TIEA311
Tietokonegrafiikan perusteet
kevät 2018

(“Principles of Computer Graphics” – Spring 2018)

TIEA311 Tietokonegrafiikan perusteet – kevät 2018 (“Principles of Computer Graphics” – Spring 2018)

Adapted from: *Wojciech Matusik*, and *Frédo Durand*: 6.837 Computer Graphics. Fall 2012. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu/.

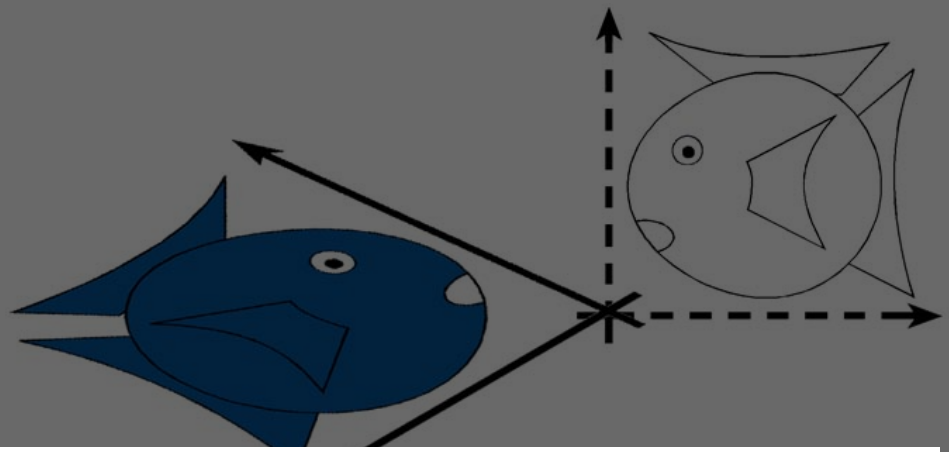
Frontpage of the local course version, held during Spring 2018 at the Faculty of Information technology, University of Jyväskylä:
`http://users.jyu.fi/~nieminen/tgp18/`

Greetings from yesterday's YPE15 (University Pedagogical Studies) – the resistance gathered to contemplate revolution:

TIEA311 teacher is now 16 hours more qualified than last time!

Linear Transformations

- $L(p + q) = L(p) + L(q)$
- $L(ap) = a\ L(p)$



Translation is not linear:

$f(p) = p+t$

$f(ap) = ap+t \neq a(p+t) = a\ f(p)$

$f(p+q) = p+q+t \neq (p+t)+(q+t) = f(p) + f(q)$

Affine space

- Points are elements of an affine space
- We denote them with a tilde $\tilde{p}$

- Affine spaces are an extension of vector spaces

Point-vector operations

- Subtracting points gives a vector

$$\tilde{p} - \tilde{q} = \vec{v}$$

- Adding a vector to a point gives a point

$$\tilde{q} + \vec{v} = \tilde{p}$$

Frames

- A frame is an origin $\tilde{o}$ plus a basis $\vec{\mathbf{b}}$
- We can obtain any point in the space by adding a vector to the origin

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b}_i$$

- using the coordinates ***c*** of the vector in $\vec{\mathbf{b}}$

Algebra notation

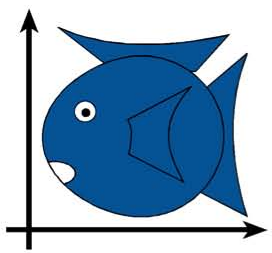
- We like matrix-vector expressions
- We want to keep track of the frame
- We're going to cheat a little for elegance and decide that 1 times a point is the point

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b}_i \quad = \quad \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} = \vec{\mathbf{f}}^t \mathbf{c}$$

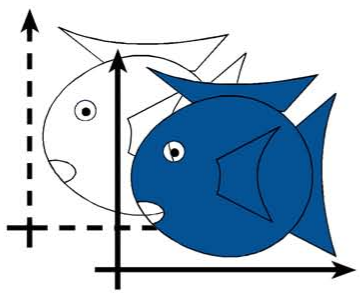
- $\tilde{p}$ is represented in $\vec{\mathbf{f}}$ by 4 coordinate, where the extra dummy coordinate is always 1 (for now)

Affine transformations

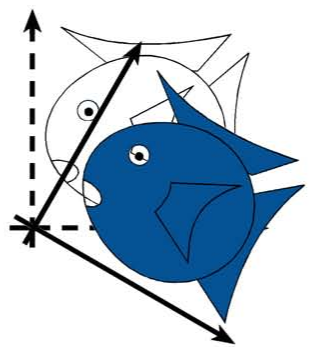
- Include all linear transformations
  - Applied to the vector basis
- Plus translation



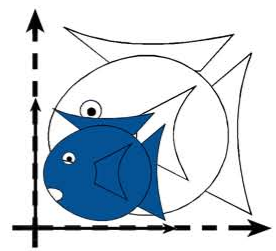
Identity



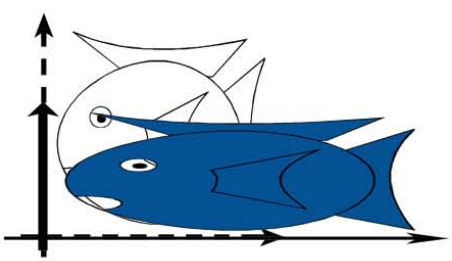
Translation



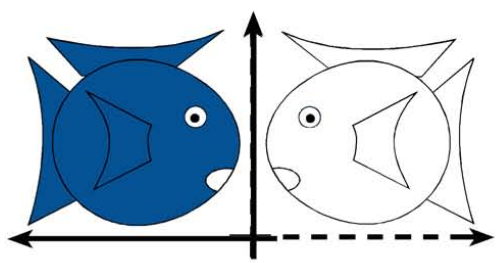
Rotation



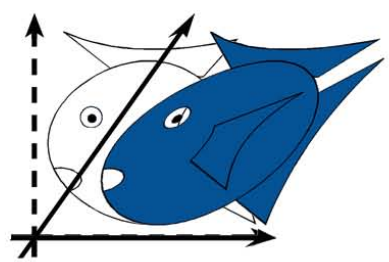
Isotropic (Uniform) Scaling



Scaling



Reflection



Shear

Matrix notation

- We know how to transform the vector basis

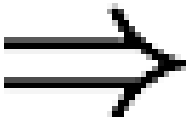
$$\begin{bmatrix} \mathcal{L}(\vec{b}_1) & \mathcal{L}(\vec{b}_2) & \mathcal{L}(\vec{b}_3) \end{bmatrix} = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix} \begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix}$$

- We will soon add translation by a vector $\vec{t}$

$$\tilde{p} \Rightarrow \tilde{p} + \vec{t}$$

Linear component

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$



$$\tilde{o} + \sum_i c_i \mathcal{L}(\vec{b_i}) = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & 0 \\ M_{21} & M_{22} & M_{23} & 0 \\ M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

- Note how we leave the fourth component alone

Translation component

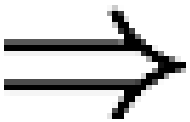
$$\tilde{p} \Rightarrow \tilde{p} + \vec{t}$$

- Express translation vector t in the basis

$$\vec{t} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} \end{bmatrix} \begin{bmatrix} M_{14} \\ M_{24} \\ M_{34} \end{bmatrix}$$

Translation

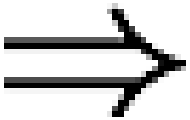
$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$



$$\tilde{o} + \vec{t} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & M_{14} \\ 0 & 1 & 0 & M_{24} \\ 0 & 0 & 1 & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

Full affine expression

$$\tilde{p} = \tilde{o} + \sum_i c_i \vec{b_i} = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$



$$\tilde{o} + \vec{t} + \sum_i c_i \mathcal{L}(\vec{b_i}) = \begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$

Which tells us both how to get a new frame ftM
or how to get the coordinates Mc after transformation

Questions?

More notation properties

- If the fourth coordinate is zero, we get a vector
- Subtracting two points:

$$\tilde{p} = \vec{f^t} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix} \qquad \tilde{p'} = \vec{f^t} \begin{bmatrix} c'_1 \\ c'_2 \\ c'_3 \\ 1 \end{bmatrix}$$

- Gives us $\tilde{p} - \tilde{p'} = \vec{f^t} \begin{bmatrix} c_1 - c'_1 \\ c_2 - c'_2 \\ c_3 - c'_3 \\ 0 \end{bmatrix}$

  a vector (last coordinate = 0)

More notation properties

- Adding a point

$$\tilde{p} = \vec{f^t} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 1 \end{bmatrix}$$ to a vector $$\vec{v} = \vec{f^t} \begin{bmatrix} c'_1 \\ c'_2 \\ c'_3 \\ 0 \end{bmatrix}$$

- Gives us $$\tilde{p} + \vec{v} = \vec{f^t} \begin{bmatrix} c_1 + c'_1 \\ c_2 + c'_2 \\ c_3 + c'_3 \\ 1 \end{bmatrix}$$

  a point (4th coordinate=1)

More notation properties

- vectors are not affected by the translation part

$$\begin{bmatrix} \vec{b_1} & \vec{b_2} & \vec{b_3} & \tilde{o} \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ 0 \end{bmatrix}$$

- because their 4th coordinate is 0
- If I rotate my moving car in the world, I want its motion to rotate
- If I translate it, motion should be unaffected

Questions?

