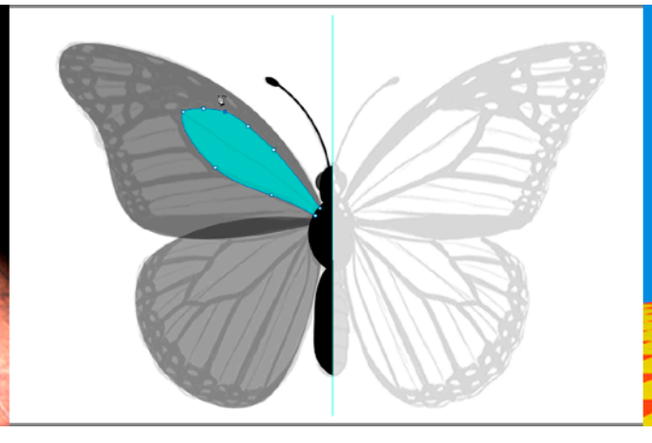
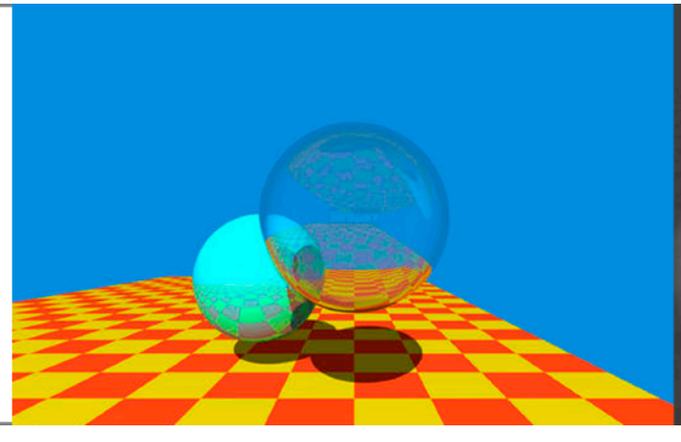
Introduction to Computer Graphics

GAMES101, Lingqi Yan, UC Santa Barbara

Lecture 11:

Geometry 2 (Curves and Surfaces)


http://www.cs.ucsb.edu/~lingqi/teaching/games101.html

Announcements

- Homework 3 deadline has been extended
  - To Thursday 23:59PM, Beijing time
- COVID-19 is getting worse in the US
  - Be careful, dude
  - Have to stream at home, network & lighting are worse

Last Lecture

- Introduction to geometry
  - Examples of geometry
  - Various representations of geometry
    - Implicit
    - Explicit

Today

- Explicit Representations
- Curves
  - Bezier curves
  - De Casteljau’s algorithm
  - B-splines, etc.
- Surfaces
  - Bezier surfaces
  - Triangles & quads
    - Subdivision, simplification, regularization

Explicit Representations in Computer Graphics

直接给出点的位置，或者可以进行参数映射;然而想要判断内外时，显式的表达就很难进行表示

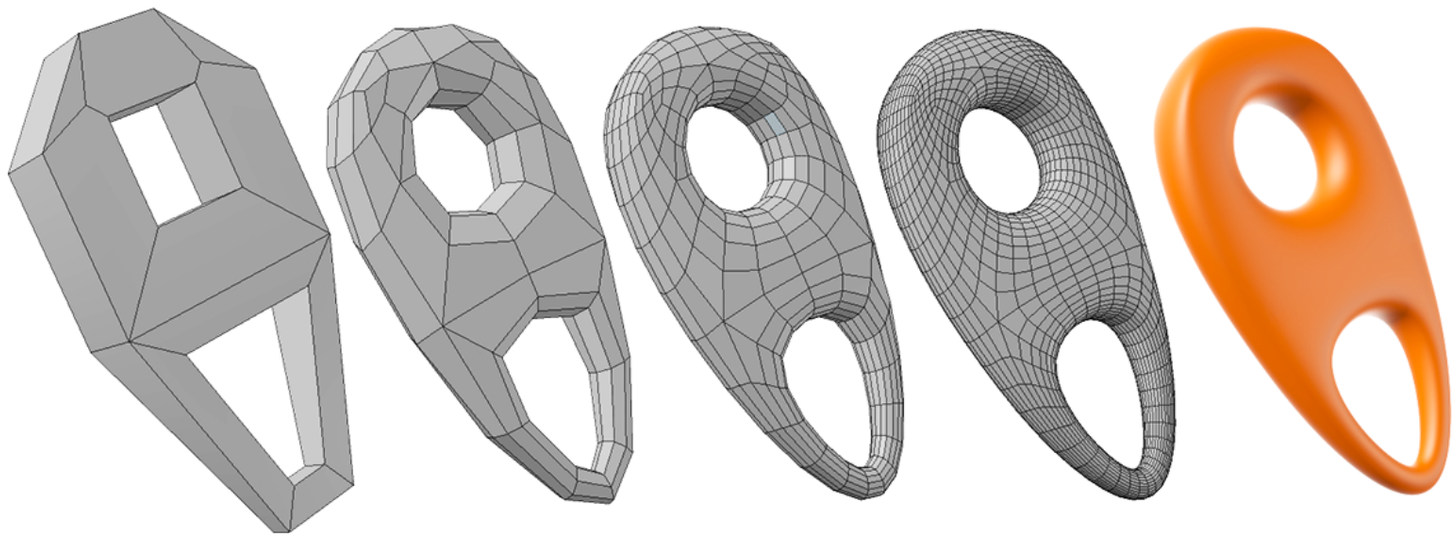
Many Explicit Representations in Graphics

triangle meshes

Bezier surfaces

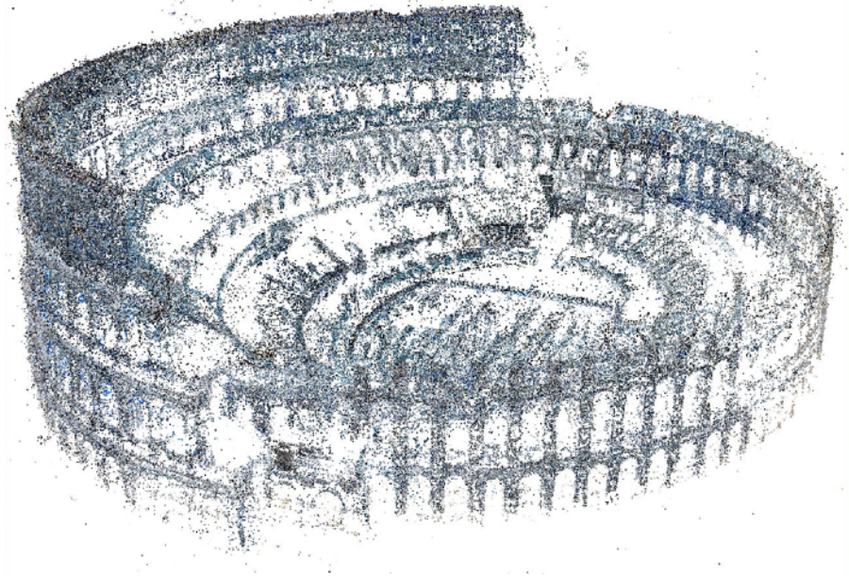
subdivision surfaces

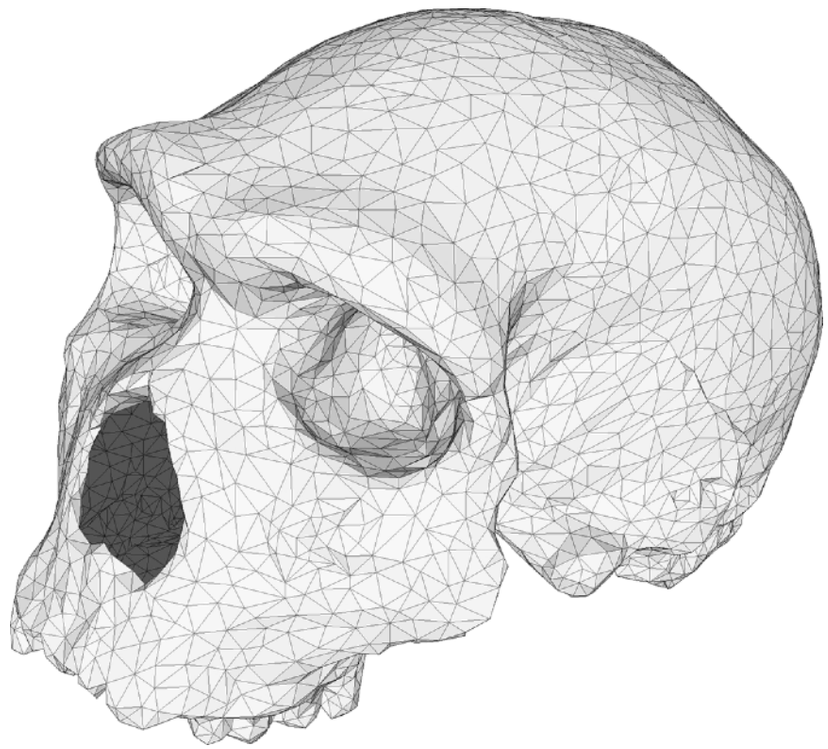
NURBS



point clouds

…





Point Cloud (Explicit)

Easiest representation: list of points (x,y,z)

