
6.837 Computer Graphics

Bézier Curves and Splines

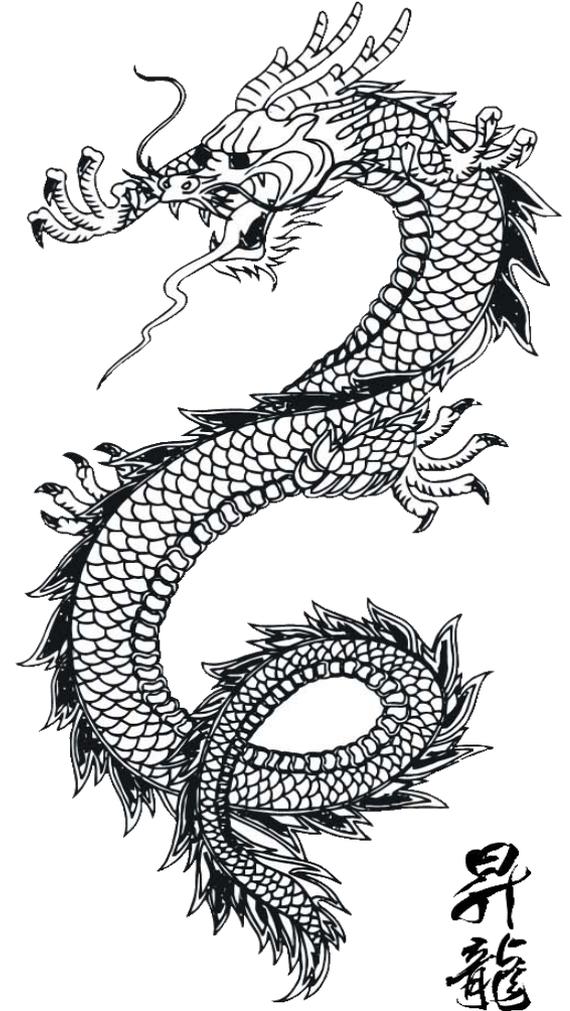
Wojciech Matusik
MIT CSAIL

Before We Begin

- Anything on your mind concerning Assignment 0?
- Any questions about the course?
- Assignment 1 (Curves & Surfaces)
- Linear algebra review session

Today

- Smooth curves in 2D
 - Useful in their own right
 - Provides basis for surface editing

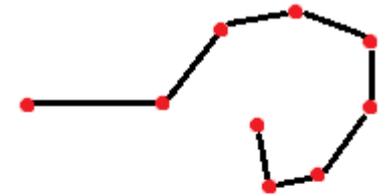


This image is in the public domain
Source: [Wikimedia Commons](#)

Modeling 1D Curves in 2D

- **Polylines**

- Sequence of vertices connected by straight line segments
- Useful, but not for smooth curves
- This is the representation that usually gets drawn in the end (a curve is converted into a polyline)



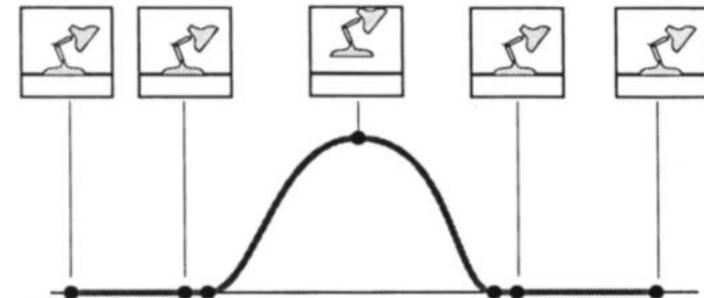
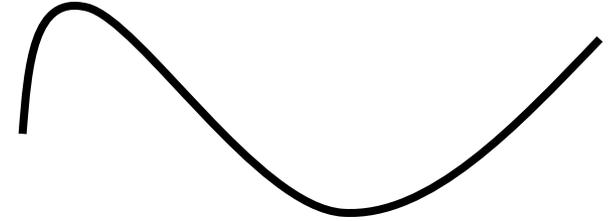
- **Smooth curves**

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



Splines

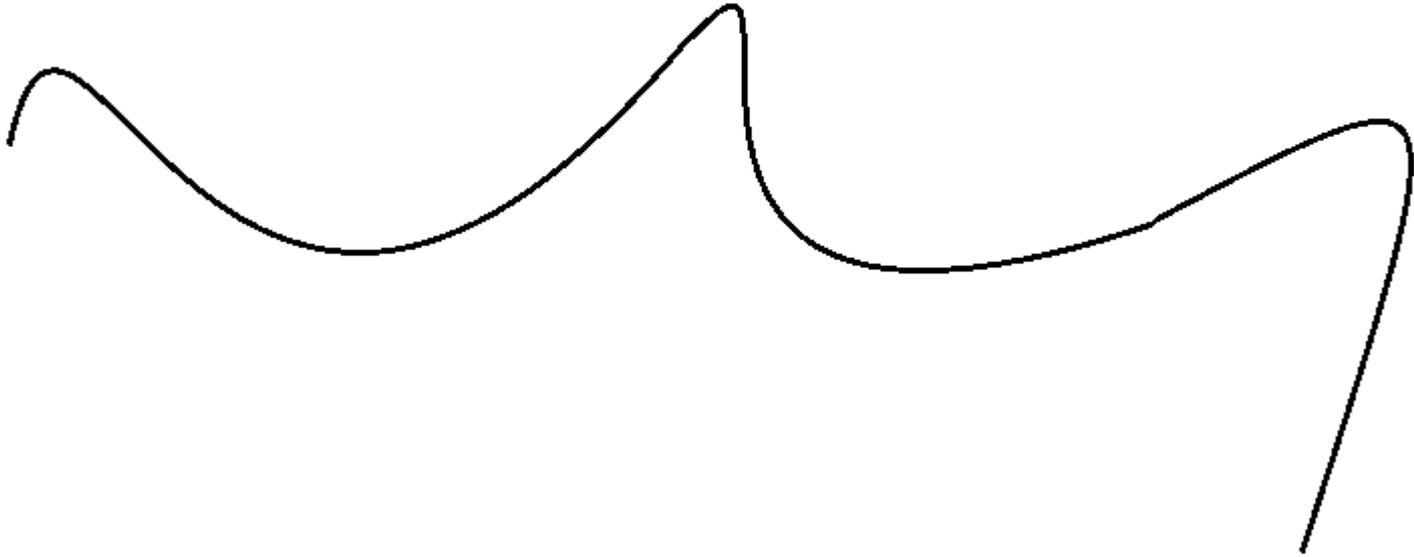
- A type of smooth curve in 2D/3D
- Many different uses
 - 2D illustration (e.g., Adobe Illustrator)
 - Fonts (e.g., PostScript, TrueType)
 - 3D modeling
 - Animation: trajectories
- In general: interpolation and approximation



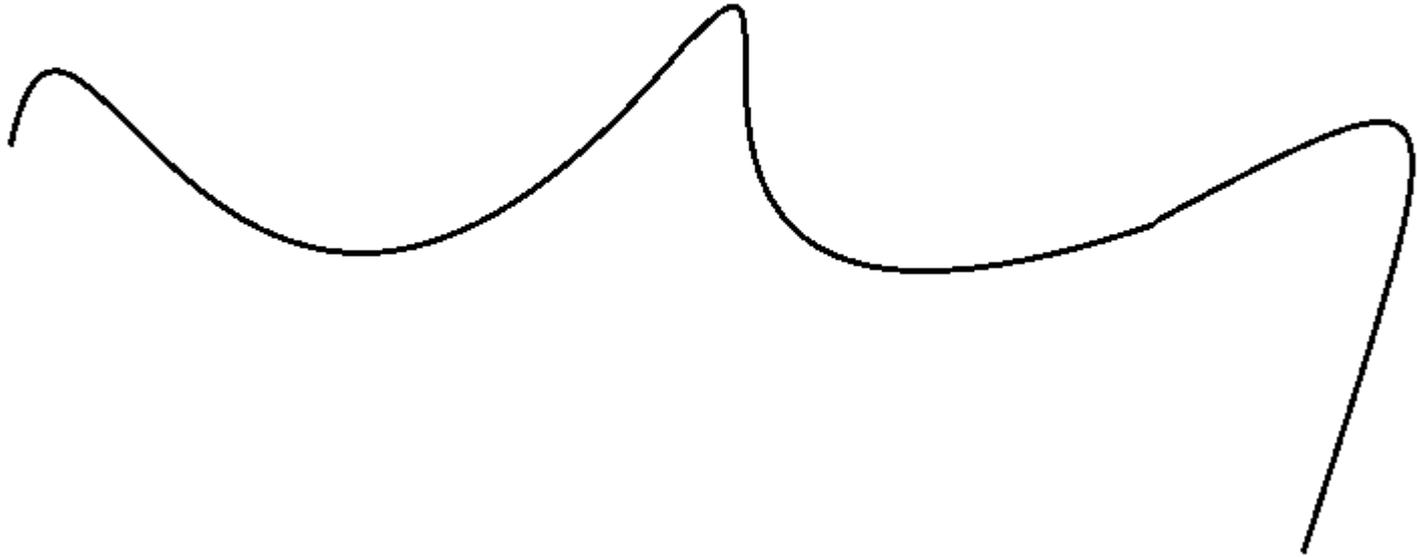
ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

Demo

How Many Dimensions?