Further Examples (Affine transforms)

Look at the implementation of the Matrix4f class in our example codes, found in the files `Matrix4f.cpp` and `Matrix4f.h`

Really, do it. (And keep doing it all the time!)

Observe how the fourth coordinate is used to implement 3D frames and affine transforms of points. And linear transforms of directions. A straightforward way to do many things is to build a proper 4x4 matrix and then multiply. **Not much code**, actually!

In Assignment 1 you will **avoid a lot of tears** by figuring out how (and when and why) to use the provided constructors and the operator $*$ to transform points and frames suitably.

Also, this stuff **is a key thing** in **computer graphics**!

TIEA311 - Status on Lecture 5 (Jan 25, 2018):

This year we started with linear algebra, vectors, and matrices. At this point you will have used your power tools (brain, pen, and paper) to get a hang of it – **haven't you?!** Now we come back to geometry and curves. Some of it was covered on the first lecture, so we'll have a very fast recap and go forward.

The following slides are reproduced verbatim from the MIT course. . .

Warning: subscripts and superscripts are occasionally lost in translation, $P2$ should be P_2, $t3$ should be t^3 etc.

Resolution: Use your power tools! Rewrite the equations nicely – you'll understand more with your power tools anyway.

Remember: If the slide says "verify" (and even if it doesn't) you should keep scribbling until you understand!

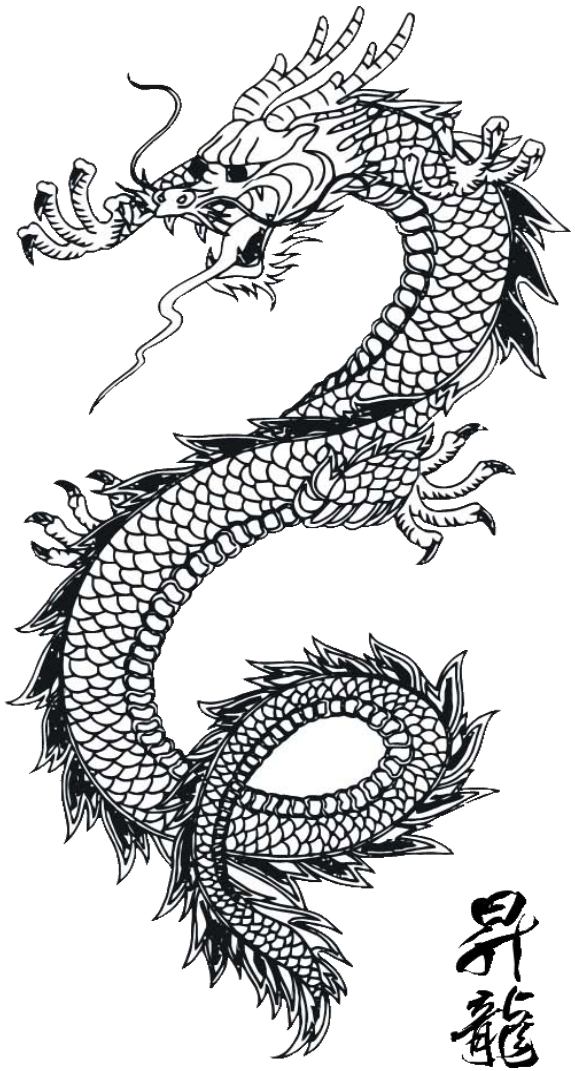
6.837 Computer Graphics

Bézier Curves and Splines

Wojciech Matusik
MIT CSAIL

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

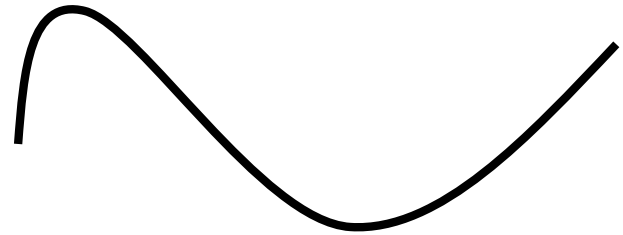
Modeling 1D Curves in 2D

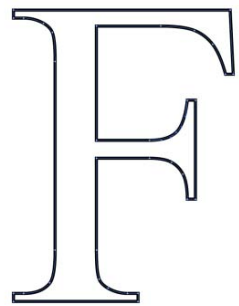
- Polylines
  - Sequence of vertices connected by straight line segments
  - Useful, but not for smooth curves
  - This is the representation that usually gets drawn in the end (a curve is converted into a polyline)

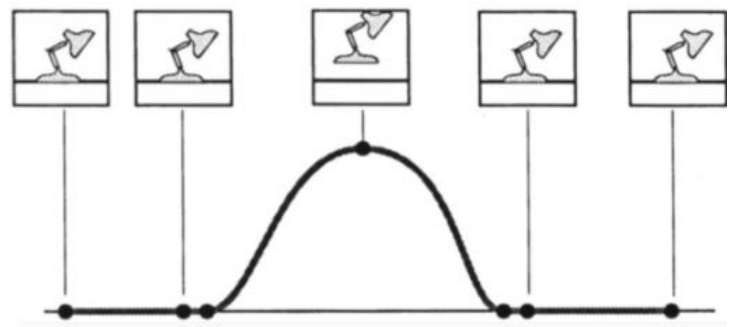
- **Smooth curves**
  - How do we specify them?
  - A little harder (but not too much)

