Curves and Surfaces

Explicit Representations in Computer Graphics

Many Explicit Representations in Graphics

triangle meshes

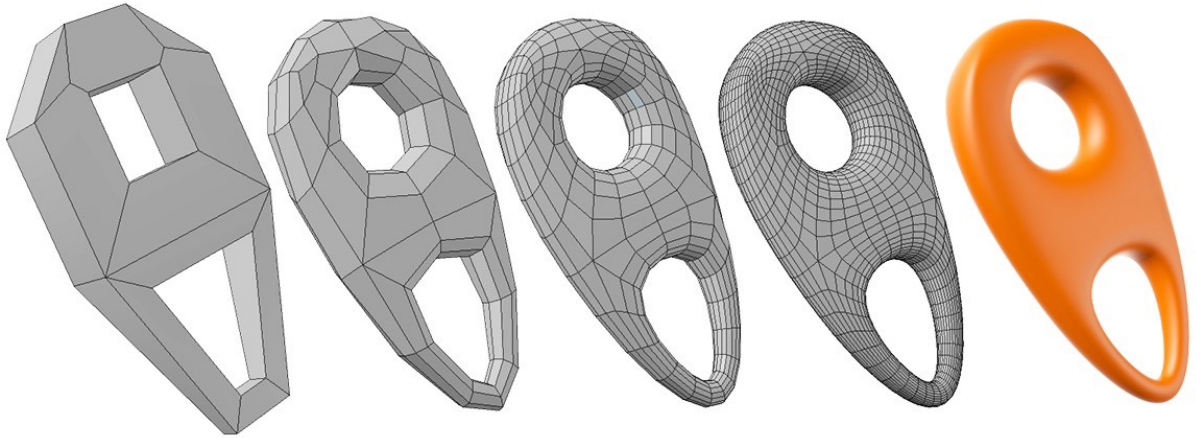
Bezier surfaces

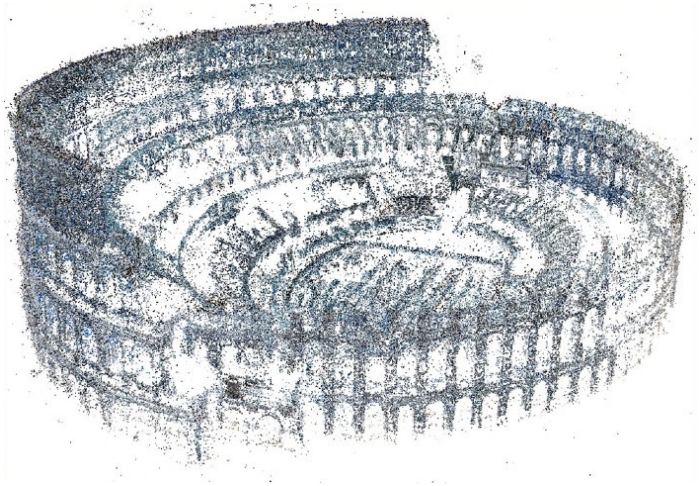
subdivision surfaces

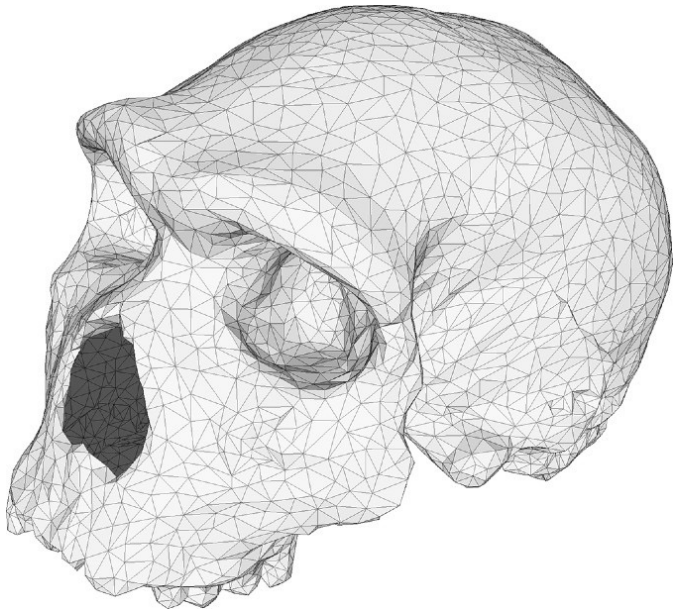
NURBS

point clouds

...

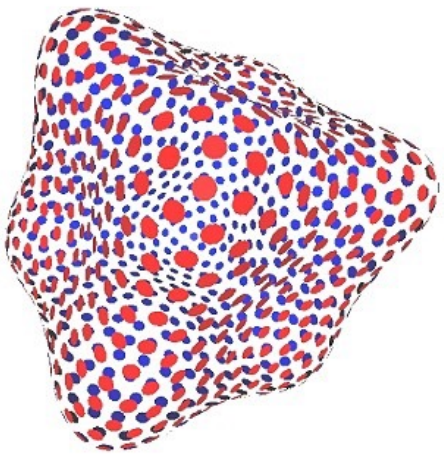


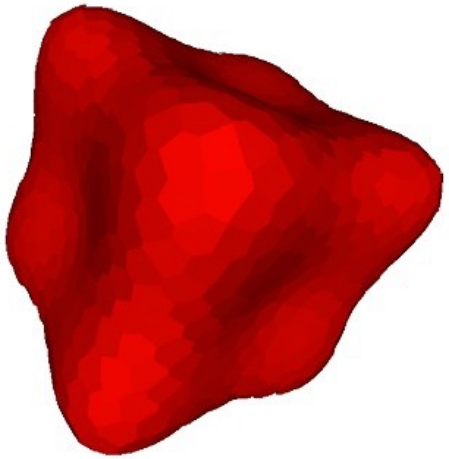


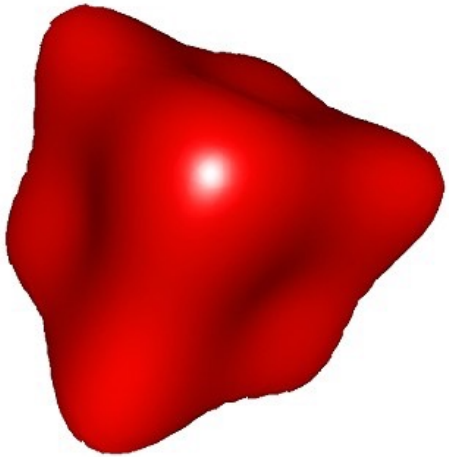


Point Cloud (Explicit)

Easiest representation: list of points (x,y,z) Easily represent any kind of geometry Useful for LARGE datasets (>>1 point/pixel) Often converted into polygon mesh Difficult to draw in undersampled regions







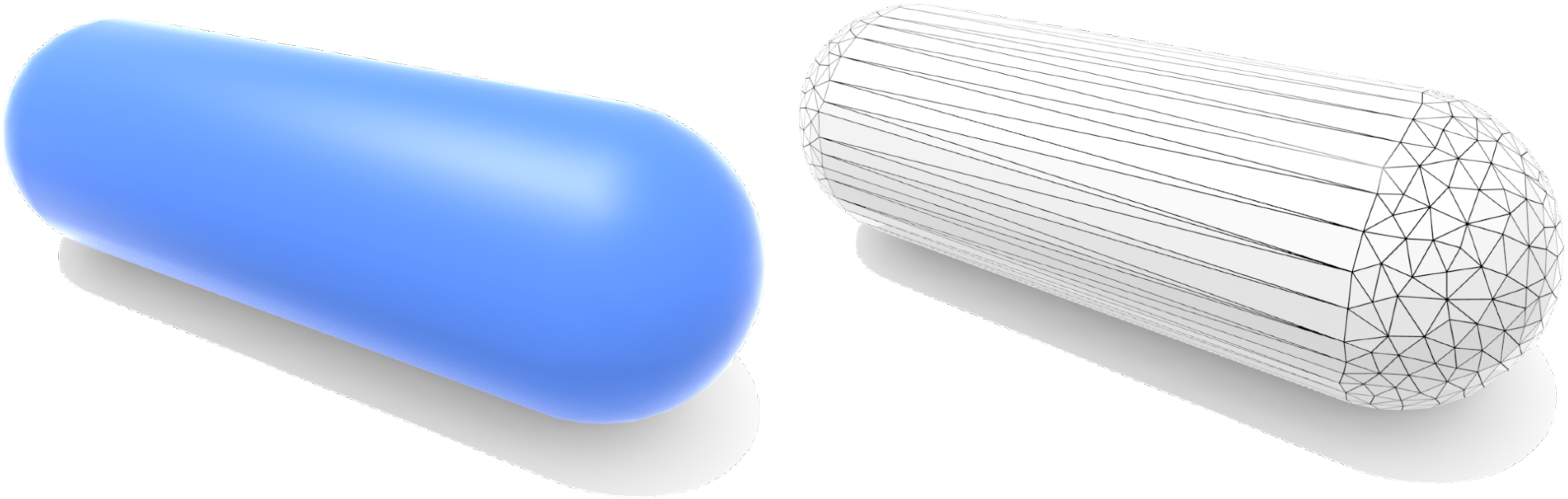
Polygon Mesh (Explicit)

Store vertices & polygons (often triangles or quads)

Easier to do processing / simulation, adaptive sampling

More complicated data structures

Perhaps most common representation in graphics



The Wavefront Object File (.obj) Format

Commonly used in Graphics research

Just a text file that specifies vertices, normals, texture coordinates **and their connectivities**

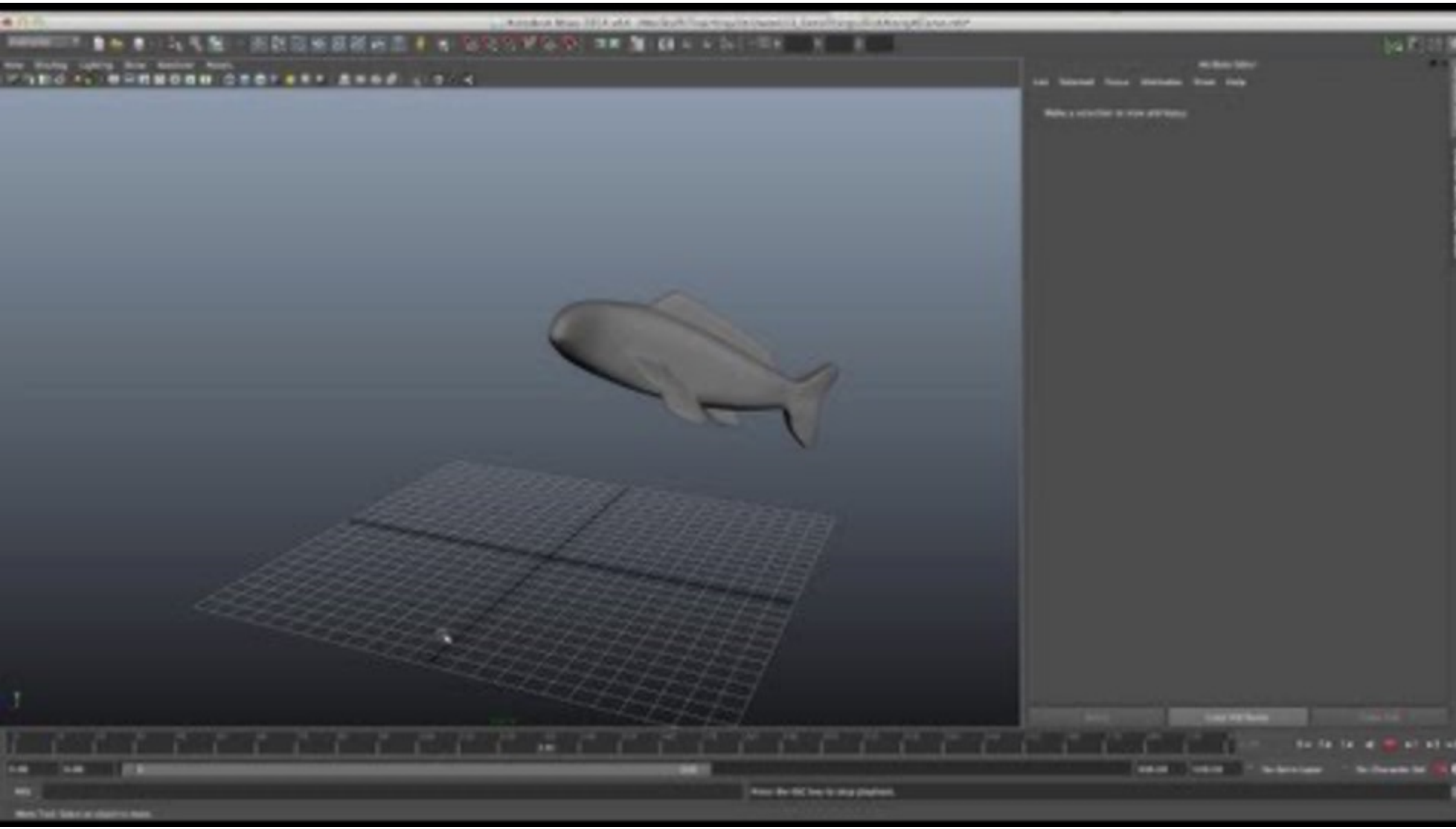
```
# This is a comment

v 1.000000 -1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
v 0.999999 1.000000 1.000001
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000

vt 0.748573 0.750412
vt 0.749279 0.501284
vt 0.999110 0.501077
vt 0.999455 0.750380
vt 0.250471 0.500702
vt 0.249682 0.749677
vt 0.001085 0.750380
vt 0.001517 0.499994
vt 0.499422 0.500239
vt 0.500149 0.750166
vt 0.748355 0.998230
vt 0.500193 0.998728
vt 0.498993 0.250415
vt 0.748953 0.250920

vn 0.000000 0.000000 -1.000000
vn -1.000000 -0.000000 -0.000000
vn -0.000000 -0.000000 1.000000
vn -0.000001 0.000000 1.000000
vn 1.000000 -0.000000 0.000000
vn 1.000000 0.000000 0.000001
vn 0.000000 1.000000 -0.000000
vn -0.000000 -1.000000 0.000000

f 5/1/1 1/2/1 4/3/1
f 5/1/1 4/3/1 8/4/1
f 3/5/2 7/6/2 8/7/2
f 3/5/2 8/7/2 4/8/2
f 2/9/3 6/10/3 3/5/3
f 6/10/4 7/6/4 3/5/4
f 1/2/5 5/1/5 2/9/5
f 5/1/6 6/10/6 2/9/6
f 5/1/7 8/11/7 6/10/7
f 8/11/7 7/12/7 6/10/7
f 1/2/8 2/9/8 3/13/8
f 1/2/8 3/13/8 4/14/8
```

