\ 1+t,\ 1+t+t^2,\ 1+t-t^2+t^3$
  - $t^3,\ t^3+t^2,\ t^3+t,\ t^3+1$

Change-of-basis matrix

"Canonical" monomial basis

These relationships hold for each value of *t*

$$\begin{pmatrix} 1 \\ 1+t \\ 1+t+t^2 \\ 1+t-t^2+t^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

$$\begin{pmatrix} t^3 \\ t^3+t^2 \\ t^3+t \\ t^3+1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

Matrix-Vector Notation For Polynomials

- For example:
  - $1,\ 1+t,\ 1+t+t^2,\ 1+t-t^2+t^3$
  - $t^3,\ t^3+t^2,\ t^3+t,\ t^3+1$

Change-of-basis matrix

"Canonical" monomial basis

$$\begin{pmatrix} 1 \\ 1+t \\ 1+t+t^2 \\ 1+t-t^2+t^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

$$\begin{pmatrix} t^3 \\ t^3+t^2 \\ t^3+t \\ t^3+1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

Not any matrix will do! If it's singular, the basis set will be linearly dependent, i.e., redundant.

Matrix Form of Bernstein

Cubic Bernstein:

- $B_1(t)=(1-t)^3$
- $B_2(t)=3t(1-t)^2$
- $B_3(t)=3t^2(1-t)$
- $B_4(t)=t^3$

Expand these out and collect powers of *t*. The coefficients are the entries in the matrix B!

$$\begin{pmatrix} B_1(t) \\ B_2(t) \\ B_3(t) \\ B_4(t) \end{pmatrix} = \overbrace{\begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}^{\boldsymbol{B}} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

More Matrix-Vector Notation

- Remember:
  $\mathbf{P}(t) = \mathbf{P}_1 \mathrm{B}_1(t) + \mathbf{P}_2 \mathrm{B}_2(t) + \mathbf{P}_3 \mathrm{B}_3(t) + \mathbf{P}_4 \mathrm{B}_4(t)$
  is a linear combination of control points

- or, in matrix-vector notation

Bernstein polynomials (4x1 vector)

$$P(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} \begin{pmatrix} B_1(t) \\ B_2(t) \\ B_3(t) \\ B_4(t) \end{pmatrix}$$

point on curve (2x1 vector)

matrix of control points (2 x 4)

More Matrix-Vector Notation

$$\begin{pmatrix} B_1(t) \\ B_2(t) \\ B_3(t) \\ B_4(t) \end{pmatrix} = \overbrace{\begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}^{\boldsymbol{B}} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

<= Flasback from two slides ago, let's combine with below:

Bernstein polynomials (4x1 vector)

$$P(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} \begin{pmatrix} B_1(t) \\ B_2(t) \\ B_3(t) \\ B_4(t) \end{pmatrix}$$

point on curve (2x1 vector)

matrix of control points (2 x 4)

Phase 3: Profit

- Combined, we get cubic Bézier in matrix notation

point on curve (2x1 vector)

$$\mathrm{P(t)} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} =$$

Canonical monomial basis

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

"Geometry matrix" of control points $P_1..P_4$ (2 x 4)

"Spline matrix" (Bernstein)

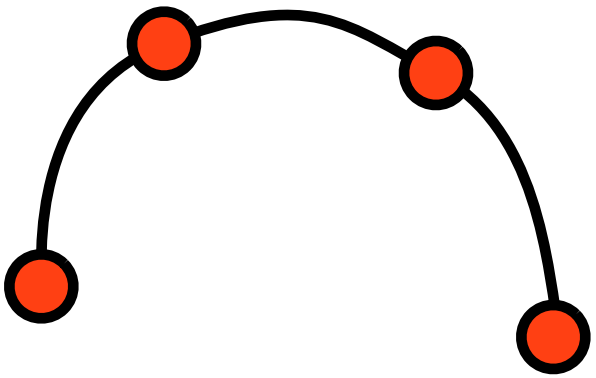
General Spline Formulation

$$Q(t) = \mathbf{GBT(t)} = \text{Geometry } \mathbf{G} \cdot \text{Spline Basis } \mathbf{B} \cdot \text{Power Basis } \mathbf{T(t)}$$

- Geometry: control points coordinates assembled into a matrix (P_1, P_2, …, P_{n+1})
- Spline matrix: defines the type of spline
  - Bernstein for Bézier
- Power basis: the monomials (1, t, ..., t^n)
- Advantage of general formulation
  - Compact expression
  - Easy to convert between types of splines
  - Dimensionality (plane or space) doesn't really matter

What can we do with this?

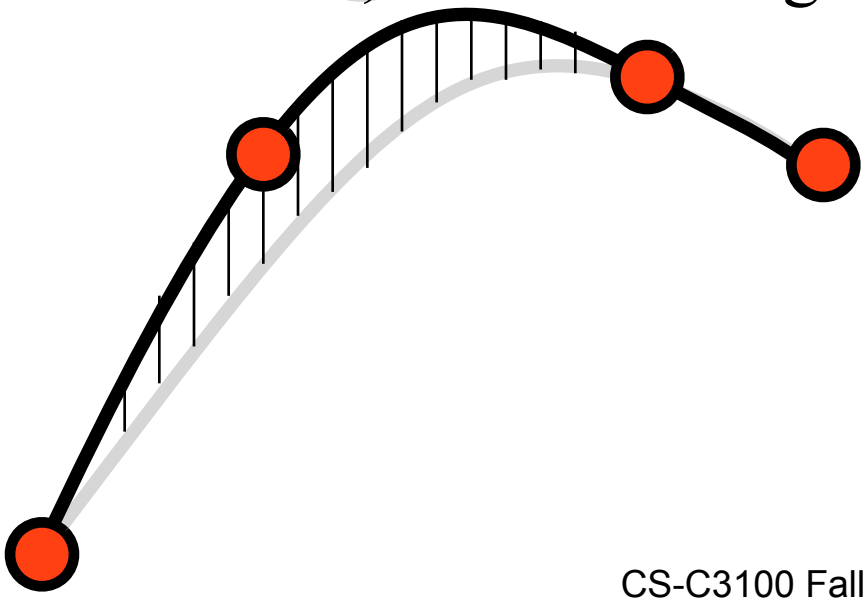
- Cubic polynomials form a vector space.
- Bernstein basis is canonical for Bézier.
- Can do other bases as well: Catmull-Rom splines interpolate all the control points, for instance



$$\mathbf{B}_{\mathrm{CR}} = \frac{1}{2}\begin{pmatrix} 0 & -1 & 2 & -1 \\ 2 & 0 & -5 & 3 \\ 0 & 1 & 4 & -3 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

3D Spline has 3D control points

- The 2D control points can be replaced by 3D points – this yields space curves.
  - All the above math stays the same except with the addition of 3rd row to control point matrix (z coords)
  - In fact, can do homogeneous coordinates as well!



Easy Brain Teaser for Next Time

- Can you derive formulas for conversion between two different cubic spline matrices, say Bézier and Catmull-Rom?
  - Both are cubics, so you can represent the other curve exactly with new control points $\mathbf{G}_2$

$$\mathbf{G}_1\mathbf{B}_1\mathbf{T}(t) \equiv \mathbf{G}_2\mathbf{B}_2\mathbf{T}(t)$$

$$\mathbf{G}_2 = ???$$

Change of Basis, Other Direction

- Given B1...B4, how to get back to canonical 1, t, t^2, t^3 ?

$$\begin{pmatrix} B_1(t) \\ B_2(t) \\ B_3(t) \\ B_4(t) \end{pmatrix} = \overbrace{\begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}^{\boldsymbol{B}} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

Change of Basis, Other Direction

That's right, with the inverse matrix!

- Given B1...B4, how to get back to canonical 1, t, t^2, t^3 ?

$$\begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix} = \overbrace{\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}}^{\boldsymbol{B}^{-1}} \begin{pmatrix} B_1(t) \\ B_2(t) \\ B_3(t) \\ B_4(t) \end{pmatrix}$$

Recap

- Cubic polynomials form a vector space.
- Bernstein basis is canonical for Bézier.
  - Can be seen as influence function of data points
  - Or data points are coordinates of the curve in the Bernstein basis
- We can change between basis with matrices.

Questions?

A Cubic Only Gets You So Far

- What if you want more control?

Higher-Order Bézier Curves

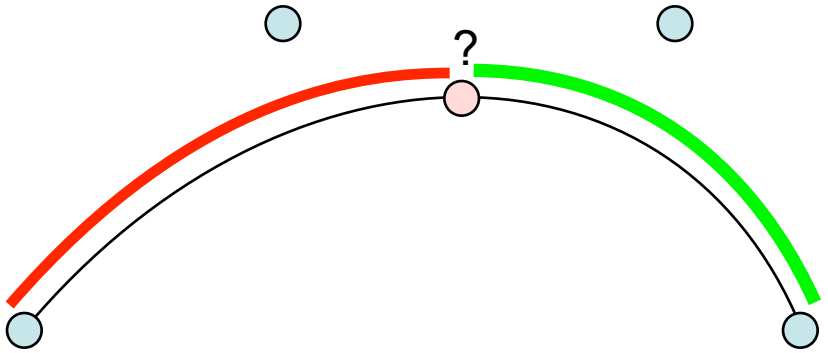
- > 4 control points
- Bernstein Polynomials as basis functions
  - For polynomial order n, i^{th} basis function is

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

- Every control point affects entire curve
  - Not simple local effect
  - More difficult to control for modeling
- You will not need this in this class

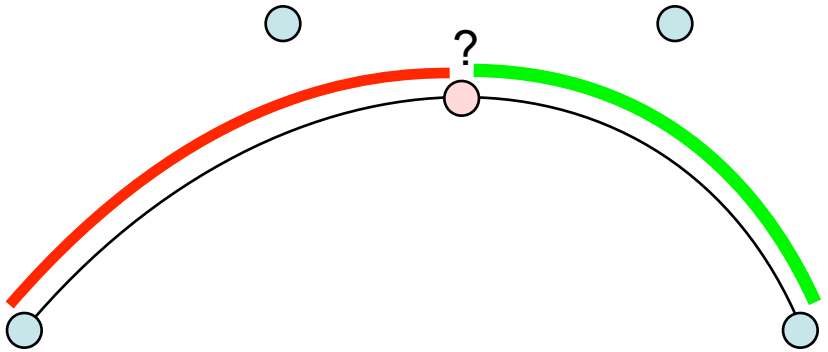
Subdivision of a Bézier curve

- Can we split a Bezier curve into two in the middle, using two new Bézier curves?
  - Would be useful for adding detail, as a single cubic doesn't get you very far, and higher-order curves are nasty.