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



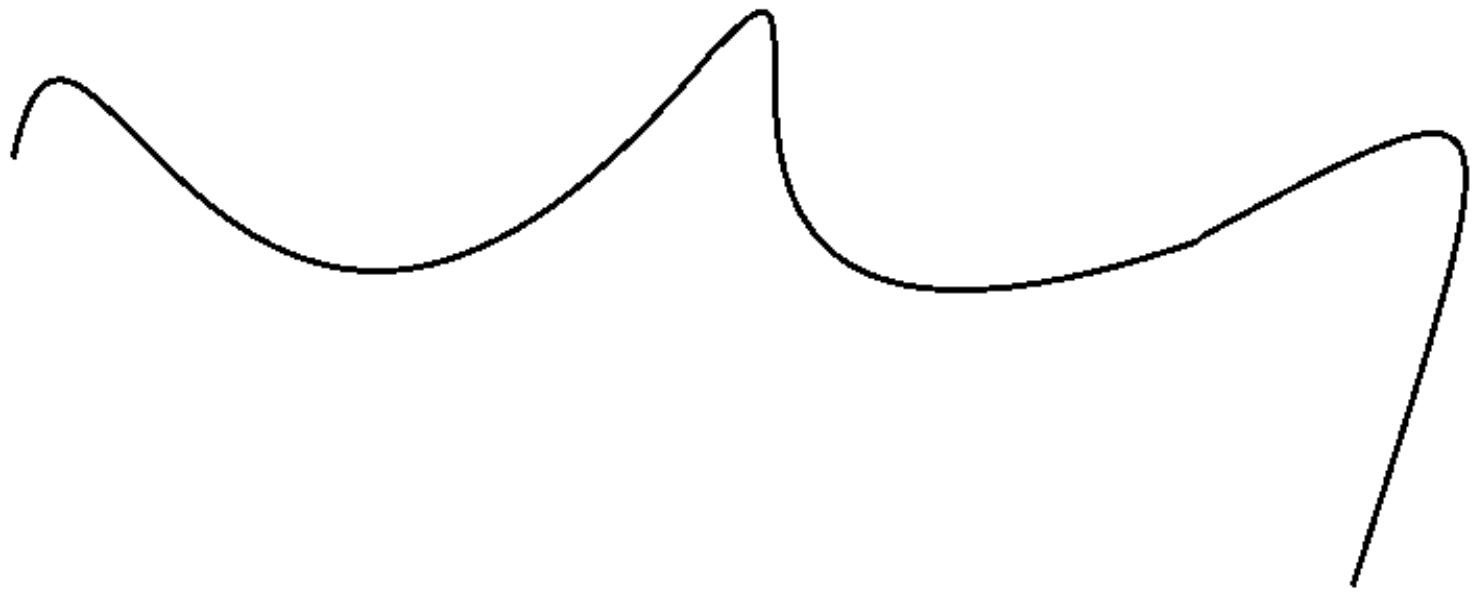




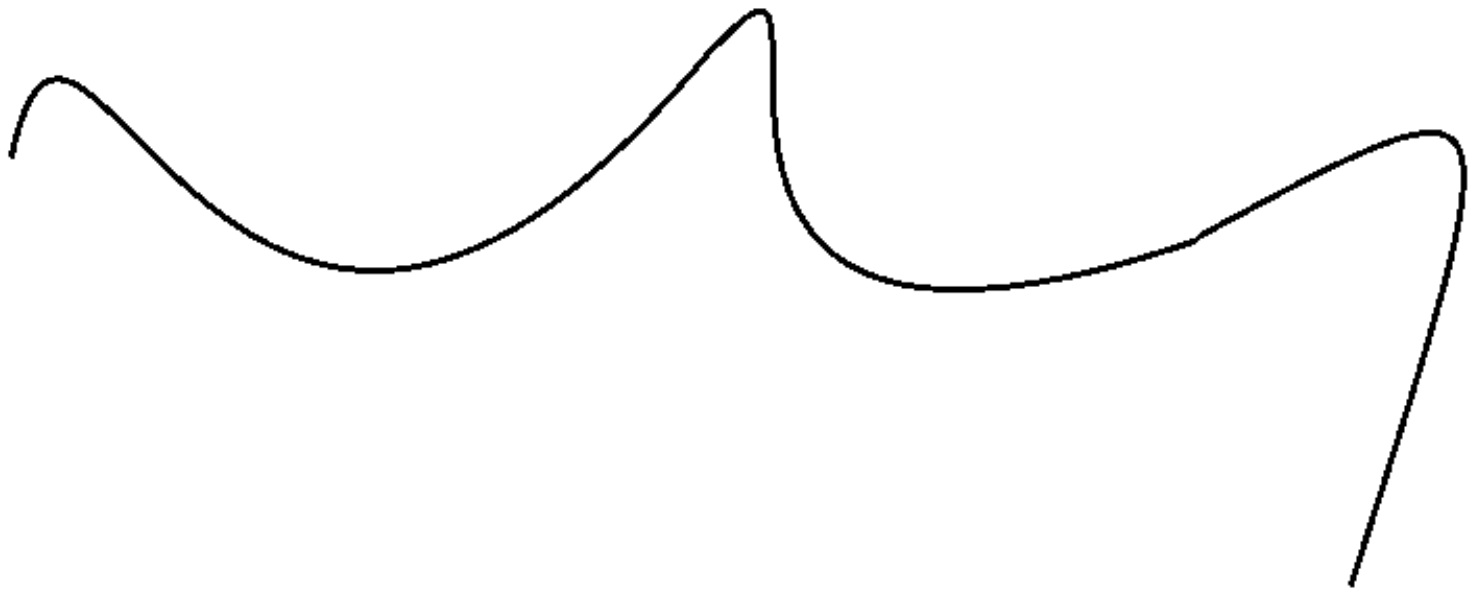
ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

Demo

How Many Dimensions?

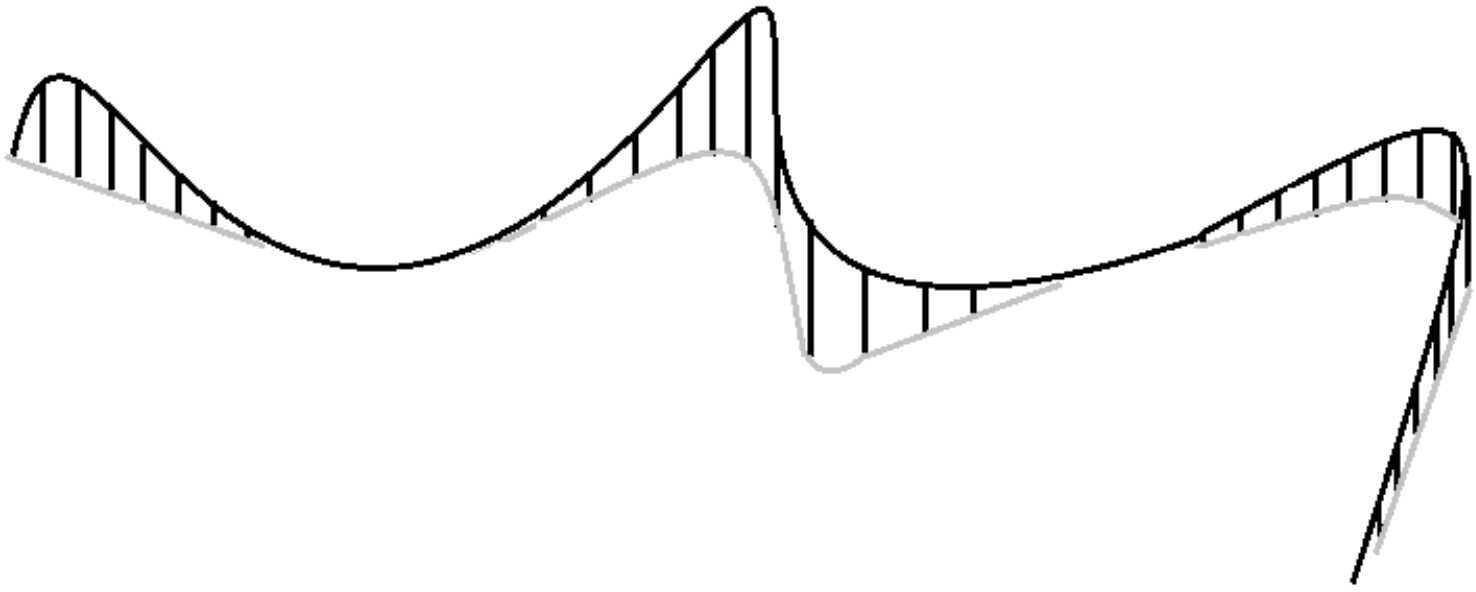


How Many Dimensions?



**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

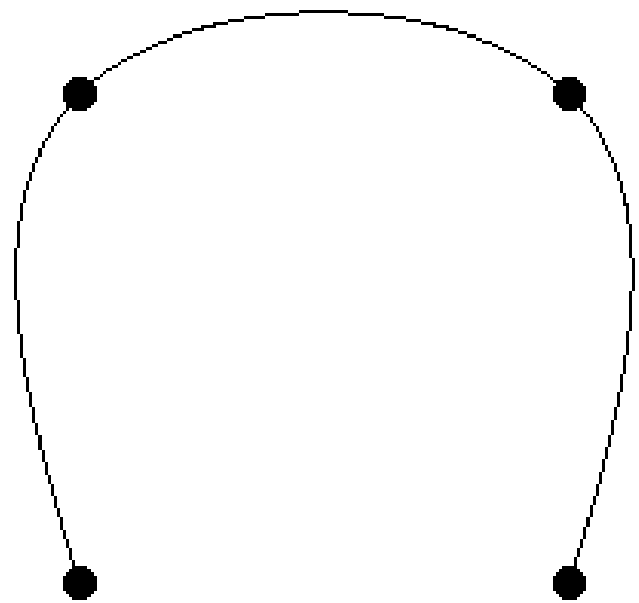
- A continuous 1D set of points in 2D (or 3D)
- A mapping from an interval S onto the plane
  - That is, *P(t)* is the point of the curve at parameter *t*

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

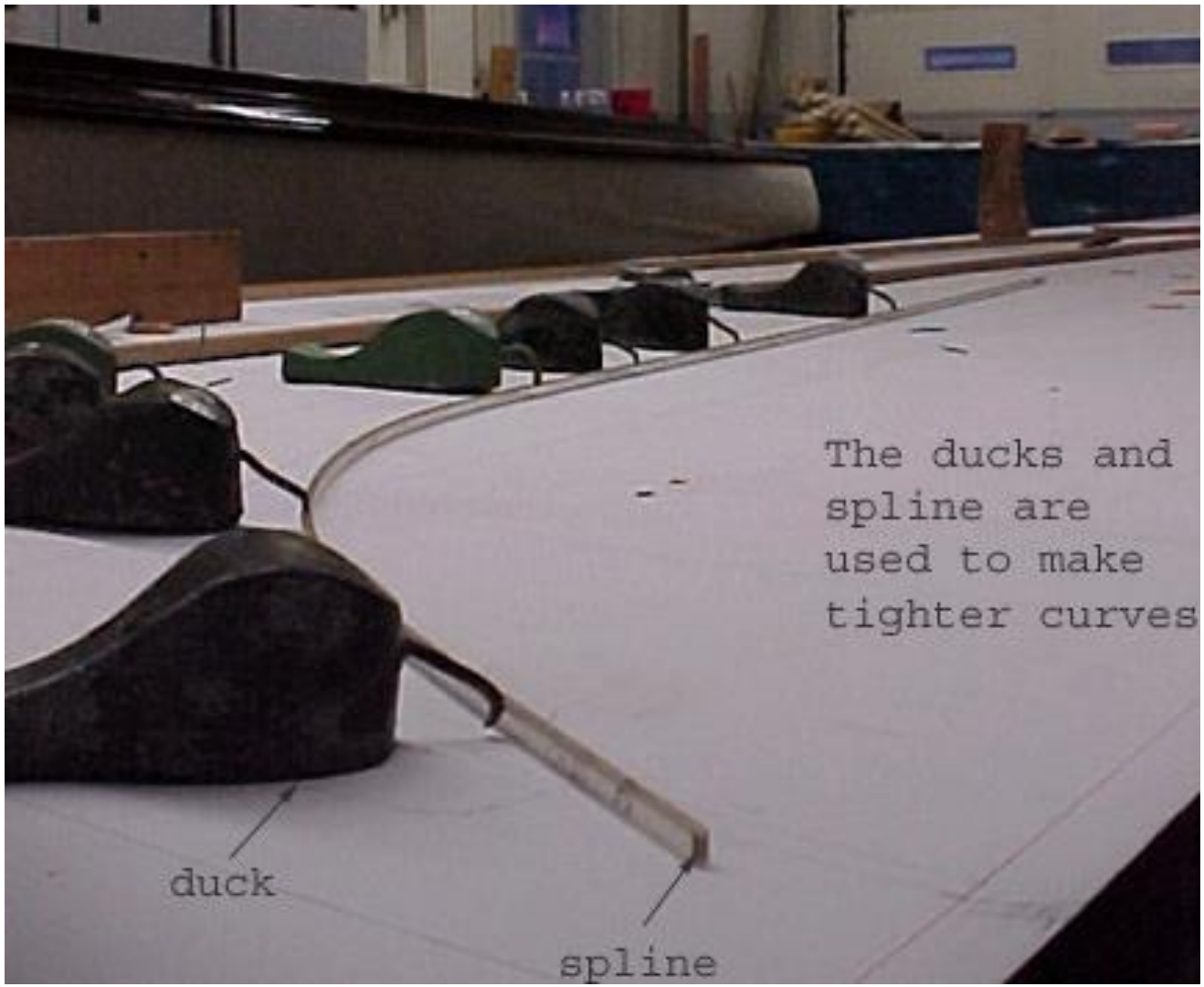
- Big differences
  - It is easy to generate points on the curve from the 2nd
  - The second definition can describe trajectories, the speed at which we move on the curve