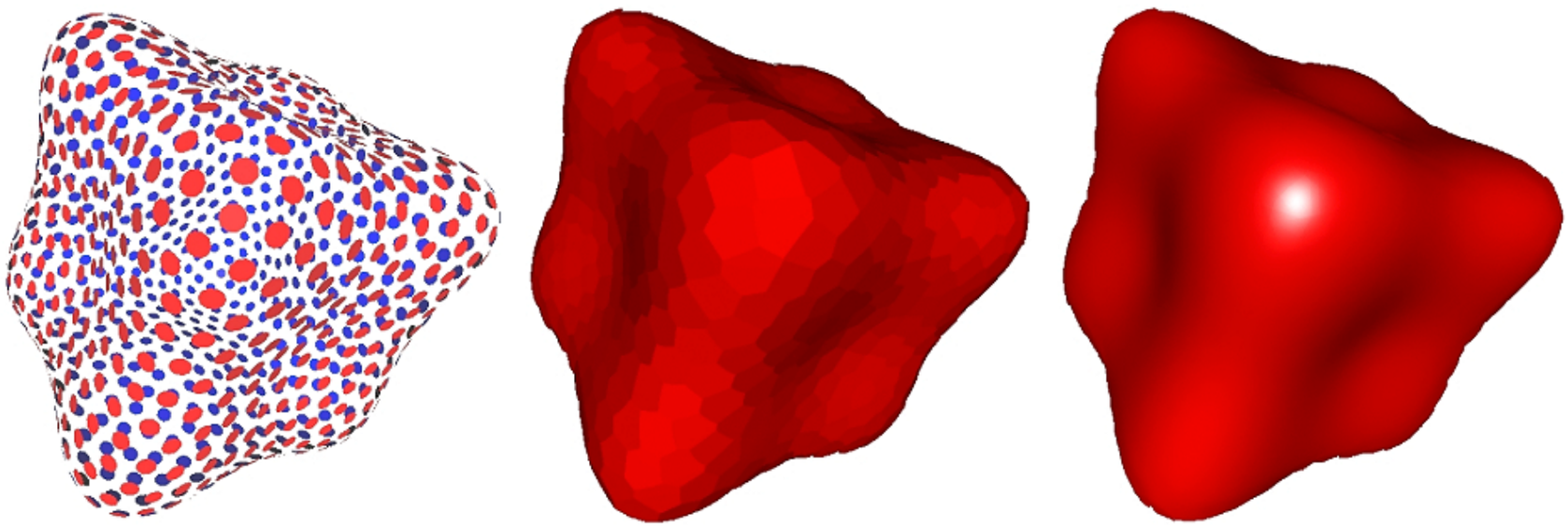
Easily represent any kind of geometry

Useful for LARGE datasets (>>1 point/pixel)

Often converted into polygon mesh

Difficult to draw in undersampled regions

用空间中一堆点的集合来表示物体，只要点足够密集，就看不到点与点之间的空隙，理论上可以表示任何几何，通常三维扫描得到的结果就是点云(点云可以转变为三角形)



Polygon Mesh (Explicit)

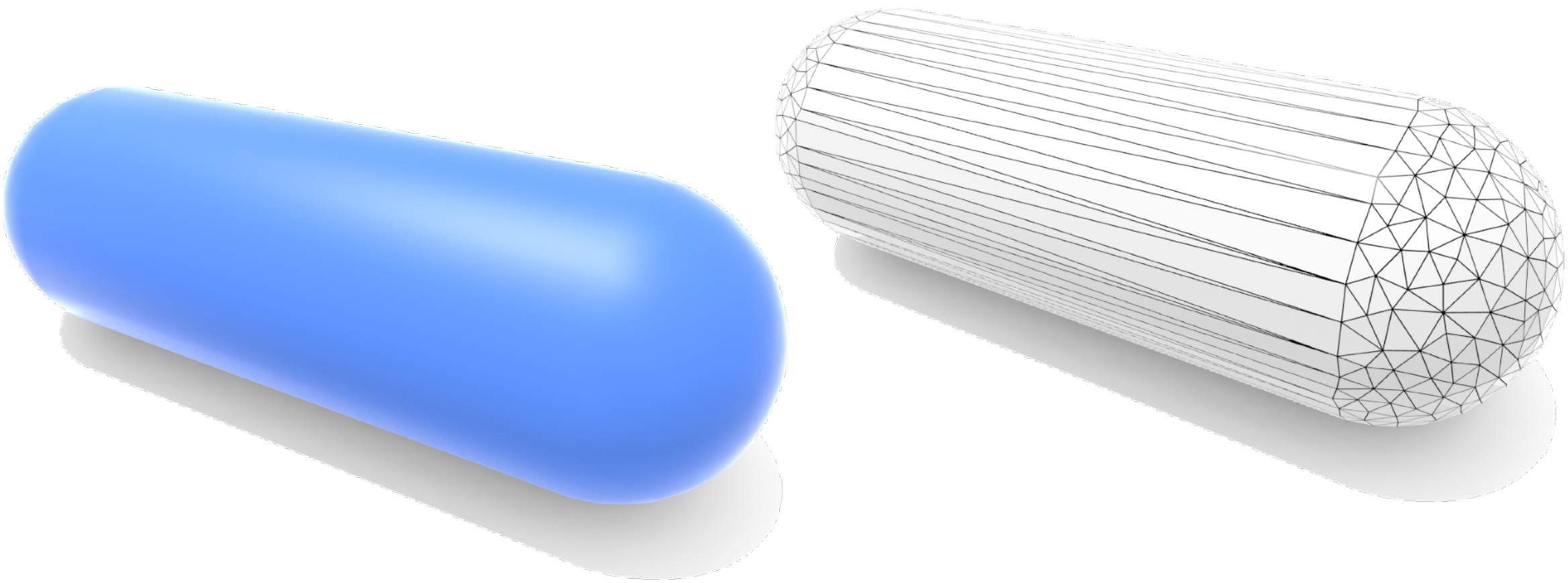
Store vertices & polygons (often triangles or quads)

Easier to do processing / simulation, adaptive sampling

More complicated data structures

Perhaps most common representation in graphics

用很多三角形或者多边形组成想要的曲面，是得到广泛运用的显示表示



The Wavefront Object File (.obj) Format

这里表示了一个立方体，上面有4大块内容首先3-10行定义了立方体8个顶点的位置信息，12-25表示了纹理坐标27-34表示了6个面的法线信息（重复的可忽略），36-47表示了各个三角形面的信息，比如其中5/1/1表示的是第5个顶点纹理坐标为1法线坐标为1，三个顶点的信息组成一个面f