Curves

Camera Paths

Flythrough of proposed Perth Citylink subway, https://youtu.be/rIJMuQPwr3E

Animation Curves

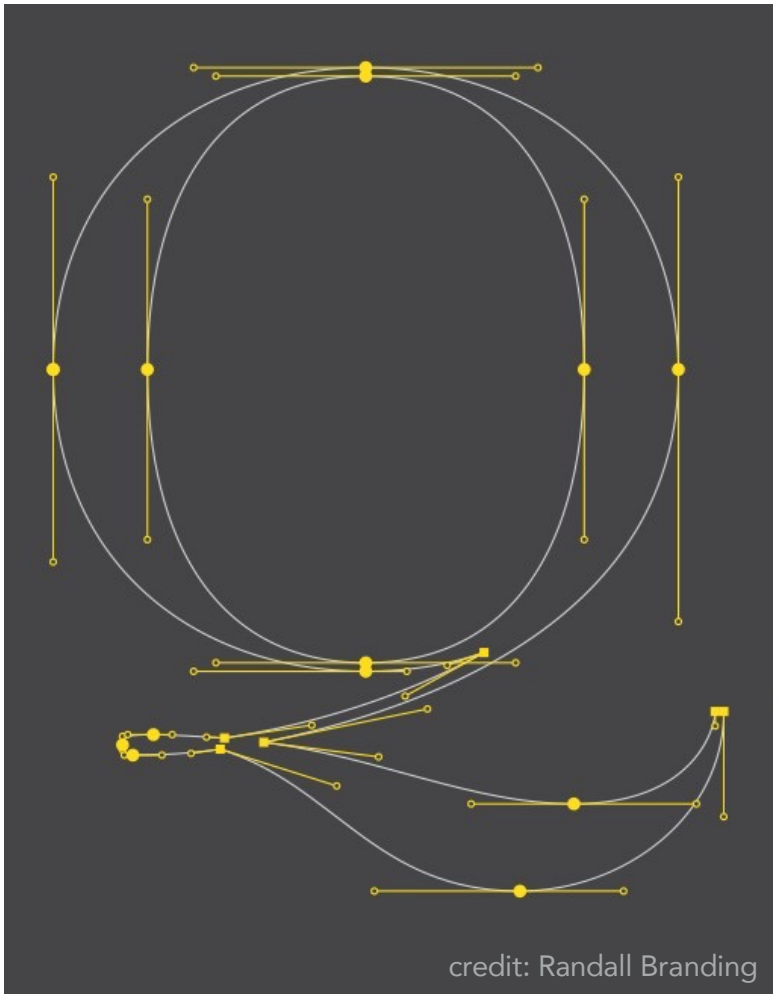


Maya Animation Tutorial: https://youtu.be/b-o5wtZlJPc

Vector Fonts

The Quick Brown
Fox Jumps Over
The Lazy Dog

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

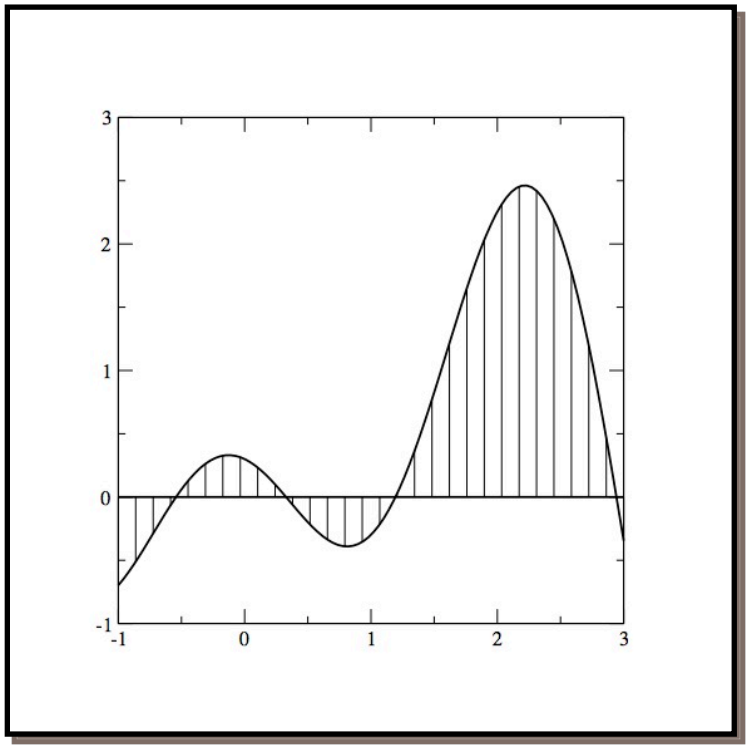


Baskerville font - represented as piecewise cubic Bézier curves

Discretization

Arbitrary curves have an uncountable number of parameters

i.e. specify function value at all points on real number line



Polynomial Basis

Power Basis

$$x(u) = \sum_{i=0}^{d} c_i \, u^i \qquad \boldsymbol{C} = [c_0, c_1, c_2, \ldots, c_d]$$

$$x(u) = \boldsymbol{C} \cdot \boldsymbol{\mathcal{P}}^d \qquad \boldsymbol{\mathcal{P}}^d = [1, u, u^2, \ldots, u^d]$$

The elements of $\boldsymbol{\mathcal{P}}^d$ are *linearly independant*

i.e. no good approximation

$$u^k \not\approx \sum_{i \neq k} c_i \, u^i$$

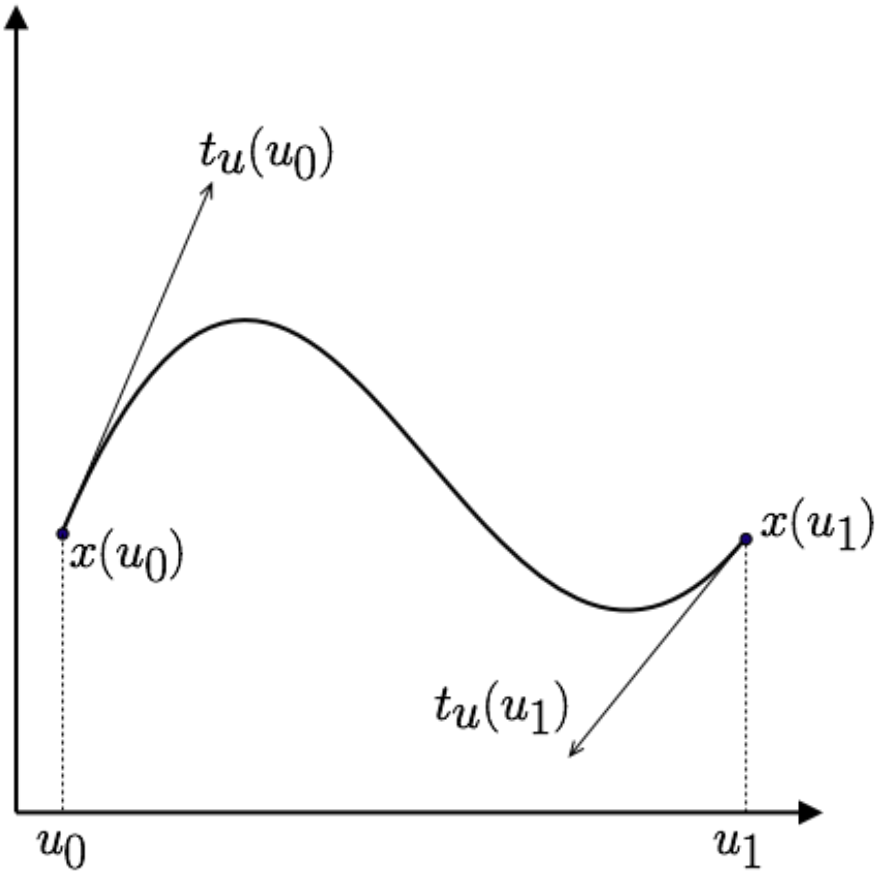
Skipping something would lead to bad results... odd stiffness

Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

For now, assume

$u_0 = 0 \quad u_1 = 1$



Specifying a Curve

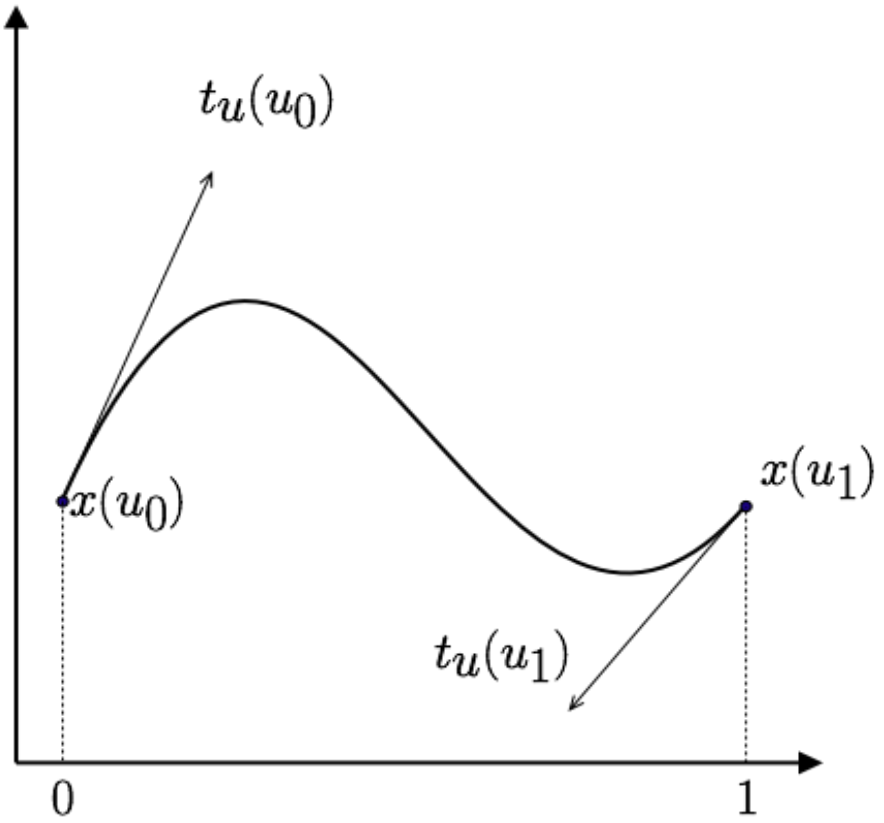
Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$x(0) = c_0 \quad = x_0$$

$$x(1) = \Sigma\, c_i \;= x_1$$

$$x'(0) = c_1 \quad = x'_0$$

$$x'(1) = \Sigma\, i\, c_i = x'_1$$

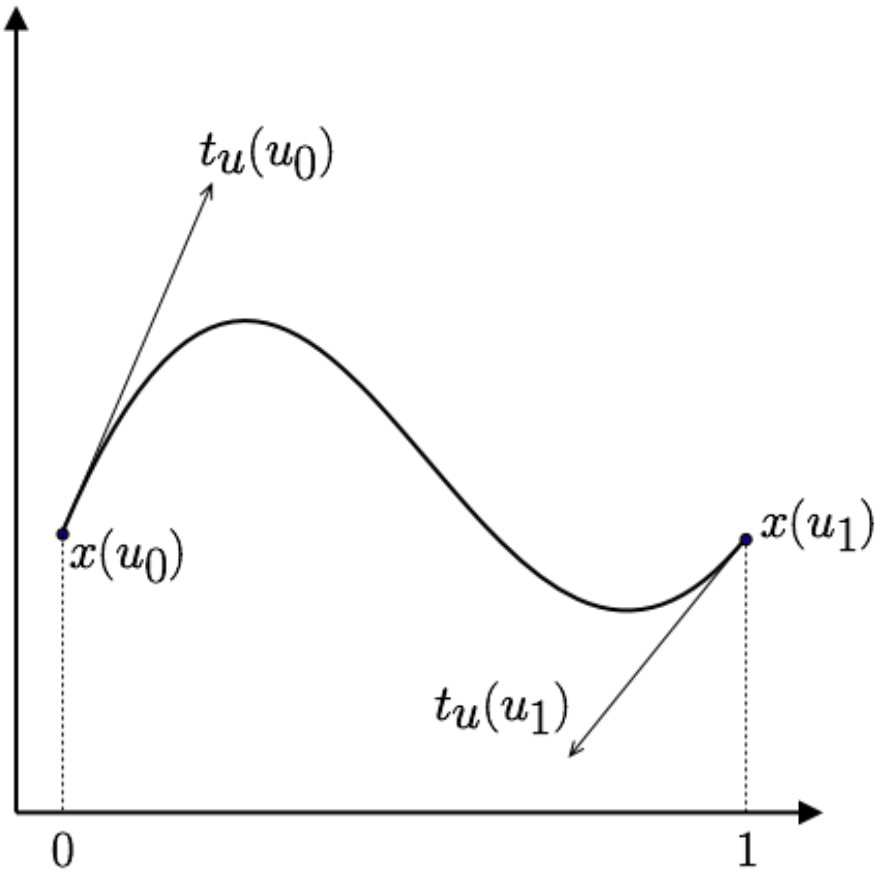


Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$\begin{bmatrix} x_0 \\ x_1 \\ x'_0 \\ x'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\mathbf{p} = \mathbf{B} \cdot \mathbf{c}$$

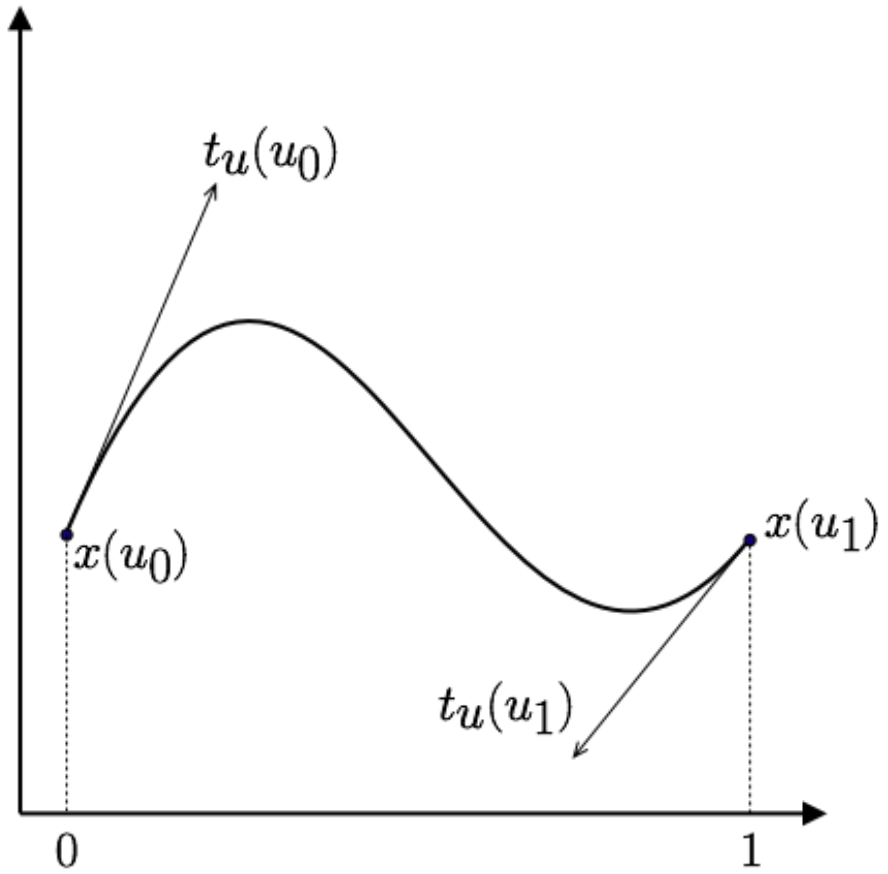


Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$\mathbf{c} = \beta_{\text{H}} \cdot \mathbf{p}$$

$$\boldsymbol{\beta}_{\text{H}} = \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$