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



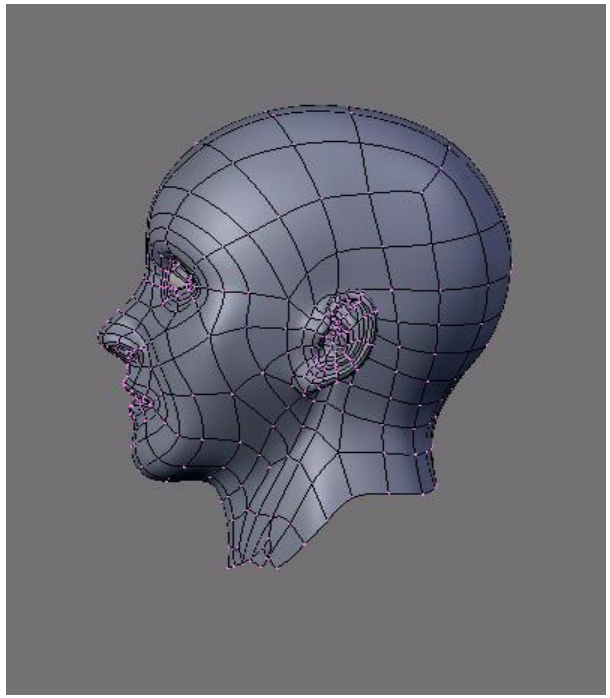
Courtesy of The Antique Boat Museum.

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

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.



Questions?

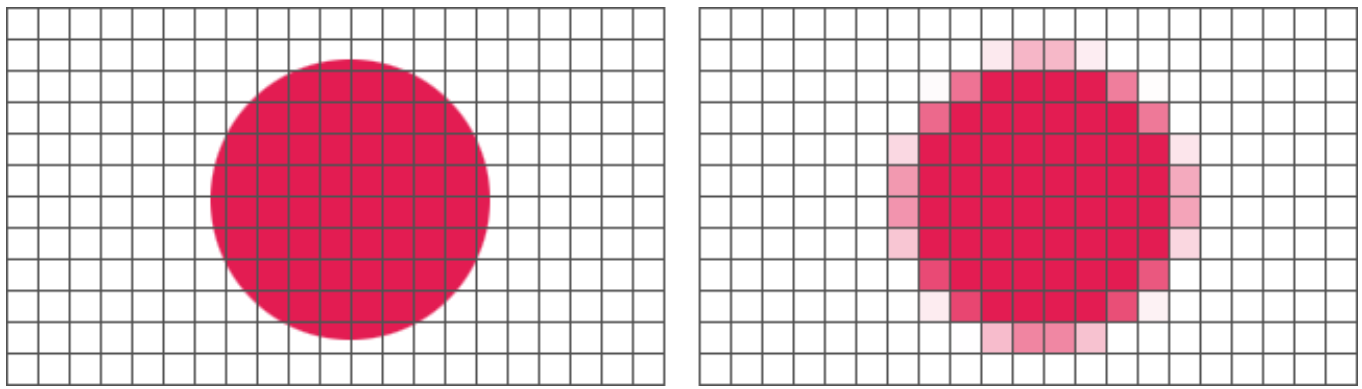
Splines

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

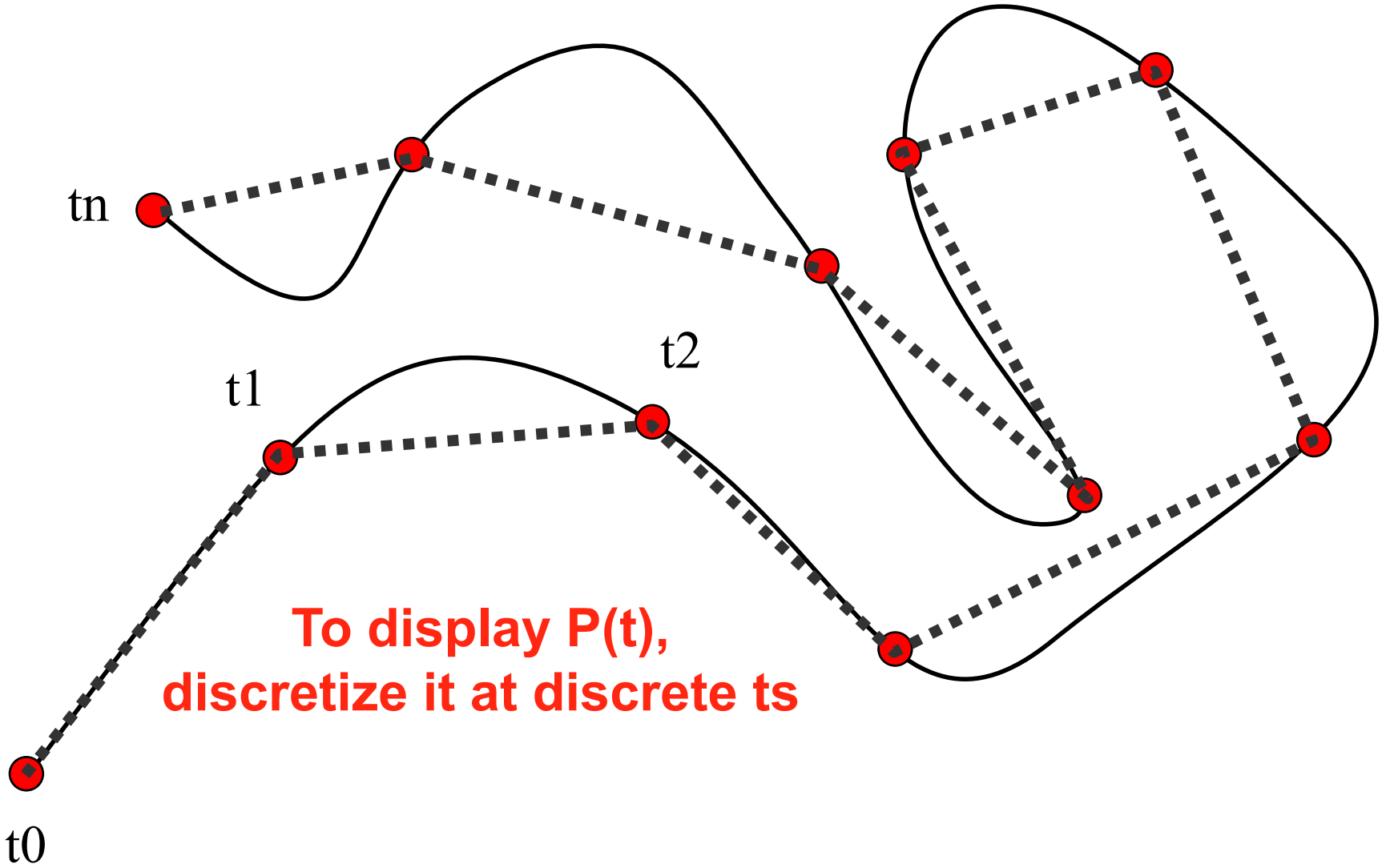
- Results in a smooth parametric curve P(t)
  - Just means that we specify $x(t)$ and $y(t)$
  - In practice: low-order polynomials, chained together
  - Convenient for animation, where t is time
  - Convenient for *tessellation* because we can discretize t and approximate the curve with a polyline