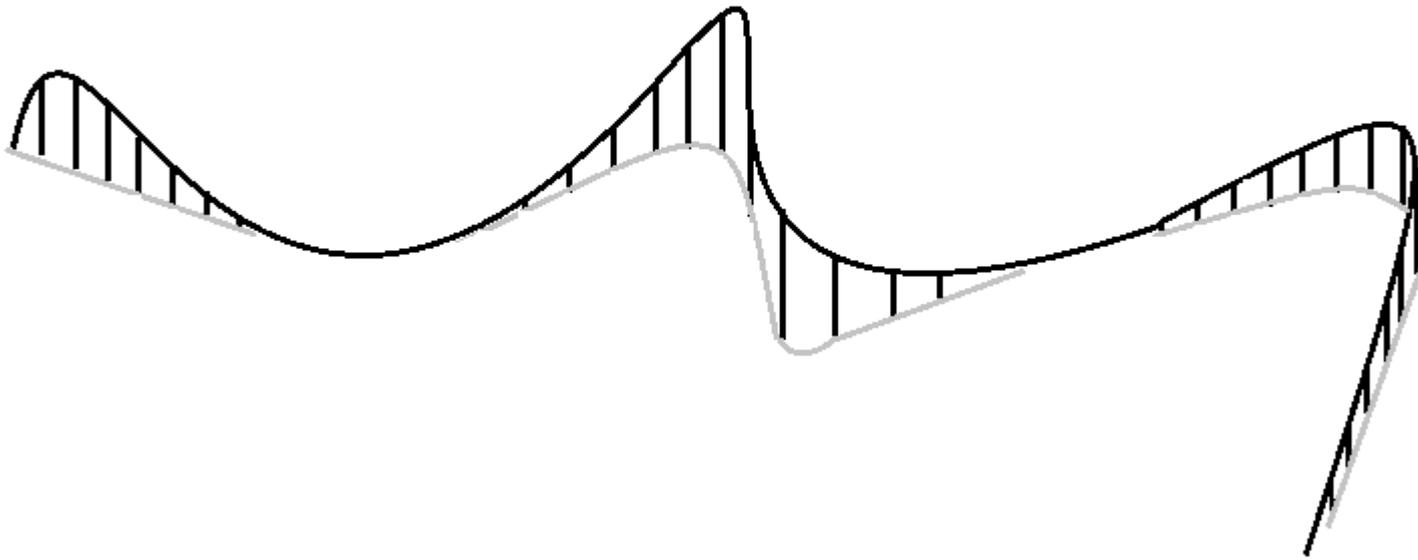


How Many Dimensions?



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

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

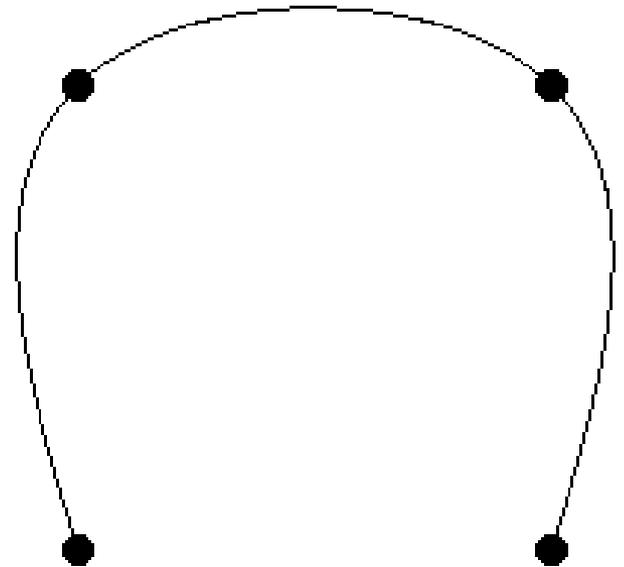
- A continuous 1D set of points in 2D (or 3D)
- A mapping from an interval S onto the plane
 - That is, $P(t)$ is the point of the curve at parameter t

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

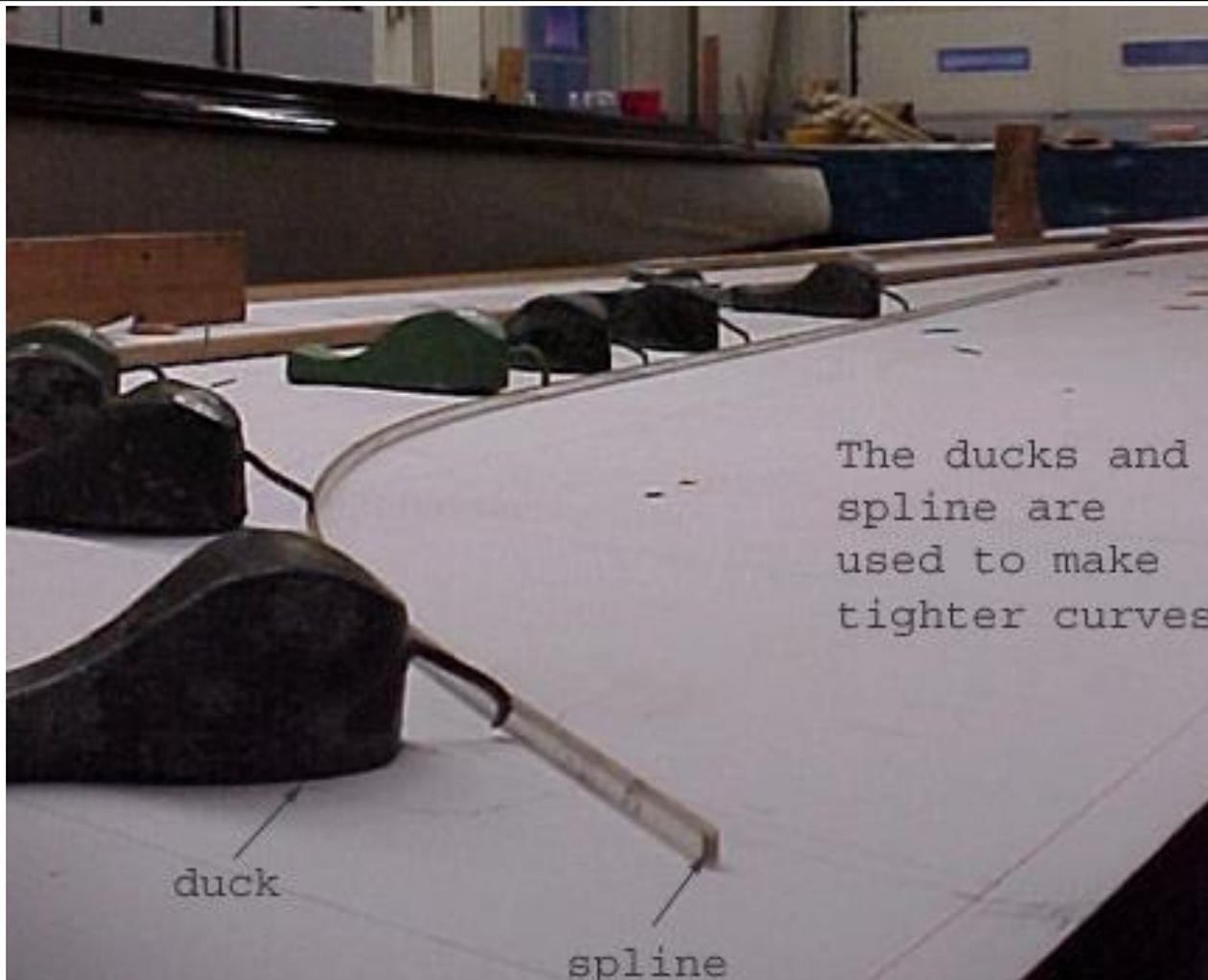
- Big differences
 - It is easy to generate points on the curve from the 2nd
 - The second definition can describe trajectories, the speed at which we move on the curve

General Principle of Splines

- User specifies **control points**
- We will interpolate the control points by a smooth curve
 - The curve is completely determined by the control points.



Physical Splines



Courtesy of The Antique Boat Museum.

[See http://en.wikipedia.org/wiki/Flat_spline](http://en.wikipedia.org/wiki/Flat_spline)

Two Application Scenarios

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

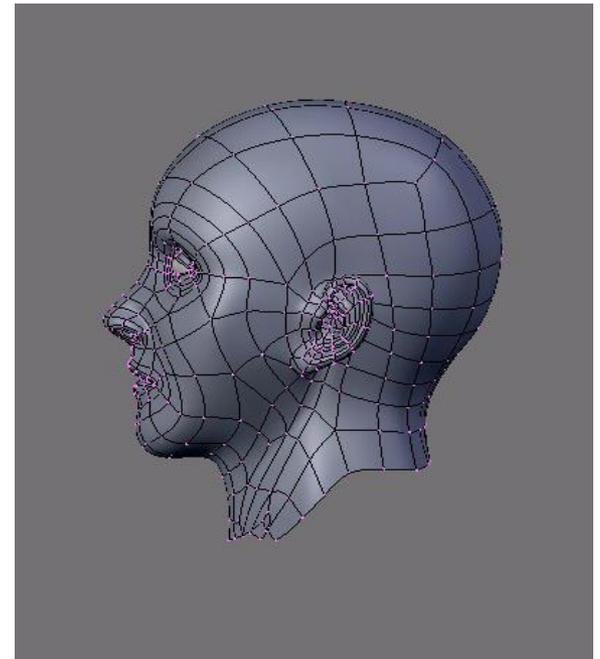


Image courtesy of [Saphires](#) on Wikimedia Commons. License: CC-BY-SA. This content is excluded from our Creative Commons license. For more information, see <http://ocw.mit.edu/help/faq-fair-use/>.