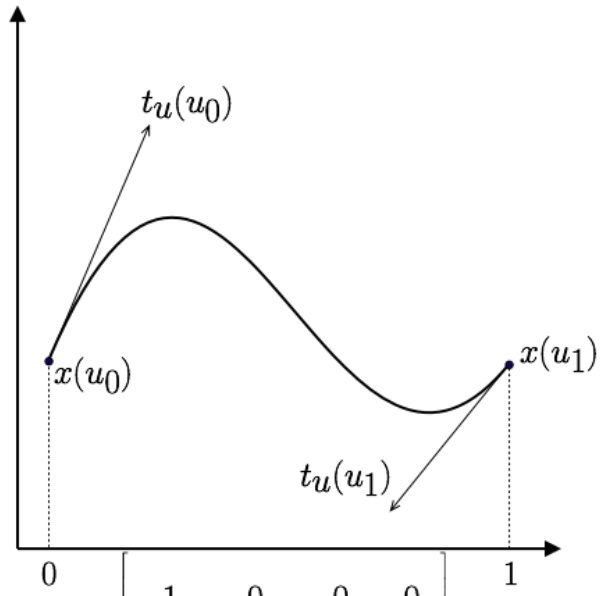
Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$\mathbf{c} = \beta_{\text{H}} \cdot \mathbf{p}$$

$$x(u) = \mathcal{P}^3 \cdot \mathbf{c} = \mathcal{P}^3 \beta_{\text{H}} \mathbf{p}$$

$$= \begin{bmatrix} 1 + 0u - 3u^2 + 2u^3 \\ 0 + 0u + 3u^2 - 2u^3 \\ 0 + 1u - 2u^2 + 1u^3 \\ 0 + 0u - 1u^2 + 1u^3 \end{bmatrix} \mathbf{p}$$



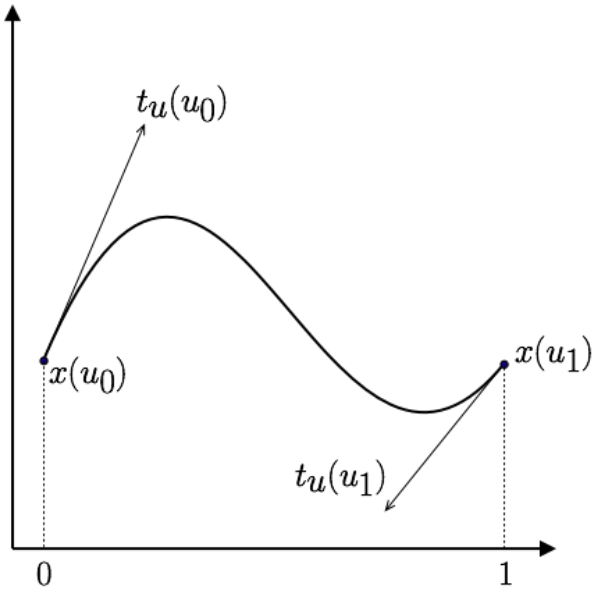
$$\beta_{\text{H}} = \mathbf{B}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & 1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?

$$\mathbf{c} = \beta_{\mathrm{H}} \cdot \mathbf{p}$$

$$x(u) = \begin{bmatrix} 1 + 0u - 3u^2 + 2u^3 \\ 0 + 0u + 3u^2 - 2u^3 \\ 0 + 1u - 2u^2 + 1u^3 \\ 0 + 0u - 1u^2 + 1u^3 \end{bmatrix} \mathbf{p}$$

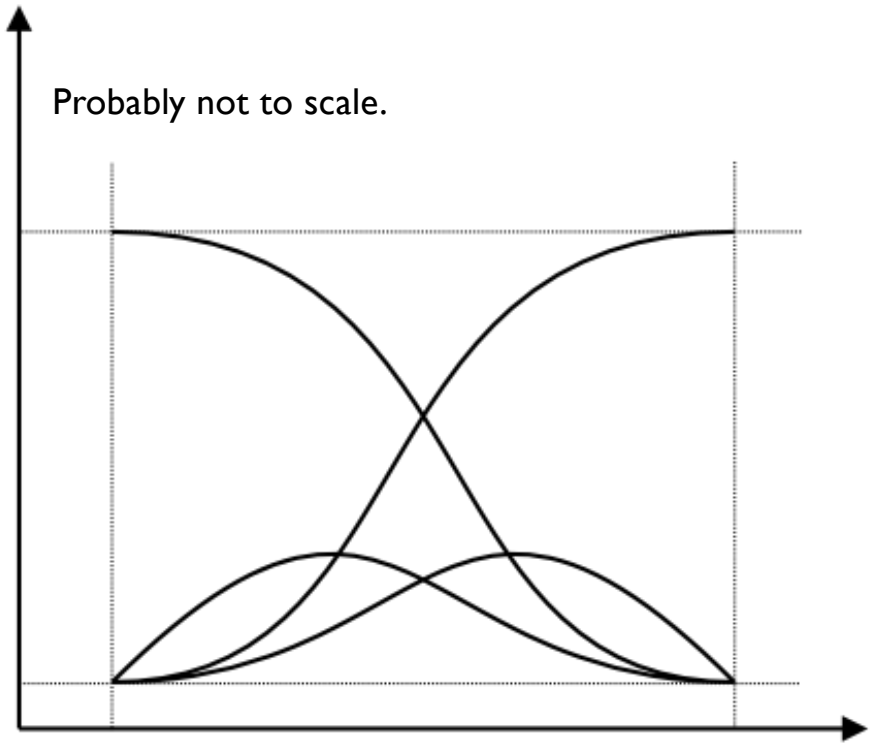


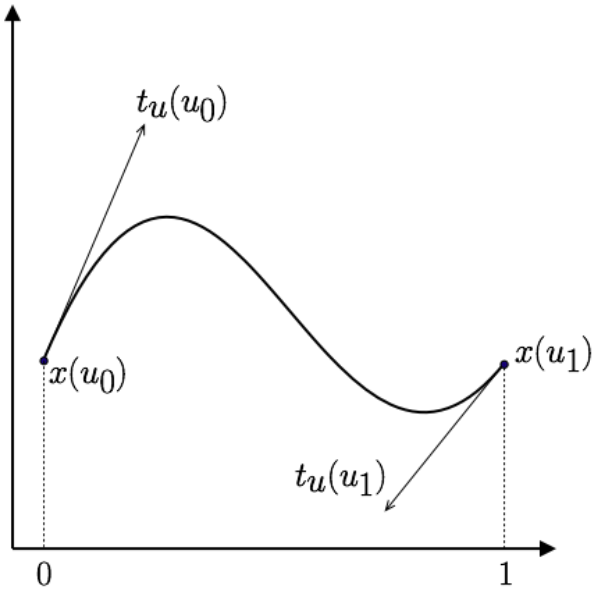
$$x(u) = \sum_{i=0}^{3} p_i b_i(u)$$

Hermite basis functions

Specifying a Curve

Given desired values (constraints) how do we determine the coefficients for cubic power basis?





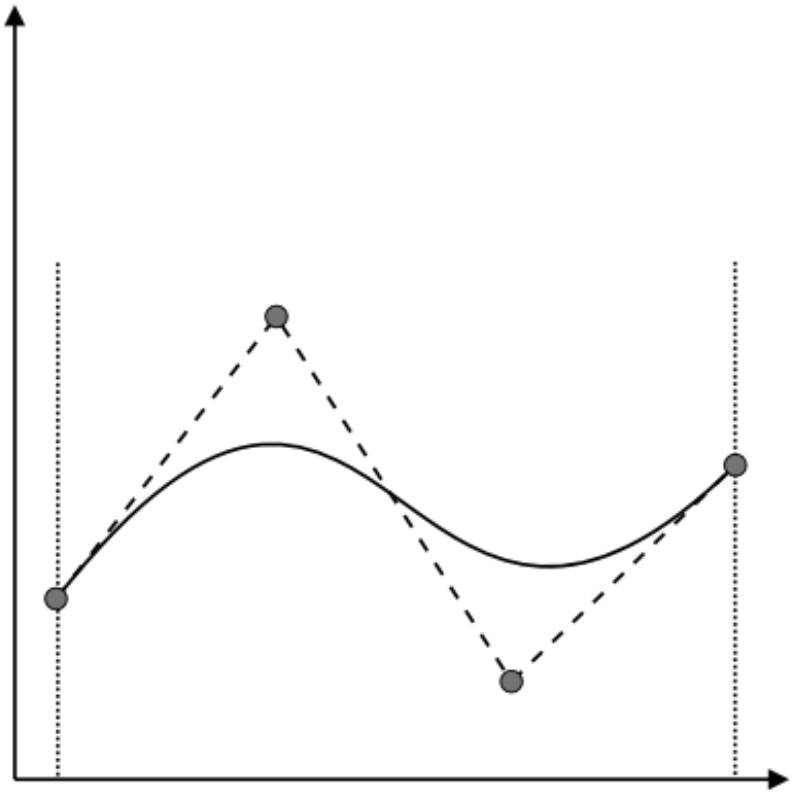
$$x(u) = \sum_{i=0}^{3} p_i b_i(u)$$

Hermite basis functions

Cubic Bézier

Similar to Hermite, but specify tangents indirectly

$$
\begin{aligned}
x_0 &= p_0 \\
x_1 &= p_3 \\
x'_0 &= 3(p_1 - p_0) \\
x'_1 &= 3(p_3 - p_2)
\end{aligned}
$$



Note: all the control points
are points in space, no tangents.

Cubic Bézier

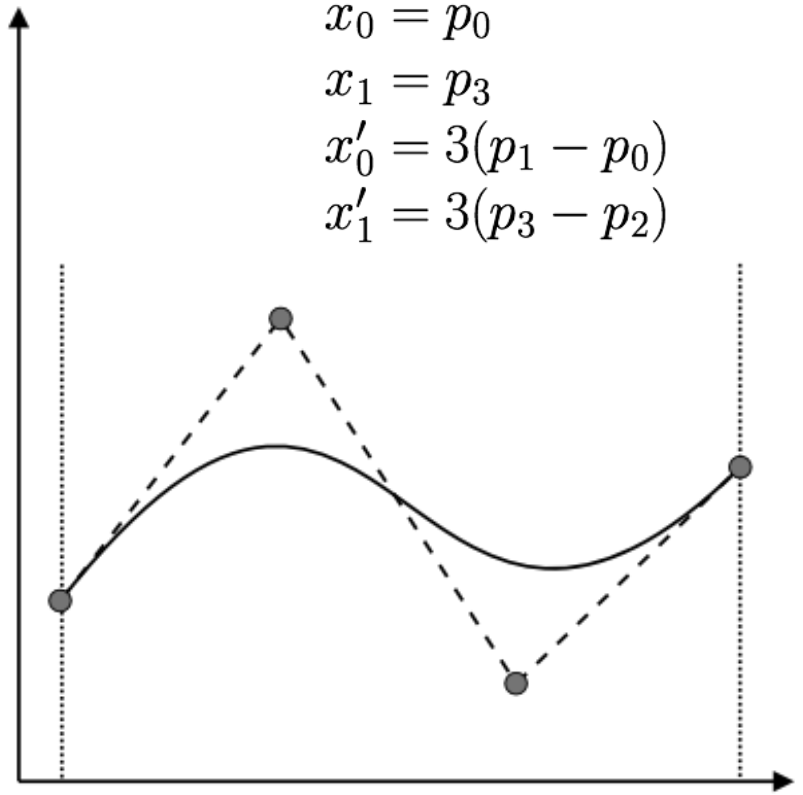
Similar to Hermite, but specify tangents indirectly

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

$$\mathbf{c} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \mathbf{p}$$

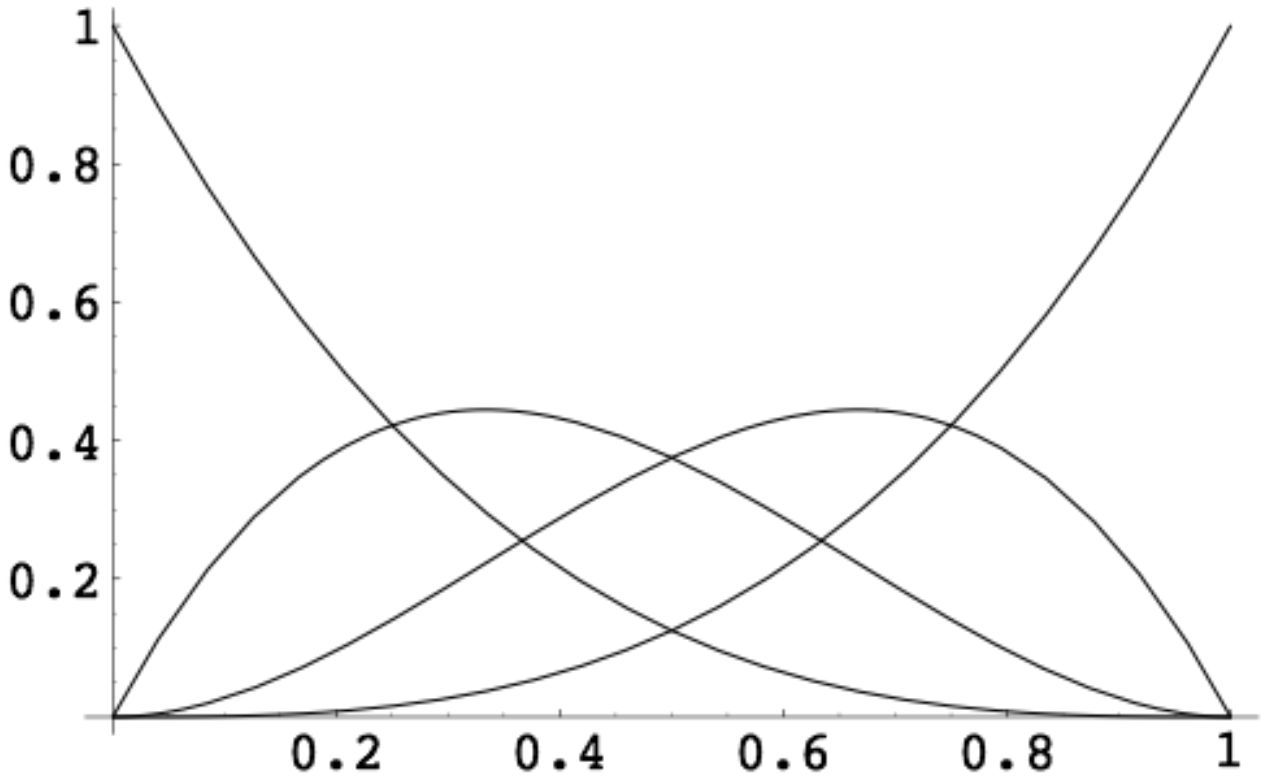
$$\mathbf{c} = \boldsymbol{\beta}_z \, \mathbf{p}$$