Tessellation

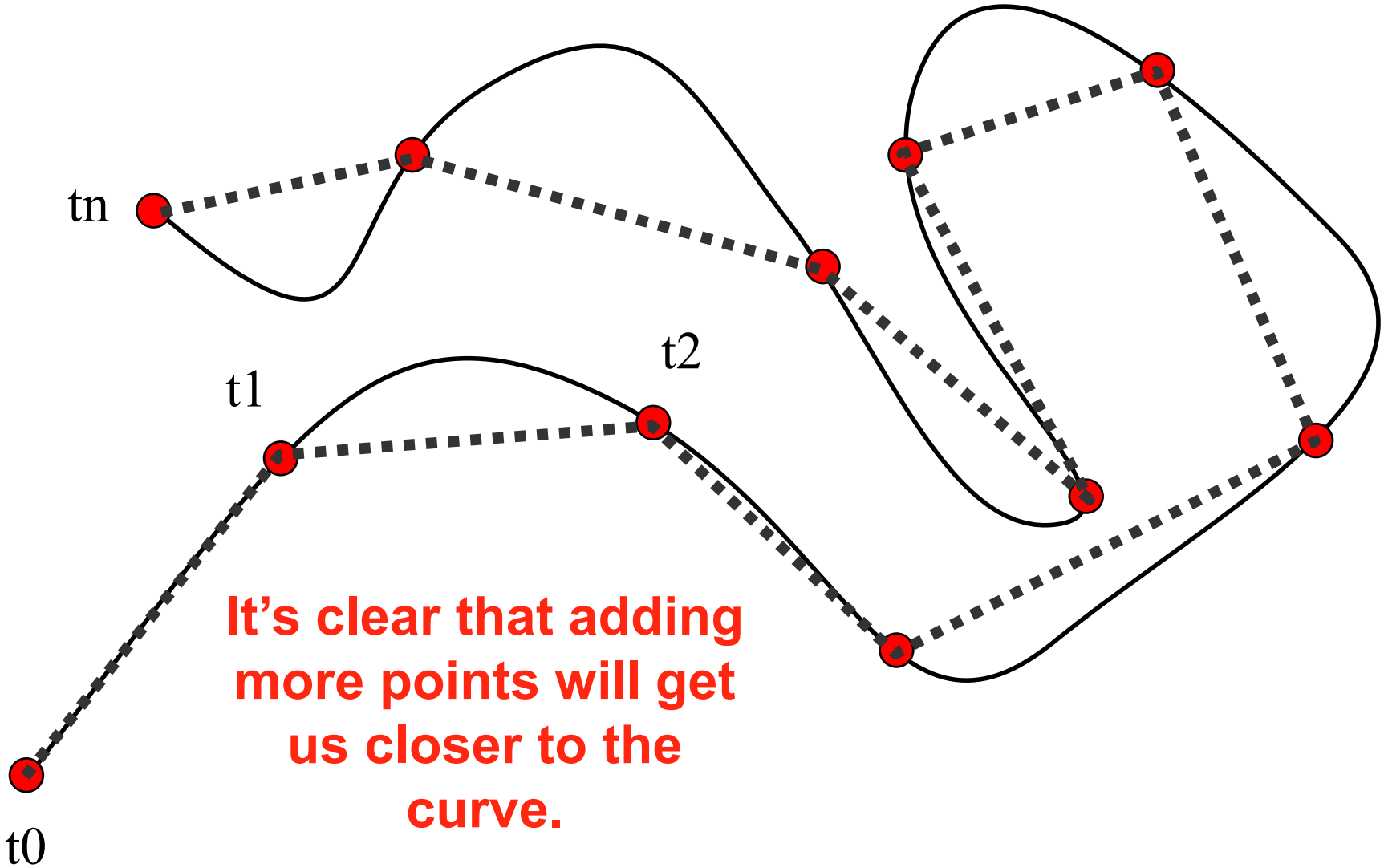
- It is easy to rasterize mathematical line segments into pixels
  - OpenGL and the graphics hardware can do it for you
- But polynomials and other parametric functions are harder



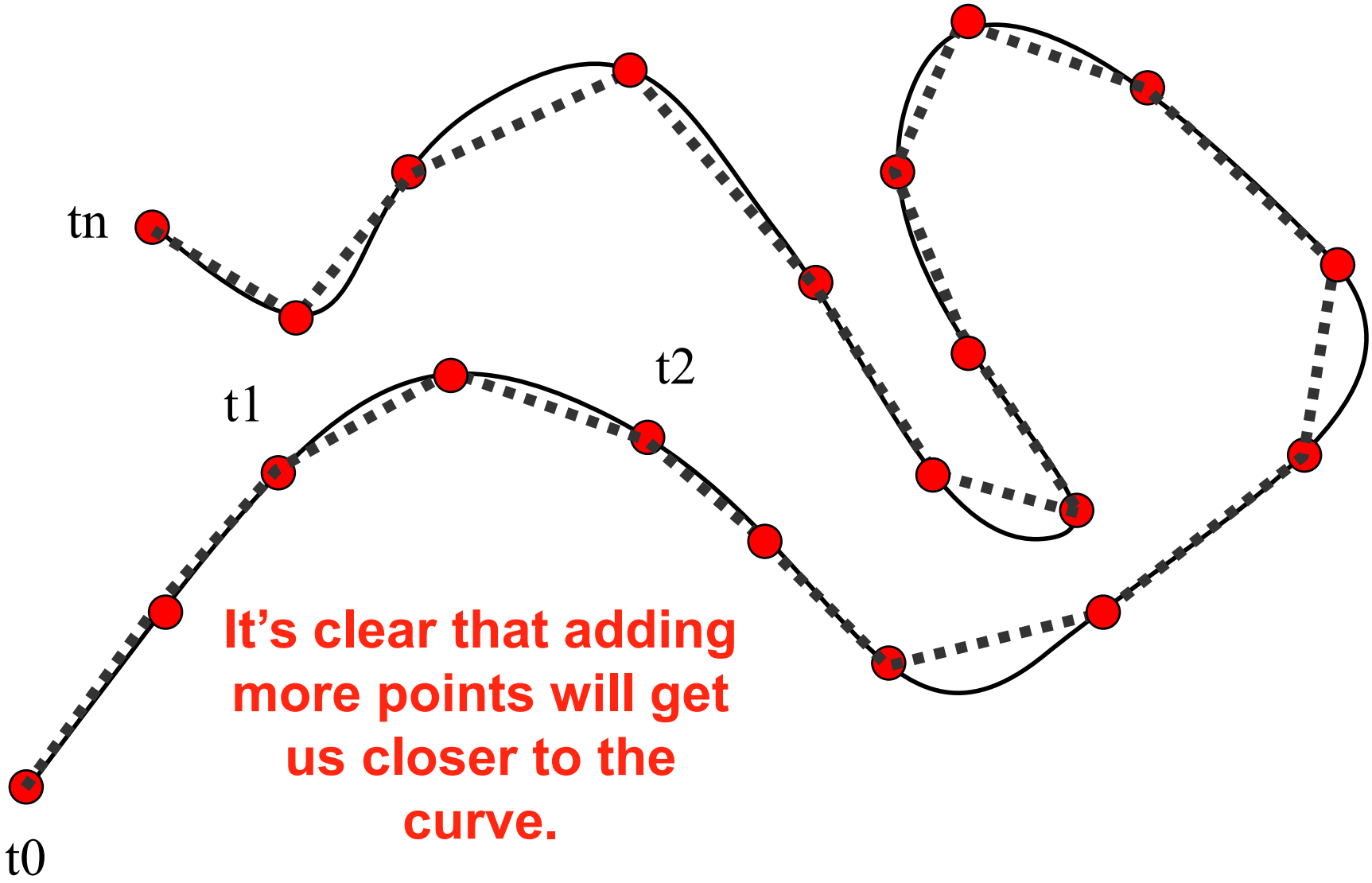
Tessellation



Tessellation

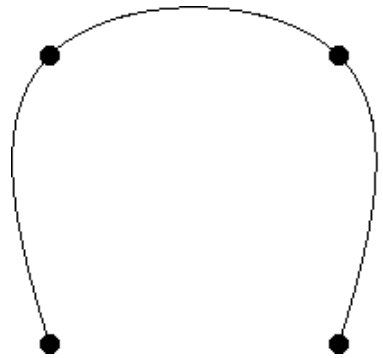


Tessellation



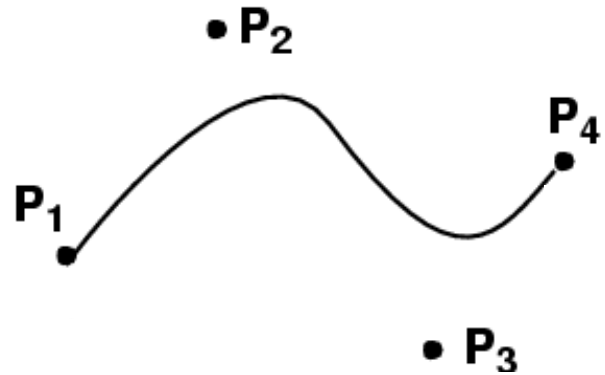
Interpolation vs. Approximation

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



Interpolation

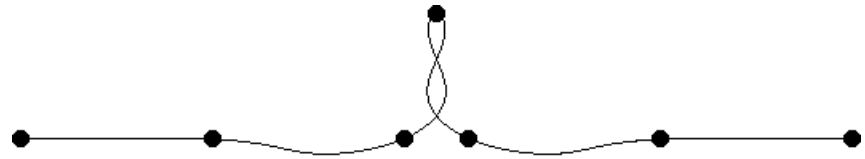
- Approximation
  - Does not go through all points



Approximation

Interpolation vs. Approximation

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



Interpolation

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

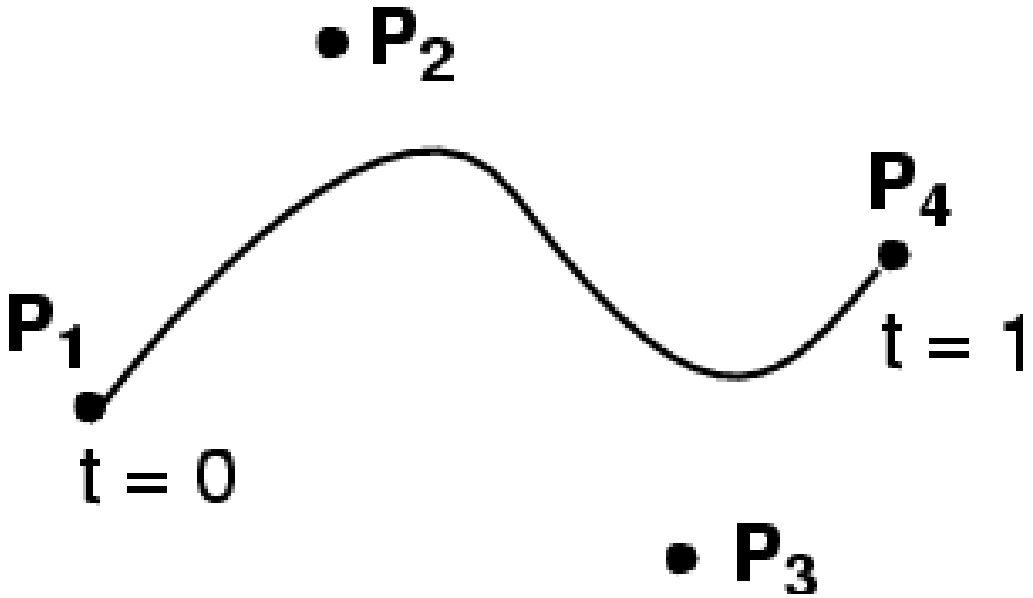
Approximation

- We will do something in between.