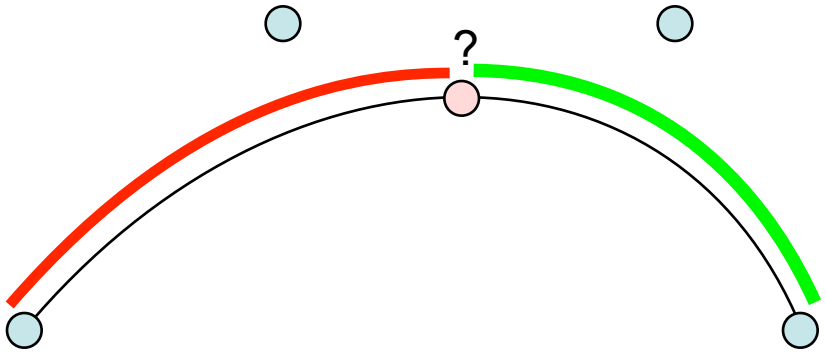
Subdivision of a Bezier curve

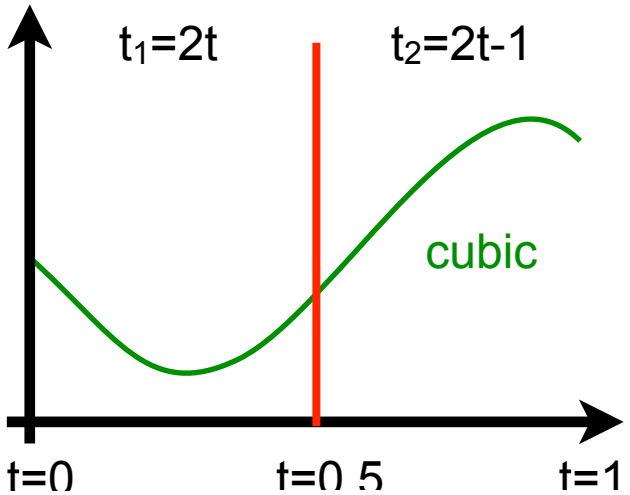
- Can we split a Bezier curve into two in the middle, using two new Bézier curves?
  - The resulting curves are again a cubic (Why?)



Subdivision of a Bezier curve

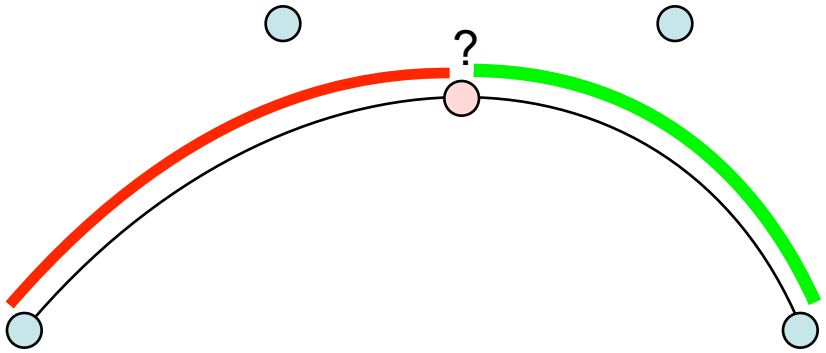
- Can we split a Bezier curve into two in the middle, using two new Bézier curves?
  - The resulting curves are again a cubic (Why? A cubic in t is also a cubic in $2t$)
  - Hence it must be representable using the Bernstein basis. So **yes, we can!**

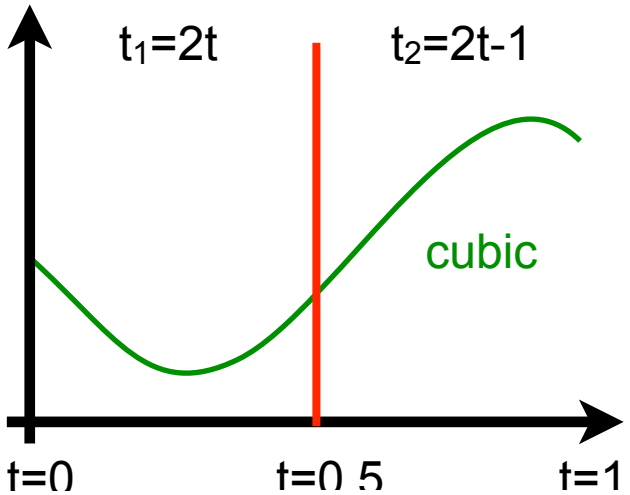




Subdivision of a Bezier curve

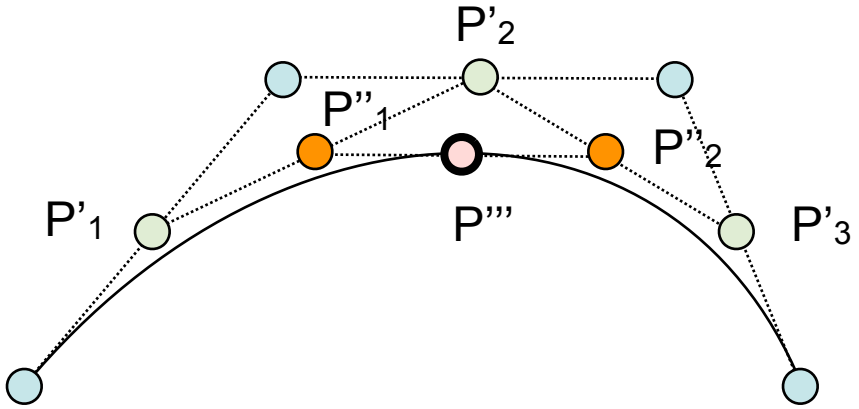
- Can we split a Bezier curve into two in the middle, using two new Bézier curves?
  - The resulting curves are again a cubic (Why? A cubic in t is also a cubic in $2t$)
  - Hence it must be representable using the Bernstein basis. So **yes, we can!**





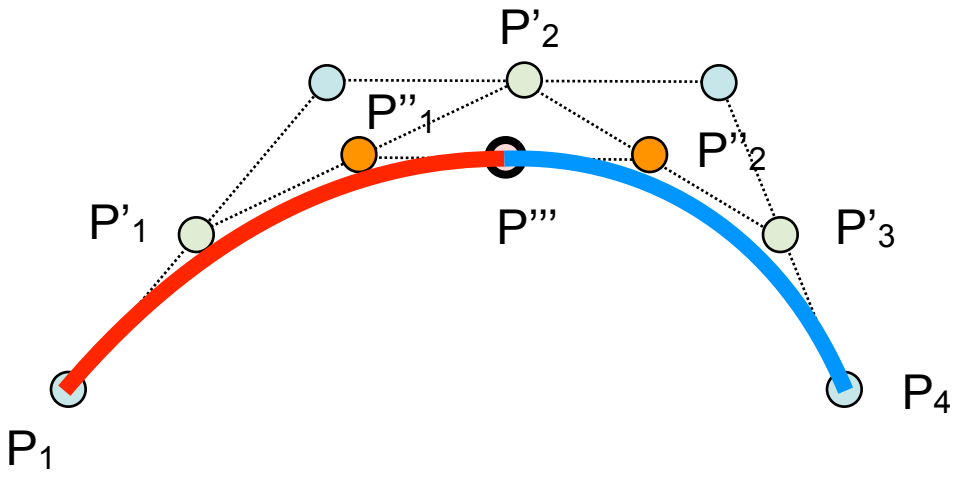
“De Casteljau Construction”

- Take the middle point of each of the 3 segments
- Construct the two segments joining them
- Take the middle of those two new segments
- Join them
- Take the middle point P’’’



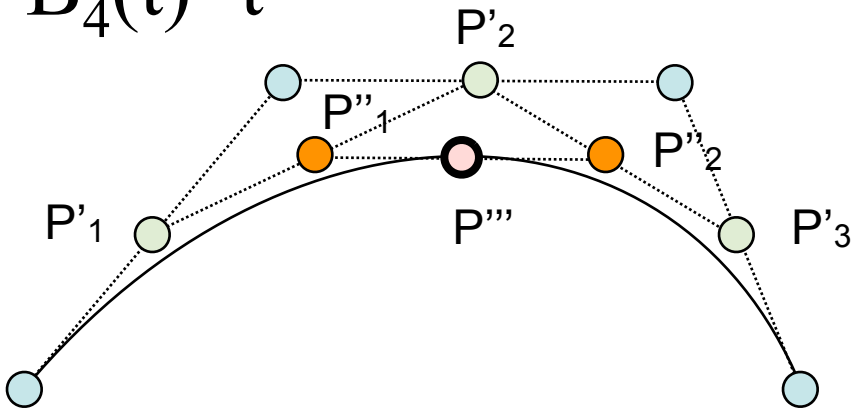
Result of Split in Middle

- The two new curves are defined by
  - P_1, P'_1, P''_1, and P'''
  - P''', P''_2, P'_3, and P_4
- Together they exactly replicate the original curve!
  - Originally 4 control points, now 7 (more control)



Sanity Check

- Do we get the middle point?
- $B_1(t)=(1-t)^3$
- $B_2(t)=3t(1-t)^2$
- $B_3(t)=3t^2(1-t)$
- $B_4(t)=t^3$