$$x_0 = p_0$$
$$x_1 = p_3$$
$$x_0' = 3(p_1 - p_0)$$
$$x_1' = 3(p_3 - p_2)$$



Cubic Bézier

Plot of Bézier basis functions



Changing Bases

Power basis, Hermite, and Bézier all are still just cubic polynomials

The three bases all span the same space

Like different axes in $\Re^3$ X $\Re^4$

Changing basis

$$\mathbf{c} = \boldsymbol{\beta}_Z \mathbf{p}_Z$$

$$\mathbf{c} = \boldsymbol{\beta}_H \mathbf{p}_H$$

$$\mathbf{p}_Z = \boldsymbol{\beta}_Z^{-1} \boldsymbol{\beta}_H \mathbf{p}_H$$

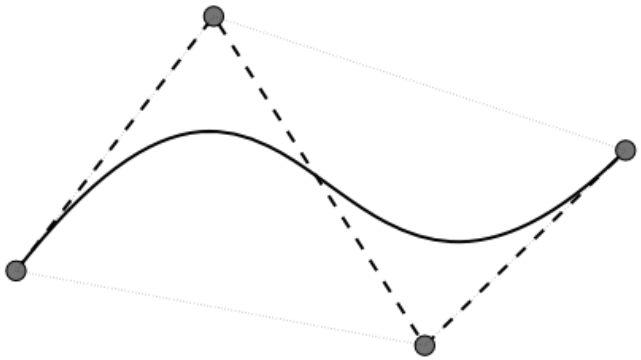
Useful Properties of a Basis

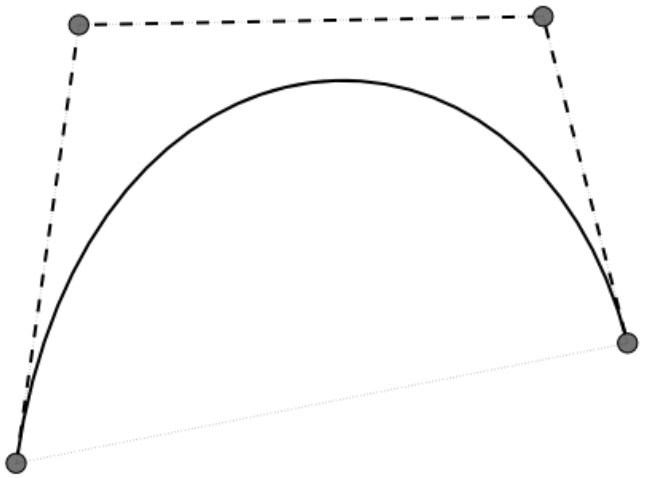
Convex Hull

All points on curve inside convex hull of control points

Bézier basis has convex hull property

$$\sum_i b_i(u) = 1 \qquad b_i(u) \geq 0 \qquad \forall u \in \Omega$$





Useful Properties of a Basis

Invariance under class of transforms

--Transforming curve is same as transforming control points

$$\boldsymbol{x}(u) = \sum_i \boldsymbol{p}_i b_i(u) \Leftrightarrow \mathcal{T}\boldsymbol{x}(u) = \sum_i (\mathcal{T}\boldsymbol{p}_i) b_i(u)$$

Bézier basis invariant for affine transforms

Bézier basis NOT invariant for perspective transforms

Useful Properties of a Basis

Local support

- Changing one control point has limited impact on entire curve

Nice subdivision rules

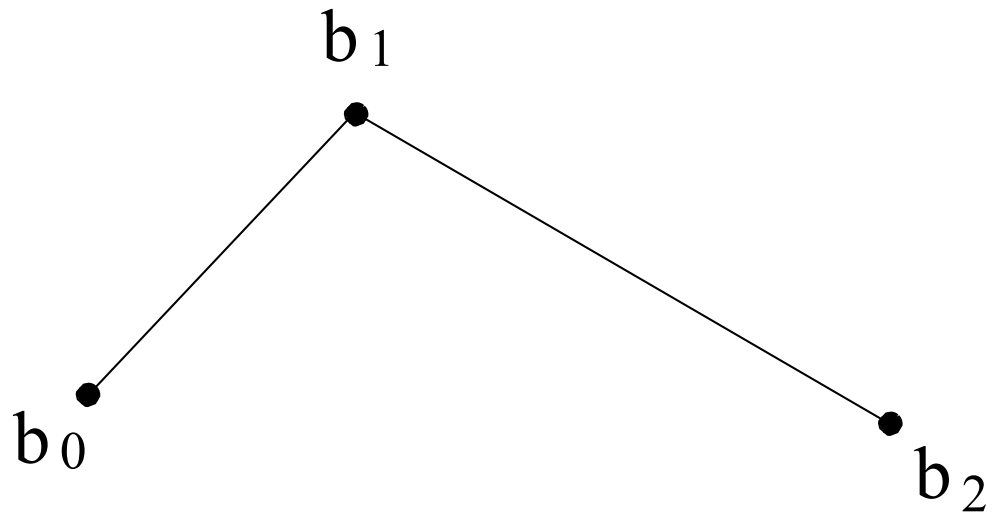
Fast evaluation scheme

Interpolation -vs- approximation

Evaluating Bézier Curves (de Casteljau Algorithm)

Bézier Curves – de Casteljau Algorithm

Consider three points (quadratic Bezier)

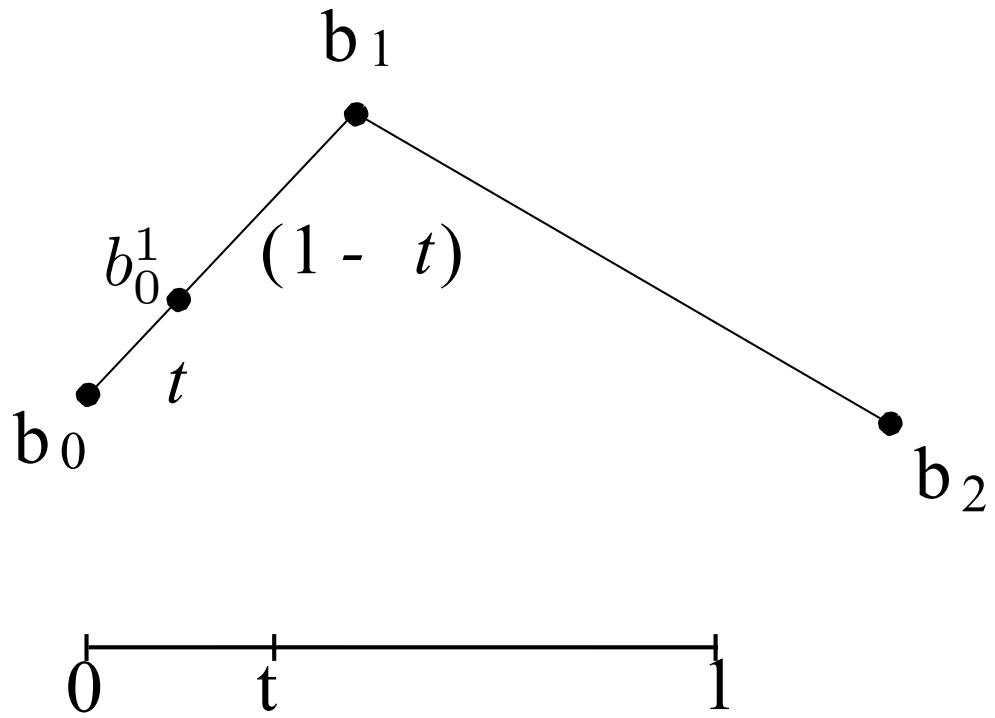


Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

Insert a point using linear interpolation

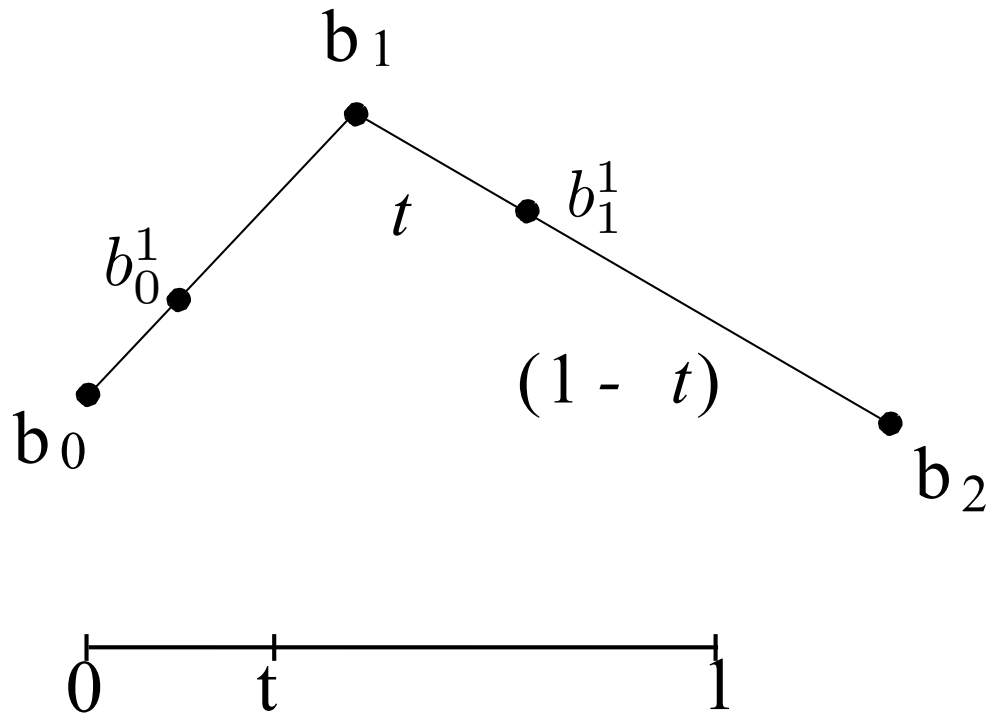


Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

Insert on both edges

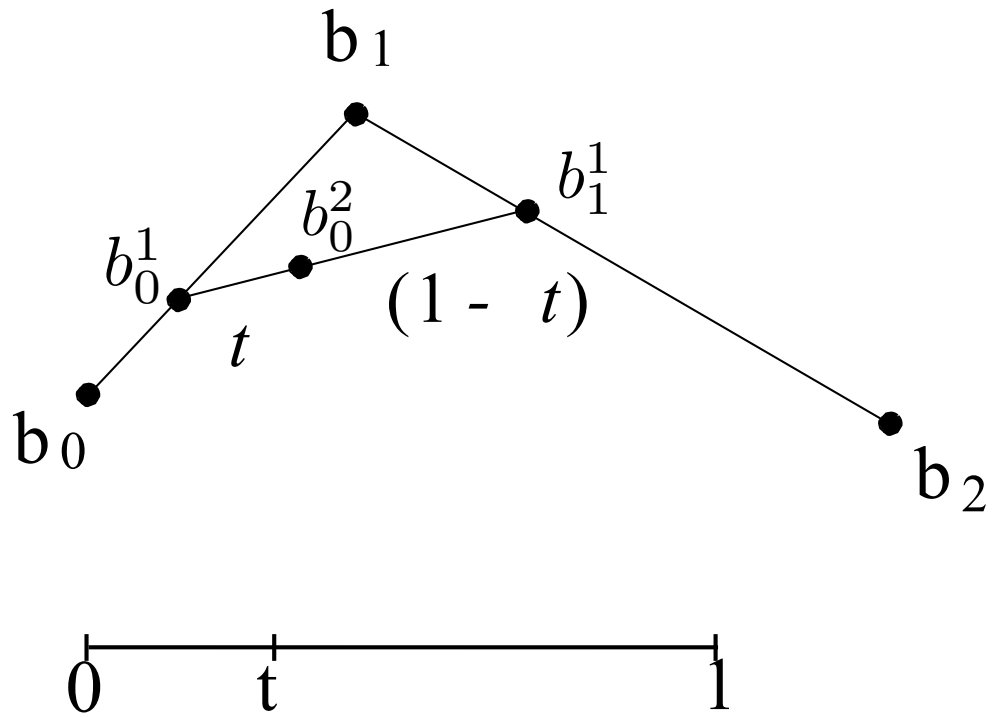


Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

Repeat recursively

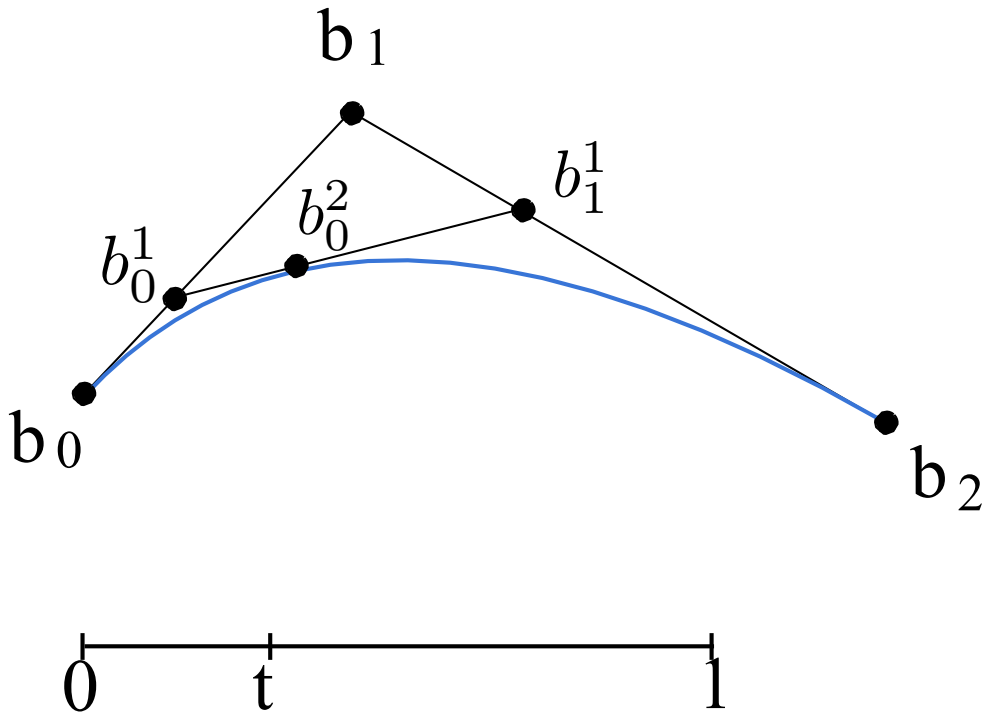


Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

Run the same algorithm for every t in [0,1]