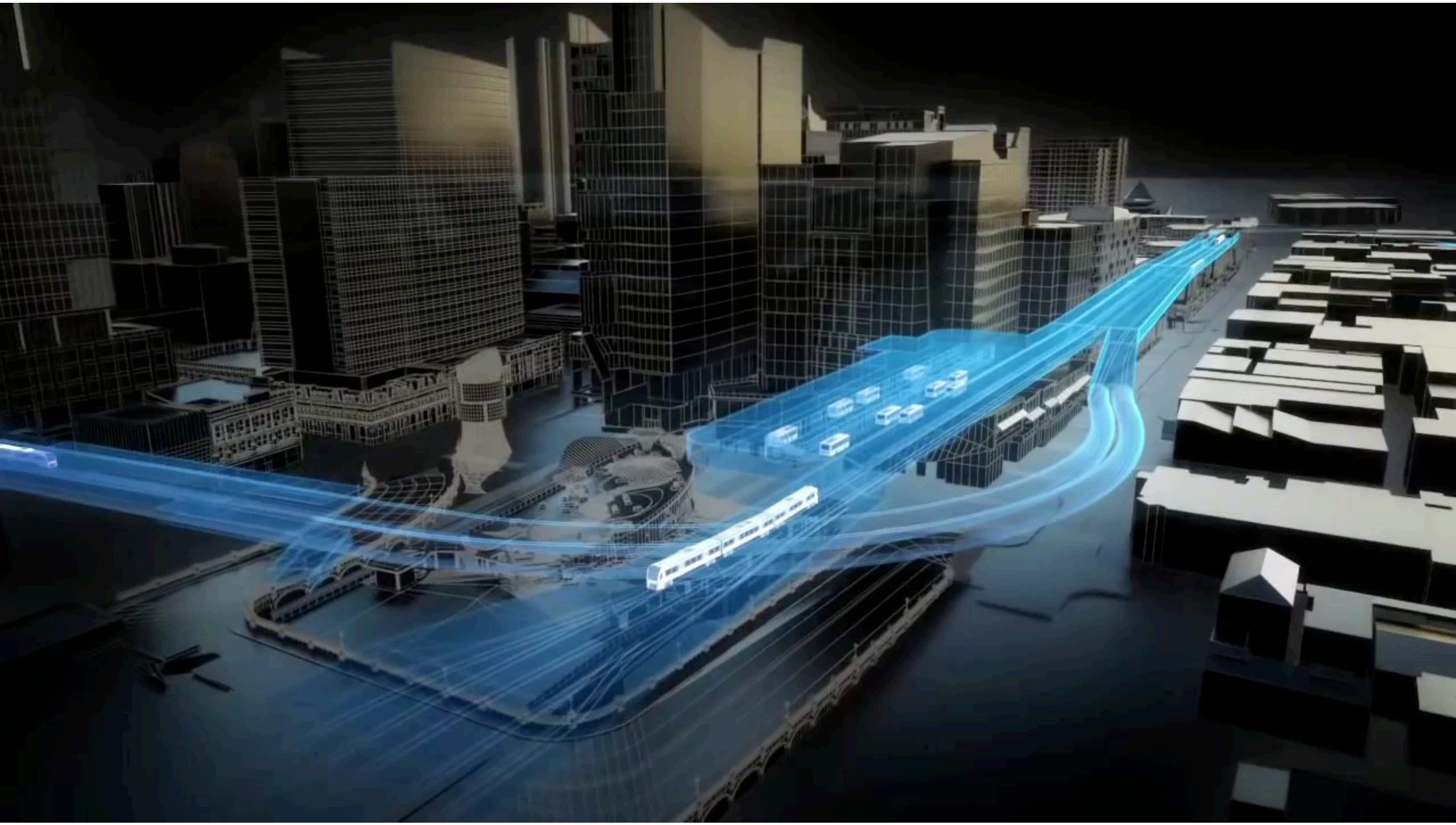
Commonly used in Graphics research

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
```

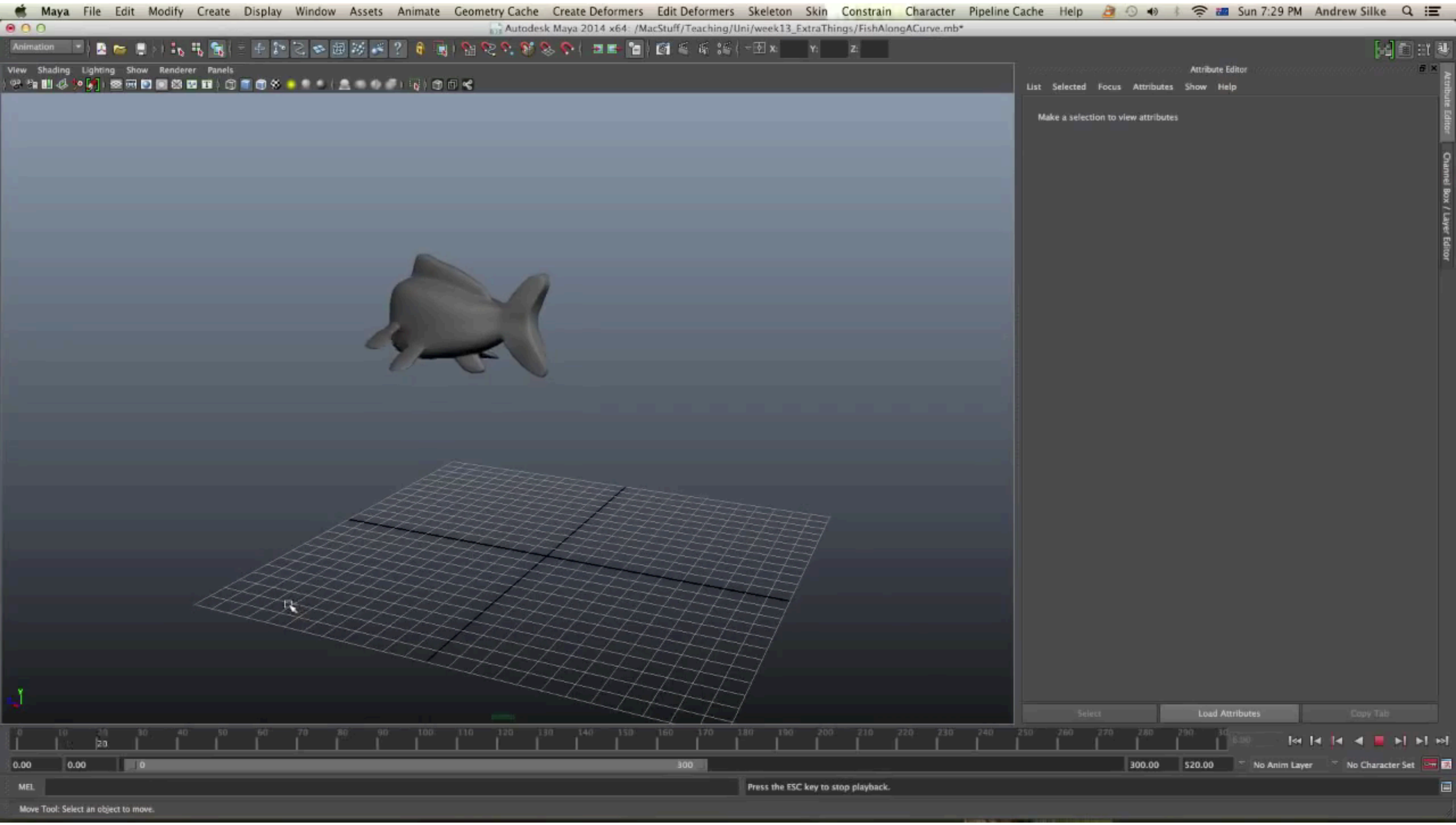
Curves

Camera Paths



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

Animation Curves

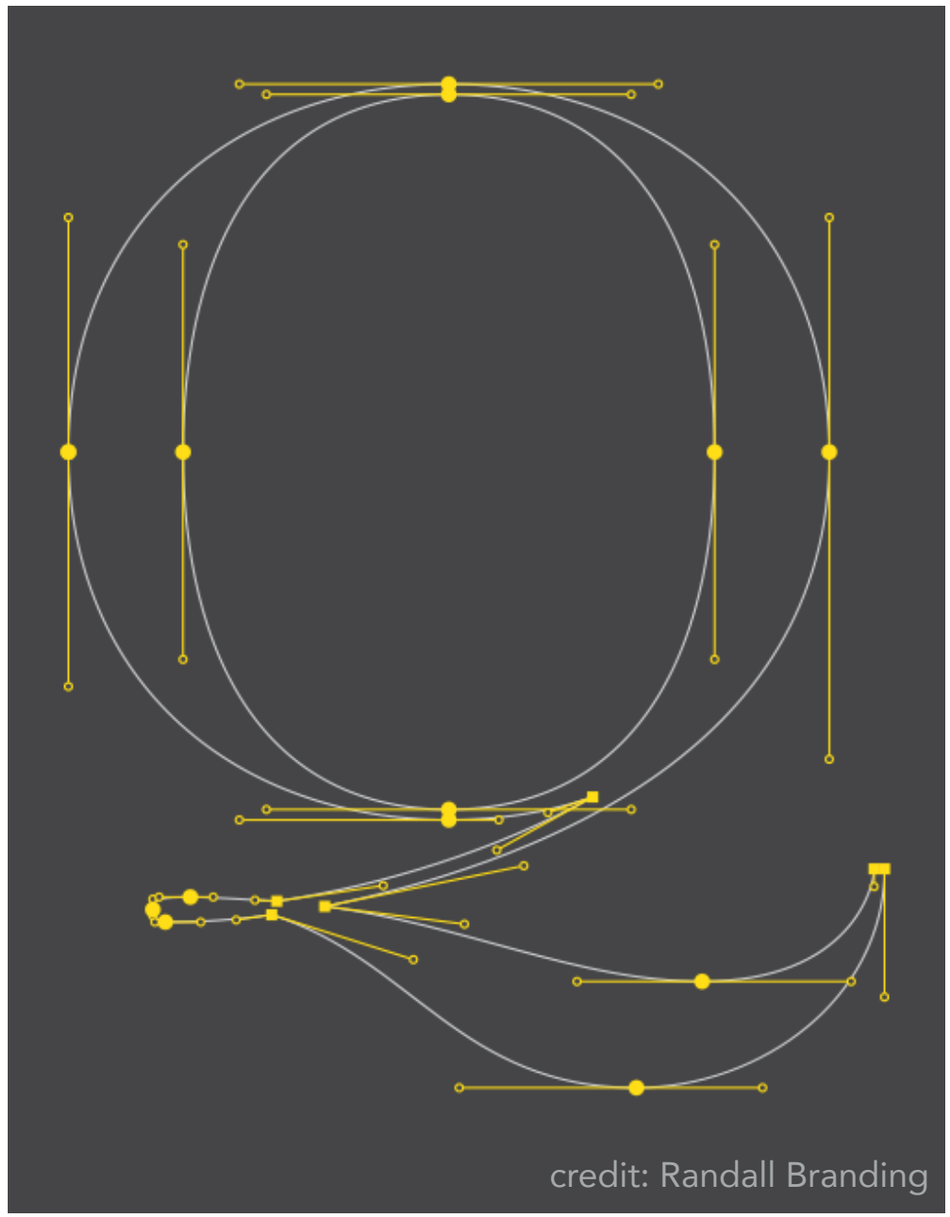


Maya Animation Tutorial: https://youtu.be/b-o5wtZlJPc

Vector Fonts

The Quick Brown Fox Jumps Over The Lazy Dog

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789



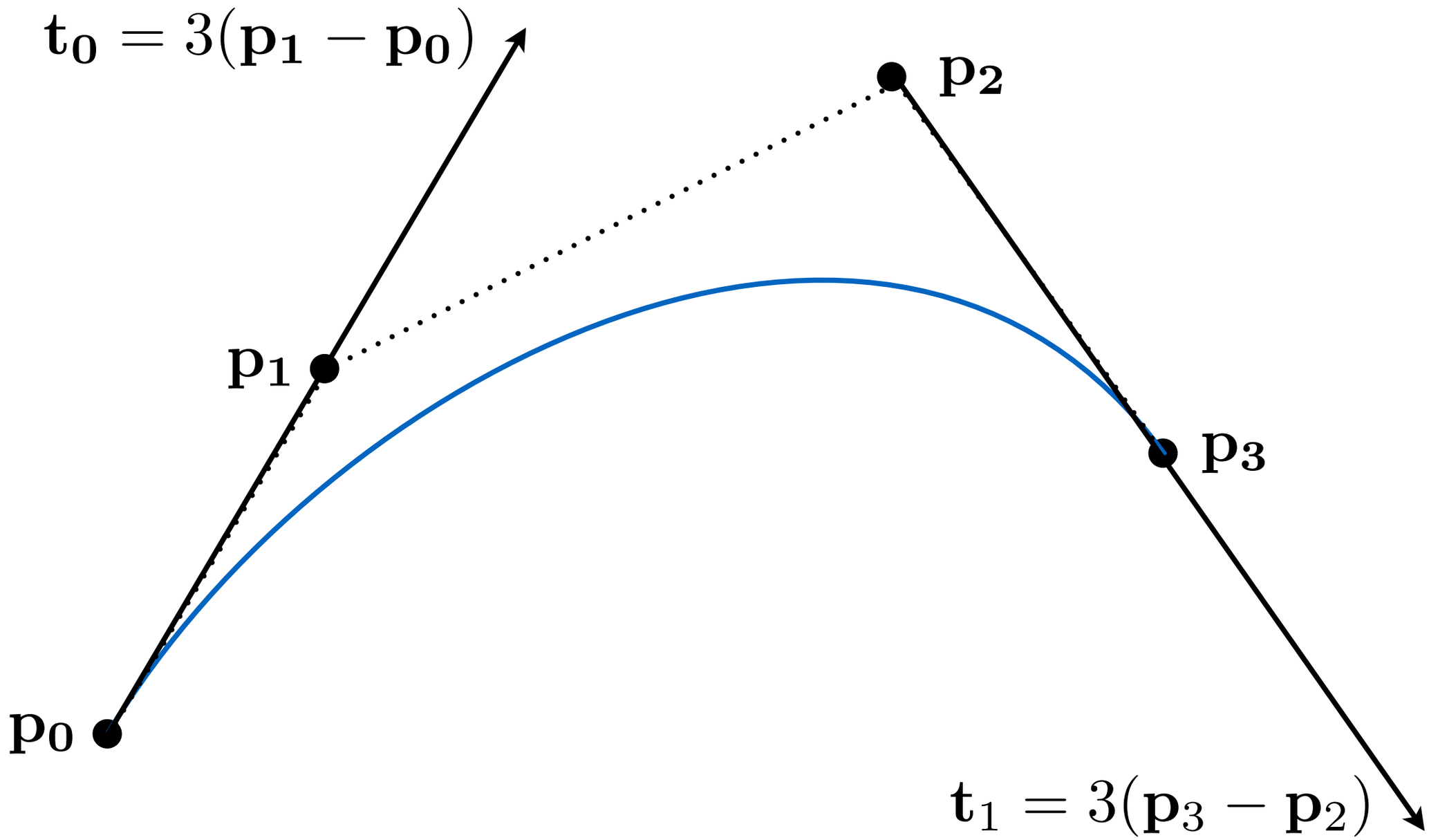
Baskerville font - represented as piecewise cubic Bézier curves

Bézier Curves

(贝塞尔曲线)

Defining Cubic Bézier Curve With Tangents

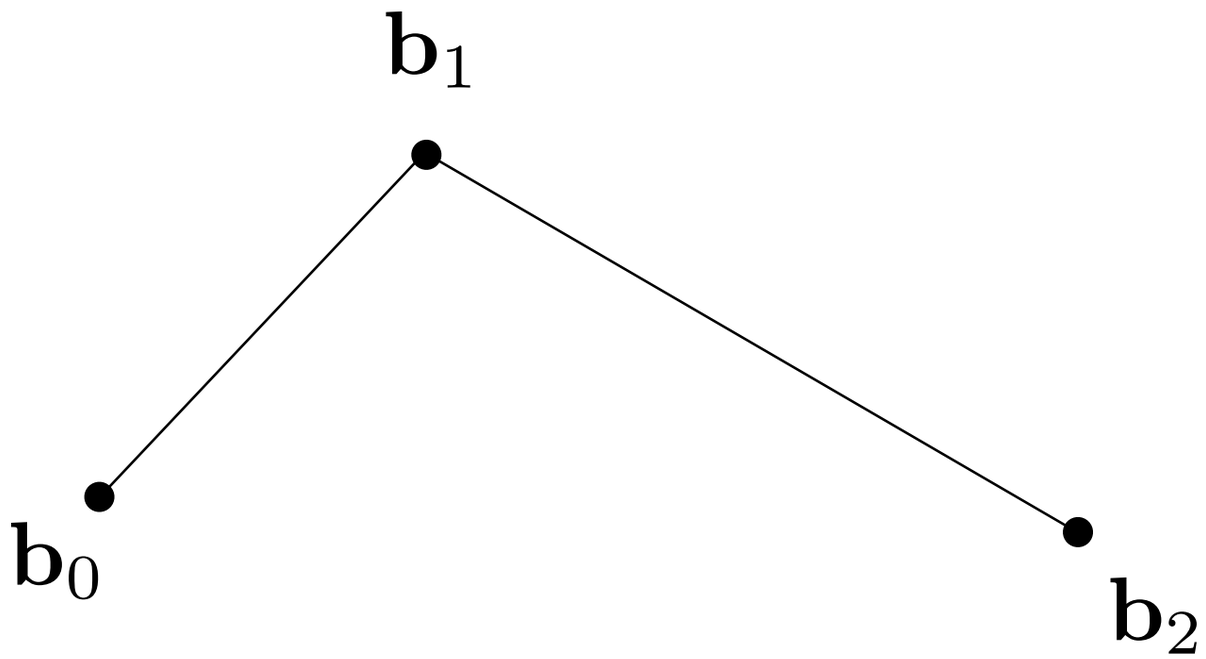
贝塞尔曲线是用一系列控制点来定义一条曲线，如图，通过这四个控制点可得曲线必须经过起始点和终点p0p3(不一定要经过所有点)并且起始方向和结束方向必须沿着t0t1切线。