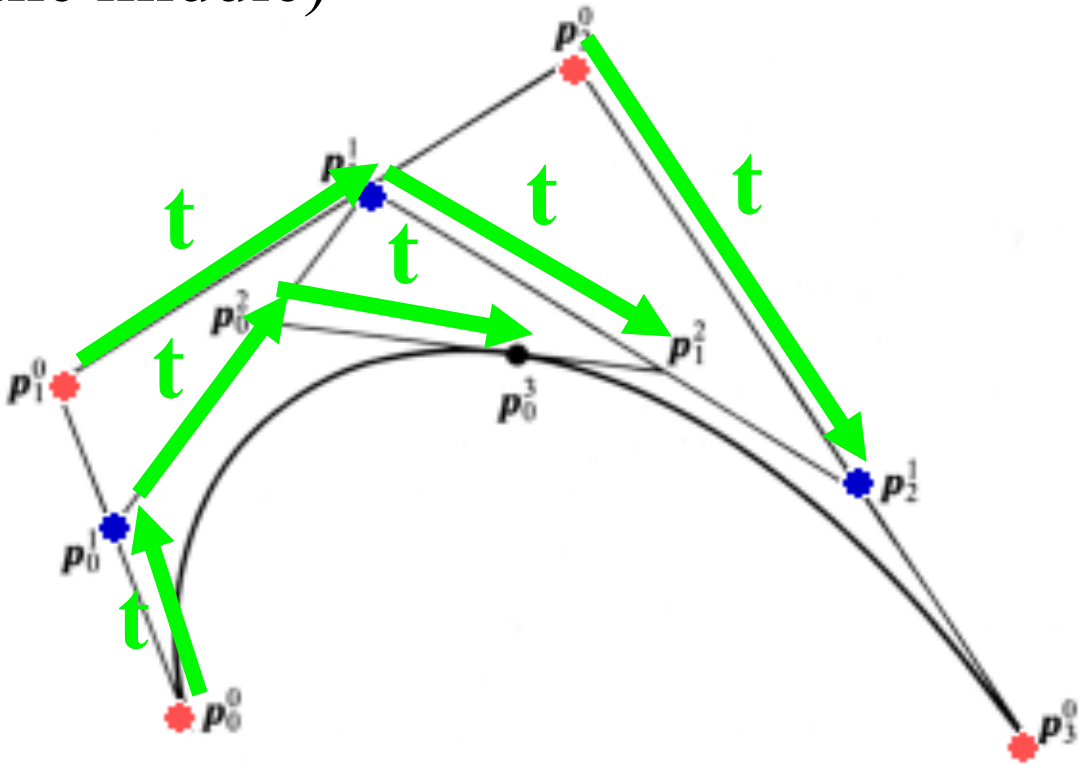
$$P_1' = 0.5(P_1 + P_2)$$
$$P_2' = 0.5(P_2 + P_3)$$
$$P_3' = 0.5(P_3 + P_4)$$

$$P_1'' = 0.5(P_1' + P_2')$$
$$P_2'' = 0.5(P_2' + P_3')$$

$$\begin{aligned} P''' &= 0.5(P_1'' + P_2'') \\ &= 0.5\,(0.5(P_1' + P_2') + 0.5(P_2' + P_3')) \\ &= 0.5\,(0.5\,[0.5(P_1 + P_2) + 0.5(P_2 + P_3)] + \\ &\quad 0.5\,[0.5(P_2 + P_3) + 0.5(P_3 + P_4)] \\ &= 1/8P_1 + 3/8P_2 + 3/8P_3 + 1/8P_4 \end{aligned}$$

De Casteljau Construction

- Actually works to construct a point at any *t*, not just 0.5
- Just subdivide the segments with ratio (1-*t*), *t* (not in the middle)



Questions?

Aside: Extra Credit Suggestion

- TrueType font glyphs (the images of the characters) consist of *quadratic* Bézier curves chained together.
- Why don't you write code that loads the glyphs using the Win32 API, and either
  - Convert them to cubic Bézier and display them using the drawing code you already wrote
  - Or write code for drawing quadratic curves directly.
- See this MSDN documentation page for a starting point!

Linear Transformations & Cubics

- What if we want to transform each point on the curve with a linear transformation **M**?

$$P'(t) = \mathbf{M}\left(\begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix} \right)$$

Linear Transformations & Cubics

- What if we want to transform each point on the curve with a linear transformation **M**?
  - Because everything is linear, it's the same as transforming the only the control points

$$P'(t) = \mathbf{M}\left(\begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix} \right)$$

$$= \left(\mathbf{M} \begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \end{pmatrix} \right) \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

Linear Transformations & Cubics

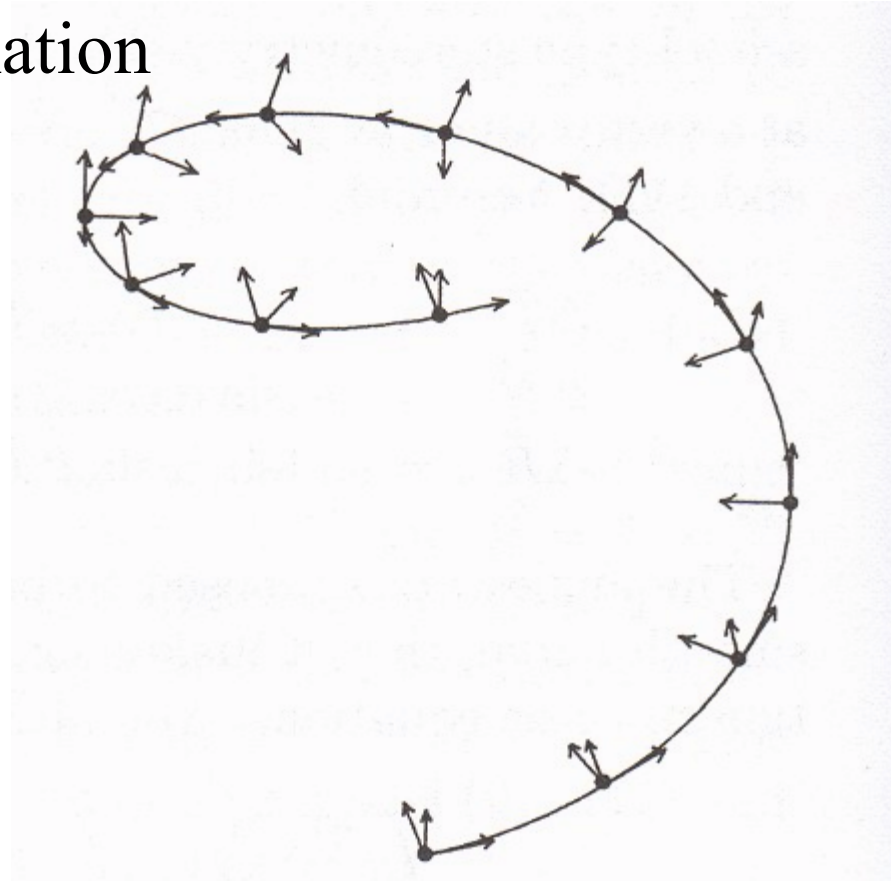
- Homogeneous coordinates also work
  - Means you can translate, rotate, shear, etc.
  - Also, changing *w* gives a "tension" parameter
    - Note though that you need to normalize P' by $1/w$

$$P'(t) = \mathbf{M}\left[\begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix} \right]$$

$$= \left[\mathbf{M} \begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \right] \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

Differential properties of curves

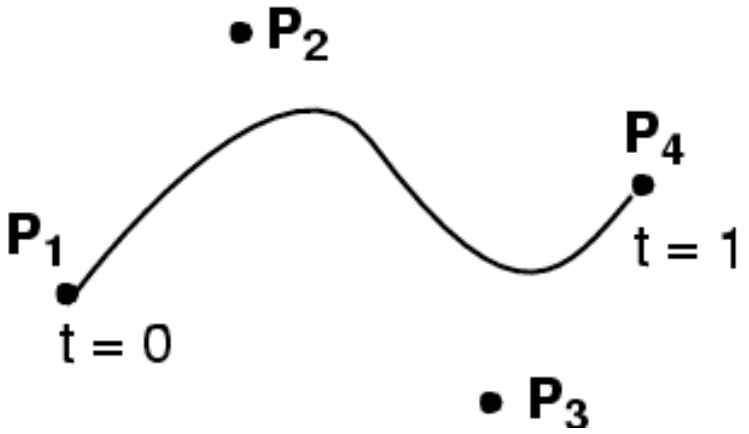
- Motivation
  - Compute normal for surfaces
  - Compute velocity for animation
  - Analyze smoothness



Velocity

- First derivative w.r.t. *t*
- Can you compute this for Bezier curves?

$$P(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

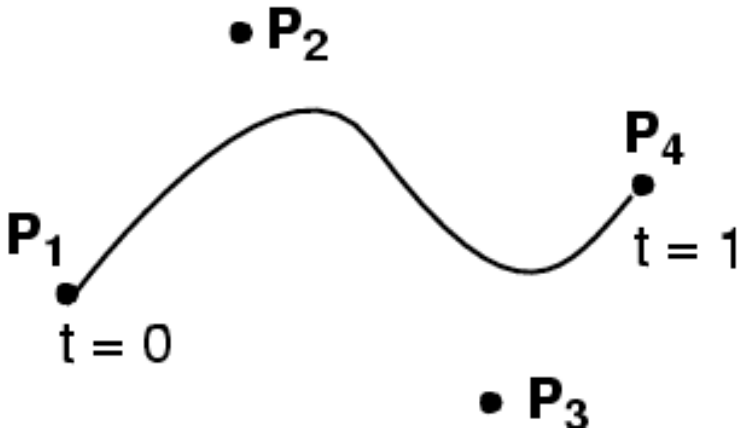


- You know how to differentiate polynomials...

Velocity

- First derivative w.r.t. *t*
- Can you compute this for Bezier curves?

$$P(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$



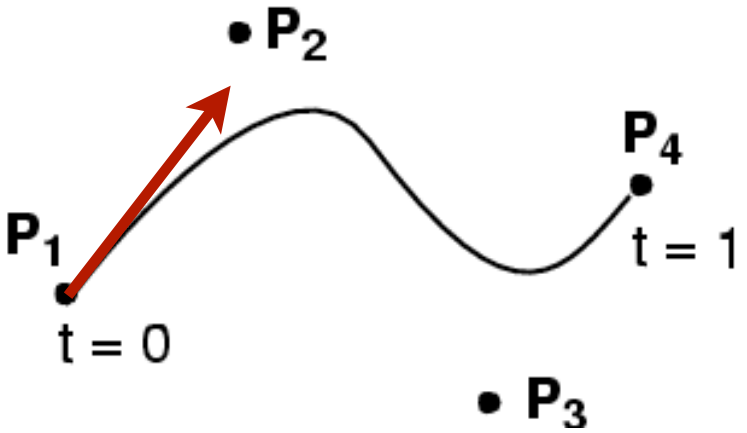
- **$P'(t) = -3(1-t)^2 P_1 + [3(1-t)^2 - 6t(1-t)] P_2 + [6t(1-t) - 3t^2] P_3 + 3t^2 P_4$**

Sanity check: t=0; t=1

Can also write this using a matrix **B'** – try it out!