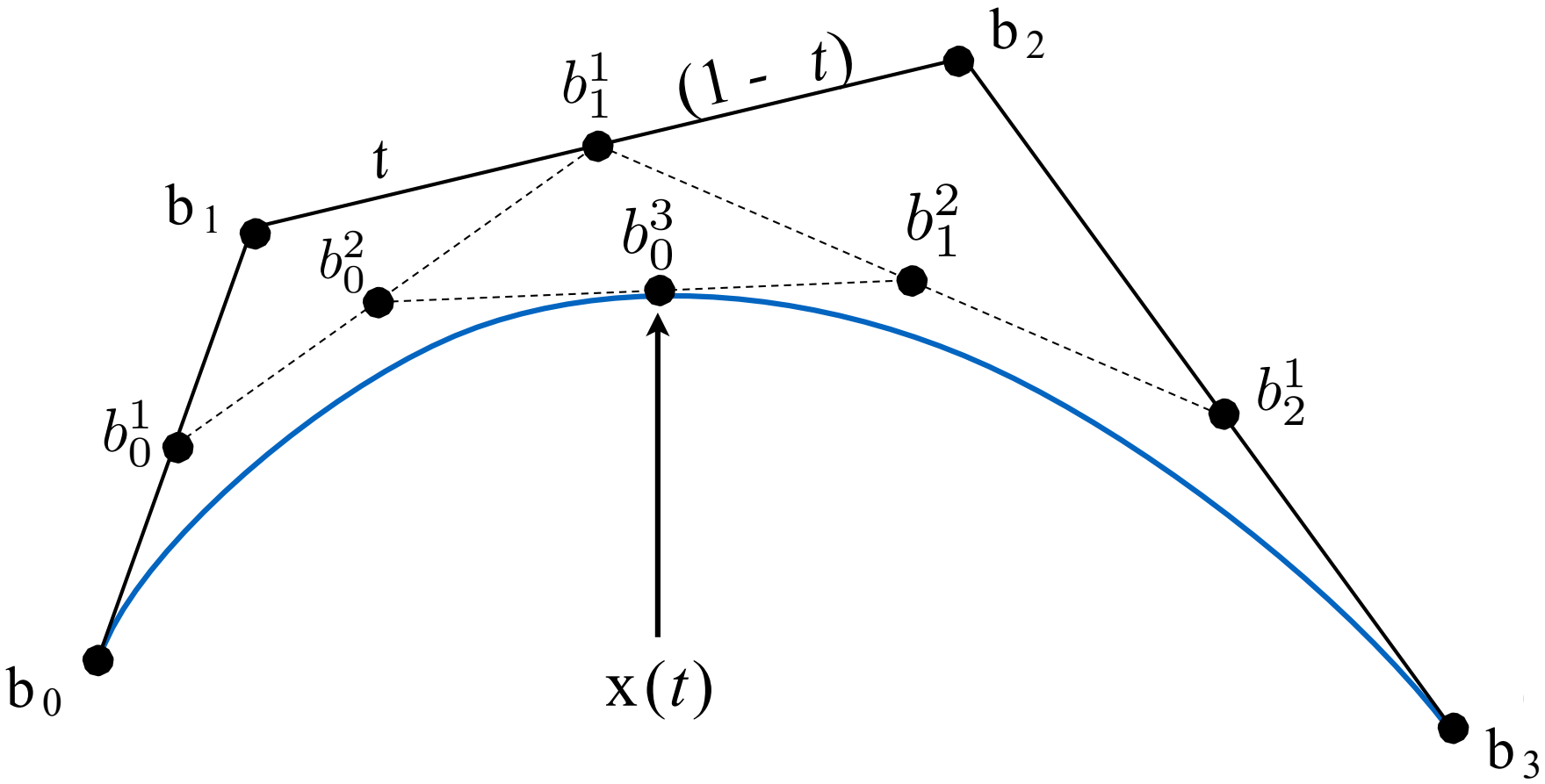
Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

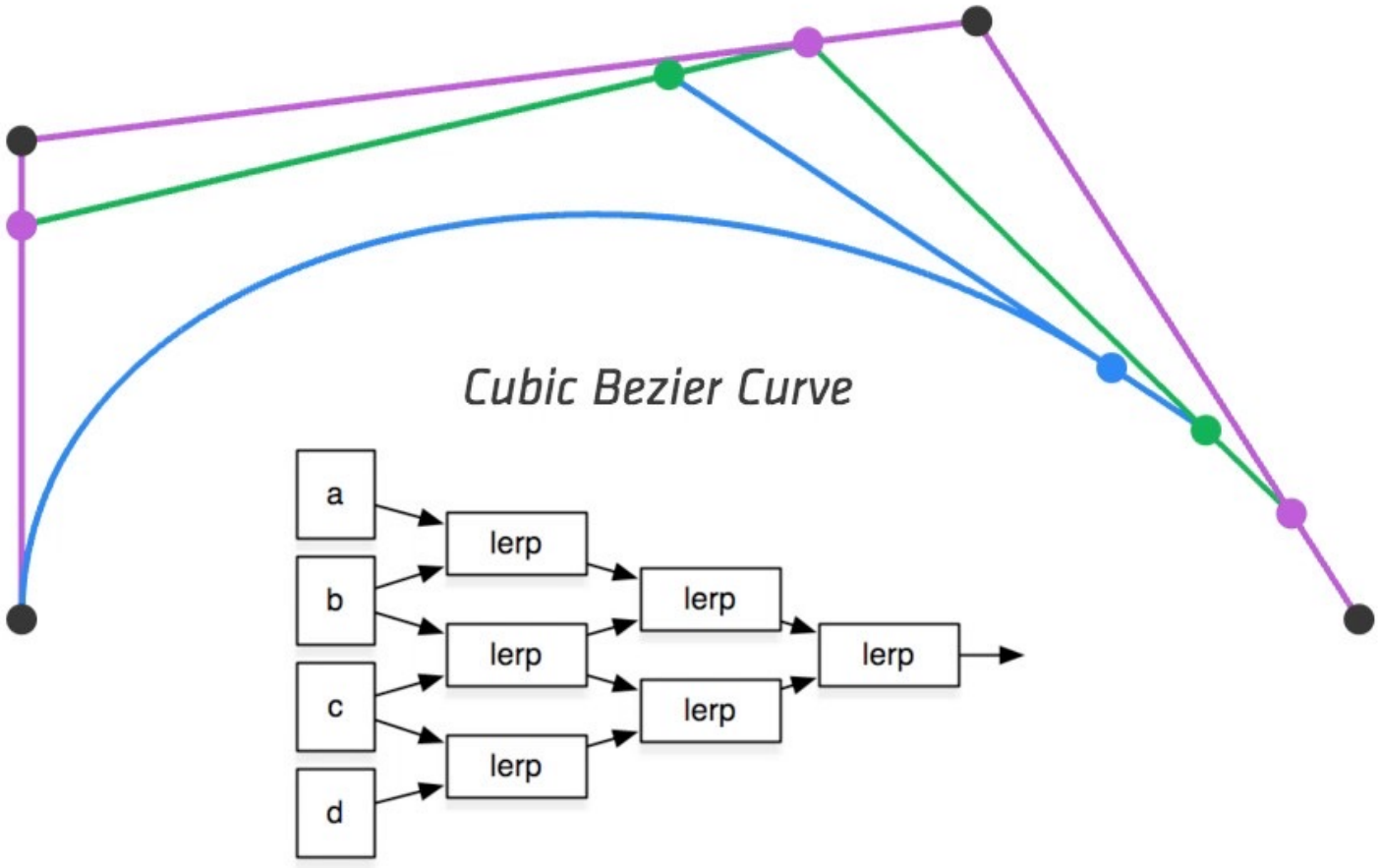
Cubic Bézier Curve – de Casteljau

Four input points in total

Same recursive linear interpolations



Visualizing de Casteljau Algorithm

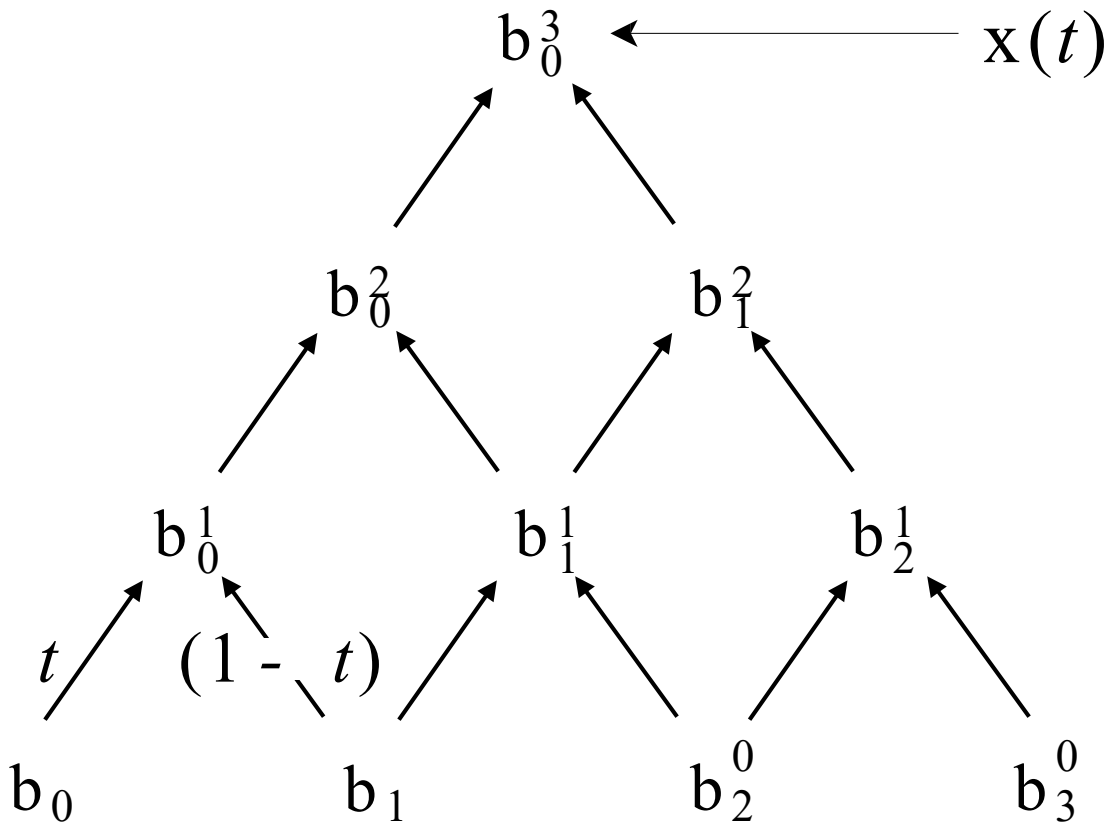


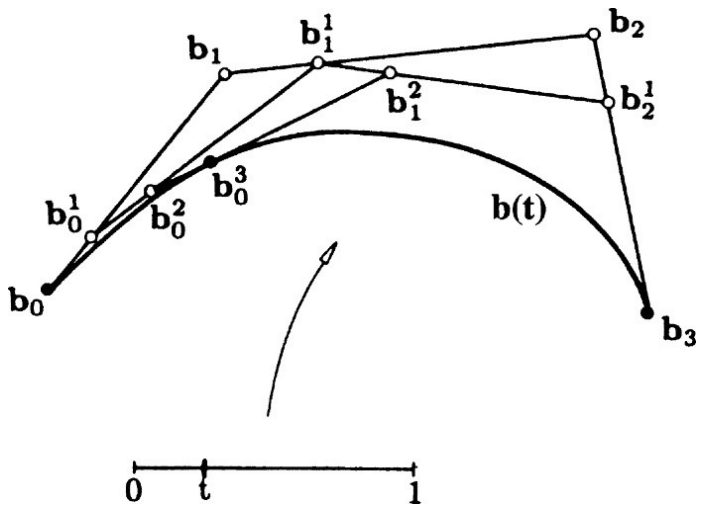
Animation: Steven Wittens, Making Things with Maths, http://acko.net

Evaluating Bézier Curves
Algebraic Formula

Bézier Curve – Algebraic Formula

de Casteljau algorithm gives a pyramid of coefficients

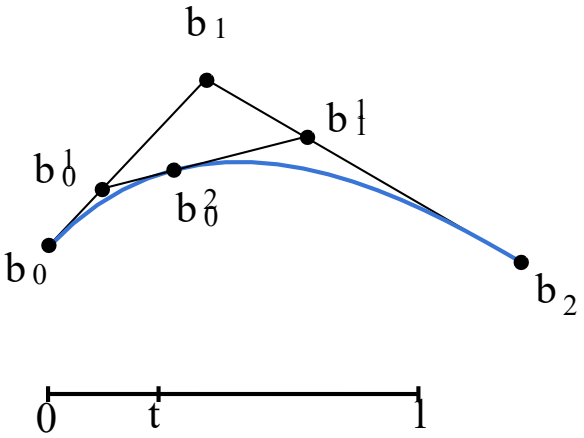




Every rightward arrow is multiplication by t,
Every leftward arrow by (1–t)

Bézier Curve – Algebraic Formula

Example: quadratic Bézier curve from three points



$$b_0^1(t) = (1 - t)b_0 + tb_1$$

$$b_1^1(t) = (1 - t)b_1 + tb_2$$

$$b_0^2(t) = (1 - t)b_0^1 + tb_1^1$$

$$\mathbf{b}_0^2(t) = (1 - t)^2\mathbf{b}_0 + 2t(1 - t)\mathbf{b}_1 + t^2\mathbf{b}_2$$

Bézier Curve – General Algebraic Formula

Bernstein form of a Bézier curve of order n:

$$\mathrm{b}^{\mathrm{n}}(t) = \mathrm{b}^{\mathrm{n}}(t) = \sum_{j=0}^{n} b_j B_j^n(t)$$

Bézier curve order n
(vector polynomial of degree n)

Bernstein polynomial
(scalar polynomial of degree n)

Bézier control points
(vector in R^N)

Bernstein polynomials:

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

Bézier Curve – Algebraic Formula: Example

Bernstein form of a Bézier curve of order n:

$$\mathrm{b}^{\mathrm{n}}(t) = \sum_{j=0}^{n} b_j B_j^n(t)$$

Example: assume n = 3 and we are in R^3

i.e. we could have control points in 3D such as:

$$\mathbf{b}_0 = (0, 2, 3),\ \mathbf{b}_1 = (2, 3, 5),\ \mathbf{b}_2 = (6, 7, 9),\ \mathbf{b}_3 = (3, 4, 5)$$

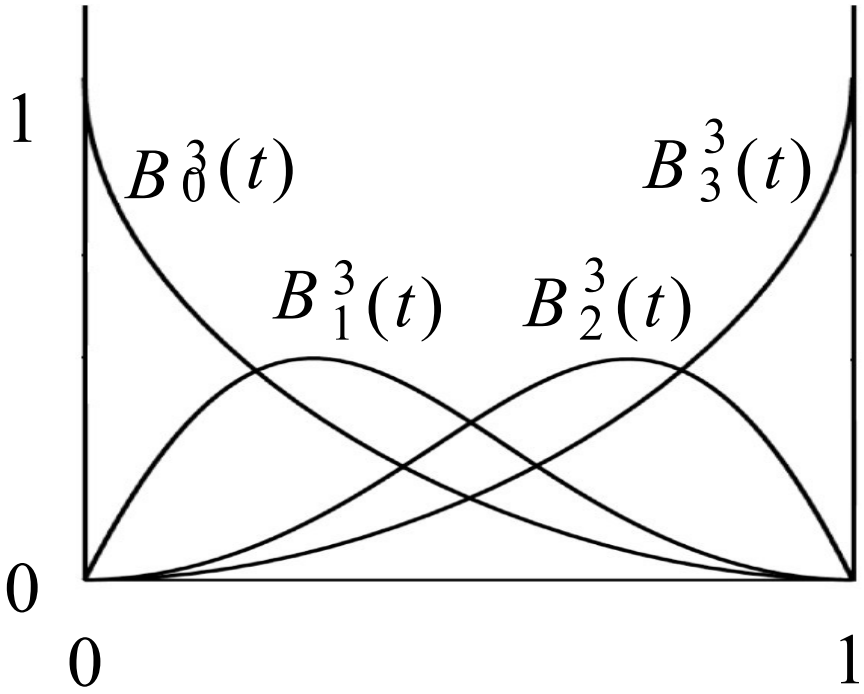
These points define a Bezier curve in 3D that is a cubic polynomial in t:

$$\mathbf{b}^n(t) = \mathbf{b}_0 (1 - t)^3 + \mathbf{b}_1 3t(1 - t)^2 + \mathbf{b}_2 3t^2(1 - t) + \mathbf{b}_3 t^3$$