Questions?

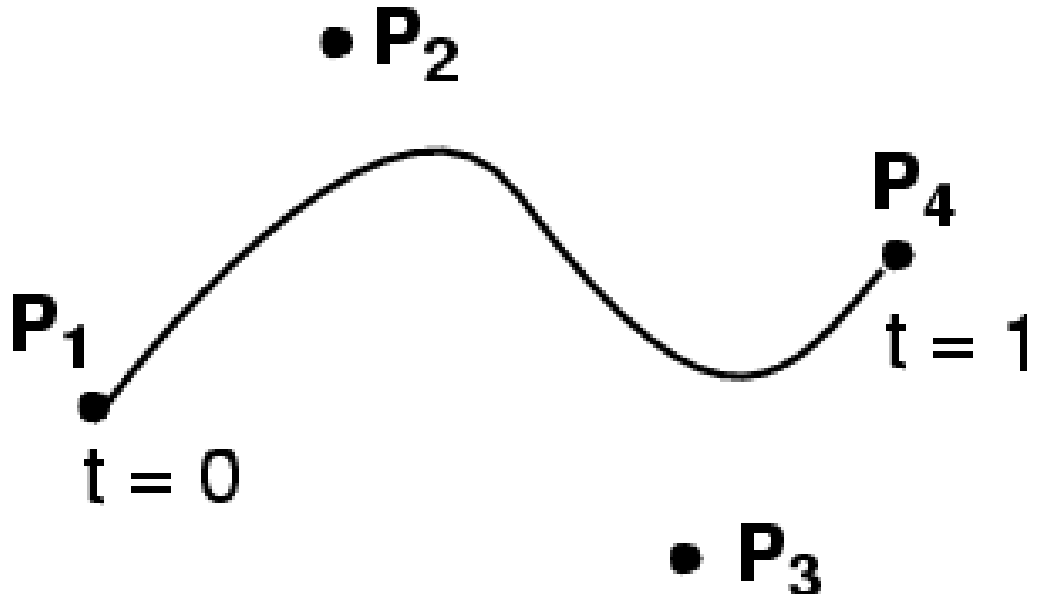
Cubic Bézier Curve

- User specifies 4 control points **P**1 ... **P**4
- Curve goes through (interpolates) the ends **P**1, **P**4
- Approximates the two other ones
- Cubic polynomial



Cubic Bézier Curve

- $P(t) = (1-t)^3 \quad \mathbf{P}1$
  $+ \quad 3t(1-t)^2 \quad \mathbf{P}2$
  $+ \quad 3t^2(1-t) \quad \mathbf{P}3$
  $+ \quad t^3 \quad \mathbf{P}4$



That is,

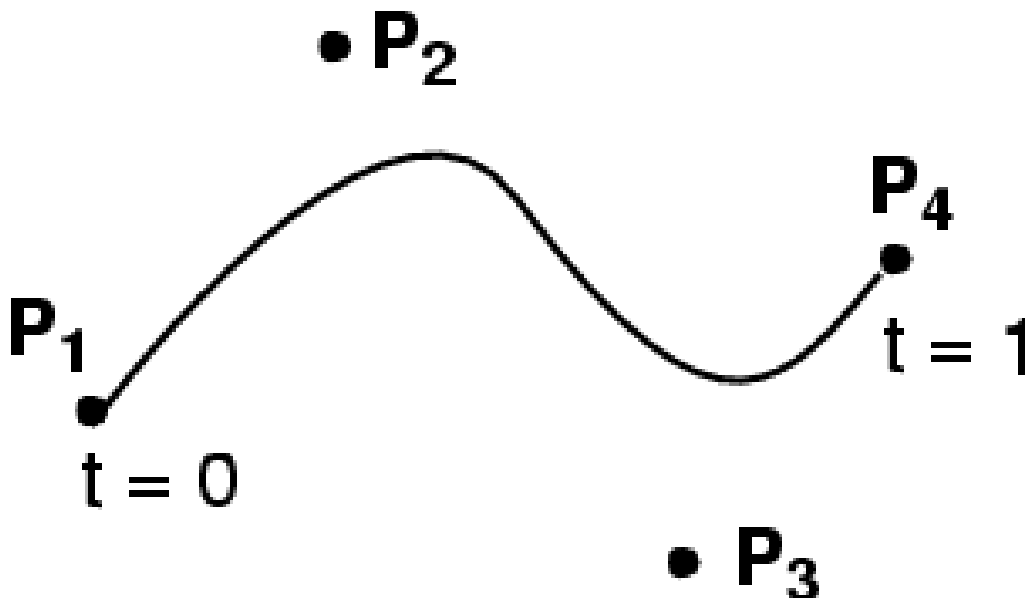
$$x(t) = (1-t)^3\, x_1 + 3t(1-t)^2\, x_2 + 3t^2(1-t)\, x_3 + t^3\, x_4$$

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

Cubic Bézier Curve

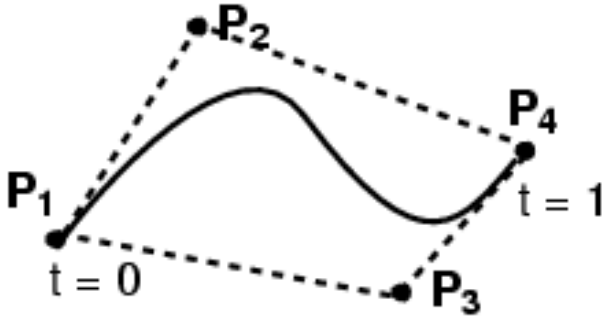
- $P(t) = (1-t)^3 \quad \mathbf{P}1$
  $+ \quad 3t(1-t)^2 \quad \mathbf{P}2$
  $+ \quad 3t^2(1-t) \quad \mathbf{P}3$
  $+ \quad t^3 \quad \mathbf{P}4$

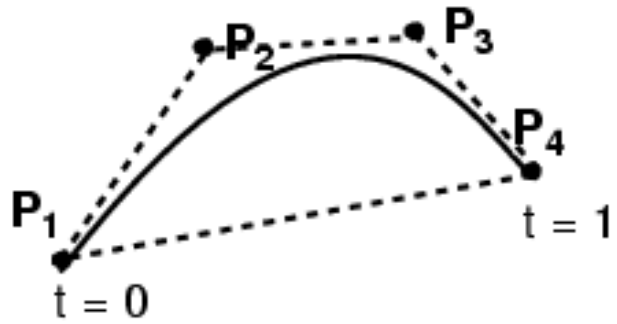
Verify what happens for t=0 and t=1

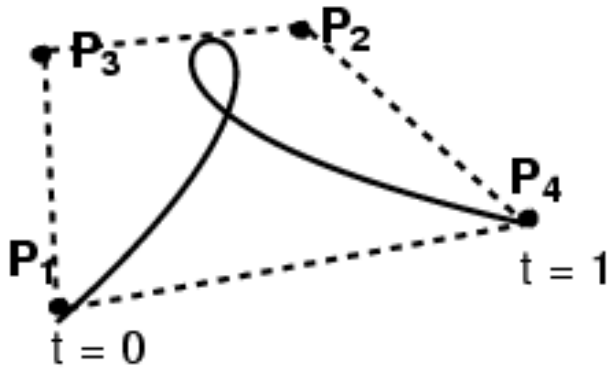


Cubic Bézier Curve

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



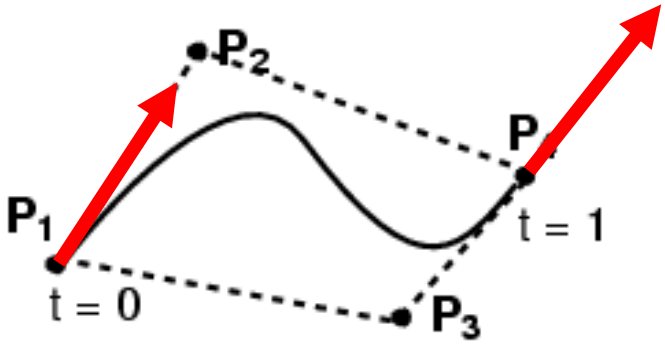


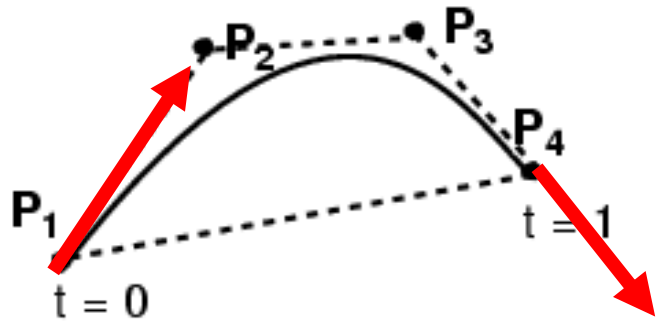


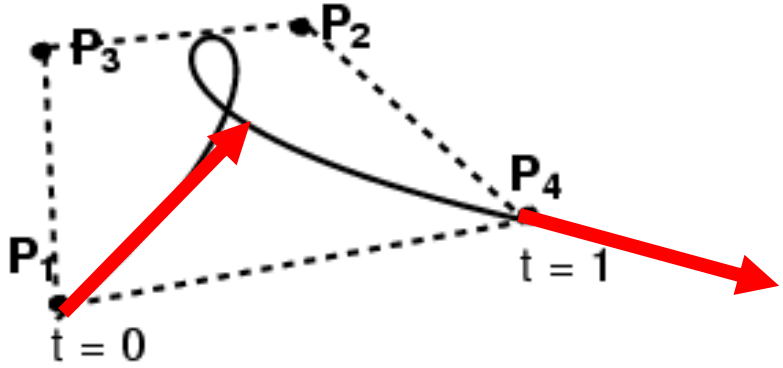
Courtesy of Seth Teller.
Used with permission.