Tangent

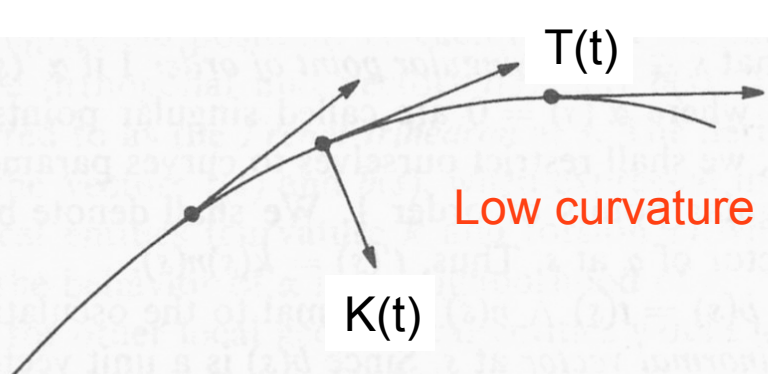
- The tangent to the curve P(t) can be defined as $T(t)=P'(t)/||P'(t)||$
  - normalized velocity, $||T(t)|| = 1$
- This provides us with one orientation for swept surfaces in a little while

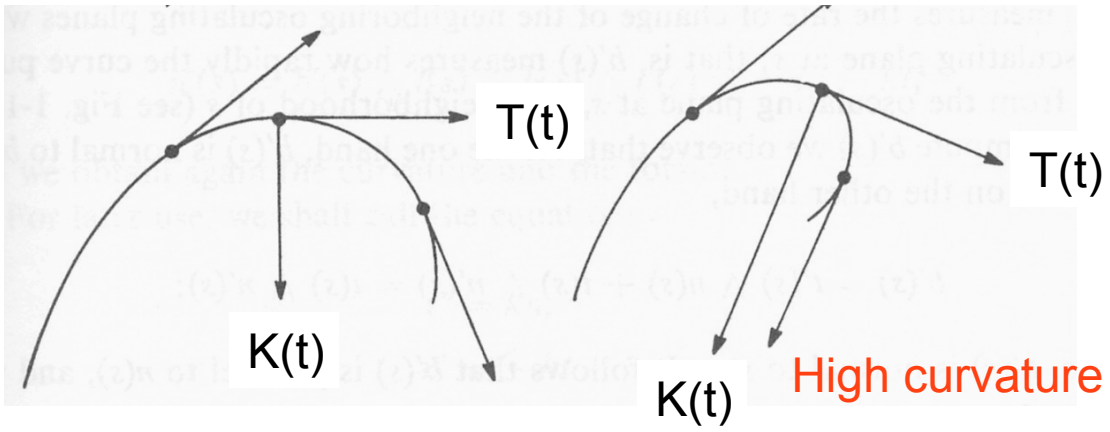


Questions?

Curvature

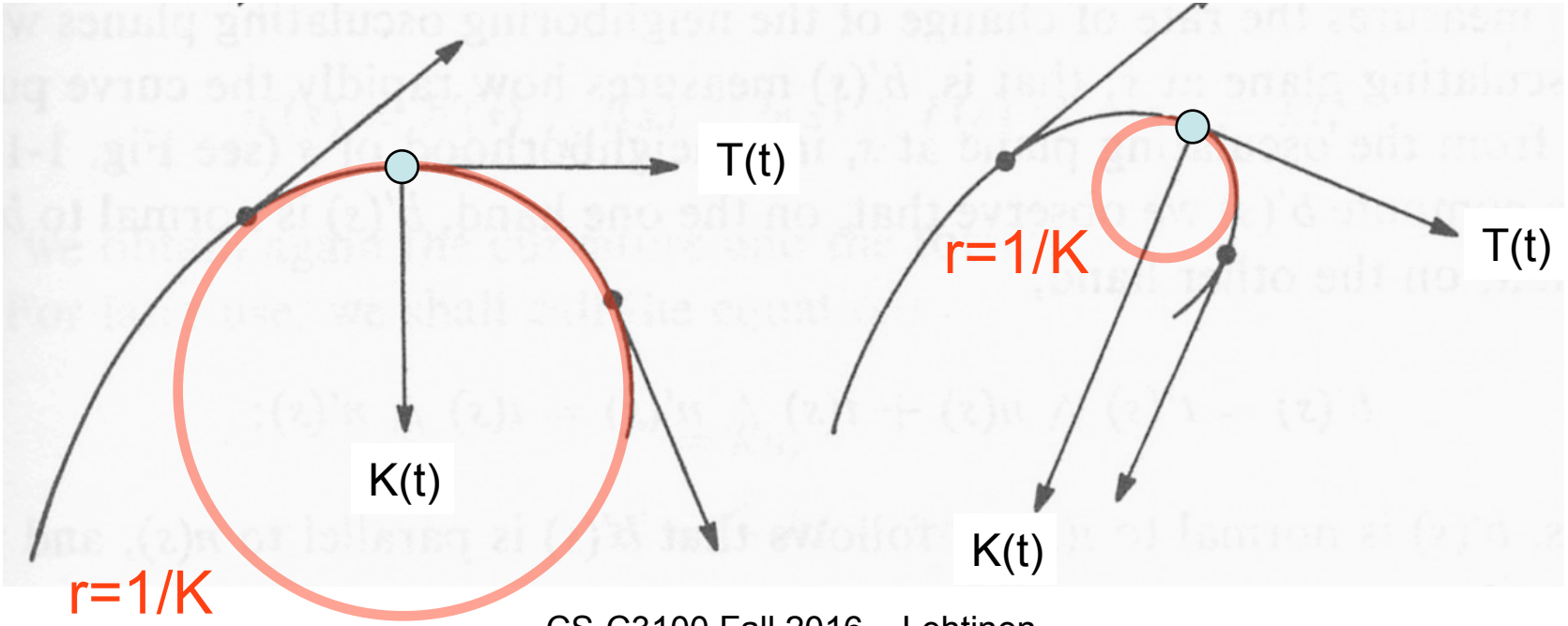
- Derivative of unit tangent
  - K(t)=T'(t)
  - Magnitude ||K(t)|| is constant for a circle
  - Zero for a straight line
- Always orthogonal to tangent, ie. $K \cdot T = 0$
  - Can you prove this? (Hints: ||T(t)||=1, (x(t)y(t))'=?)





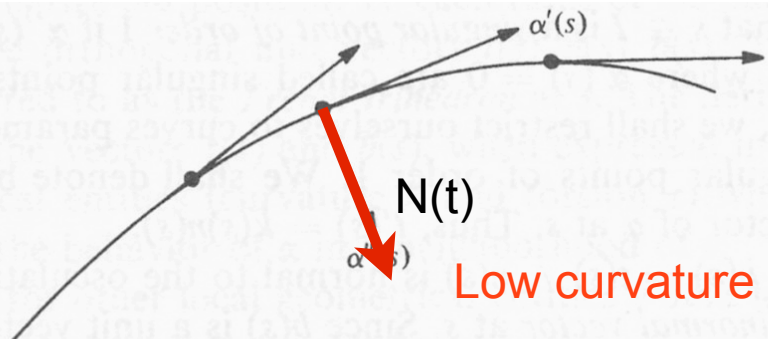
Geometric Interpretation

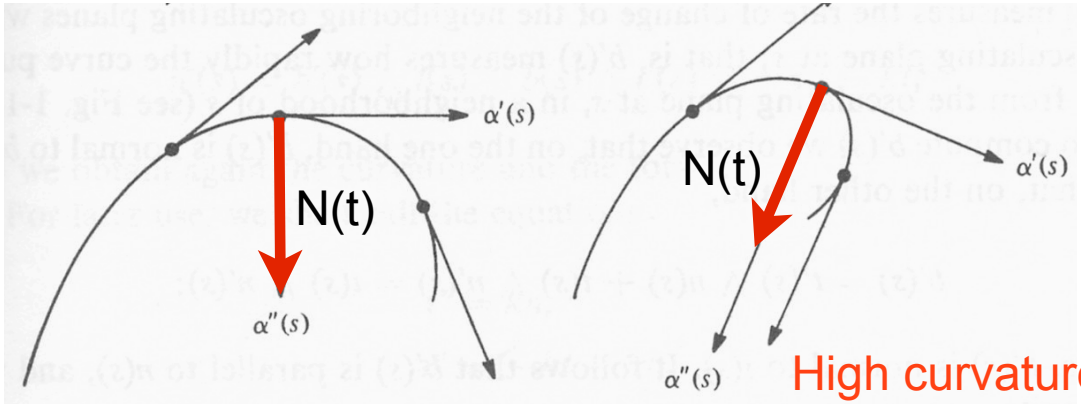
- K is zero for a line, constant for circle
  - What constant? 1/r
- 1/||K(t)|| is the radius of the circle that touches P(t) at *t* and has the same curvature as the curve