Cubic Bézier Basis Functions

Bernstein Polynomials

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



Sergei N. Bernstein
1880 – 1968

Properties of Bézier Curves

Interpolates endpoints

- For cubic Bézier: $\mathbf{b}(0) = \mathbf{b}_0; \quad \mathbf{b}(1) = \mathbf{b}_3$

Tangent to end segments

- Cubic case: $\mathbf{b}'(0) = 3(\mathbf{b}_1 - \mathbf{b}_0); \quad \mathbf{b}'(1) = 3(\mathbf{b}_3 - \mathbf{b}_2)$

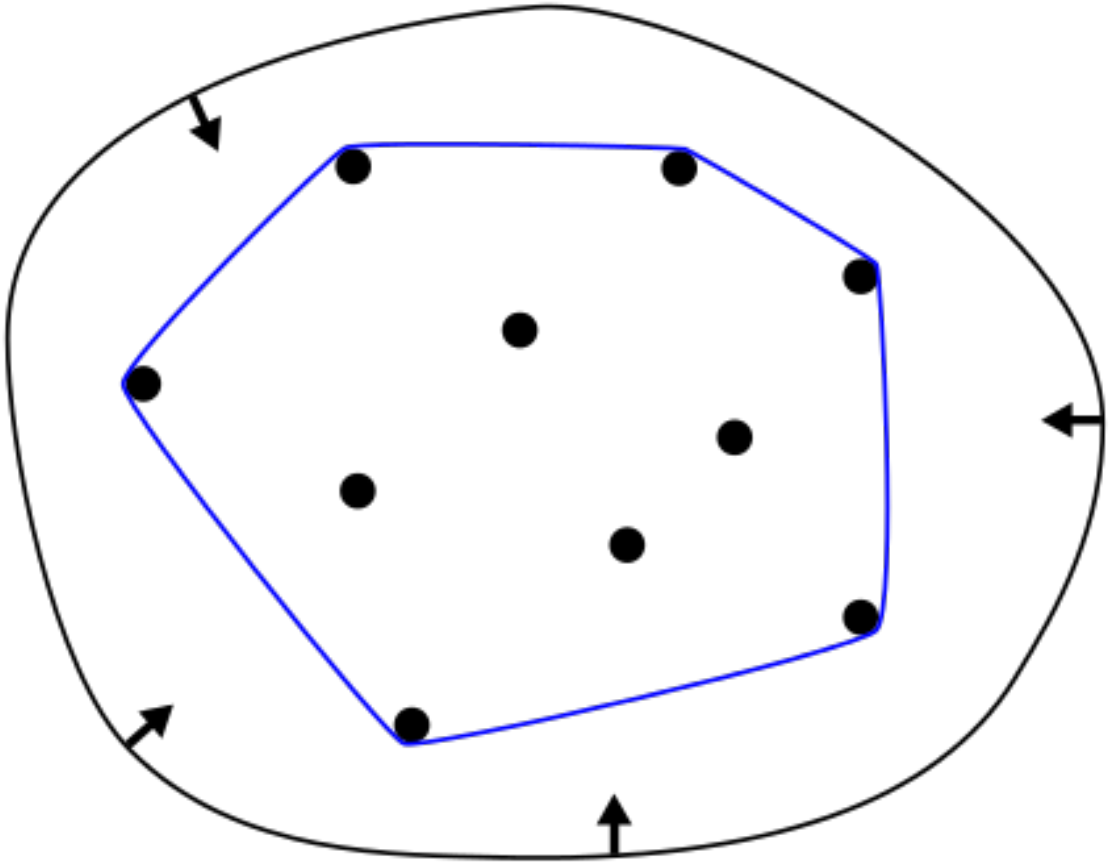
Affine transformation property

- Transform curve by transforming control points

Convex hull property

- Curve is within convex hull of control points

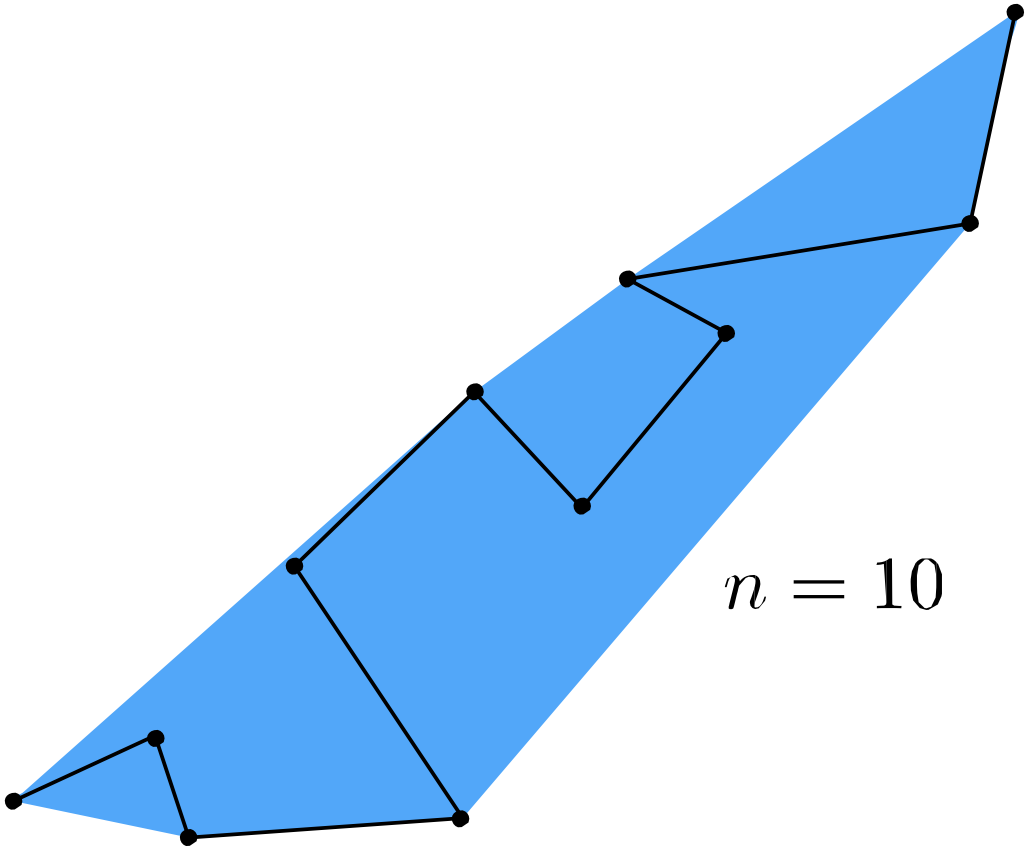
BTW: What's a Convex Hull



[from Wikipedia]

Piecewise Bézier Curves

Higher-Order Bézier Curves?



$n = 10$

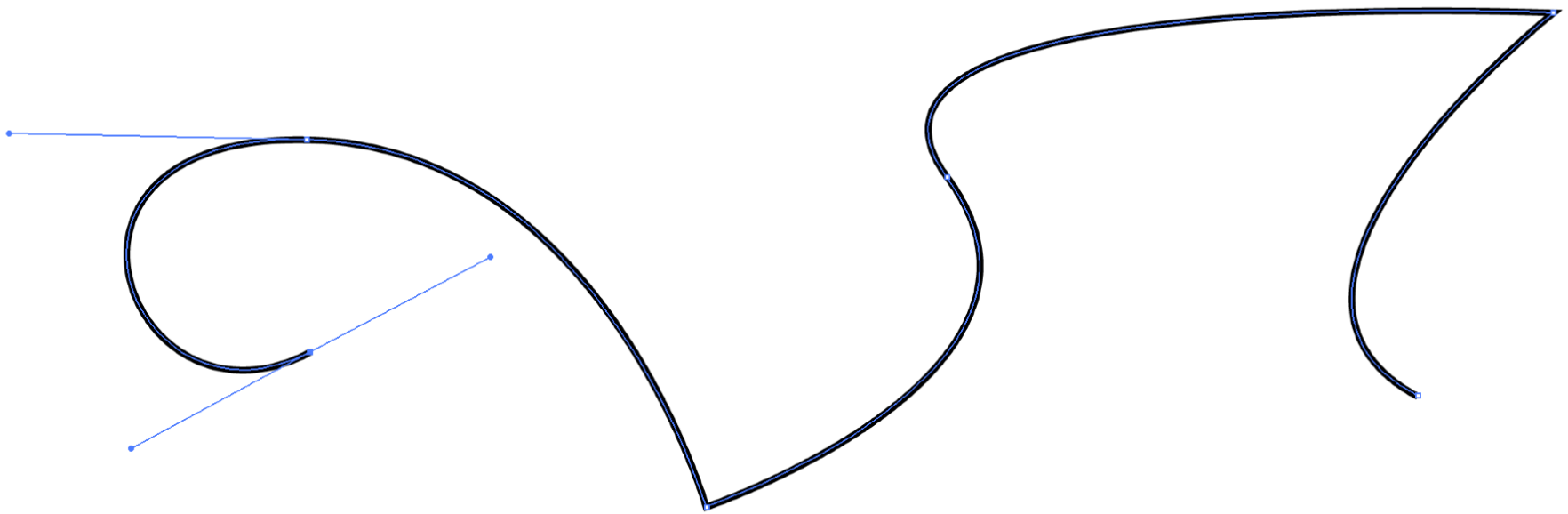
Very hard to control!

Uncommon

Piecewise Bézier Curves

Instead, chain many low-order Bézier curve

Piecewise cubic Bézier the most common technique



Widely used (fonts, paths, Illustrator, Keynote, …)

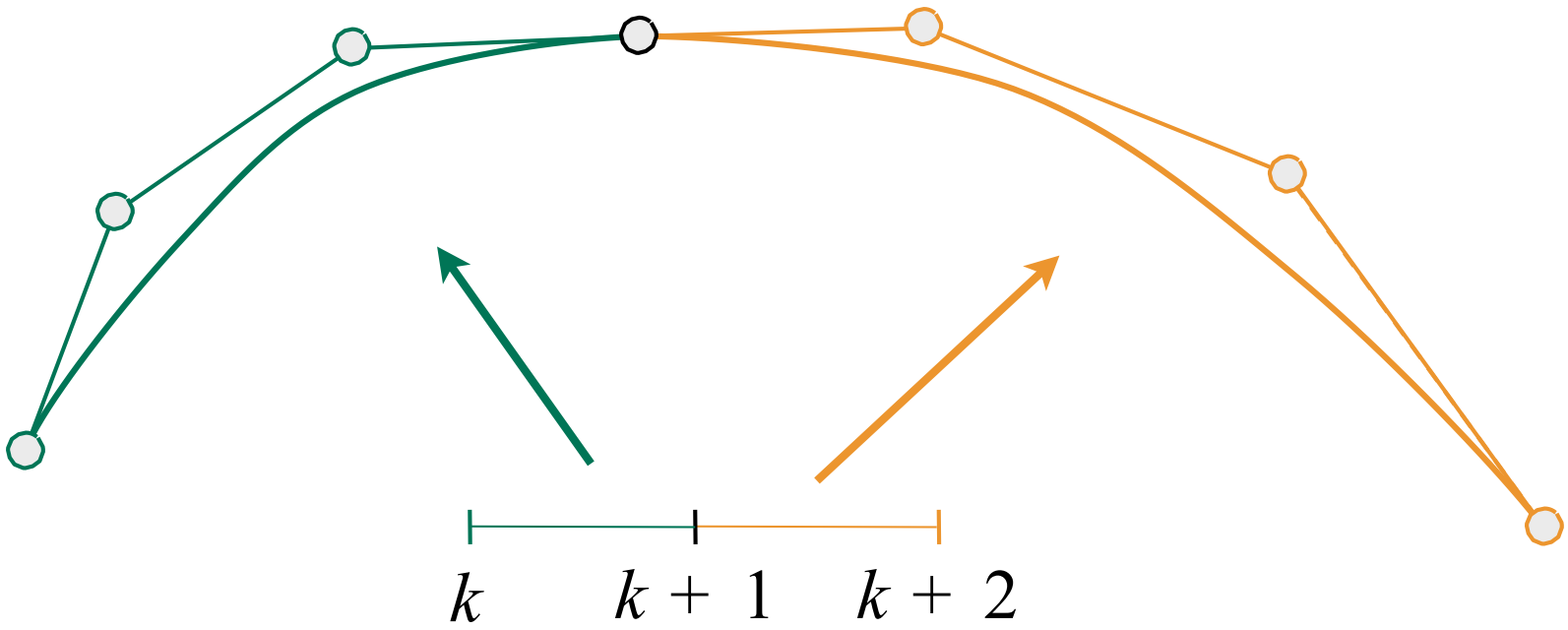
Piecewise Bézier Curve – Continuity

Two Bézier curves

$$\mathbf{a} : [k, k+1] \longrightarrow \mathbb{R}^N$$

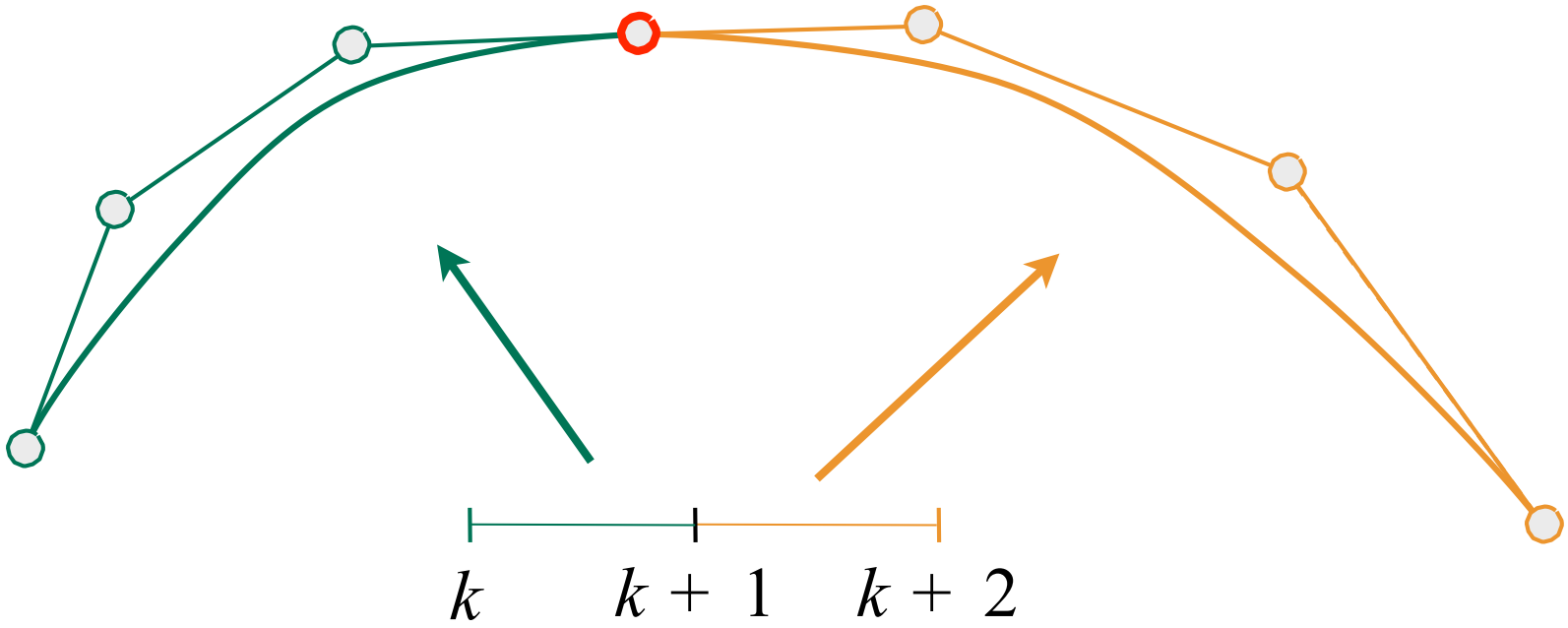
$$\mathbf{b} : [k+1, k+2] \longrightarrow \mathbb{R}^N$$