Evaluating Bézier Curves (de Casteljau Algorithm)

Bézier Curves – de Casteljau Algorithm

Consider three points (quadratic Bezier)



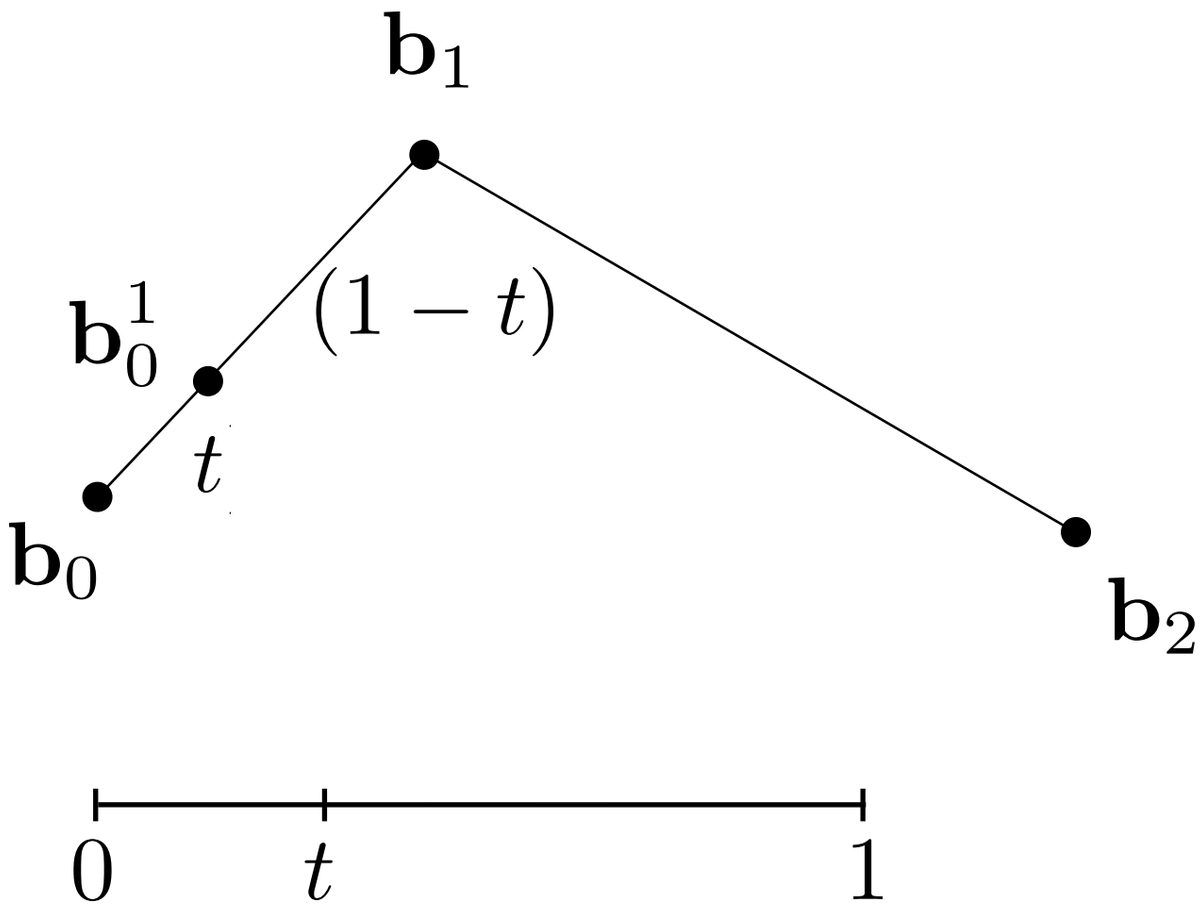
画一条曲线此时就相当于求给定时间t下的点的位置。

Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

Insert a point using linear interpolation



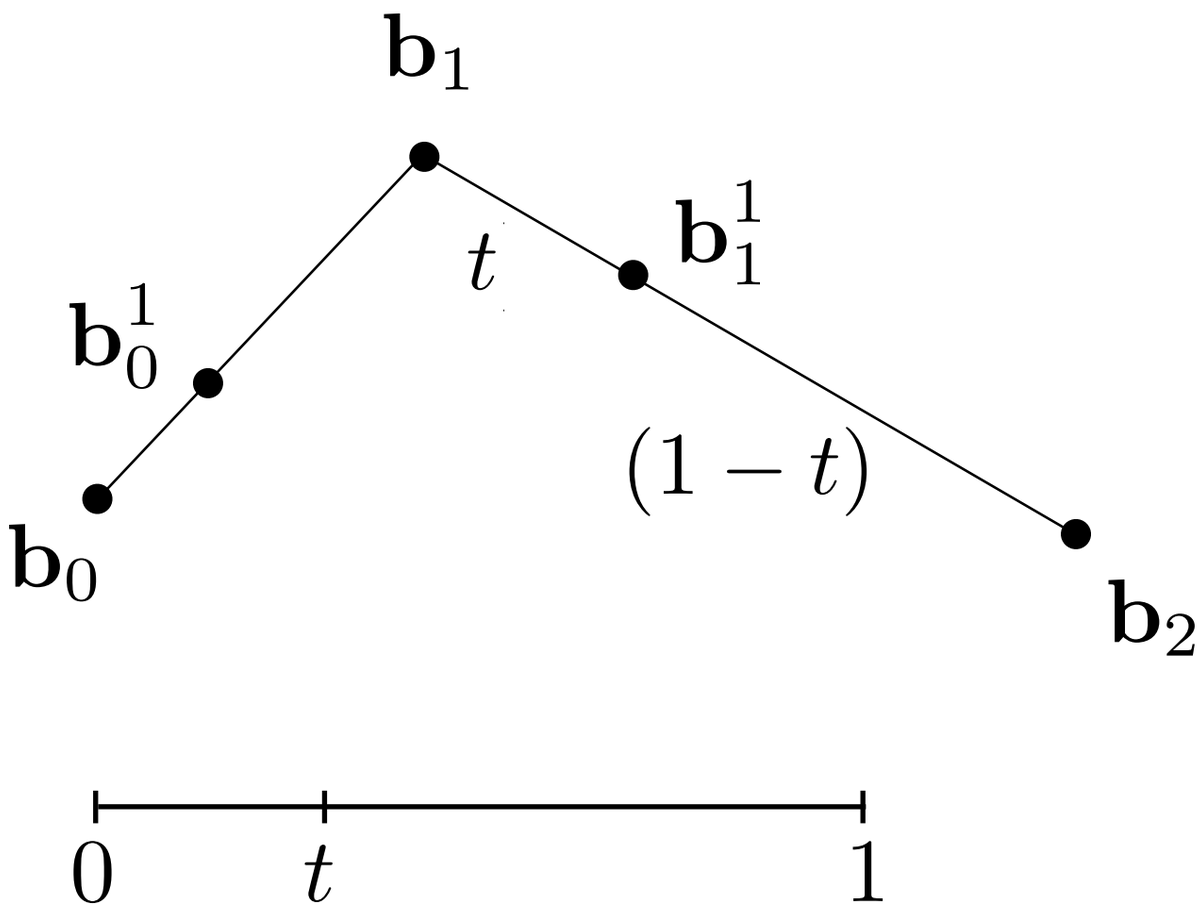
Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

我们将中间点 b1 看做是 b0b1 段的终点以及 b1b2 段的起点，找出直线段 b0b1 上 t 时刻位置的点 b01 ，以及另一线段 b1 b2 的点 b11 ，再将这两点看做起止点，再找出这两点之间的 t 时刻点 ，直到只剩唯一的一个 t 时刻点 b02，这个点就是曲线 b0b2 上 t 时刻的点

Insert on both edges

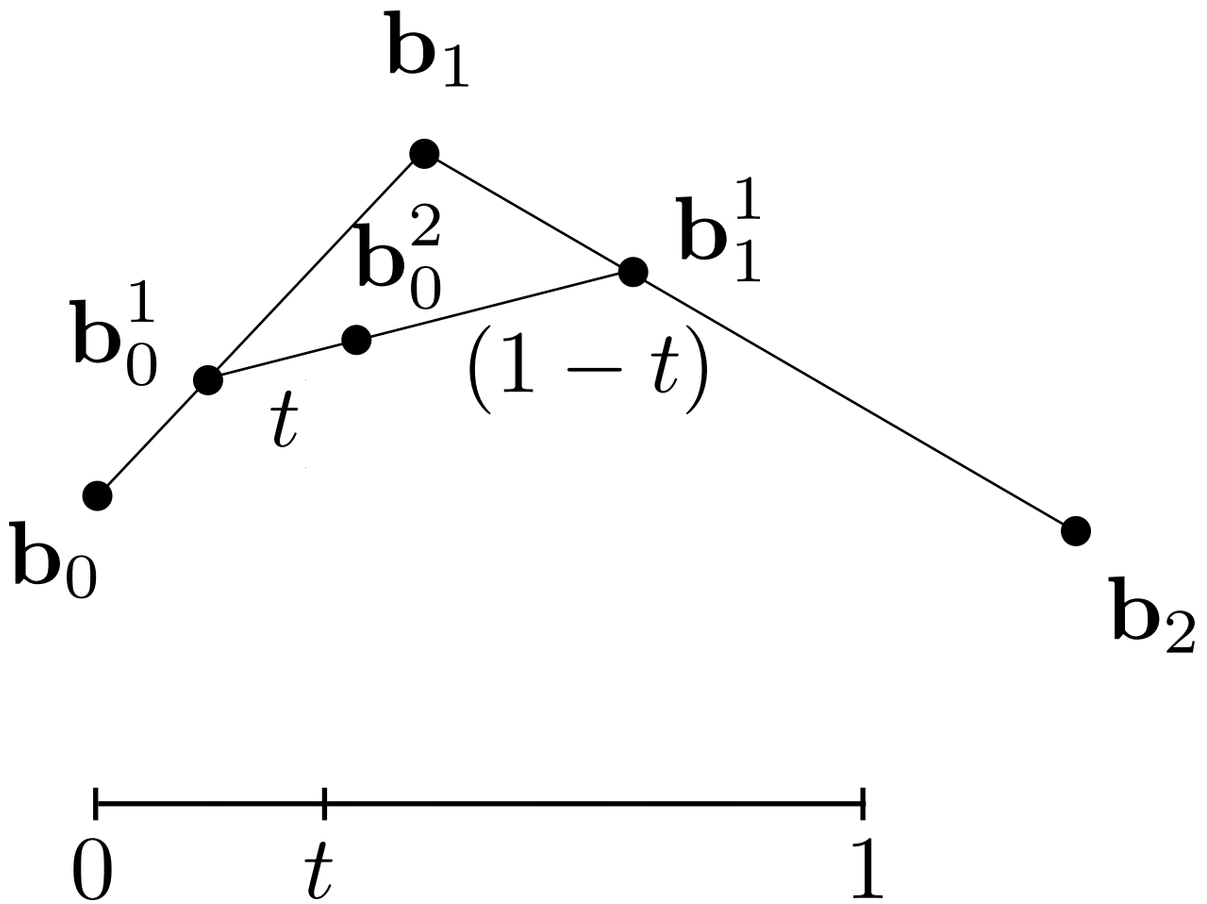


Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

Repeat recursively

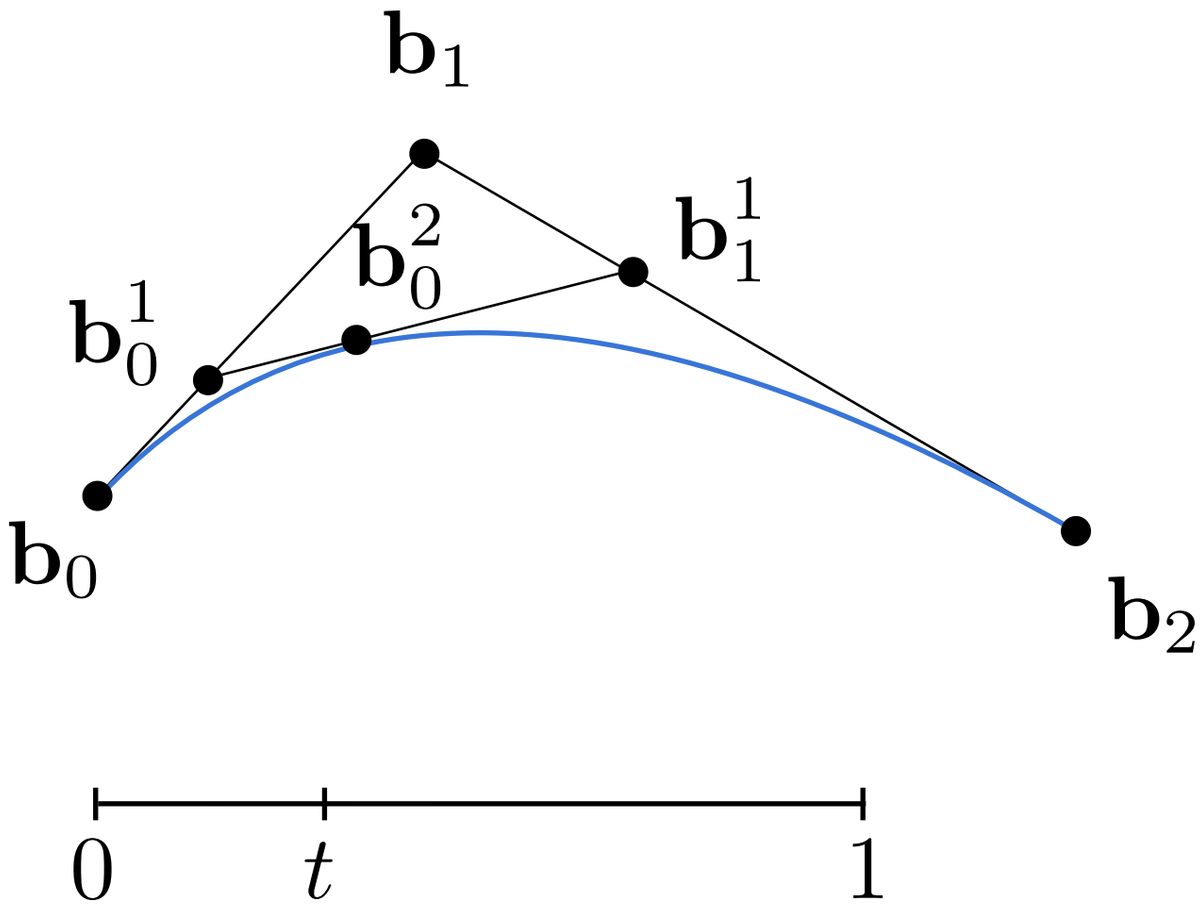


Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

Bézier Curves – de Casteljau Algorithm

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



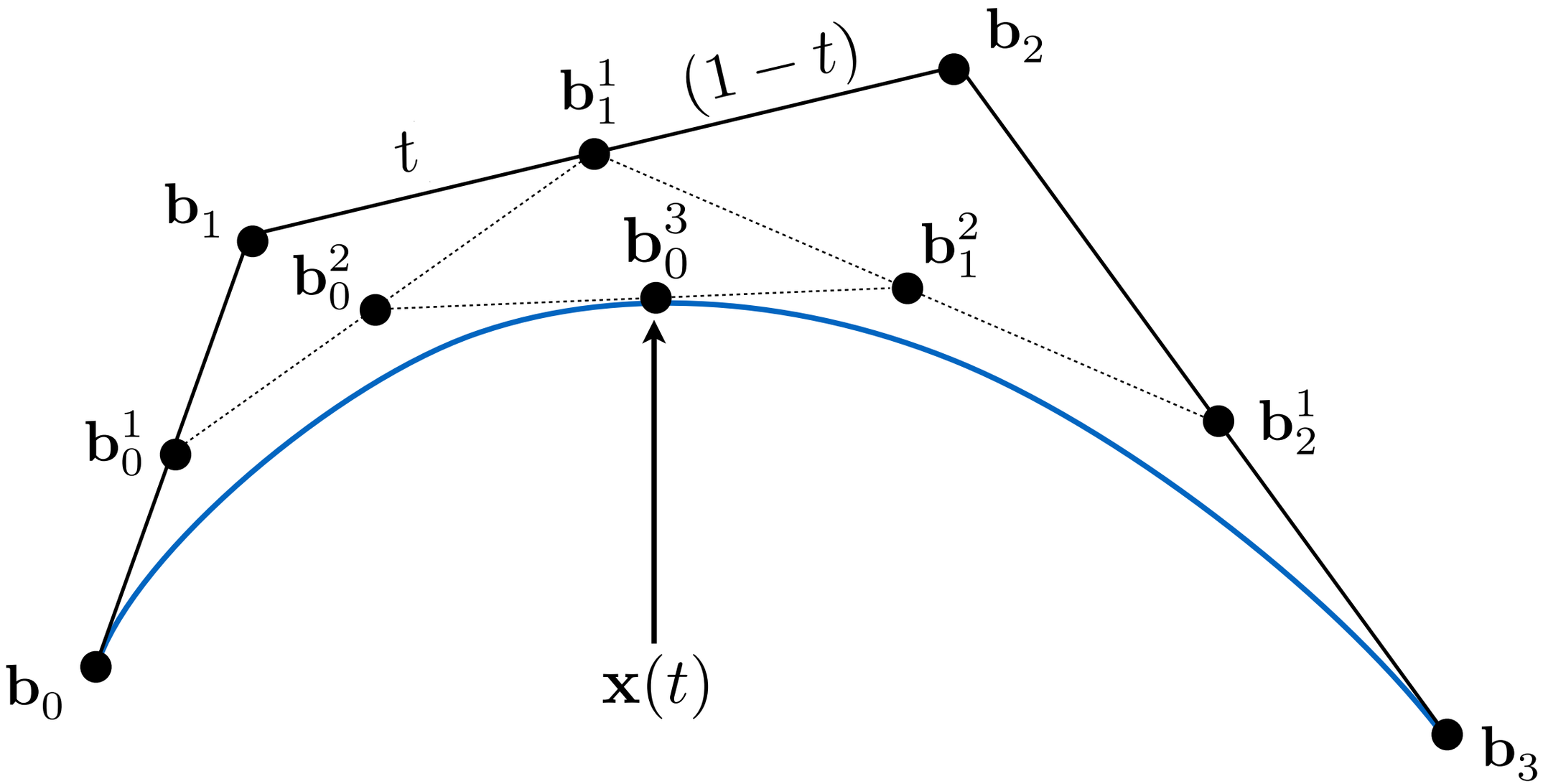
Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930

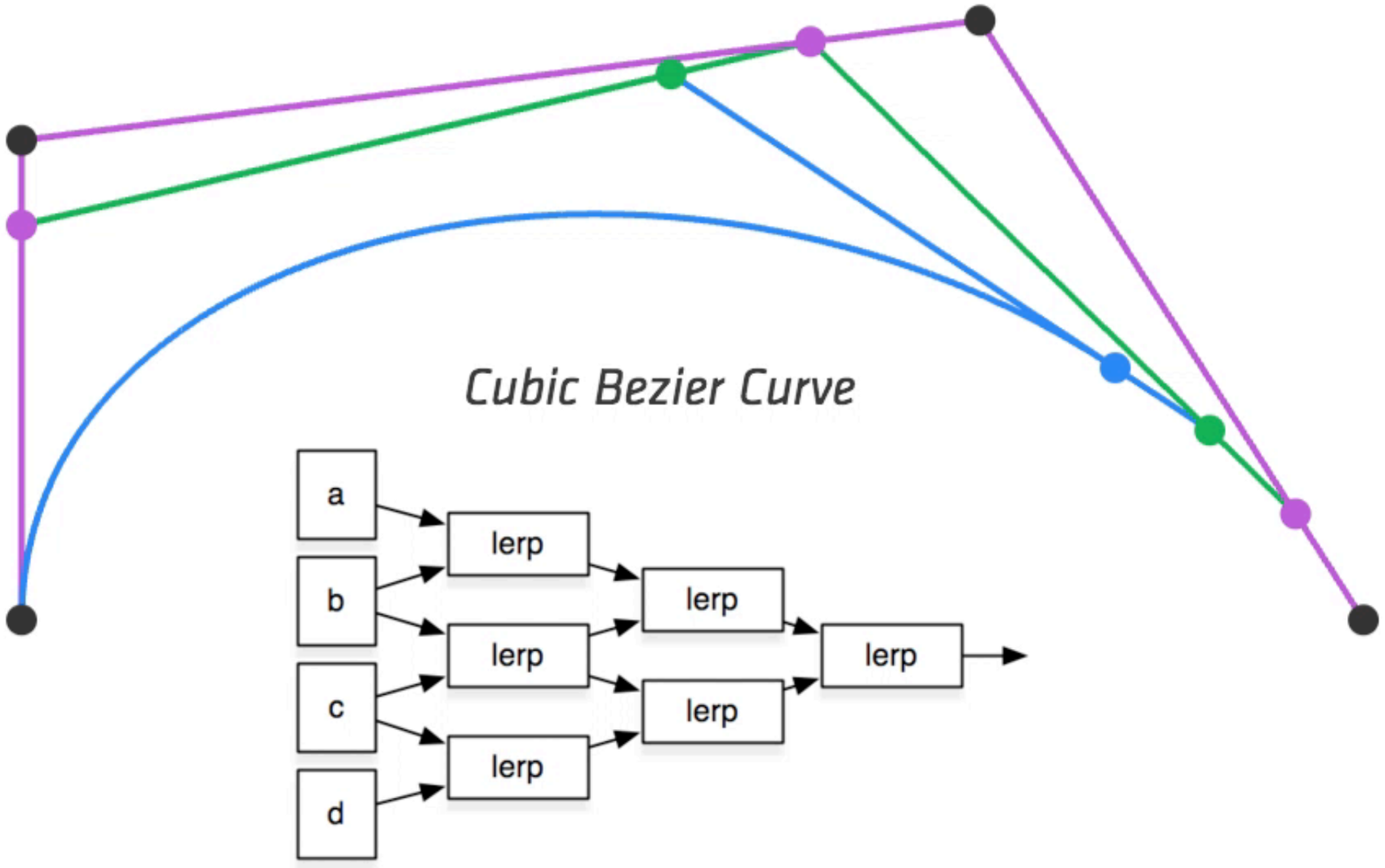
Cubic Bézier Curve – de Casteljau

Four input points in total

Same recursive linear interpolations



Visualizing de Casteljau Algorithm

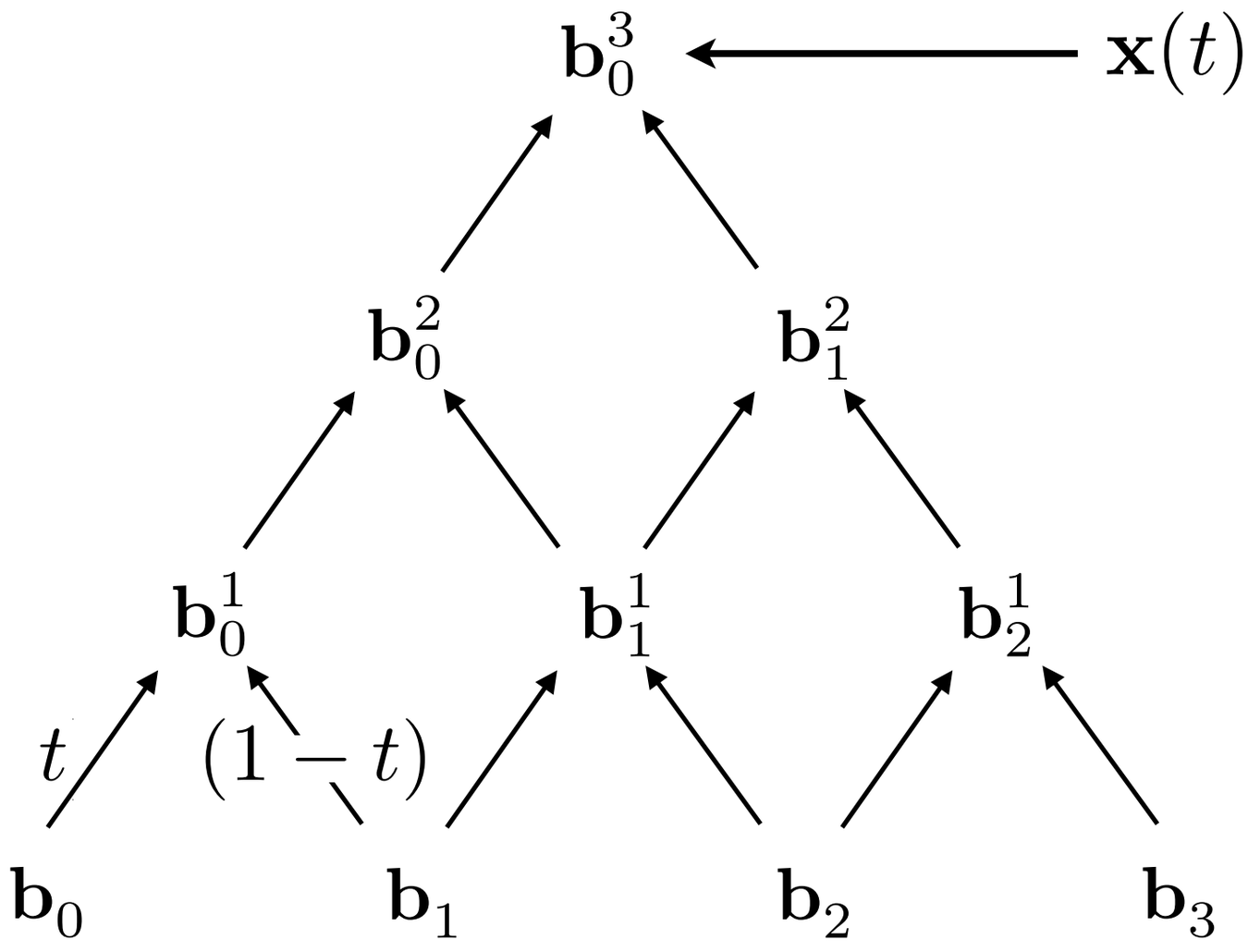


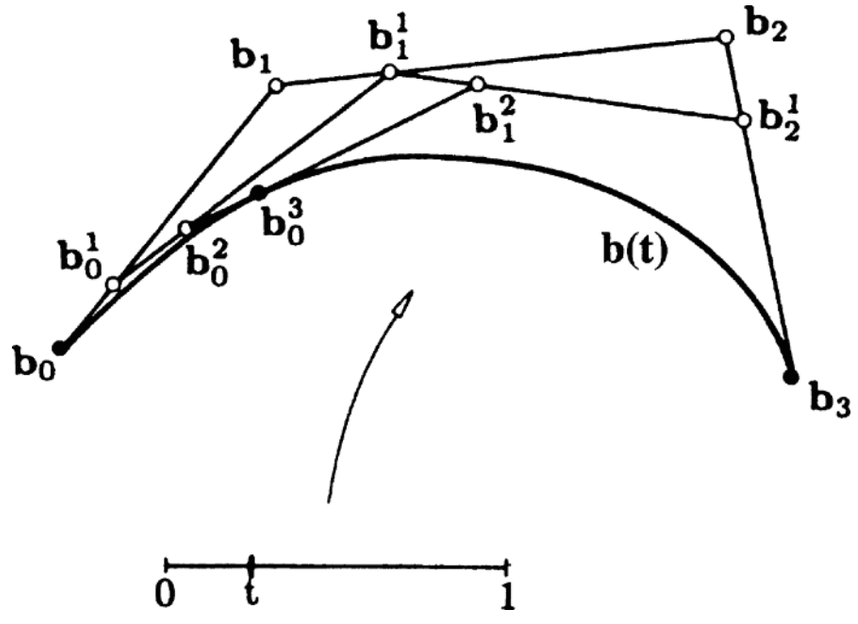
Animation: Steven Wittens, Making Things with Maths, http://acko.net

Evaluating Bézier Curves
Algebraic Formula

Bézier Curve – Algebraic Formula

de Casteljau algorithm gives a pyramid of coefficients



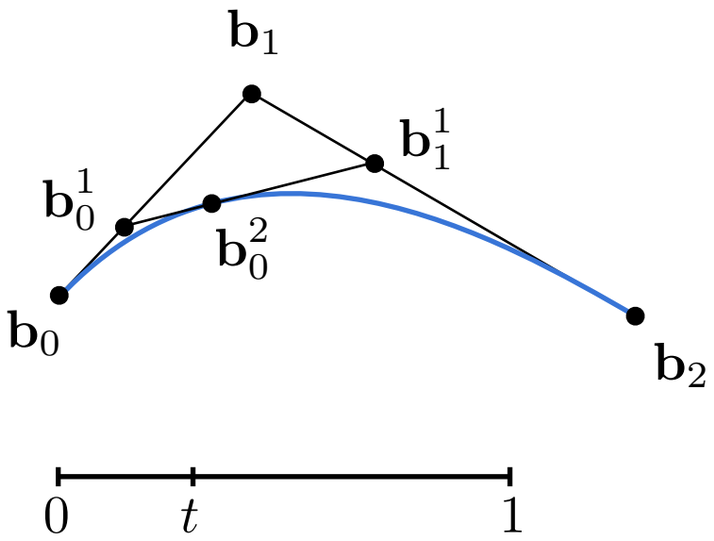