Curve Normal

- Normalized curvature: T’(t)/||T’(t)||

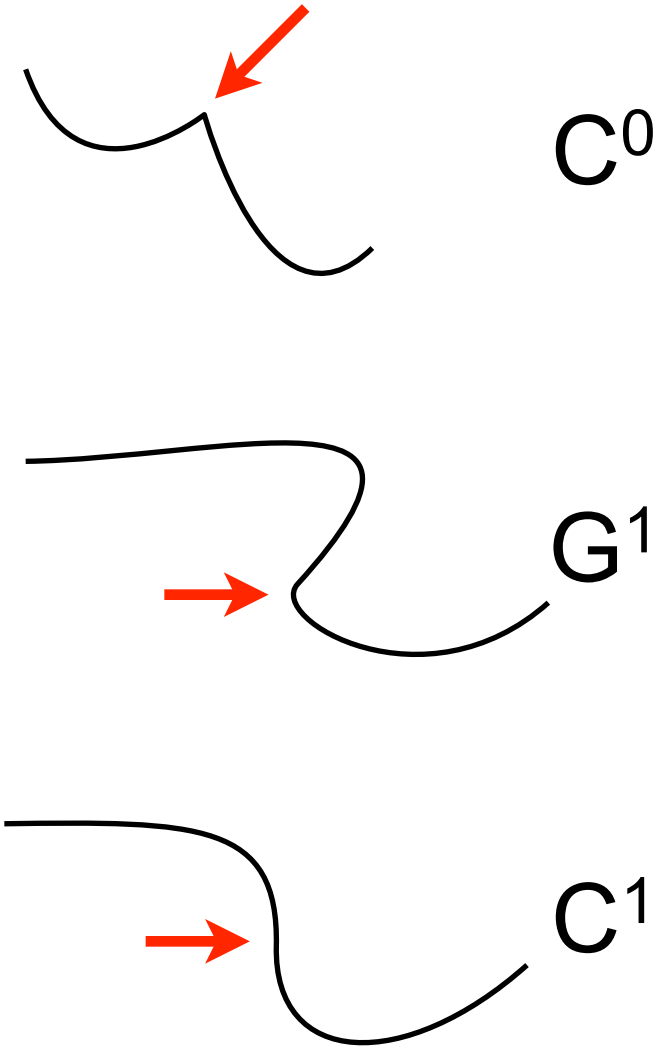




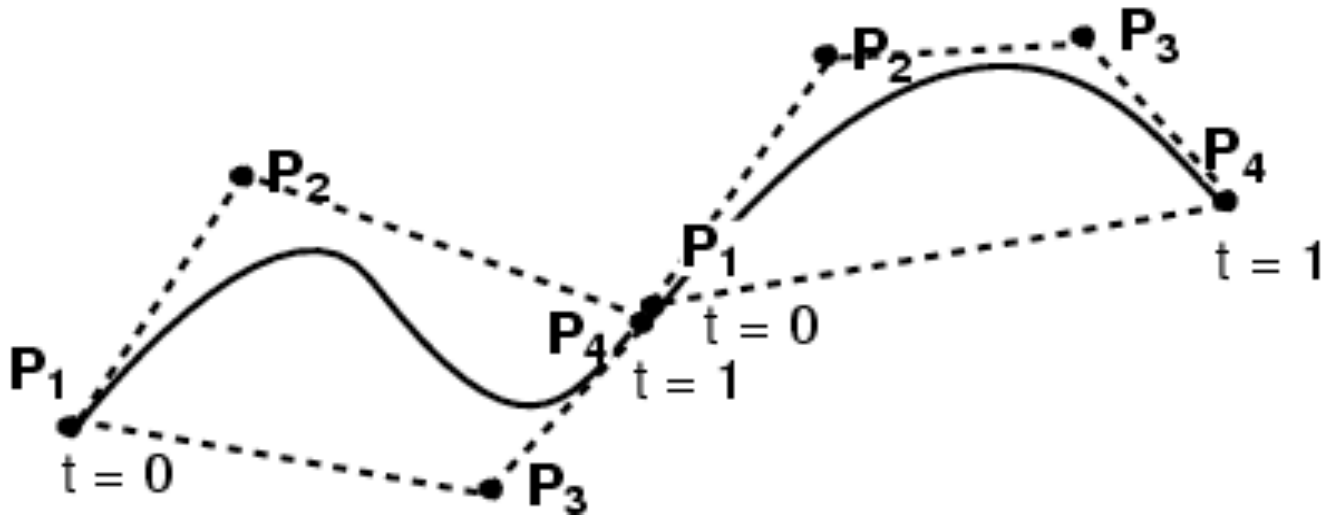
Questions?

Orders of Continuity

- C^0 = continuous
  - The seam can be a sharp kink
- G^1 = geometric continuity
  - Tangents **point to the same direction** at the seam
- C^1 = parametric continuity
  - Tangents **are the same** at the seam, implies G^1
- C^2 = curvature continuity
  - Tangents and their derivatives are the same

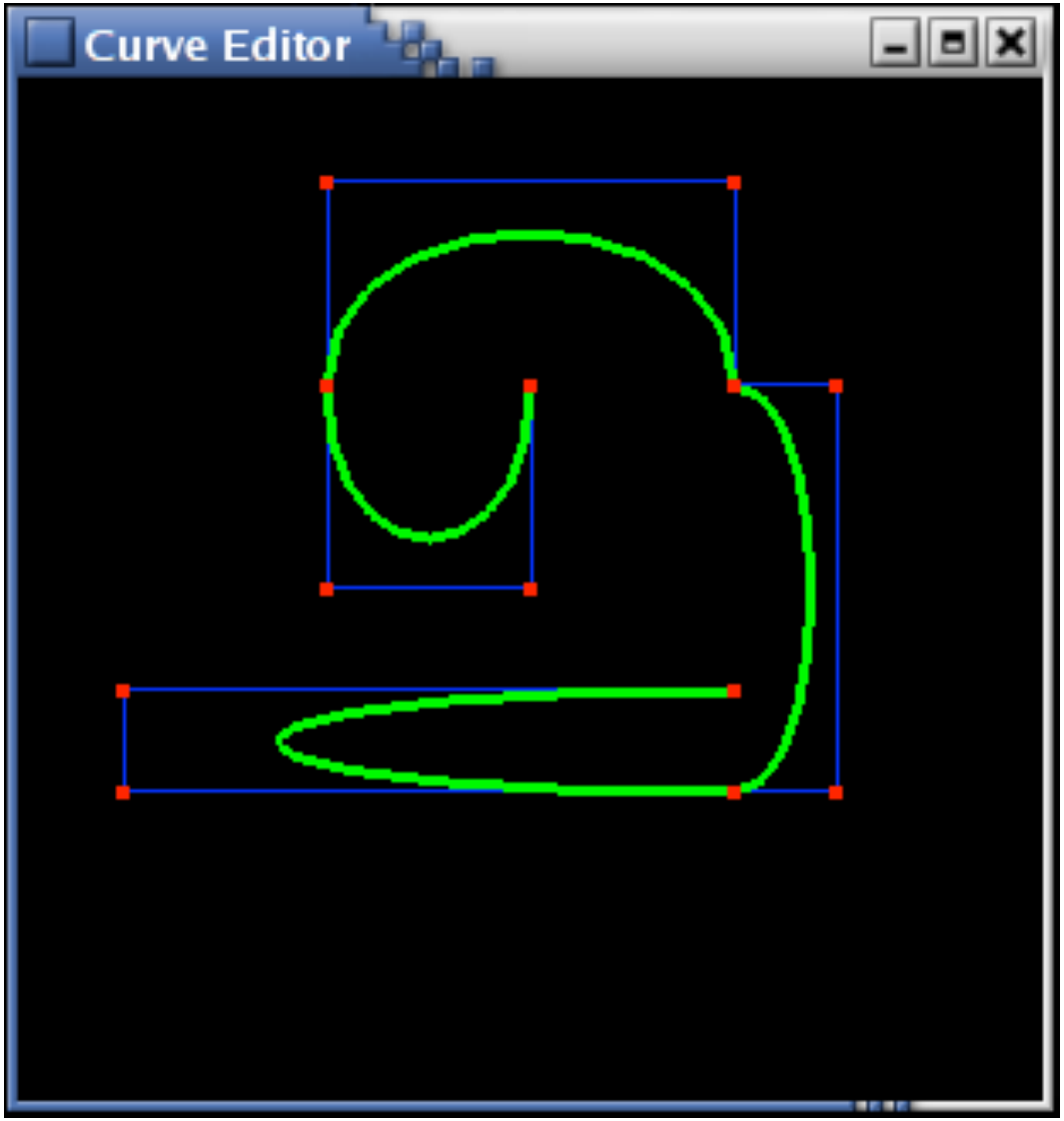


Connecting Cubic Bézier Curves



- How can we guarantee C^0 continuity?
- How can we guarantee G^1 continuity?
- How can we guarantee C^1 continuity?
- C^2 and above gets difficult

Connecting Cubic Bézier Curves

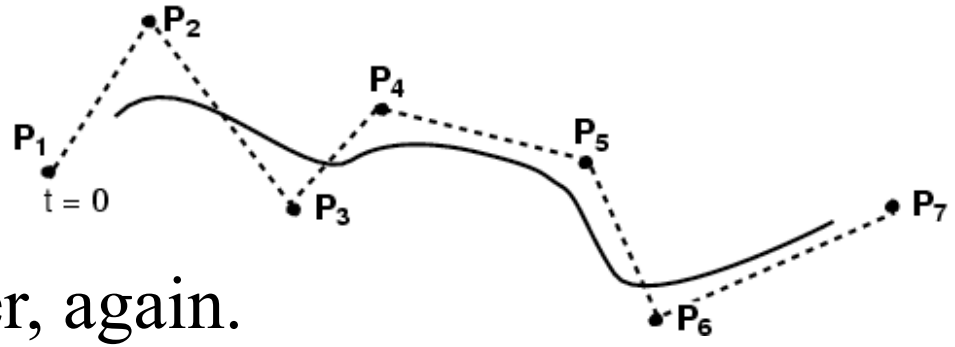


- Where is this curve
  - C^0 continuous?
  - G^1 continuous?
  - C^1 continuous?
- What's the relationship between:
  - the # of control points, and the # of cubic Bézier subcurves?

Questions?

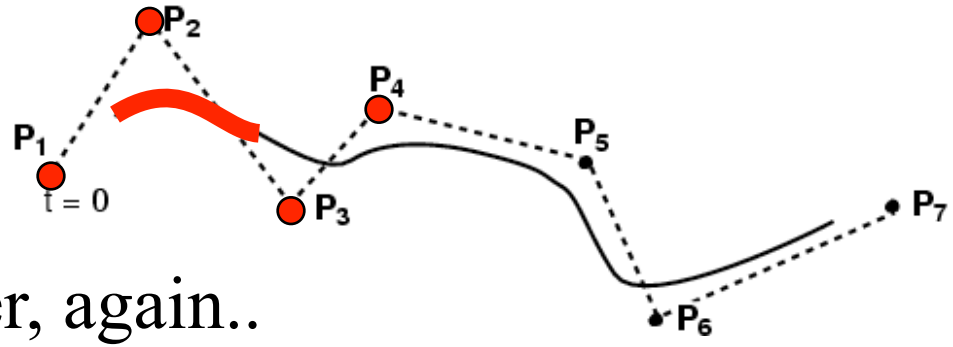
Cubic B-Splines

- ≥ 4 control points
- Locally cubic
  - Cubics chained together, again.



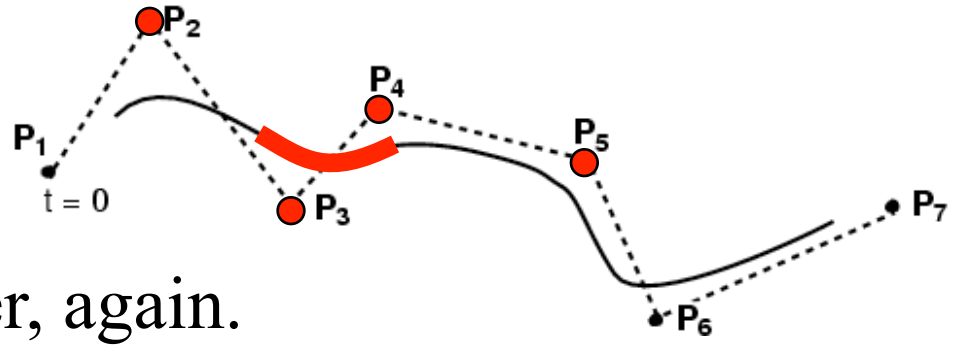
Cubic B-Splines

- ≥ 4 control points
- Locally cubic
  - Cubics chained together, again..
  - BUT with a sliding window of 4 control points!