Assuming integer partitions here, can generalize



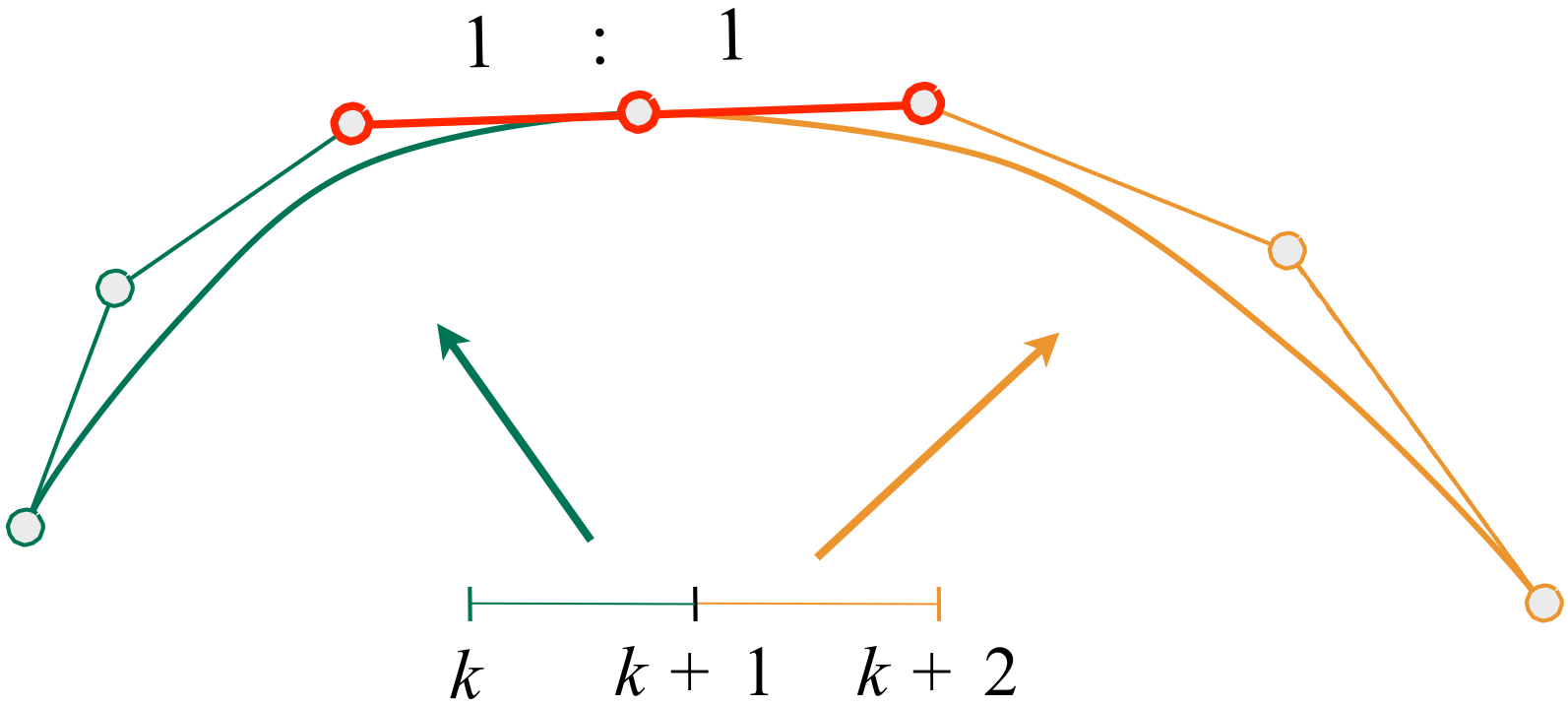
Piecewise Bézier Curve – Continuity

C^0 continuity: $\mathbf{a}_n = \mathbf{b}_0$

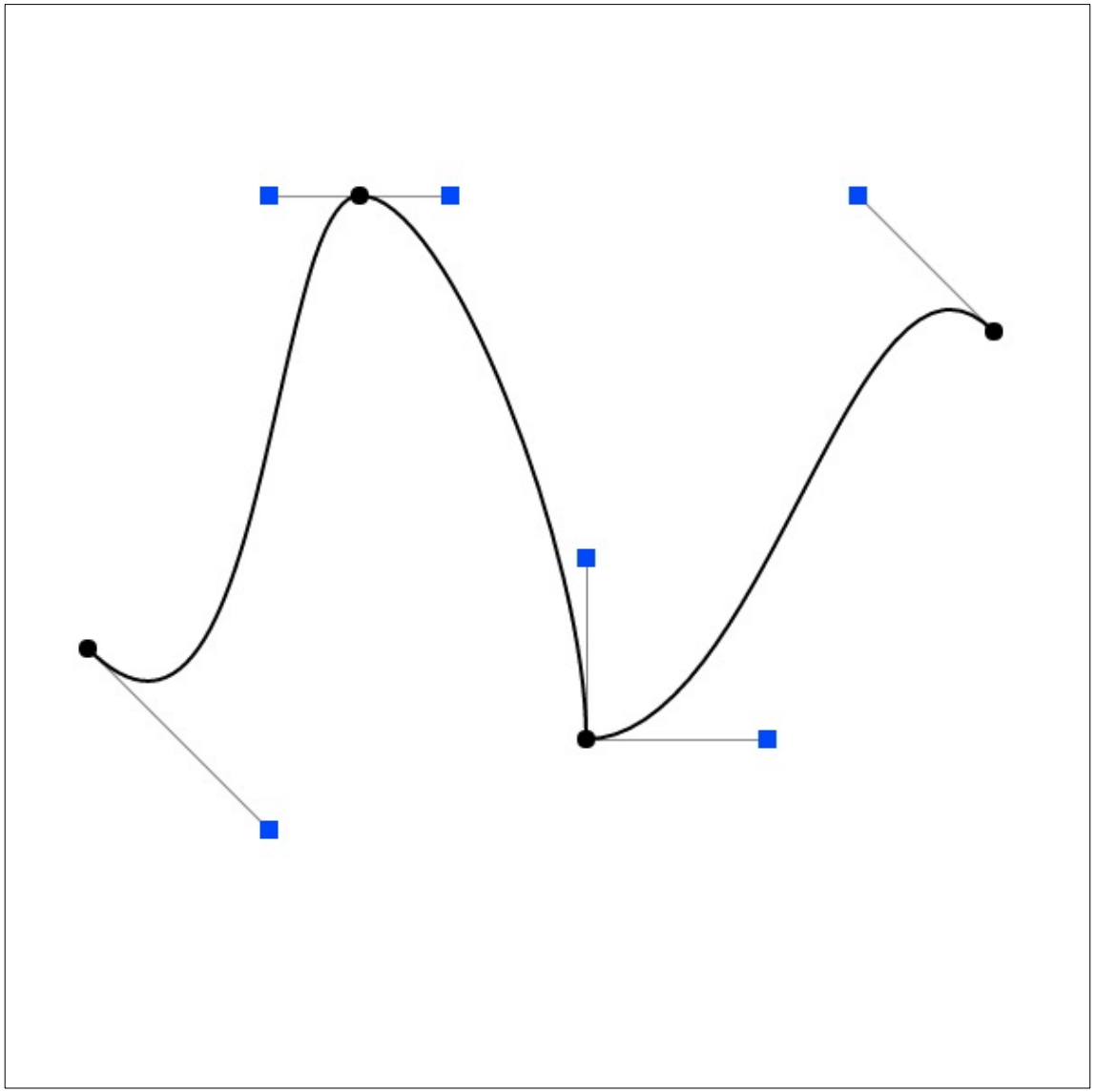


Piecewise Bézier Curve – Continuity

C^1 continuity: $a_n = b_0 = \frac{1}{2}(a_{n-1} + b_1)$



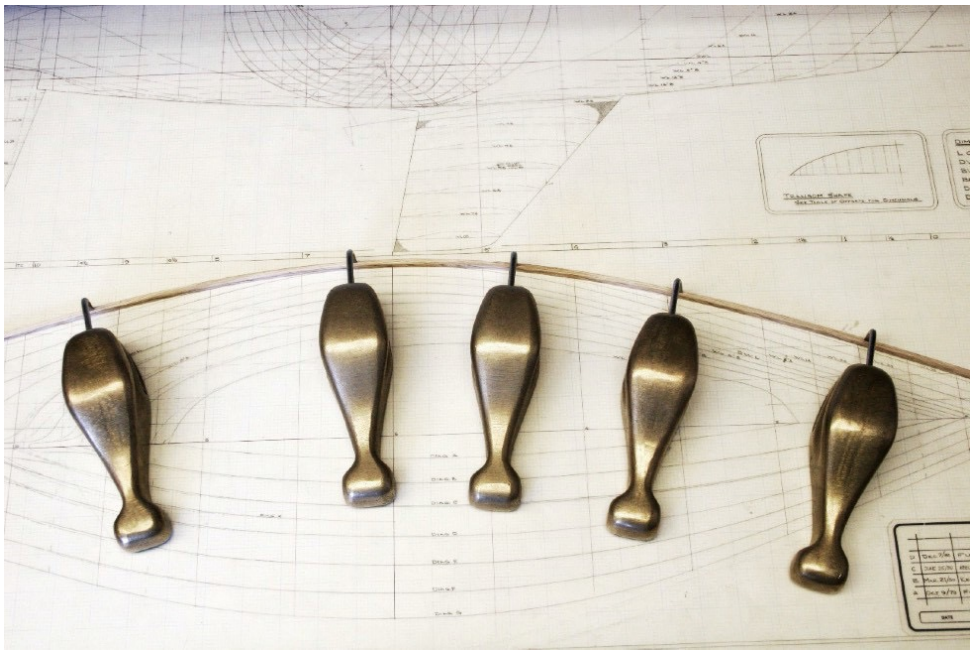
Demo – Piecewise Cubic Bézier Curve



David Eck, http://math.hws.edu/eck/cs424/notes2013/canvas/bezier.html

Other types of splines

- Spline
  - a continuous curve constructed so as to pass through a given set of points and have a certain number of continuous derivatives.
  - In short, a curve under control



Other types of splines

- B-splines
  - Short for basis splines
  - Require more information than Bezier curves
  - Satisfy all important properties that Bézier curves have (i.e. superset)

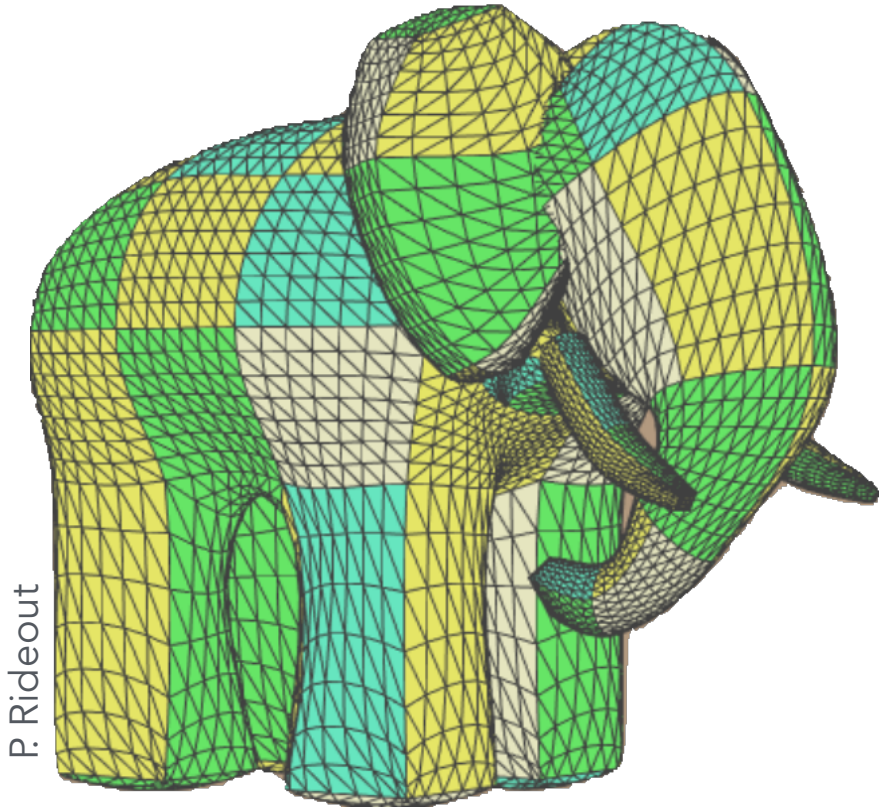
https://en.wikipedia.org/wiki/B-spline

Today

- Curves
  - Bezier curves
  - De Casteljau's algorithm
  - B-splines, etc.
- Surfaces
  - Bezier surfaces
  - Subdivision surfaces (triangles & quads)