Questions?

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

- It is easy to rasterize mathematical line segments into pixels
 - OpenGL and the graphics hardware can do it for you
- But polynomials and other parametric functions are harder

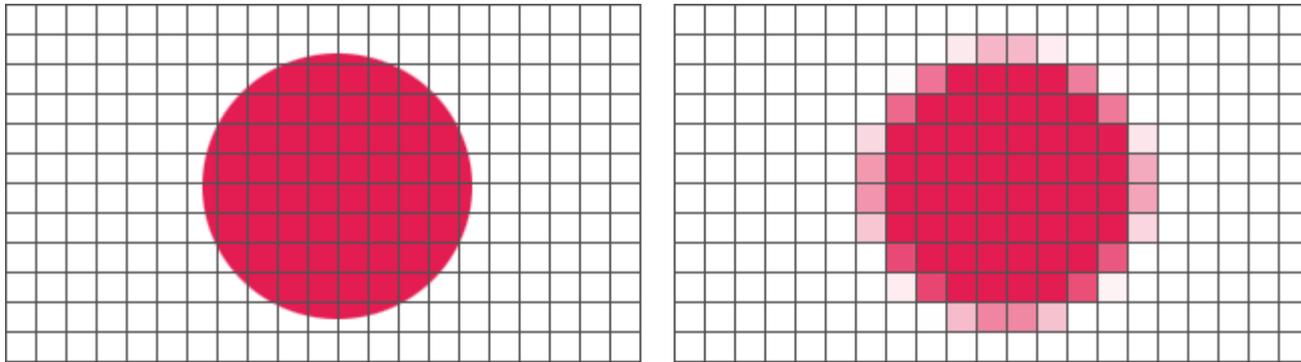
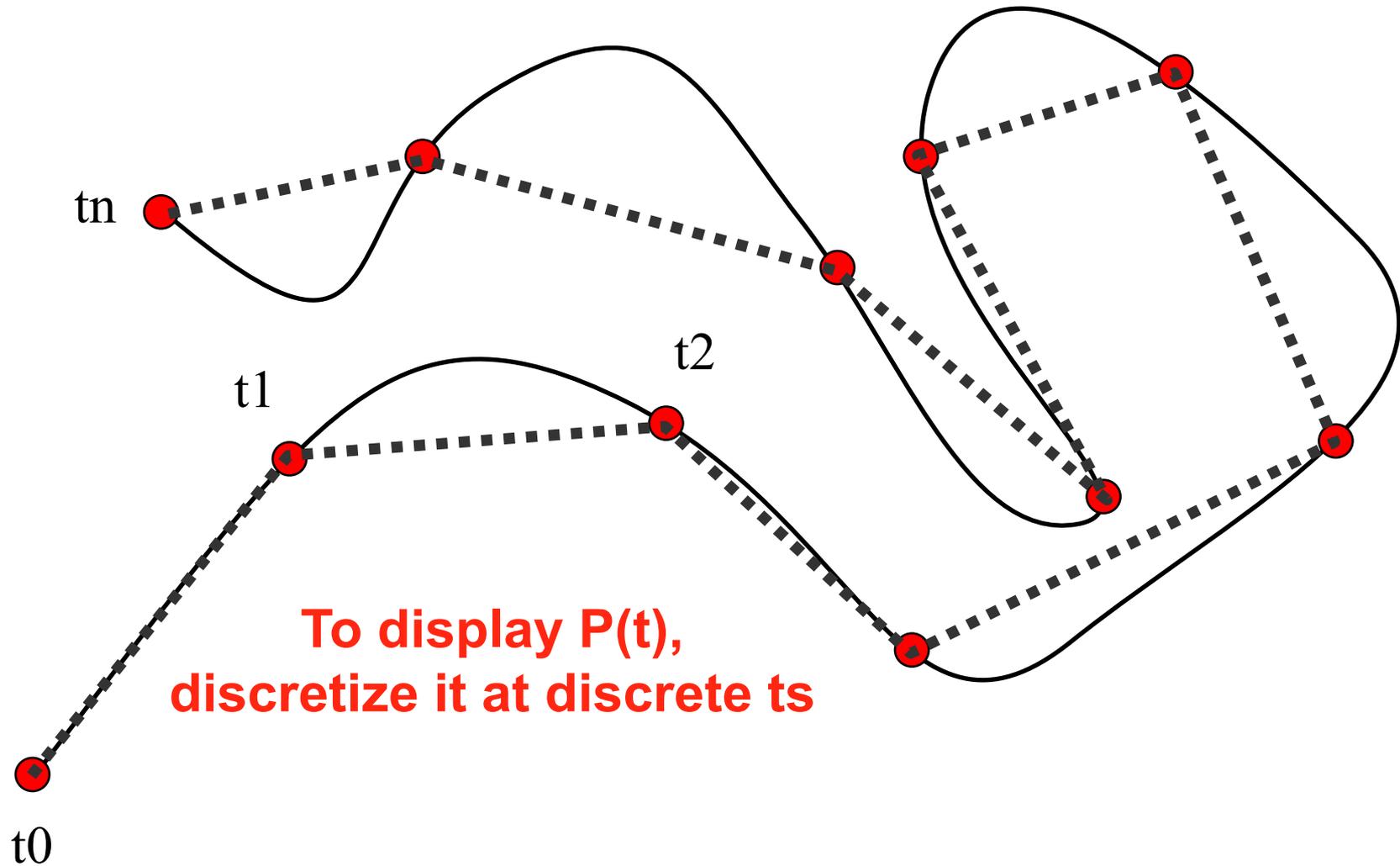
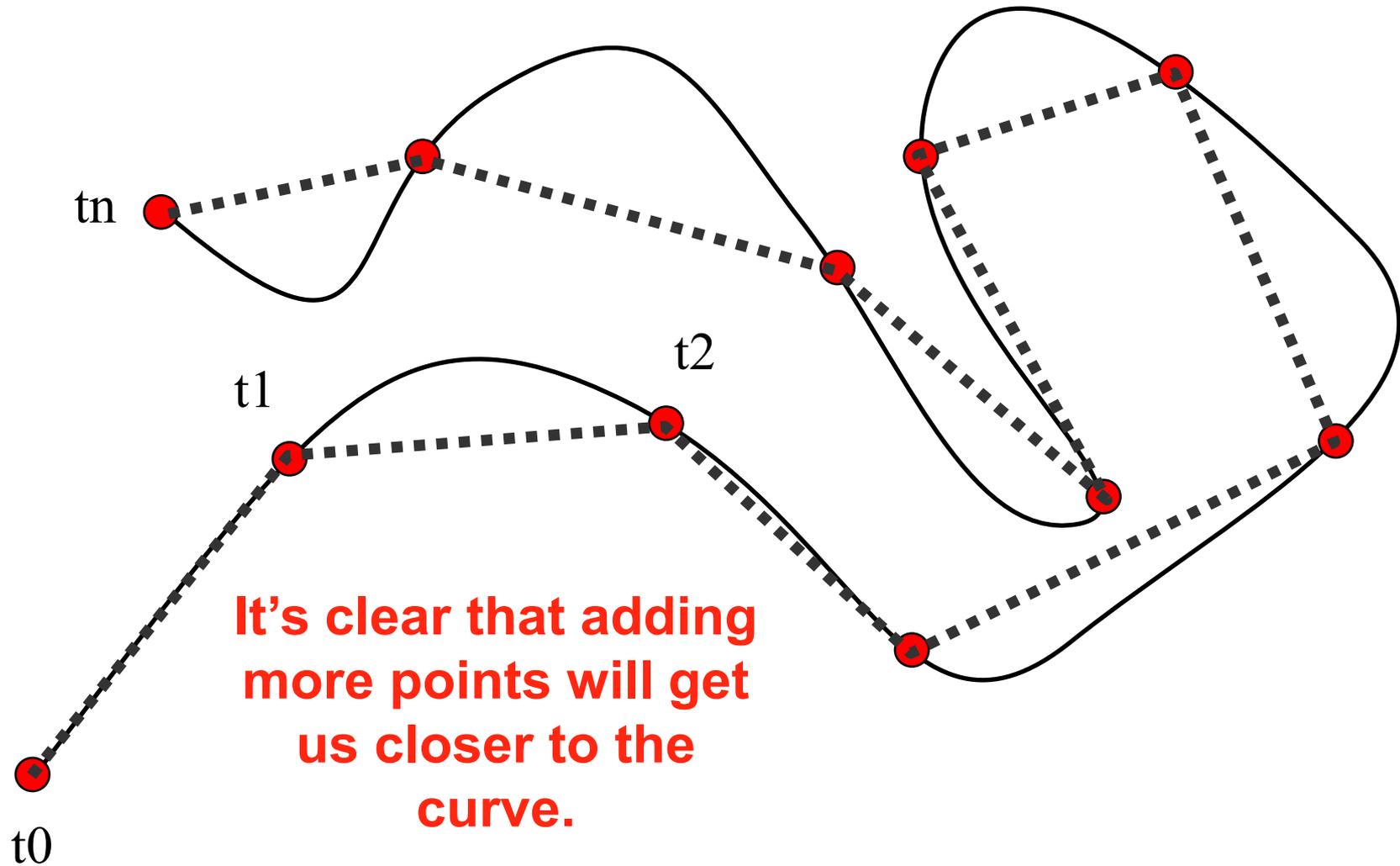


Image courtesy of [Phrood](#) on Wikimedia Commons. License: CC-BY-SA. This content is excluded from our Creative Commons license. For more information, see <http://ocw.mit.edu/help/faq-fair-use/>.