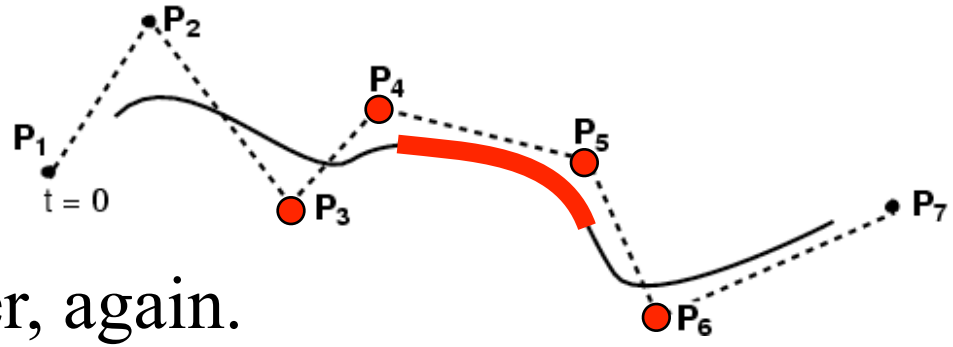
Cubic B-Splines

- ≥ 4 control points
- Locally cubic
  - Cubics chained together, again.
  - BUT with a sliding window of 4 control points!



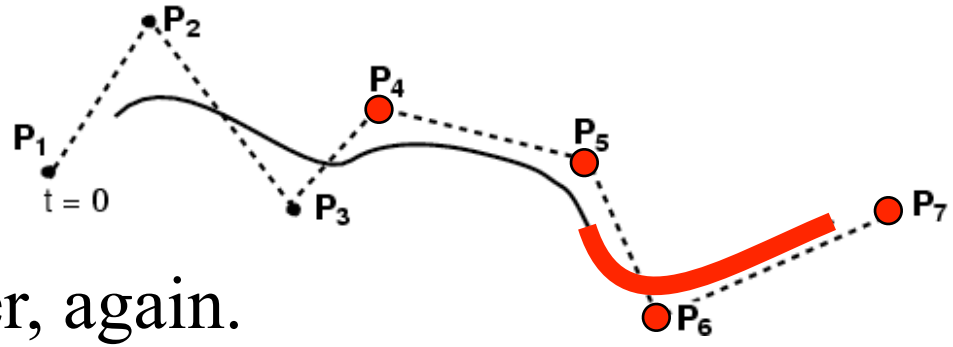
Cubic B-Splines

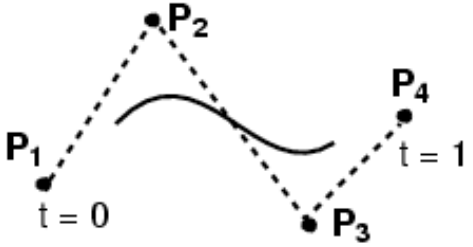
- ≥ 4 control points
- Locally cubic
  - Cubics chained together, again.
  - BUT with a sliding window of 4 control points!

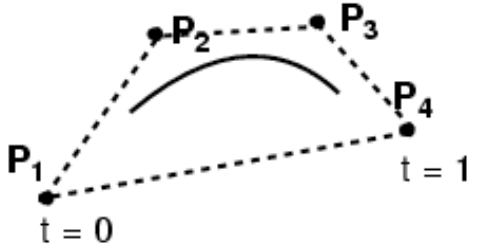


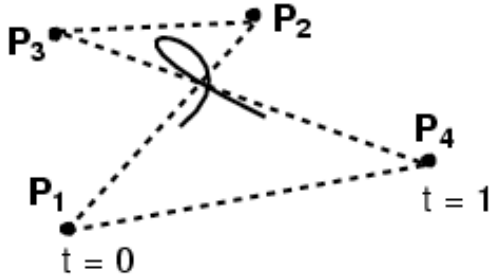
Cubic B-Splines

- ≥ 4 control points
- Locally cubic
  - Cubics chained together, again.
- Curve is not constrained to pass through any control points



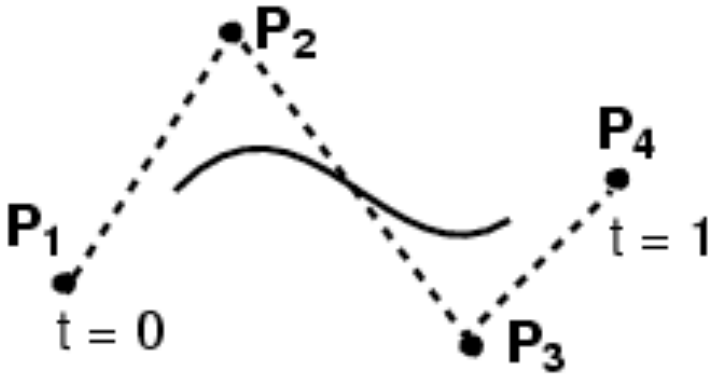


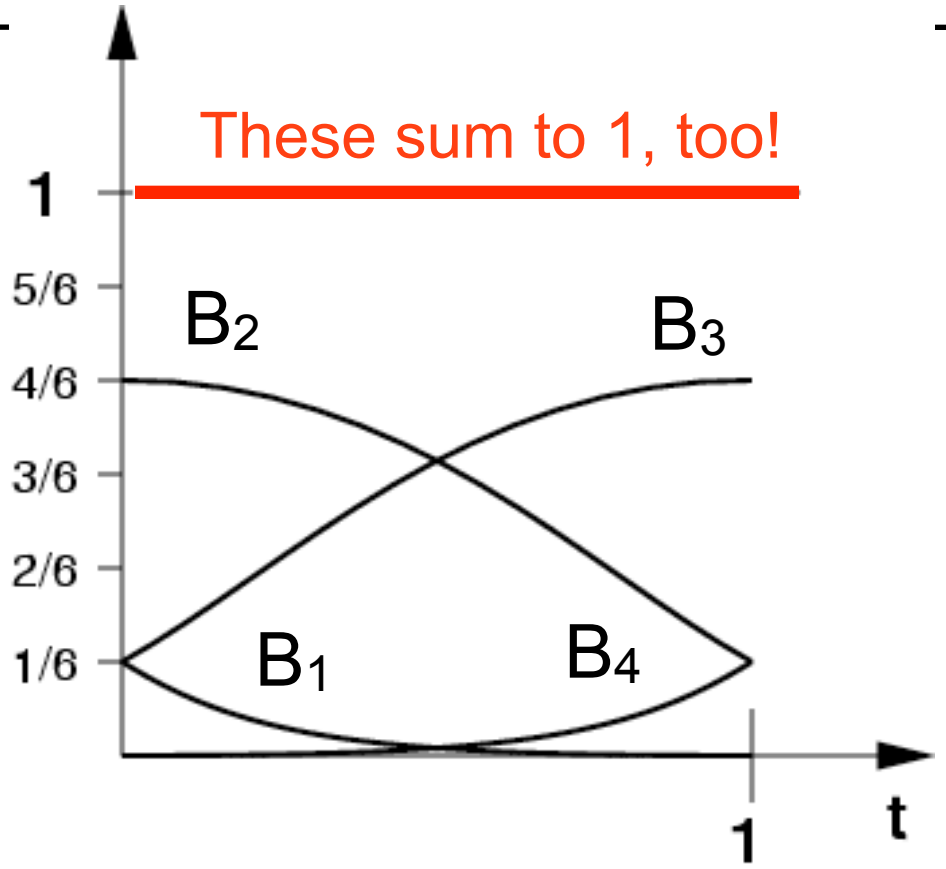




A BSpline curve is also bounded by the convex hull of its control points.

Cubic B-Splines: Basis





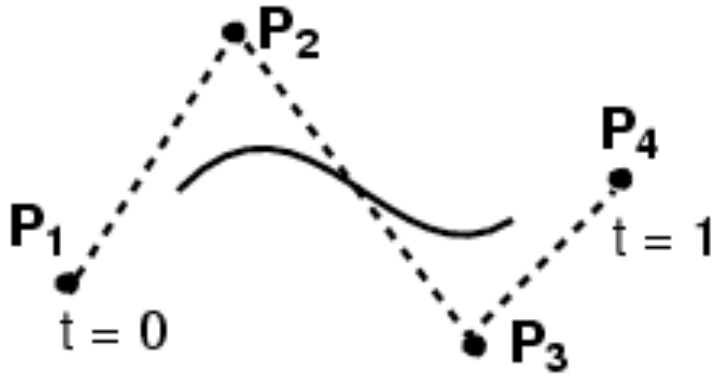
$$B_1(t) = \frac{1}{6}(1-t)^3$$

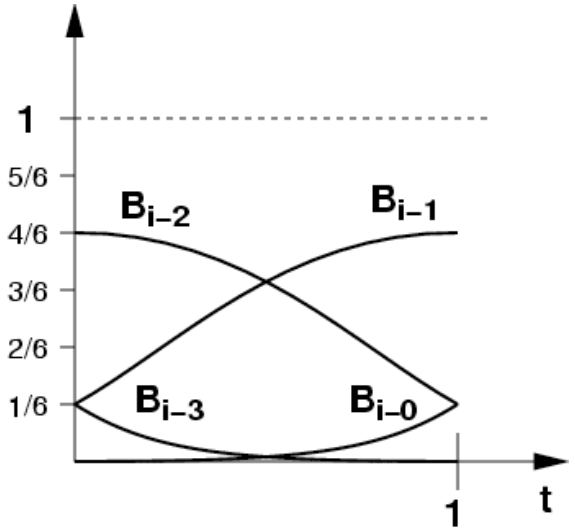
$$B_2(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

$$B_3(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1)$$

$$B_4(t) = \frac{1}{6}t^3$$

Cubic B-Splines: Basis





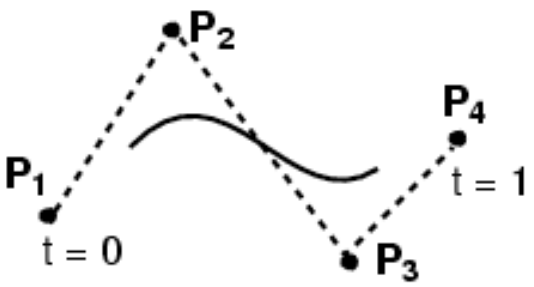
$$Q(t) = \frac{(1-t)^3}{6}P_{i-3} + \frac{3t^3 - 6t^2 + 4}{6}P_{i-2} + \frac{-3t^3 + 3t^2 + 3t + 1}{6}P_{i-1} + \frac{t^3}{6}P_i$$