Every rightward arrow is multiplication by t,
Every leftward arrow by (1–t)

Bézier Curve – Algebraic Formula

上述过程我们可以写成表达式

Example: quadratic Bézier curve from three points



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

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

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

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

Bézier Curve – General Algebraic Formula

由此我们得知，给定 n+1 个控制点，其中任意 t 时刻的点由这个多项式与控制点坐标的线性组合可以直接得到

Bernstein form of a Bézier curve of order n:

$$\mathbf{b}^n(t) = \mathbf{b}_0^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t)$$

- $\mathbf{b}^n(t) = \mathbf{b}_0^n(t)$: Bézier curve order n (vector polynomial of degree n)
- $\mathbf{b}_j$: Bézier control points (vector in R^N)
- $B_j^n(t)$: Bernstein polynomial (scalar polynomial of degree 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:

$$\mathbf{b}^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t)$$

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

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

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

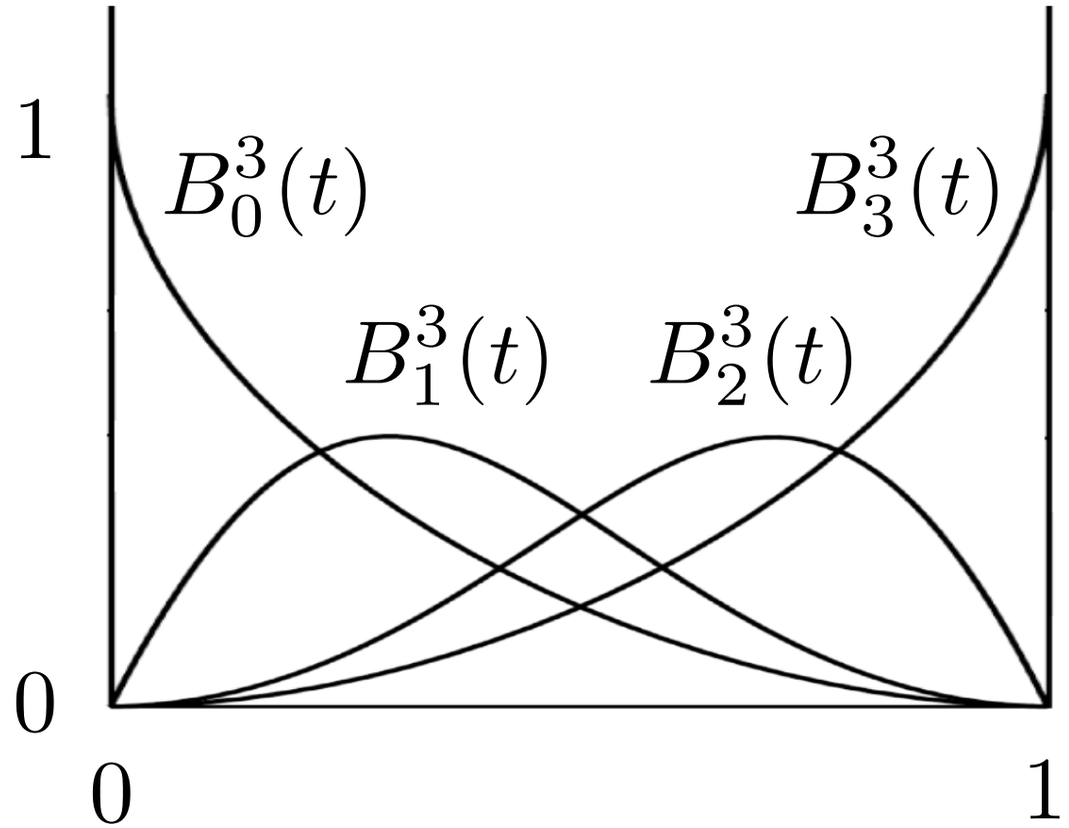
These points define a Bezier curve in 3D that is a cubic polynomial in t:

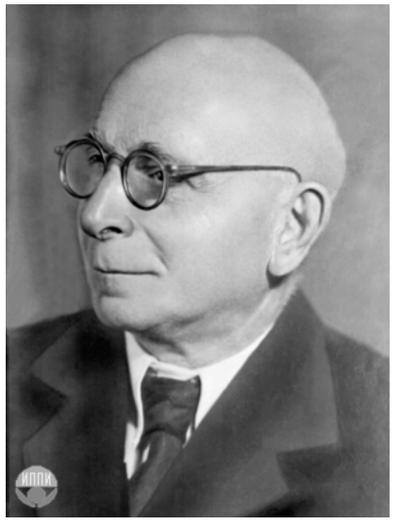
$$\mathbf{b}^n(t) = \mathbf{b}_0\,(1-t)^3 + \mathbf{b}_1\,3t(1-t)^2 + \mathbf{b}_2\,3t^2(1-t) + \mathbf{b}_3\,t^3$$

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 必过起点终点，起始切线方向为前两个点连接的方向，终止切线方向为结尾两个点连接的方向