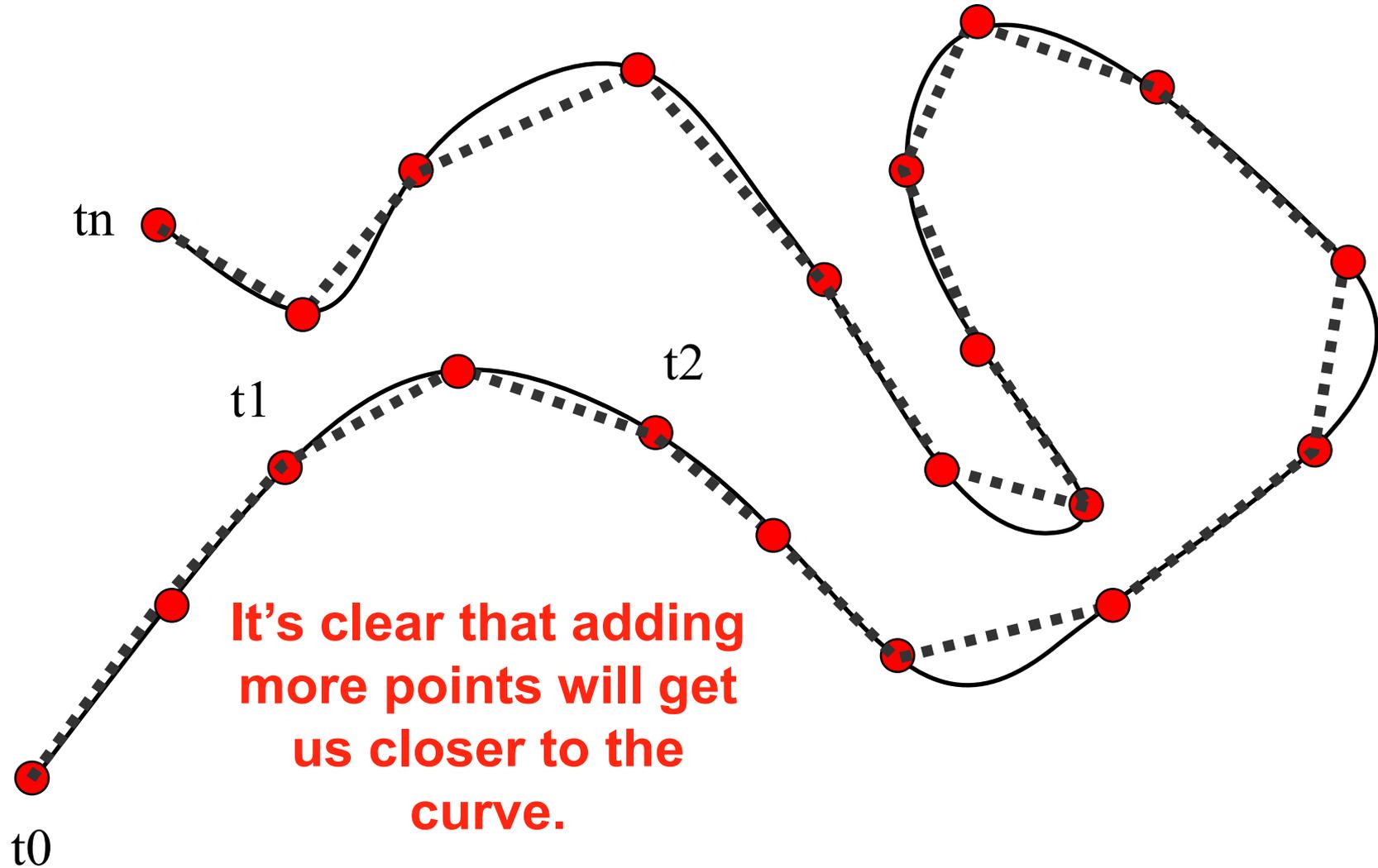
Tessellation



Tessellation

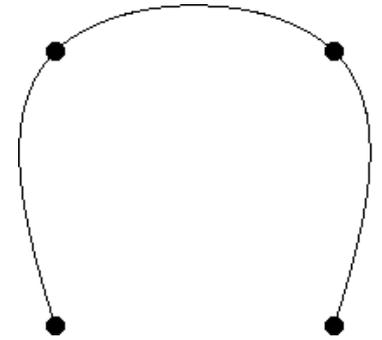


Tessellation

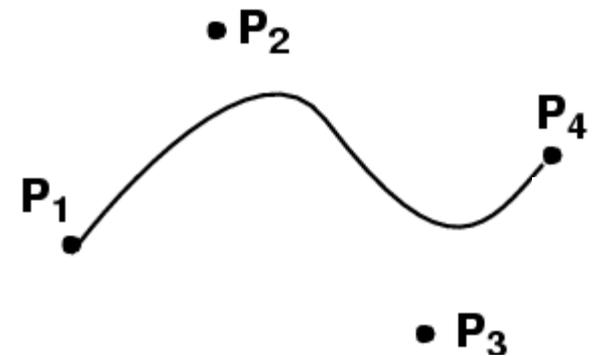


Interpolation vs. Approximation

- Interpolation
 - Goes through all specified points
 - Sounds more logical
- Approximation
 - Does not go through all points



Interpolation

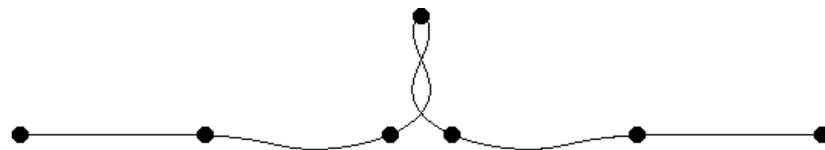


Approximation

Interpolation vs. Approximation

- Interpolation

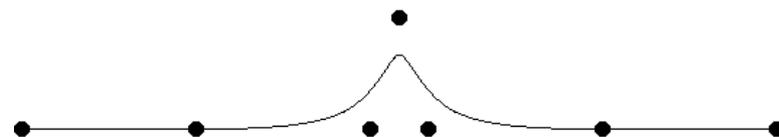
- Goes through all specified points
- Sounds more logical
- But can be more unstable



Interpolation

- Approximation

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



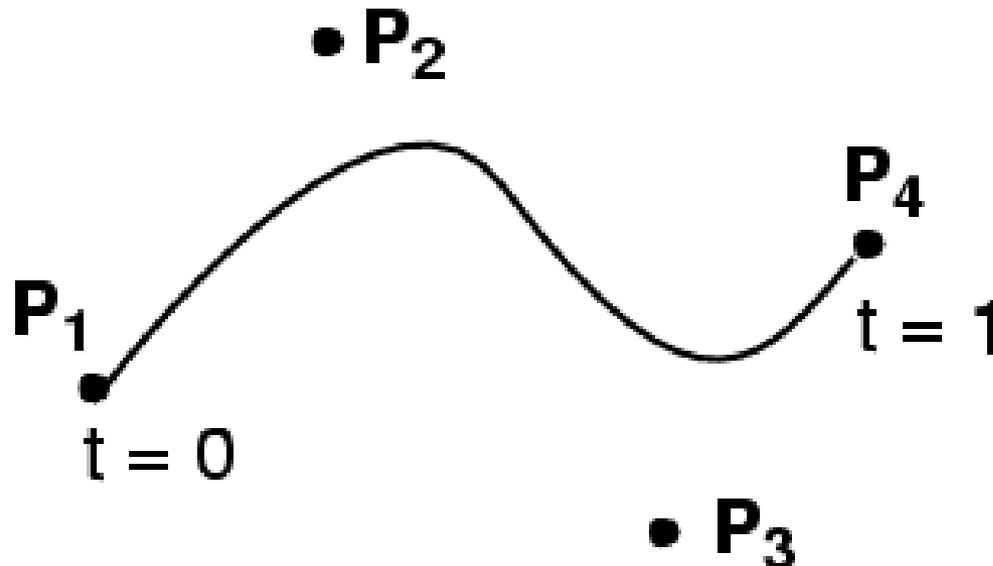
Approximation

- We will do something in between.

Questions?