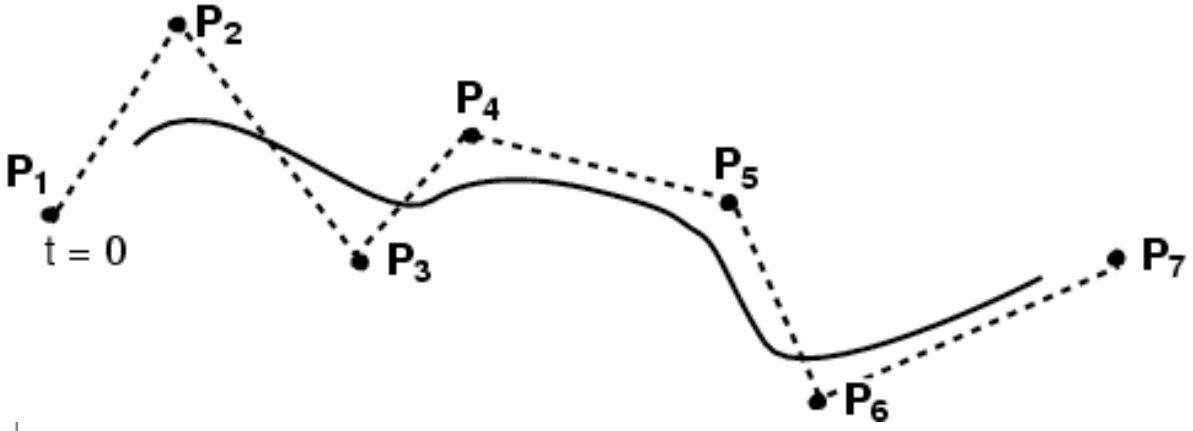
$$Q(t) = \mathbf{GBT(t)}$$

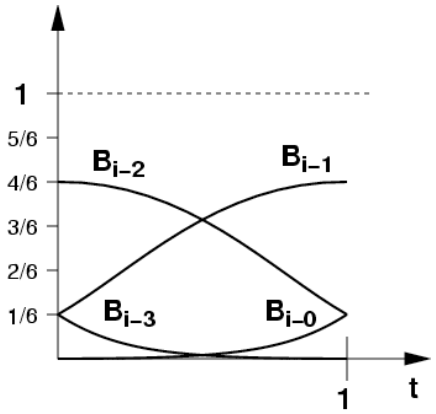
$$B_{B-Spline} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

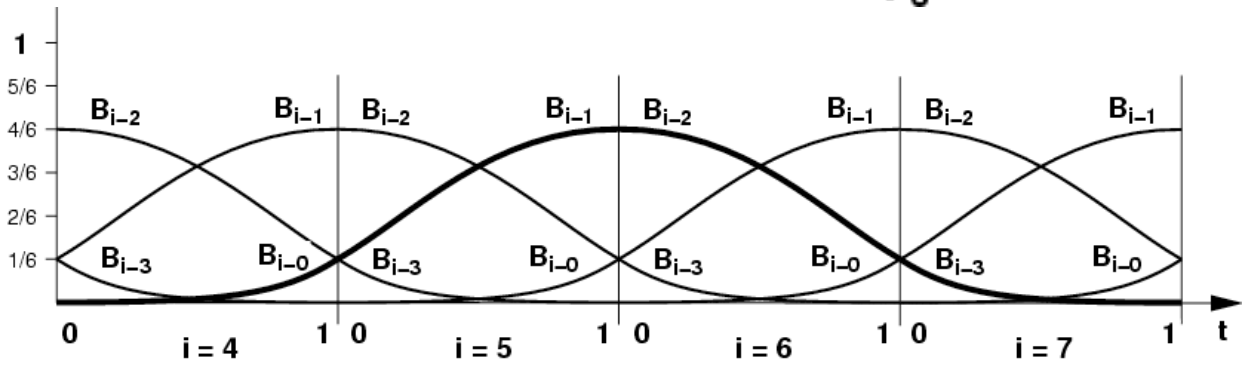
Cubic B-Splines

- Local control (windowing)
- Automatically C^2, and no need to match tangents!