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?

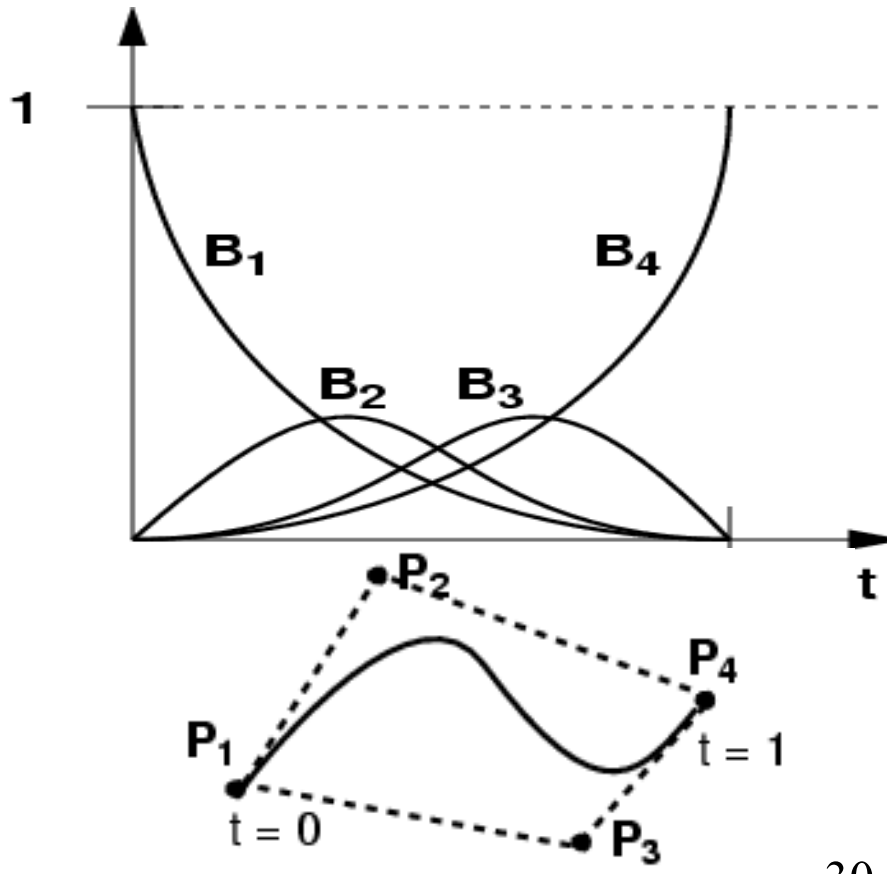
Why Does the Formula Work?

- 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:
  - $B1(t)=(1-t)^3$
  - $B2(t)=3t(1-t)^2$
  - $B3(t)=3t^2(1-t)$
  - $B4(t)=t^3$
- First, P1 is the most influential point, then P2, P3, and P4

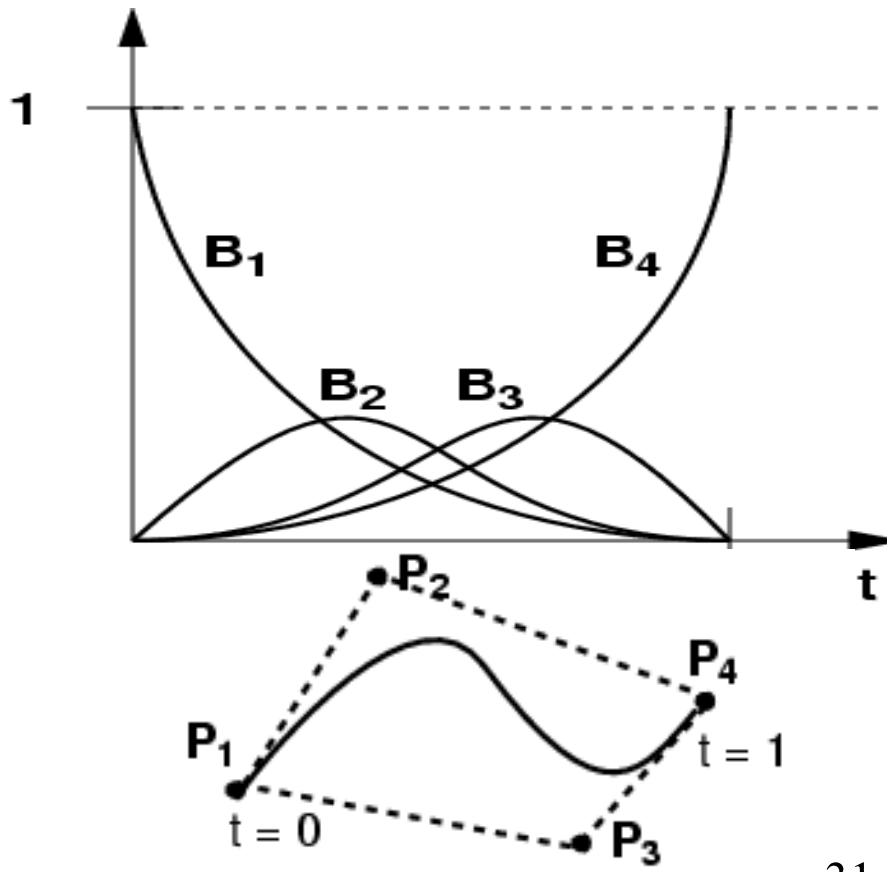
$$P(t) = (1-t)^3\,\mathbf{P}1 + 3t(1-t)^2\,\mathbf{P}2 + 3t^2(1-t)\,\mathbf{P}3 + t^3\,\mathbf{P}4$$



Weights

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

$$P(t) = (1-t)^3\,\mathbf{P}1 + 3t(1-t)^2\,\mathbf{P}2 + 3t^2(1-t)\,\mathbf{P}3 + t^3\,\mathbf{P}4$$



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!!!

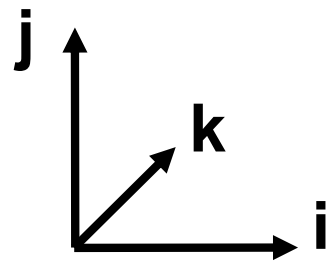
Why Study Splines as Vector Space?

- 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

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



- **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

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

- In the polynomial vector space, {1, *t*, ..., *t*n} are the basis vectors, *a*0, *a*1, ..., *a*n are the components

Questions?

Subset of Polynomials: Cubic

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

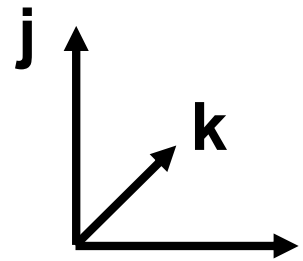
- Closed under addition & scalar multiplication
  - Means the result is still a cubic polynomial (verify!)
- Cubic polynomials also compose a vector space
  - A 4D **subspace** of the full space of polynomials
- The *x* and *y* coordinates of cubic Bézier curves belong to this subspace as functions of *t*.

Basis for Cubic Polynomials

More precisely:
What's a basis?