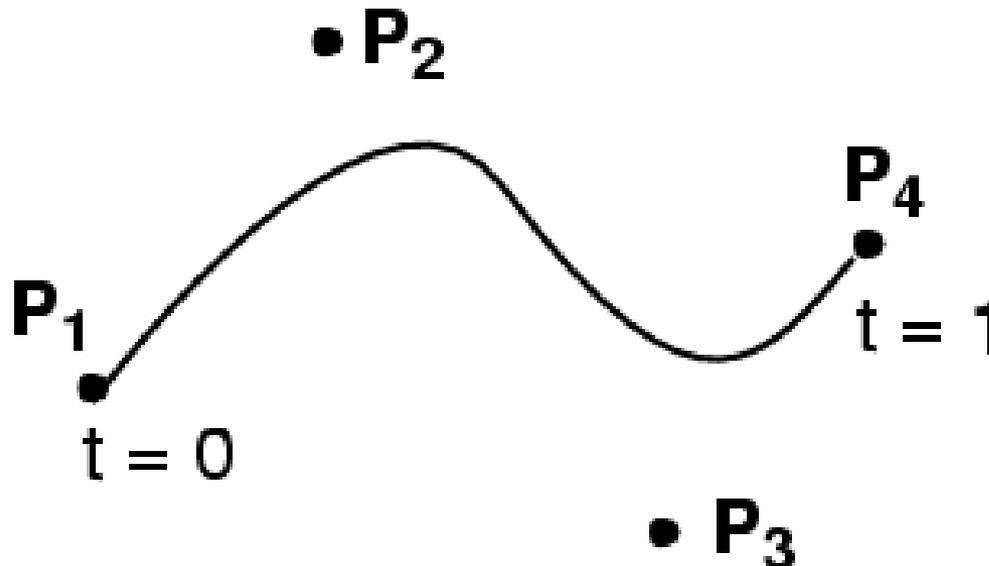
Cubic Bézier Curve

- User specifies 4 control points $P_1 \dots P_4$
- Curve goes through (interpolates) the ends P_1, P_4
- Approximates the two other ones
- Cubic polynomial



Cubic Bézier Curve

$$\begin{aligned} \bullet \quad P(t) = & (1-t)^3 & P_1 \\ & + 3t(1-t)^2 & P_2 \\ & + 3t^2(1-t) & P_3 \\ & + t^3 & P_4 \end{aligned}$$



That is,

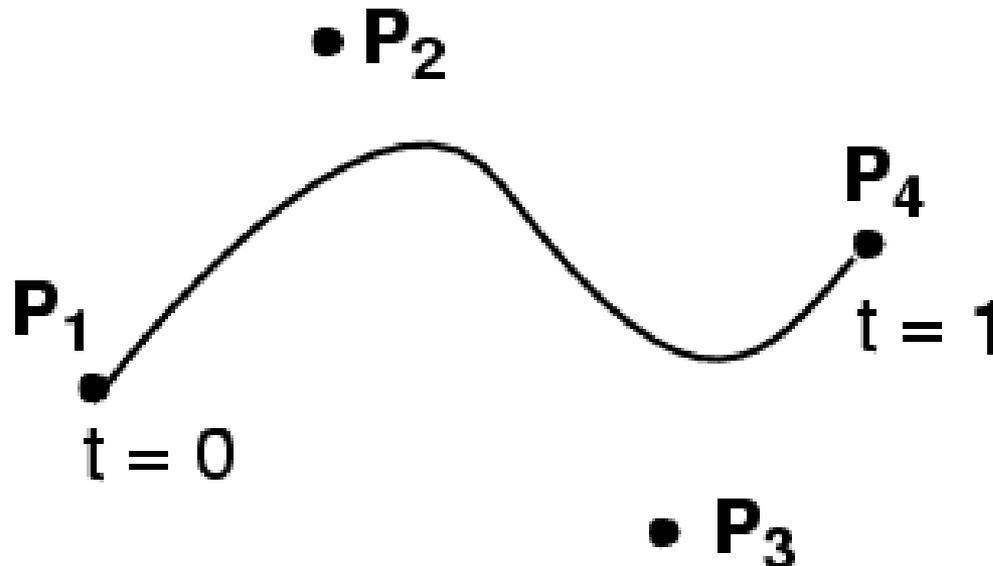
$$\begin{aligned} x(t) = & (1-t)^3 x_1 + \\ & 3t(1-t)^2 x_2 + \\ & 3t^2(1-t) x_3 + \\ & t^3 x_4 \end{aligned}$$

$$\begin{aligned} y(t) = & (1-t)^3 y_1 + \\ & 3t(1-t)^2 y_2 + \\ & 3t^2(1-t) y_3 + \\ & t^3 y_4 \end{aligned}$$

Cubic Bézier Curve

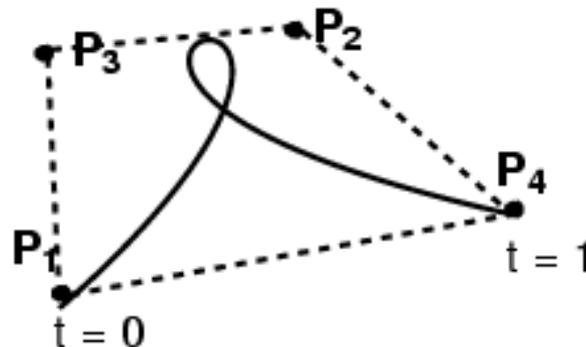
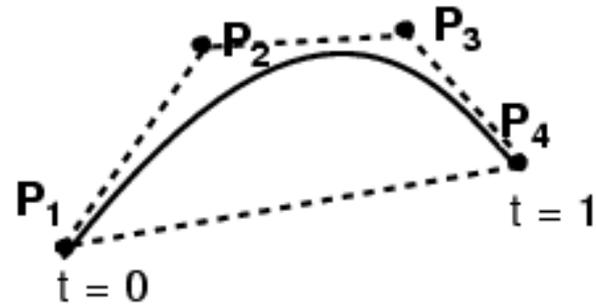
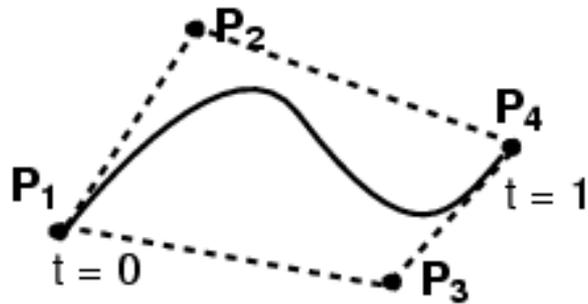
- $P(t) = (1-t)^3 \mathbf{P}_1$
+ $3t(1-t)^2 \mathbf{P}_2$
+ $3t^2(1-t) \mathbf{P}_3$
+ $t^3 \mathbf{P}_4$

Verify what happens
for $t=0$ and $t=1$



Cubic Bézier Curve

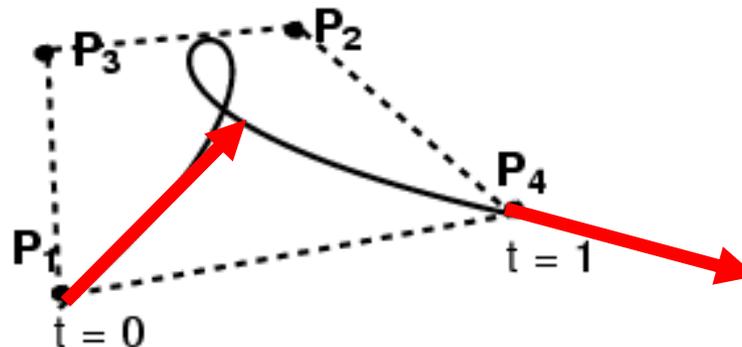
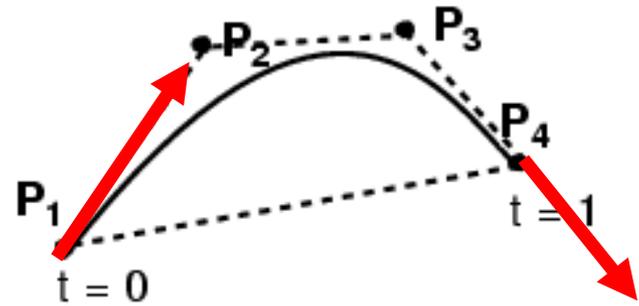
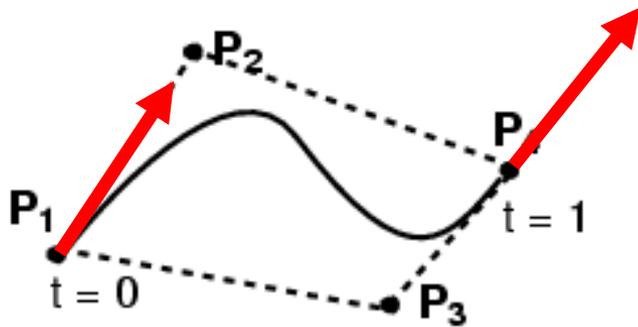
- 4 control points
- Curve passes through first & last control point



Courtesy of Seth Teller.
Used with permission.

Cubic Bézier Curve

- 4 control points
- Curve passes through first & last control point
- Curve is tangent at **P1** to **(P1-P2)** and at **P4** to **(P4-P3)**



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

Questions?

Why Does the Formula Work?

- Explanation 1:
 - Magic!
- Explanation 2:
 - These are smart weights that describe the influence of each control point
- Explanation 3:
 - It is a linear combination of *basis polynomials*.