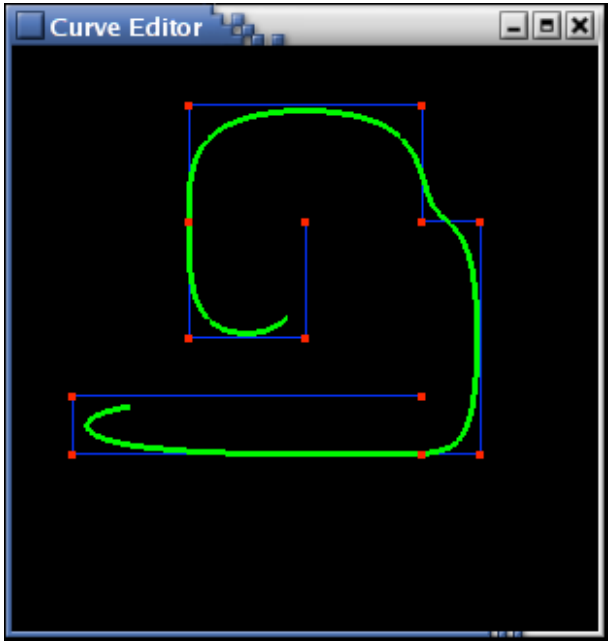




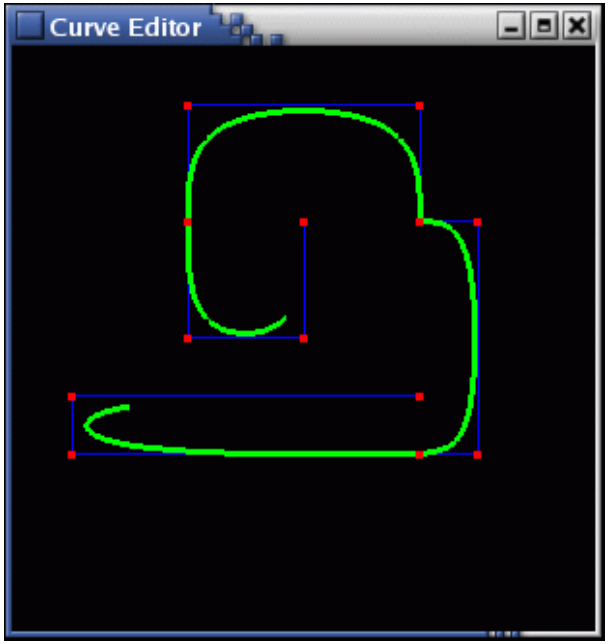




B-Spline Curve Control Points

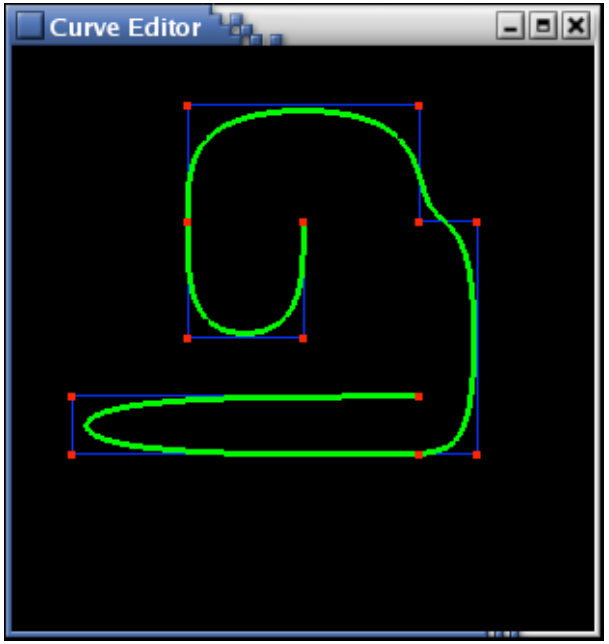


Default BSpline



BSpline with derivative discontinuity

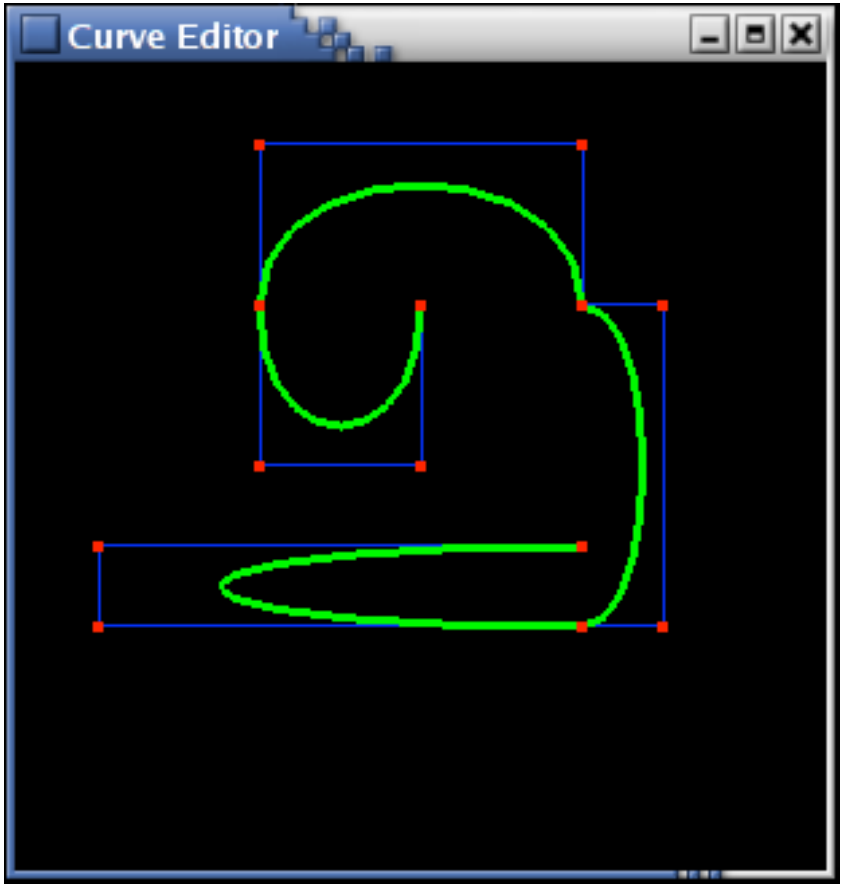
Repeat interior control point



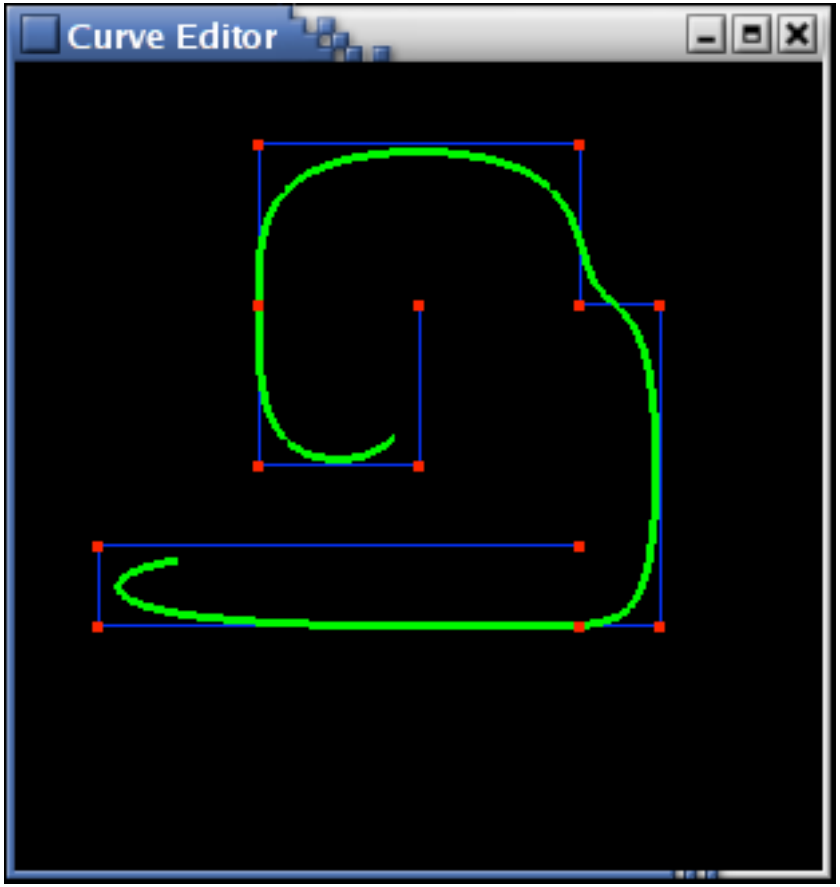
BSpline which passes through end points

Repeat end points

Bézier ≠ B-Spline



Bézier



BSpline

But both are cubics, so one can be converted into the other!

Easy Brain Teaser

- Can you derive formulas for conversion between two different cubic spline matrices, say Bézier and Catmull-Rom?
  - Both are cubics, so you can represent the other curve exactly with new control points $\mathbf{G}_2$

$$\mathbf{G}_1\mathbf{B}_1\mathbf{T}(t) \equiv \mathbf{G}_2\mathbf{B}_2\mathbf{T}(t)$$

$$\mathbf{G}_2 = ???$$

Solution

$$\mathbf{G}_1\mathbf{B}_1\mathbf{T}(t) \equiv \mathbf{G}_2\mathbf{B}_2\mathbf{T}(t)$$

$$\Leftrightarrow \mathbf{G}_1\mathbf{B}_1 = \mathbf{G}_2\mathbf{B}_2$$ **(This must hold for all t!)**

$$B_{B-Spline} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$B_{Bezier} = \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Solution

$$\mathbf{G}_1\mathbf{B}_1\mathbf{T}(t) \equiv \mathbf{G}_2\mathbf{B}_2\mathbf{T}(t)$$

$$\Leftrightarrow \mathbf{G}_1\mathbf{B}_1 = \mathbf{G}_2\mathbf{B}_2$$ **(This must hold for all t!)**

$$\Rightarrow \boxed{\mathbf{G}_2 = \mathbf{G}_1\mathbf{B}_1\mathbf{B}_2^{-1}}$$

$$B_{B-Spline} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$B_{Bezier} = \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Converting between Bézier & BSpline

- Simple with the basis matrices!
  - Note that this only works for a single segment of 4 control points

$$B_{Bezier} = \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- $P(t) = \mathbf{G}\,\mathbf{B}_1\,\mathbf{T}(t) = \mathbf{G}\,\mathbf{B}_1\,(\mathbf{B}_2^{-1}\mathbf{B}_2)\,\mathbf{T}(t) = (\mathbf{G}\,\mathbf{B}_1\,\mathbf{B}_2^{-1})\,\mathbf{B}_2\,\mathbf{T}(t)$

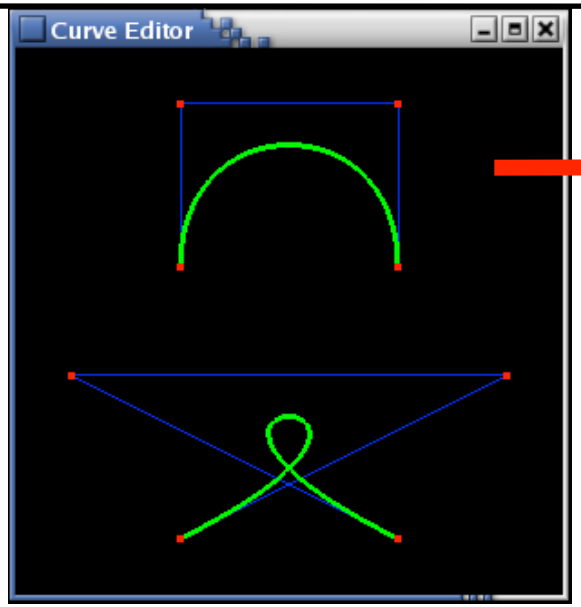
$$B_{B-Spline} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

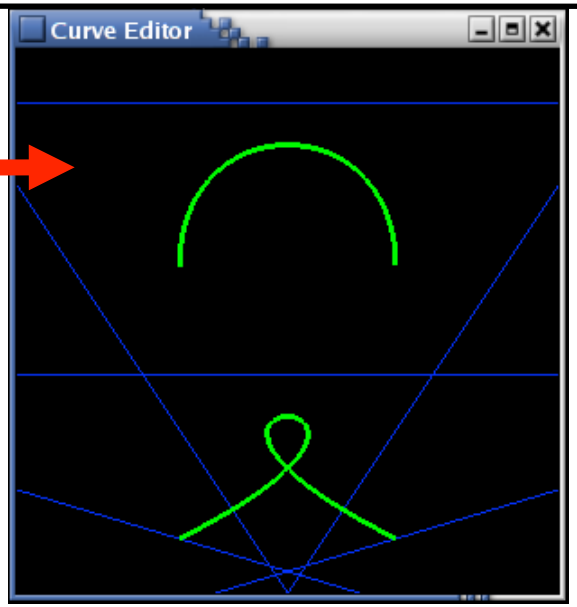
- $\mathbf{G}\,\mathbf{B}_1\,\mathbf{B}_2^{-1}$ are the control points for the segment in new basis.

$$Q(t) = \mathbf{GBT(t)} \quad = \text{Geometry } \mathbf{G} \cdot \text{Spline Basis } \mathbf{B} \cdot \text{Power Basis } \mathbf{T(t)}$$

Converting between Bézier & BSpline

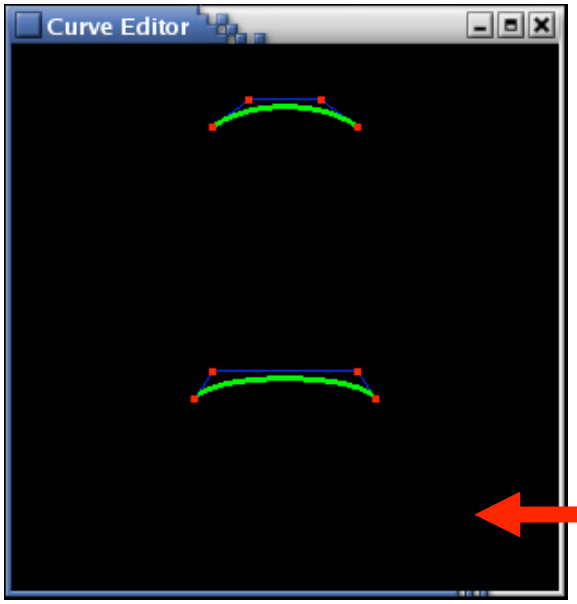
original control points as Bézier

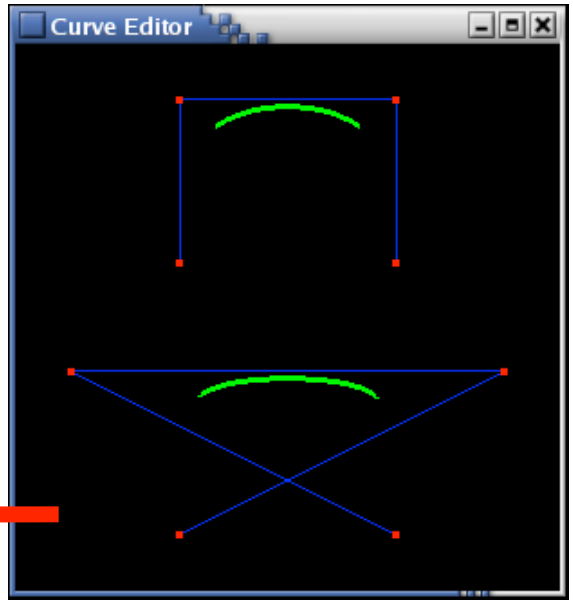




new BSpline control points to match Bézier

new Bézier control points to match BSpline

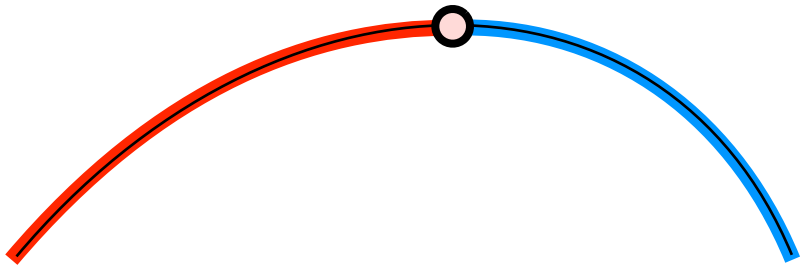




original control points as BSpline

Why Bother with B-Splines?

- Automatic C^2 is nice!
- Also, B-Splines can be split into segments of non-uniform length without affecting the global parametrization.
  - "Non-uniform B-Splines"
  - We'll not do this, but just so you know.



NURBS (Generalized B-Splines)

- Rational cubics
  - Use homogeneous coordinates, just add *w* !
    - Provides a "tension" parameter to control points

- NURBS: Non-Uniform Rational B-Spline
  - **non-uniform** = different spacing between the blending functions, a.k.a. "knots"
  - **rational** = ratio of cubic polynomials (instead of just cubic)
    - implemented by adding the homogeneous coordinate *w* into the control points.

Questions?

That's All for Today, Folks

- Fun stuff to know about function/vector spaces
  - http://en.wikipedia.org/wiki/Vector_space
  - http://en.wikipedia.org/wiki/Functional_analysis
  - http://en.wikipedia.org/wiki/Function_space

- Inkscape is an open source vector drawing program for Mac/Windows. Try it out!