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

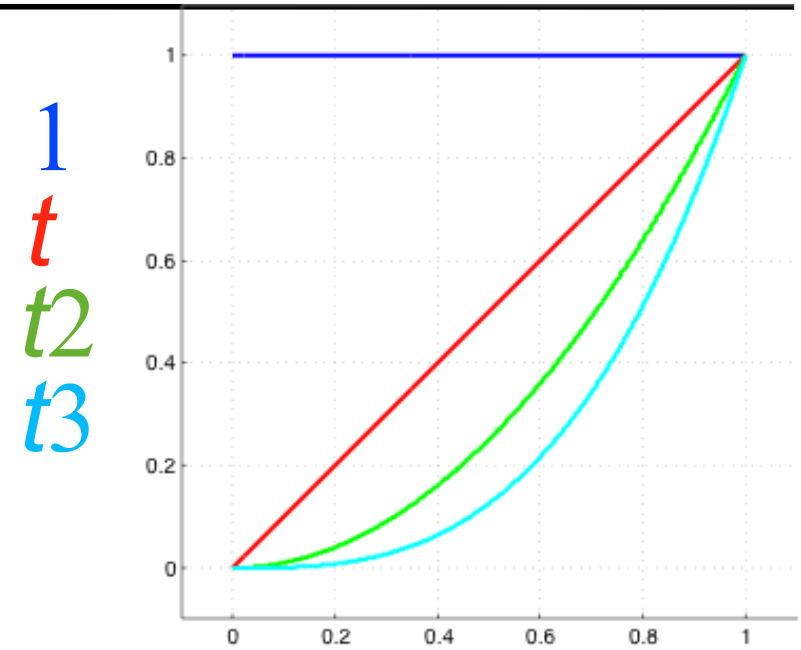
In 3D



- A set of "atomic" vectors
  - Called **basis vectors**
  - Linear combinations of basis vectors span the space
    - 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: **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:

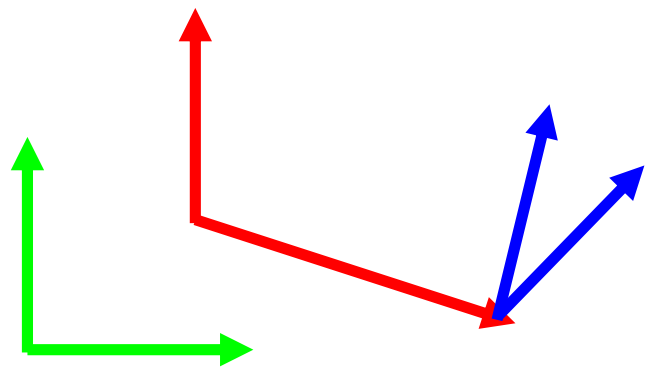
$a0+a1t+a2t2+a3t3 = a0*1+a1*t+a2*t2+a3*t3$

- They are linearly independent
  - Means you cannot write any of the four monomials as a linear combination of the others. (You can try.)

Different Basis

- For example:
  - {*1*, *1+t*, *1+t+t2*, *1+t-t2+t3*}
  - {*t3*, *t3+t2*, *t3+t*, *t3+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 R2

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+t2, 1+t-t2+t3

t3, t3+t2, t3+t, t3+1

Change-of-basis matrix

"Canonical" monomial basis

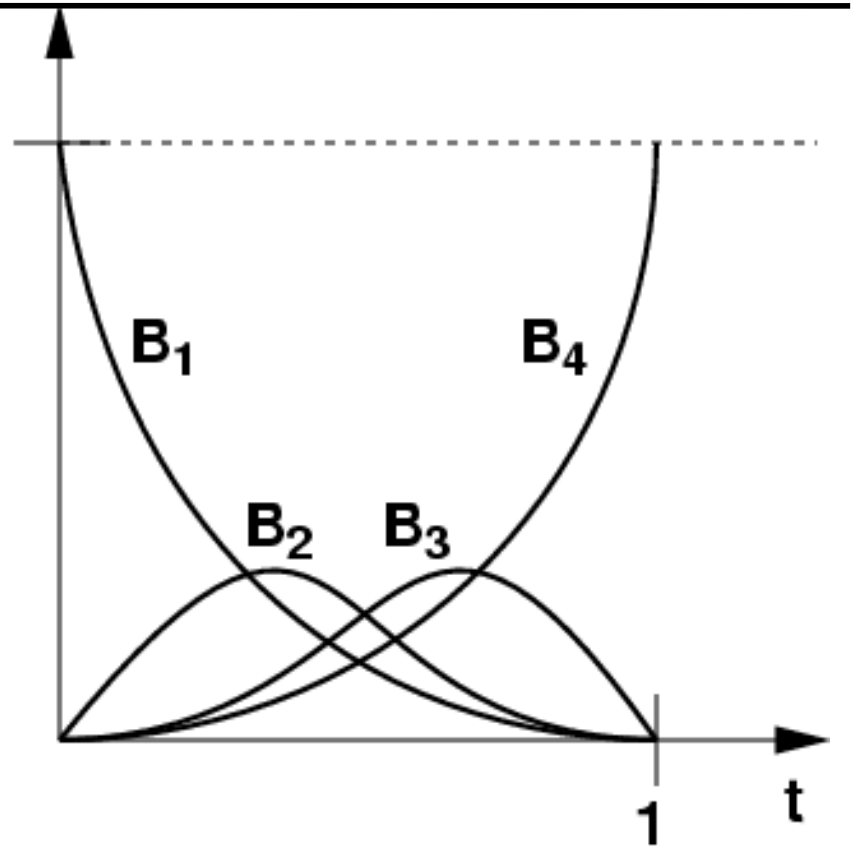
$$\begin{pmatrix} 1 \\ 1+t \\ 1+t+t^2 \\ 1+t-t^2+t^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

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

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

Bernstein Polynomials

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



- For cubic Bezier curve:

  $B1(t)=(1-t)^3$ $B2(t)=3t(1-t)^2$

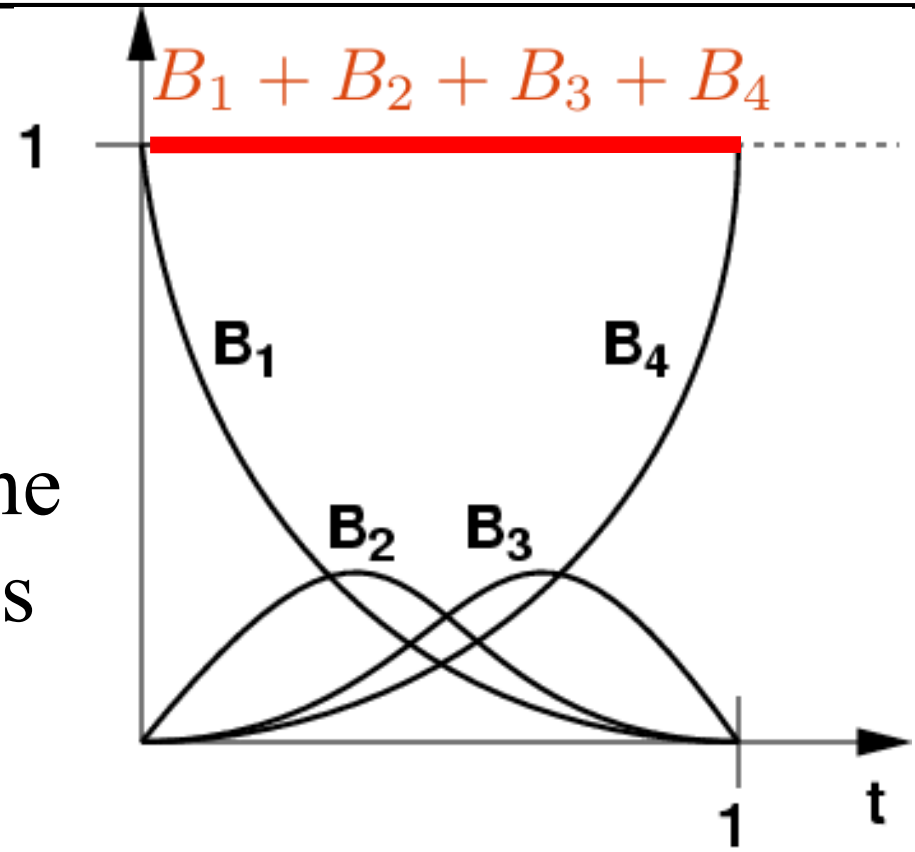
  $B3(t)=3t^2(1-t)$ $B4(t)=t^3$

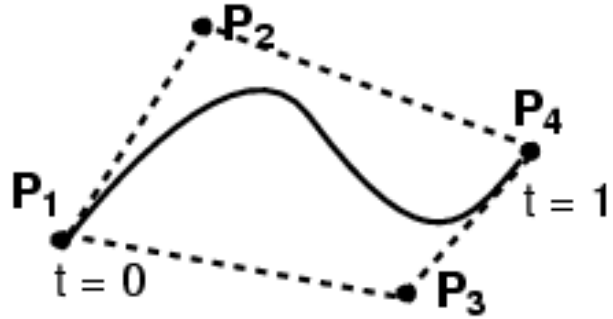
  (careful with indices, many authors start at 0)

- 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
- B1(0) =1
  - Bezier curve interpolates P1
- B4(1) =1
  - Bezier curve interpolates P4





Bézier Curves in Bernstein Basis

- $P(t) = P1B1(t) + P2B2(t) + P3B3(t) + P4B4(t)$
  - Pi are 2D points (xi, yi)
- $P(t)$ is a linear combination of the control points with weights equal to Bernstein polynomials at t
- But at the same time, the control points (P1, P2, P3, P4) 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!

- $B1(t)=(1-t)^3$
- $B2(t)=3t(1-t)^2$
- $B3(t)=3t^2(1-t)$
- $B4(t)=t^3$

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

New basis vectors

How You Get the Matrix

Cubic Bernstein:

- $B1(t)=(1-t)^3$
- $B2(t)=3t(1-t)^2$
- $B3(t)=3t^2(1-t)$
- $B4(t)=t^3$

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 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 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

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

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 P1..P4
(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 (P1, P2, …, Pn+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) does not really matter

Questions?

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!

TIEA311 towards next week

- Computer classroom today 16:15-17:45
- Koodaamo tomorrow 12:15-16:00
- Start **at least looking** at Assignment 1. Our lectures have covered only Bezier curve points. We will cover the derivatives (tangent, normal) and B-splines **next week** and surfaces even later!
- **If you start coding** for Assignment 1, aim towards using the general spline formulation $GBT(t)$ with suitably constructed G, B, and $T(t)$
- Remember the Power Tools!
- During the weekend, sleep well and dream of vector spaces!