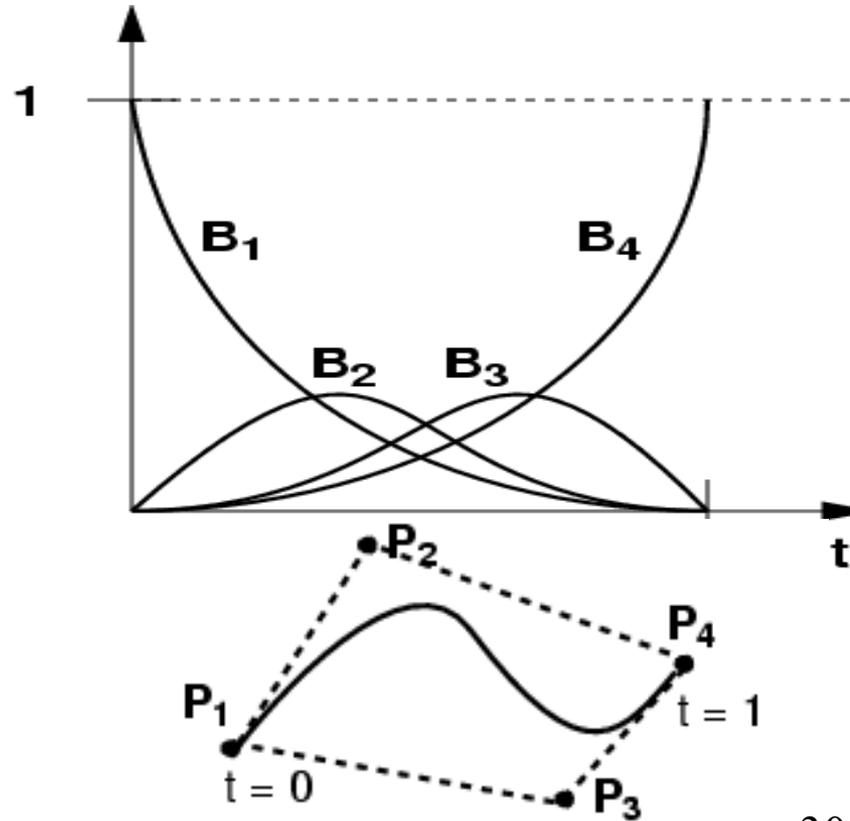
Weights

- $P(t)$ is a weighted combination of the 4 control points with weights:

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

- First, P_1 is the most influential point, then P_2 , P_3 , and P_4

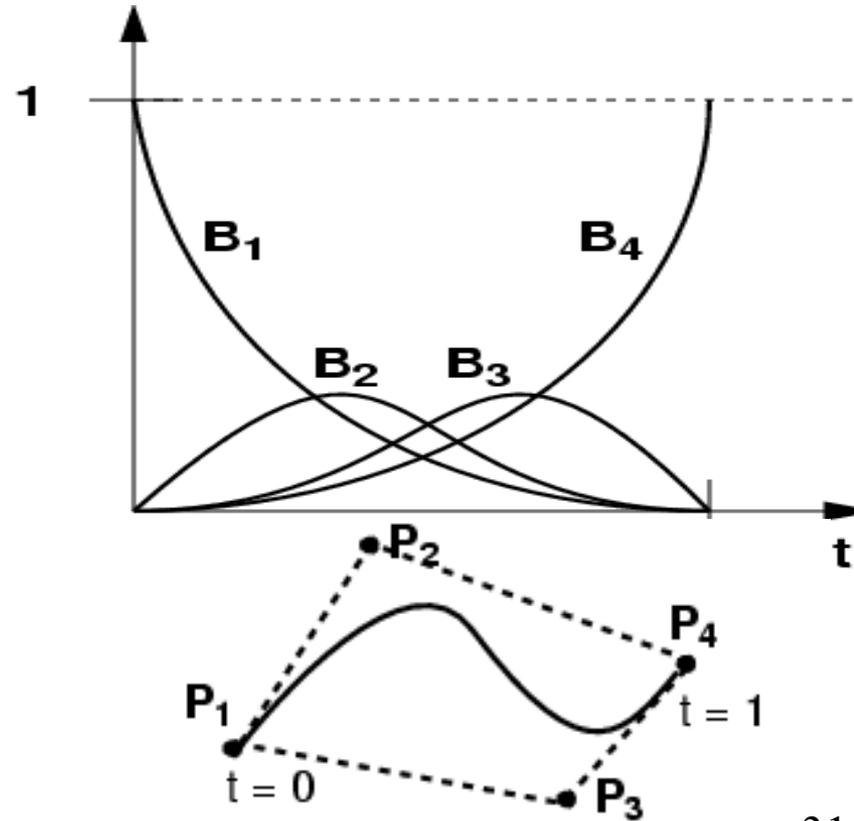
$$\begin{aligned} P(t) = & (1-t)^3 P_1 \\ & + 3t(1-t)^2 P_2 \\ & + 3t^2(1-t) P_3 \\ & + t^3 P_4 \end{aligned}$$



Weights

- P2 and P3 never have full influence
 - Not interpolated!

$$\begin{aligned} P(t) = & (1-t)^3 \quad \mathbf{P1} \\ & + 3t(1-t)^2 \quad \mathbf{P2} \\ & + 3t^2(1-t) \quad \mathbf{P3} \\ & + t^3 \quad \mathbf{P4} \end{aligned}$$



Questions?

Why Does the Formula Work?

- Explanation 1:
 - Magic!
- Explanation 2:
 - These are smart weights that describe the influence of each control point
- Explanation 3:
 - It is a linear combination of *basis polynomials*.
 - *The opposite perspective:*
control points are the weights of polynomials!!!

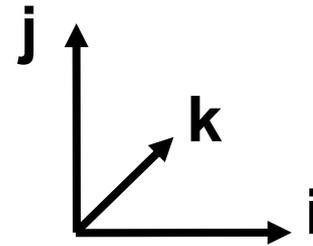
Why Study Splines as Vector Space?

- Understand relationships between types of splines
 - Conversion
- Express what happens when a spline curve is transformed by an affine transform (rotation, translation, etc.)
- Cool simple example of non-trivial vector space
- Important to understand for advanced methods such as finite elements

Usual Vector Spaces

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

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



- $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are basis vectors
- Vector addition: just add components
- Scalar multiplication: just multiply components

Polynomials as a Vector Space

- Polynomials $y(t) = a_0 + a_1t + a_2t^2 + \dots + a_nt^n$
- Can be added: just add the coefficients

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

- Can be multiplied by a scalar: multiply the coefficients

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

Polynomials as a Vector Space

- Polynomials $y(t) = a_0 + a_1t + a_2t^2 + \dots + a_nt^n$

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

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!)
- Cubic polynomials also compose a vector space
 - A 4D **subspace** of the full space of polynomials
- The x and y coordinates of cubic Bézier curves belong to this subspace as functions of t .

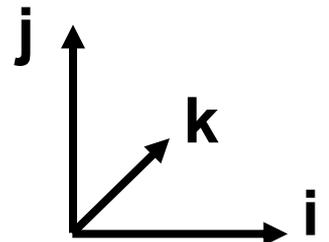
Basis for Cubic Polynomials

More precisely:
What's a basis?

- A set of “atomic” vectors
 - Called **basis vectors**
 - Linear combinations of basis vectors span the space
 - i.e. any cubic polynomial is a sum of those basis cubics
- Linearly independent
 - Means that no basis vector can be obtained from the others by linear combination
 - Example: $\mathbf{i}, \mathbf{j}, \mathbf{i}+\mathbf{j}$ is not a basis (missing \mathbf{k} direction!)

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

In 3D



Canonical Basis for Cubics

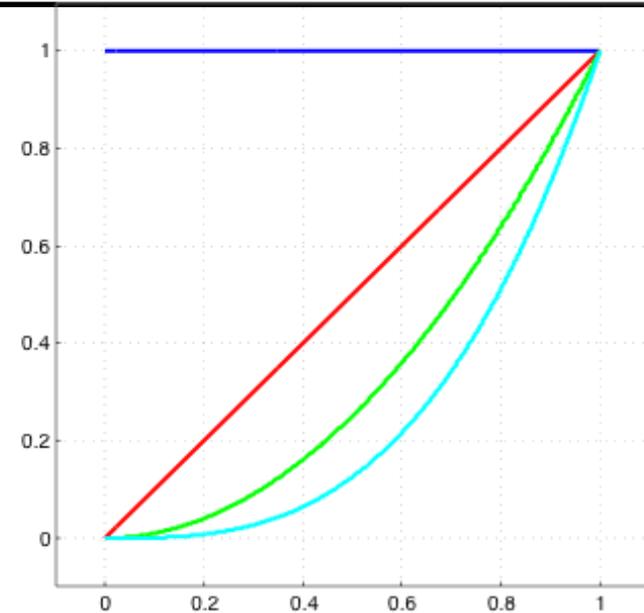
$$\{1, t, t^2, t^3\}$$

- 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 cannot write any of the four monomials as a linear combination of the others. (You can try.)

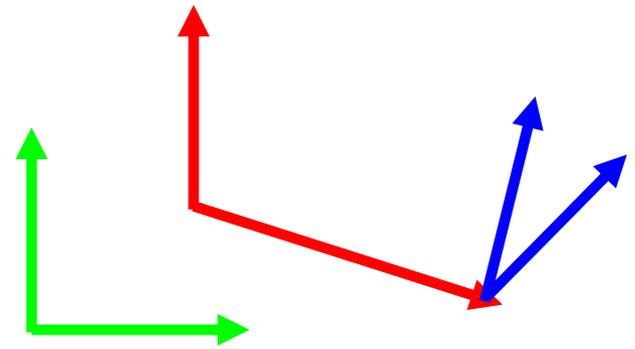
1
t
t²
t³



Different Basis

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

2D examples



- 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 \mathbb{R}^2

Matrix-Vector Notation

- 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 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