Bézier Surfaces

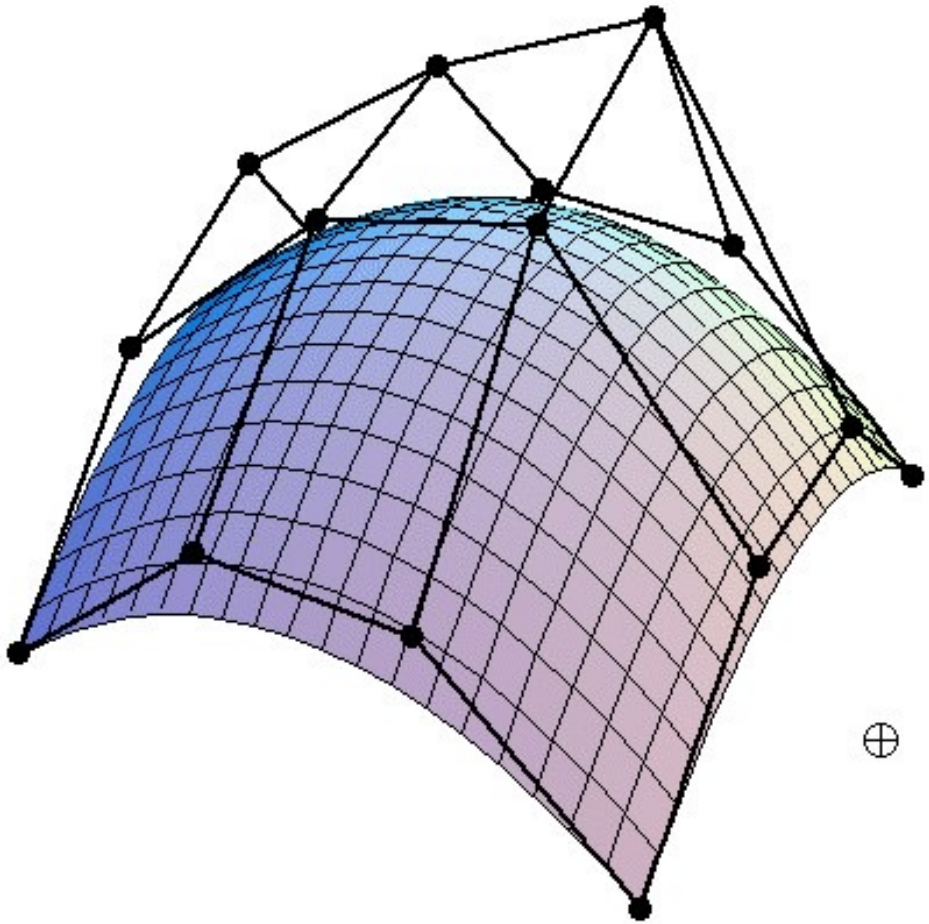
Extend Bézier curves to surfaces



Ed Catmull's "Gumbo" model

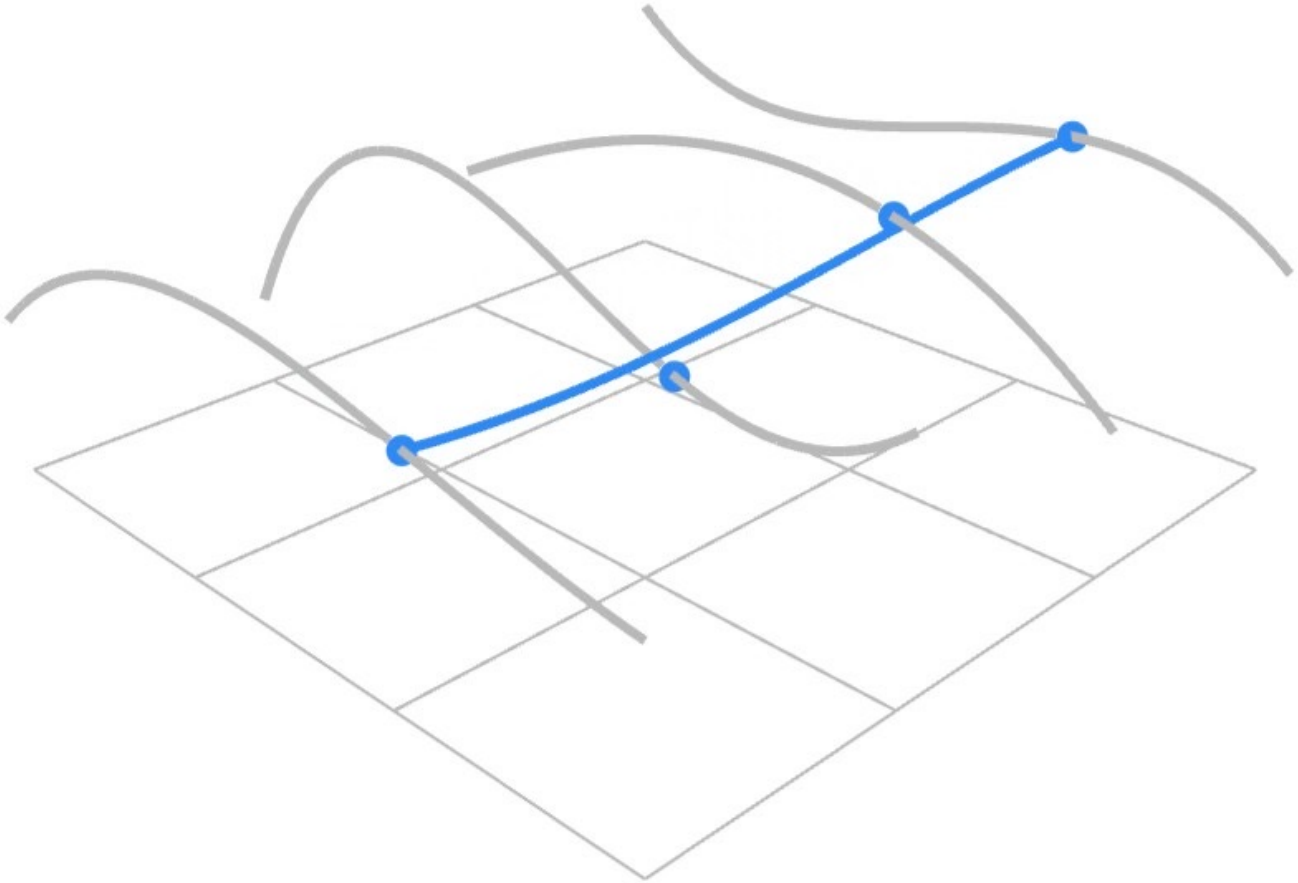
Utah Teapot

Bicubic Bézier Surface Patch



Bezier surface and 4 x 4 array of control points

Visualizing Bicubic Bézier Surface Patch



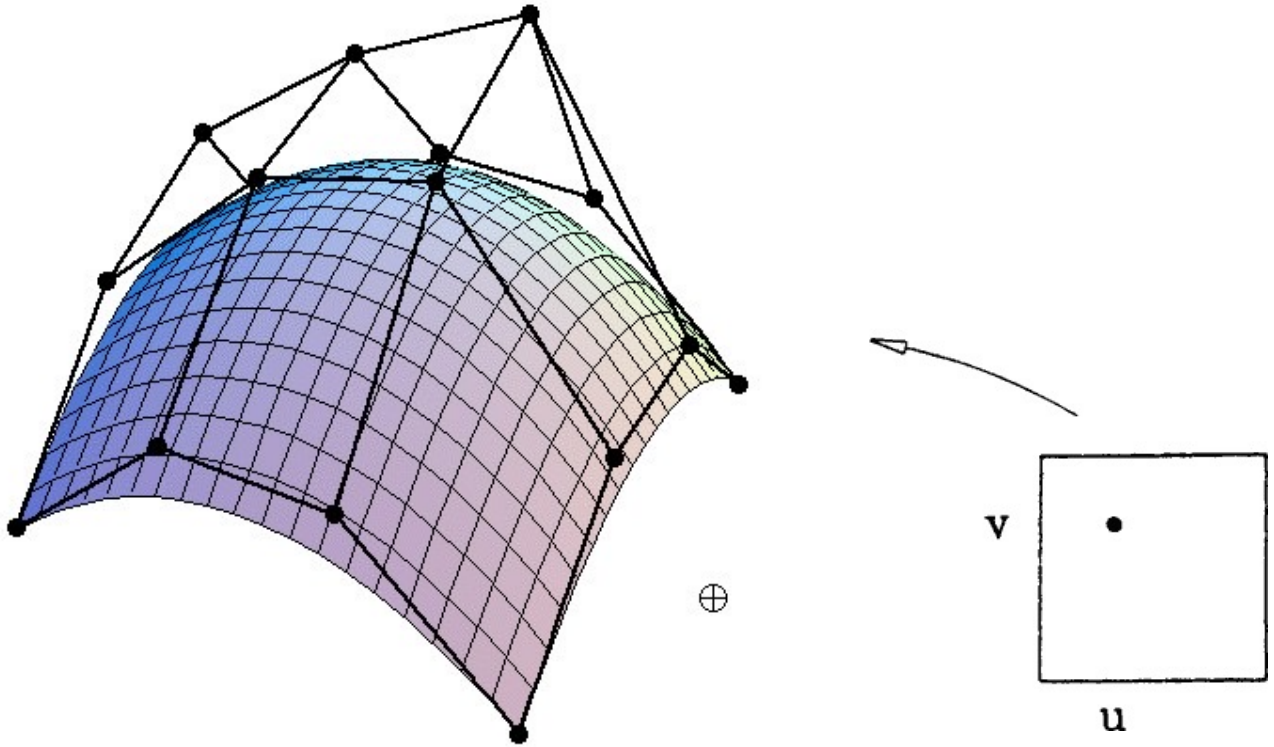
Evaluating Bézier Surfaces

Evaluating Surface Position For Parameters (u,v)

For bi-cubic Bezier surface patch,

Input: 4x4 control points

Output is 2D surface parameterized by (u,v) in $[0,1]^2$

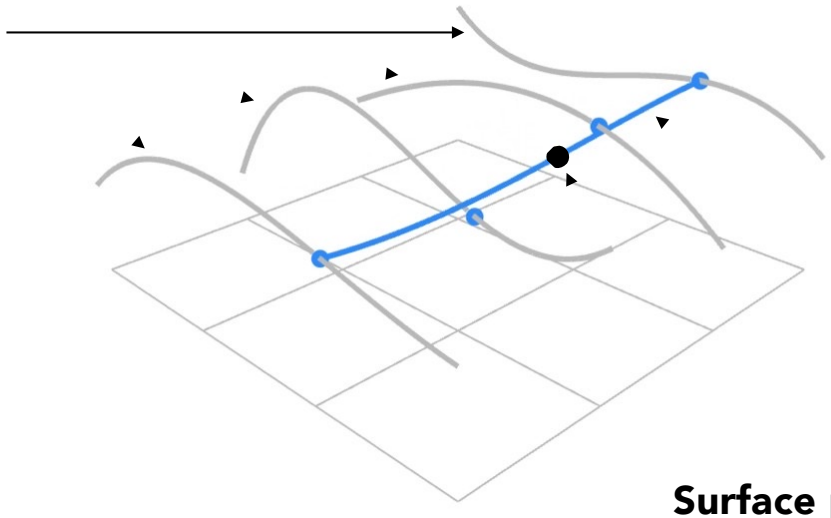


Method: Separable 1D de Casteljau Algorithm

Goal: Evaluate surface position corresponding to (u,v)

(u,v)-separable application of de Casteljau algorithm

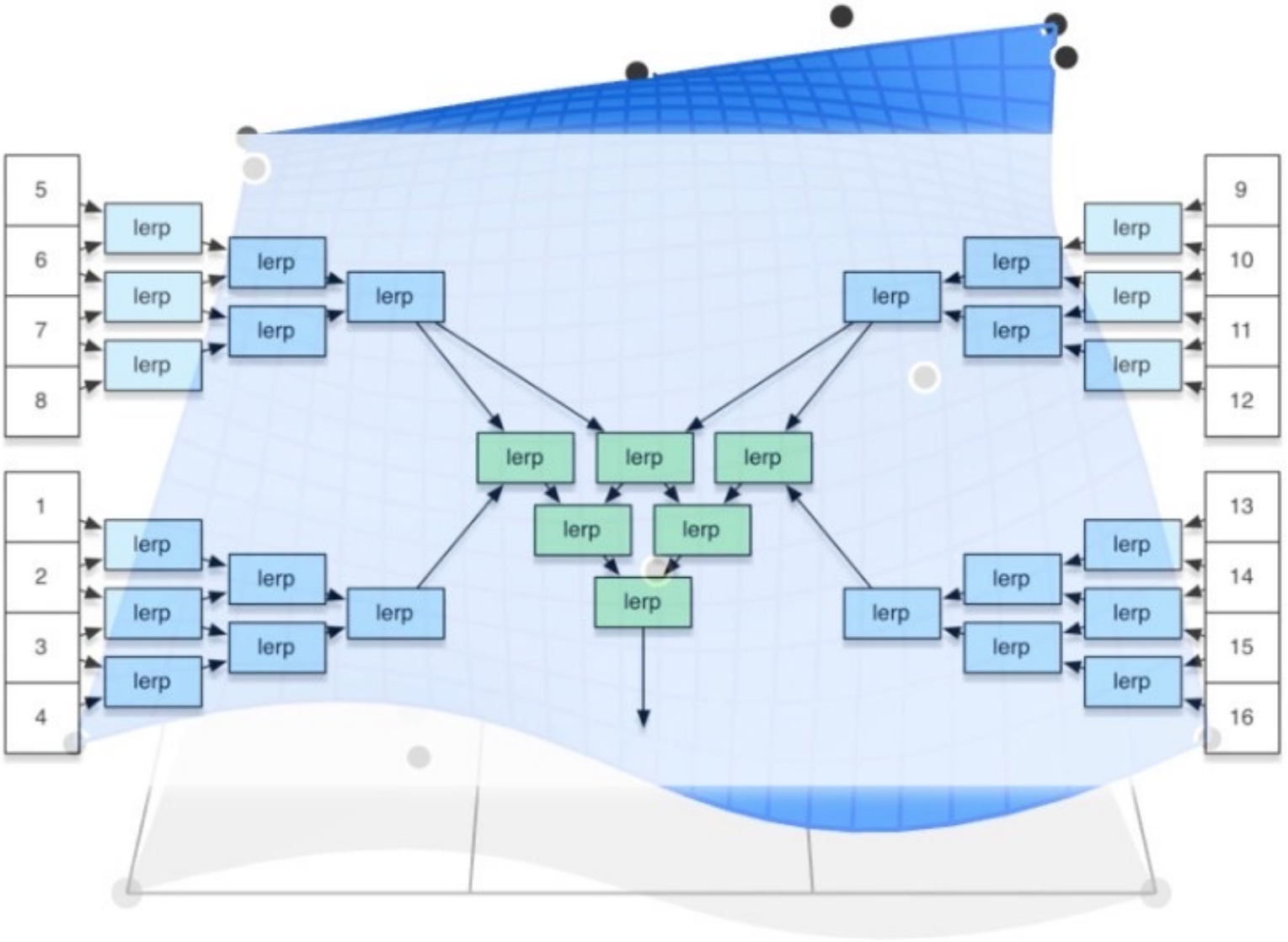
- Use de Casteljau to evaluate point u on each of the 4 Bezier curves in u. This gives 4 control points for the "moving" Bezier curve
- Use 1D de Casteljau to evaluate point v on the "moving" curve



"Moving" Bezier curve

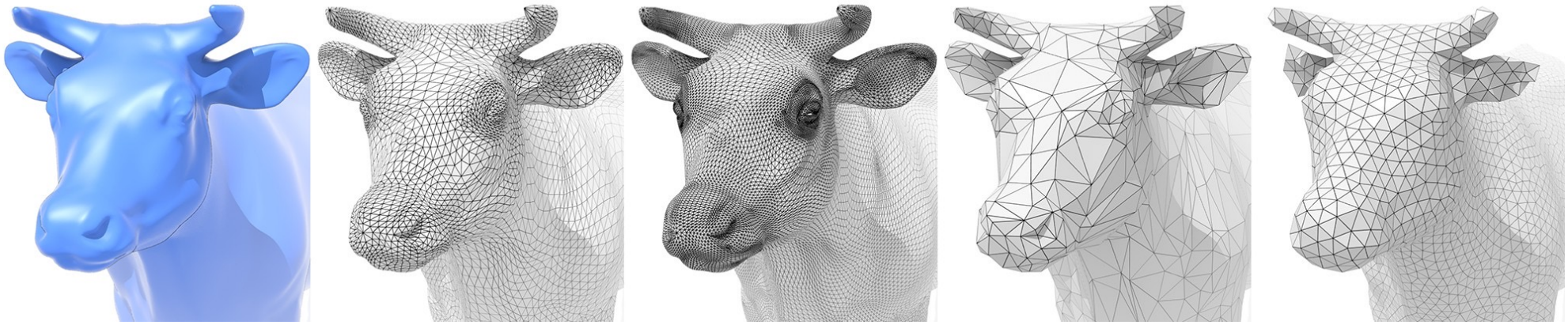
Surface point (u,v)

Method: Separable 1D de Casteljau Algorithm



Mesh Operations: Geometry Processing

- Mesh subdivision
- Mesh simplification
- Mesh regularization



Thank you!

(And thank Prof. Lingqi Yan, Prof. Ravi Ramamoorthi and Prof. Ren Ng for many of the slides!)