↓ ↓

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

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

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

Bernstein Polynomials

- For Bézier curves, the basis polynomials/vectors are Bernstein polynomials

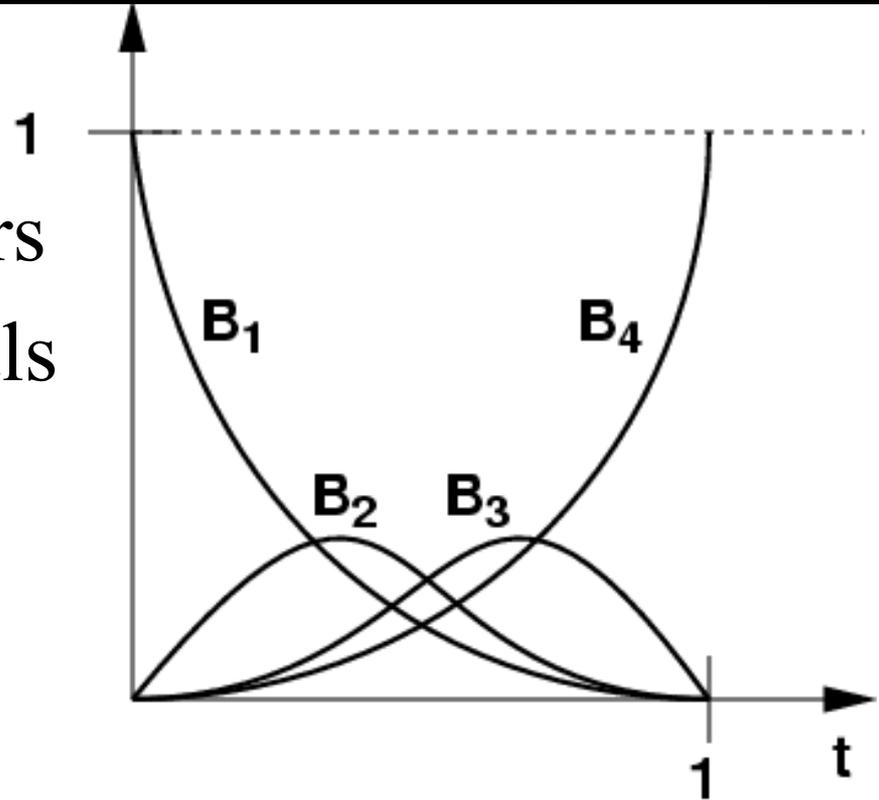
- For cubic Bezier curve:

$$B_1(t) = (1-t)^3 \quad B_2(t) = 3t(1-t)^2$$

$$B_3(t) = 3t^2(1-t) \quad B_4(t) = t^3$$

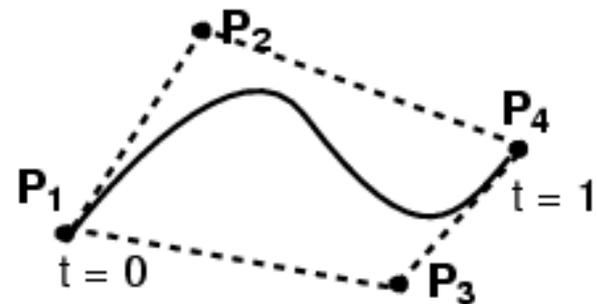
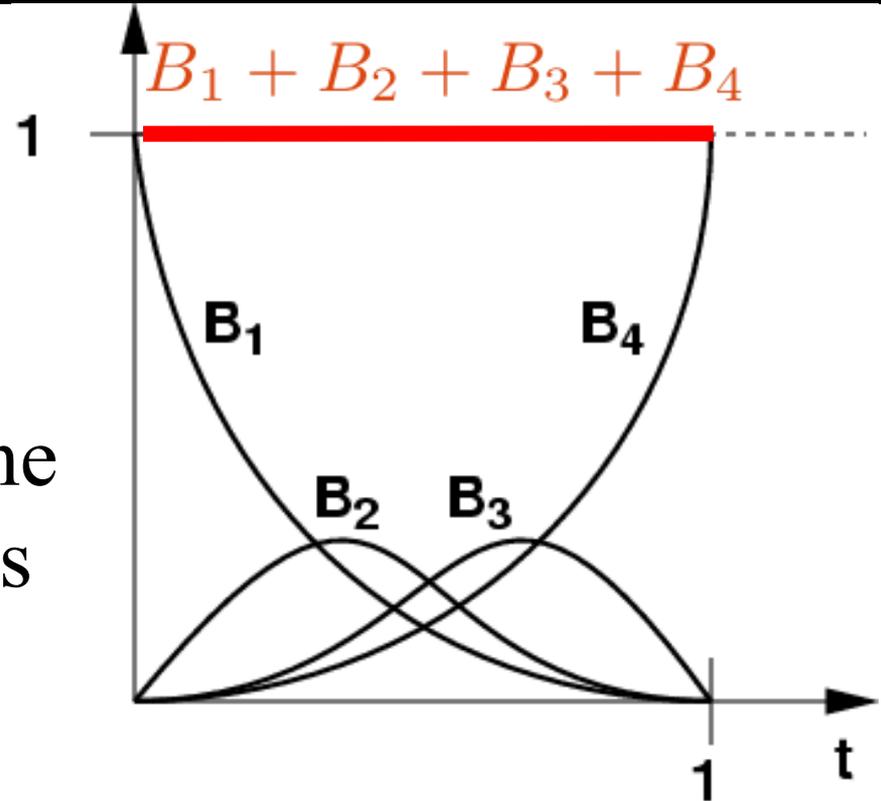
(careful with indices, many authors start at 0)

- 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
- $B_1(0) = 1$
 - Bezier curve interpolates P_1
- $B_4(1) = 1$
 - Bezier curve interpolates P_4



Bézier Curves in Bernstein Basis

- $P(t) = P_1B_1(t) + P_2B_2(t) + P_3B_3(t) + P_4B_4(t)$
 - P_i are 2D points (x_i, y_i)
- $P(t)$ is a linear combination of the control points with weights equal to Bernstein polynomials at t
- But at the same time, the control points (P_1, P_2, P_3, P_4) are the “coordinates” of the curve in the Bernstein basis
 - In this sense, specifying a Bézier curve with control points is exactly like specifying a 2D point with its x and y coordinates.

Two Different Vector Spaces!!!

- The plane where the curve lies, a 2D vector space
- The space of cubic polynomials, a 4D space
- Don't be confused!
- The 2D control points can be replaced by 3D points – this yields space curves.
 - The math stays the same, just add $z(t)$.
- The cubic basis can be extended to higher-order polynomials
 - Higher-dimensional vector space
 - More control points

Questions?

Change of Basis

- How do we go from Bernstein basis to the canonical monomial basis

$1, t, t^2, t^3$ and back?

– With a matrix!

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

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

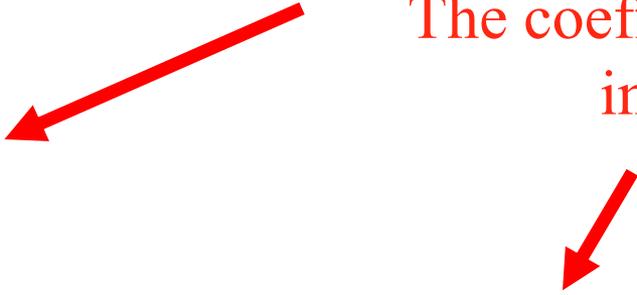
New basis vectors

How You Get the Matrix

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 !


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

Change of Basis, Other Direction

- Given $B_1 \dots B_4$, how to get back to canonical $1, t, t^2, t^3$?

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

Change of Basis, Other Direction

That's right, with the inverse matrix!

- Given $B_1 \dots B_4$, how to get back to canonical $1, t, t^2, t^3$?

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

Recap

- Cubic polynomials form a 4D 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?

More Matrix-Vector Notation

$$P(t) = \sum_{i=1}^4 P_i B_i(t) = \sum_{i=1}^4 \left[\begin{pmatrix} x_i \\ y_i \end{pmatrix} B_i(t) \right]$$

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)

Flashback

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

Cubic Bézier in Matrix Notation

point on curve
(2x1 vector)

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

“Geometry matrix”
of control points P1..P4
(2 x 4)

“Spline matrix”
(Bernstein)

Canonical
monomial basis

General Spline Formulation

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

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

Questions?

A Cubic Only Gets You So Far

- What if you want more control?

Higher-Order Bézier Curves

- > 4 control points
- Bernstein Polynomials as the basis functions
 - For polynomial of order n , the i th basis function is

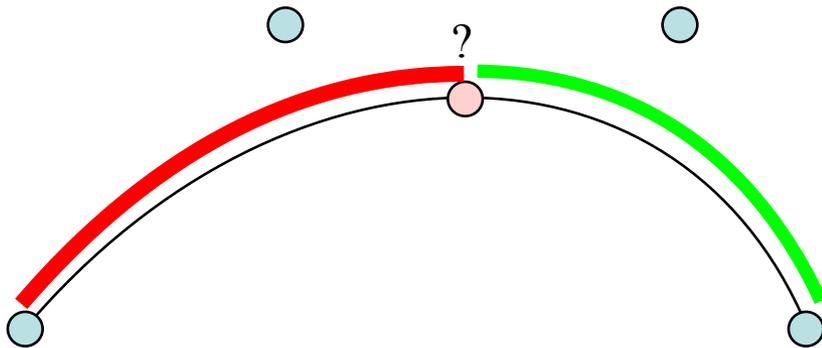
$$B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}$$

Courtesy of Seth Teller. Used with permission.

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

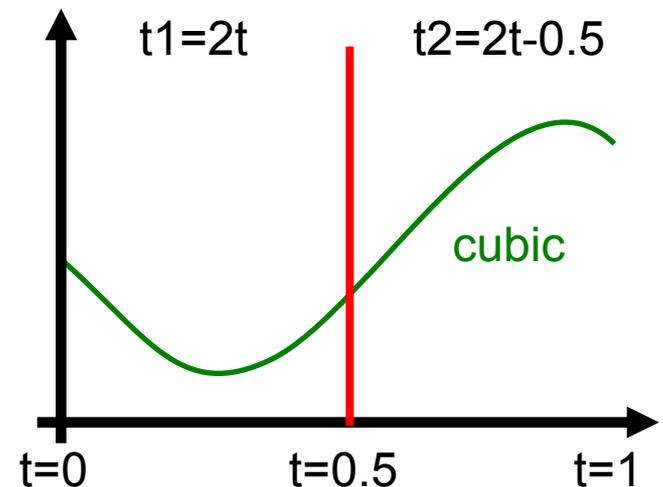
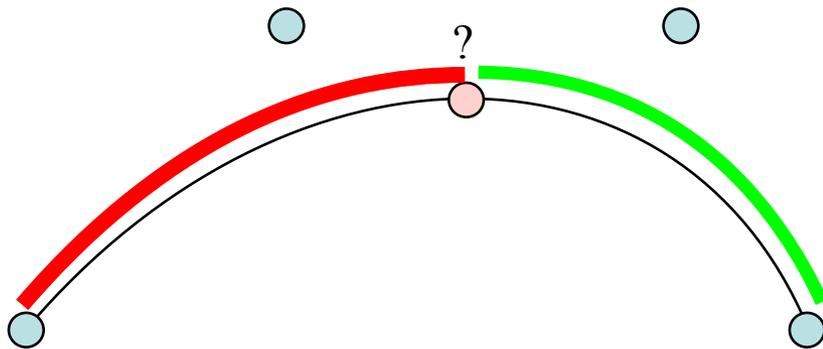
Subdivision of a Bezier Curve

- Can we split a Bezier curve in the middle into two Bézier curves?
 - This is useful for adding detail
 - It avoids using nasty higher-order curves



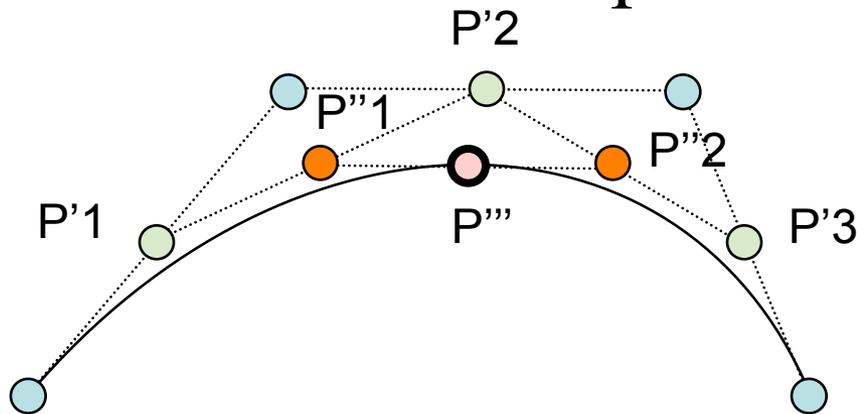
Subdivision of a Bezier Curve

- Can we split a Bezier curve in the middle into two 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!



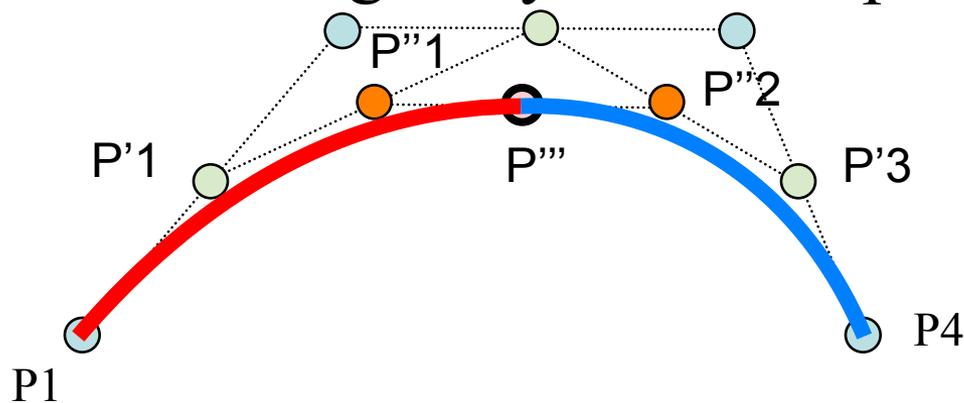
De Casteljau Construction

- 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'''



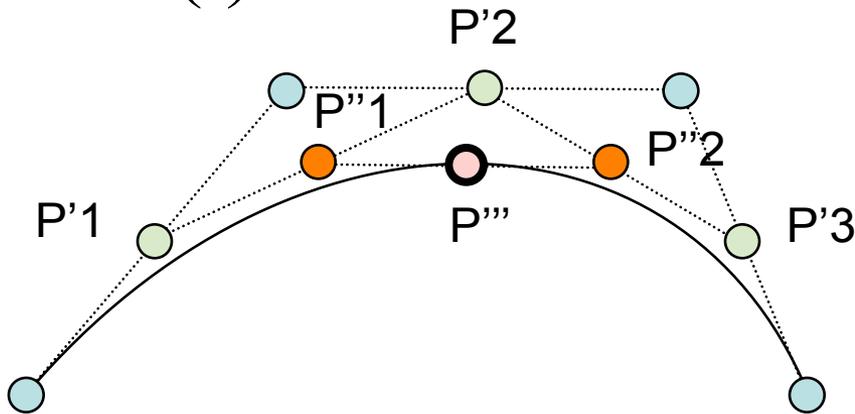
Result of Split in Middle

- The two new curves are defined by
 - $P1, P'1, P''1, \text{ and } P'''$
 - $P''' , P''2, P'3, \text{ and } P4$
- Together they exactly replicate the original curve!
 - Originally 24 control points, now 7 (more control)



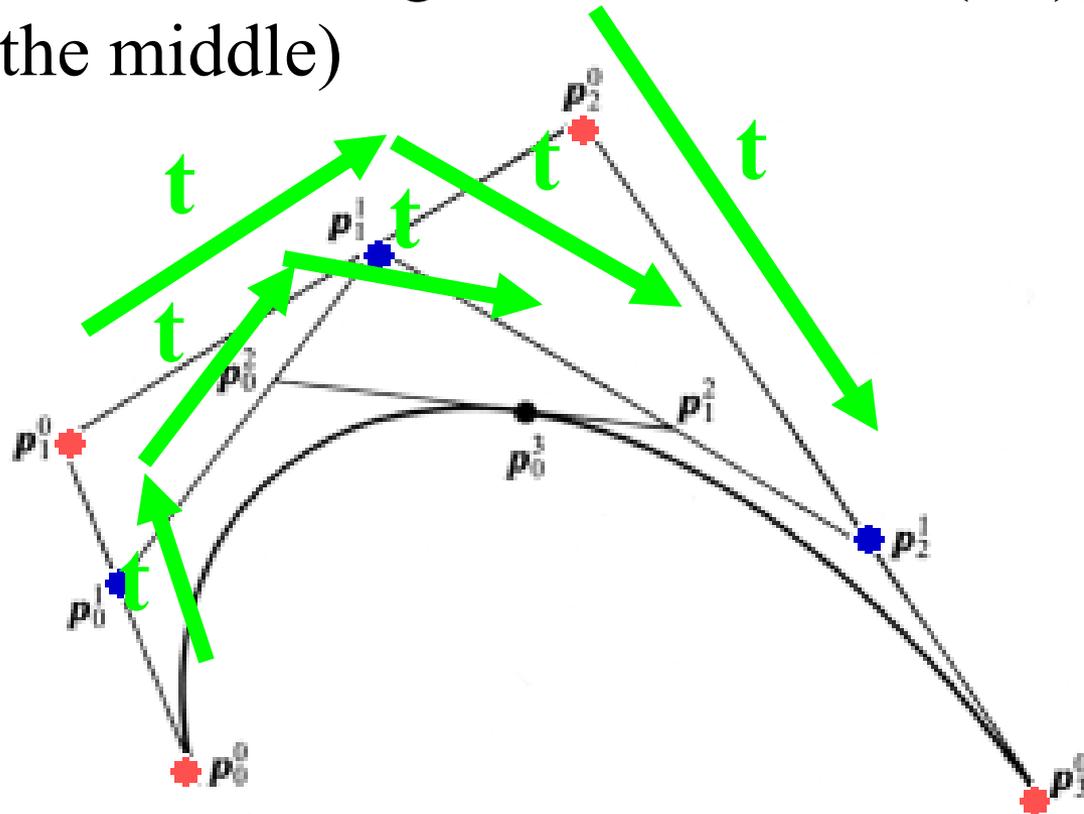
Sanity Check

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



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)



Recap

- Bezier curves: piecewise polynomials
- Bernstein polynomials
- Linear combination of basis functions
 - Basis: control points weights: polynomials
 - Basis: polynomials weights: control points
- Subdivision by de Casteljau algorithm
- All linear, matrix algebra

Recap

- Bezier curves: piecewise polynomials
- Bernstein polynomials
- Linear combination of basis functions
 - Basis: control points weights: polynomials
 - Basis: polynomials weights: control points
- Subdivision by de Casteljau algorithm
- All linear, matrix algebra

That's All for Today, Folks

- Further reading
 - Buss, Chapters 7 and 8
 - 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!

MIT OpenCourseWare
<http://ocw.mit.edu>

6.837 Computer Graphics
Fall 2012

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.