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

Tangent to end segments 对称性:第 项系数和倒数第 项系数相同

- 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

在仿射变换下，只需要对顶点做仿射变换，就能得到这个贝塞尔曲线在仿射变换下的结果

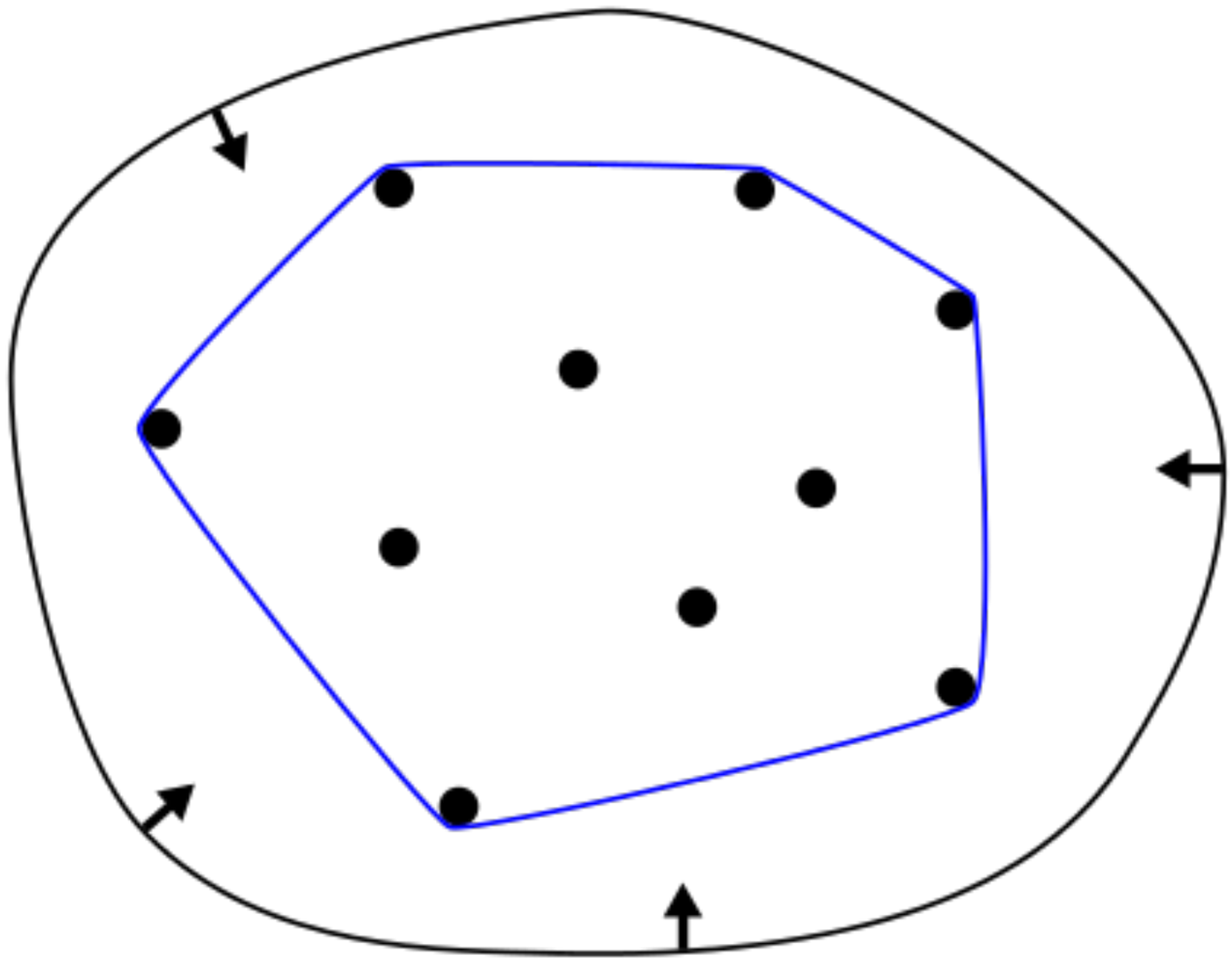
- Transform curve by transforming control points

Convex hull property

凸包性质:贝塞尔曲线始终会在包含了所有控制点的最小凸多边形中, 而不是按照控制点的顺序围成的最小多边 形

- Curve is within convex hull of control points

BTW: What's a Convex Hull

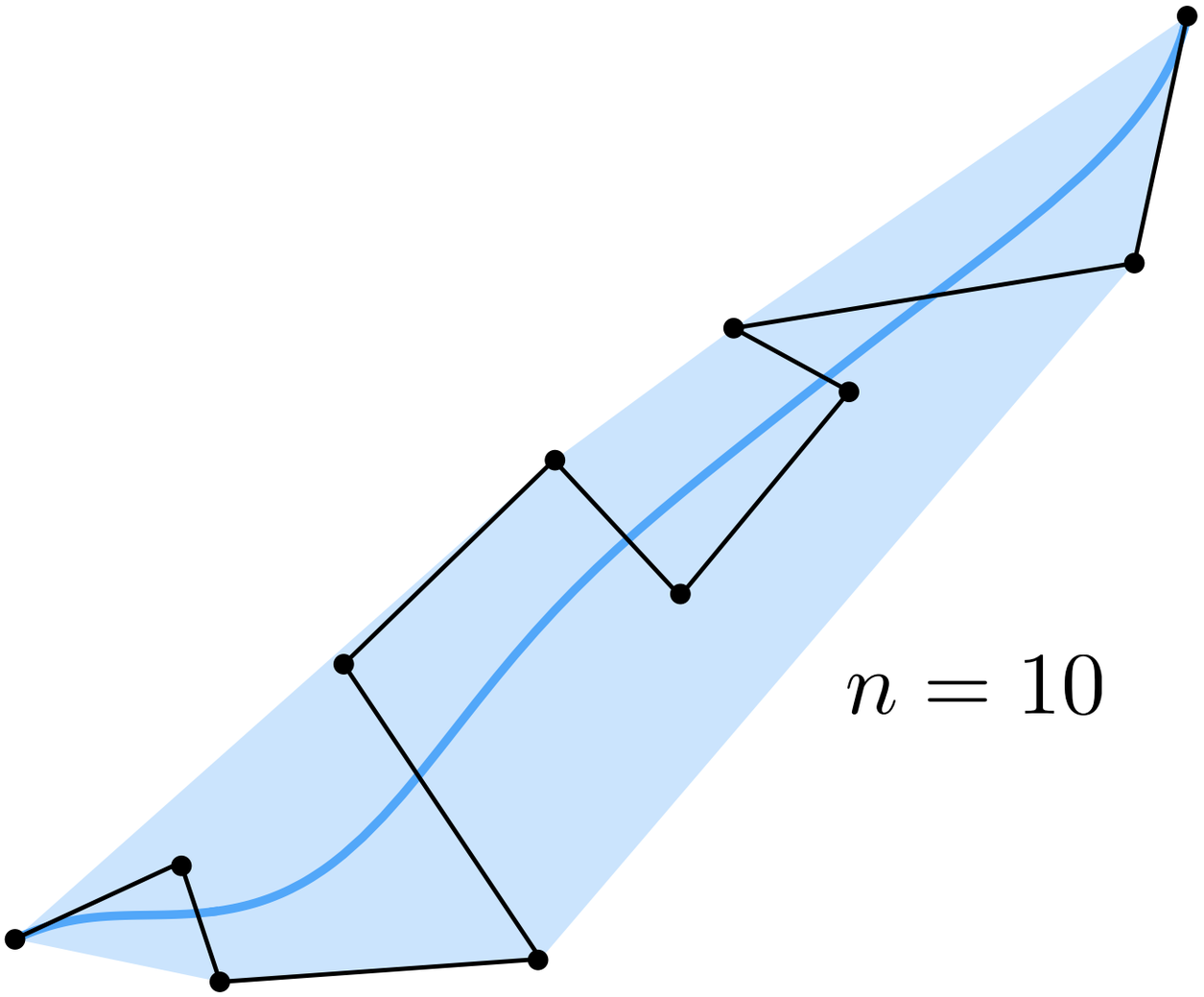


[from Wikipedia]

Piecewise Bézier Curves

对于一些比较复杂且控制点太多的贝塞尔曲线，我们可以逐段的去定义曲线再把它们连起来，这种方法叫 Piecewise Be´zier

Higher-Order Bézier Curves?



$n = 10$

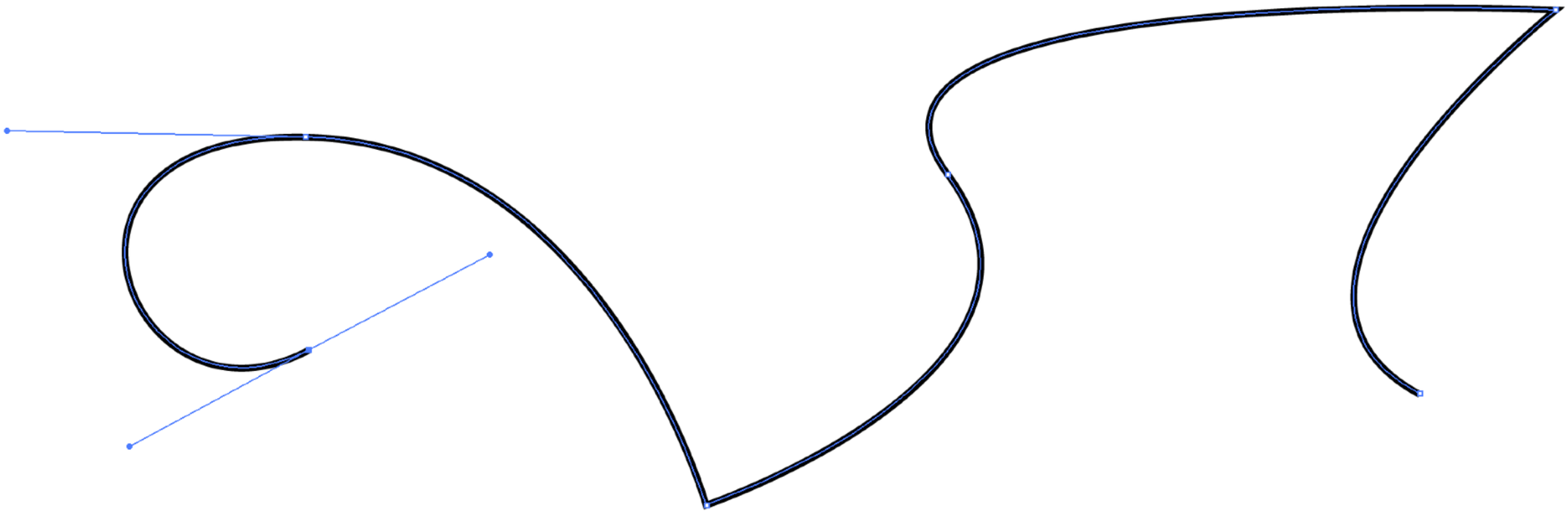
Very hard to control!

Uncommon

Piecewise Bézier Curves

Instead, chain many low-order Bézier curve

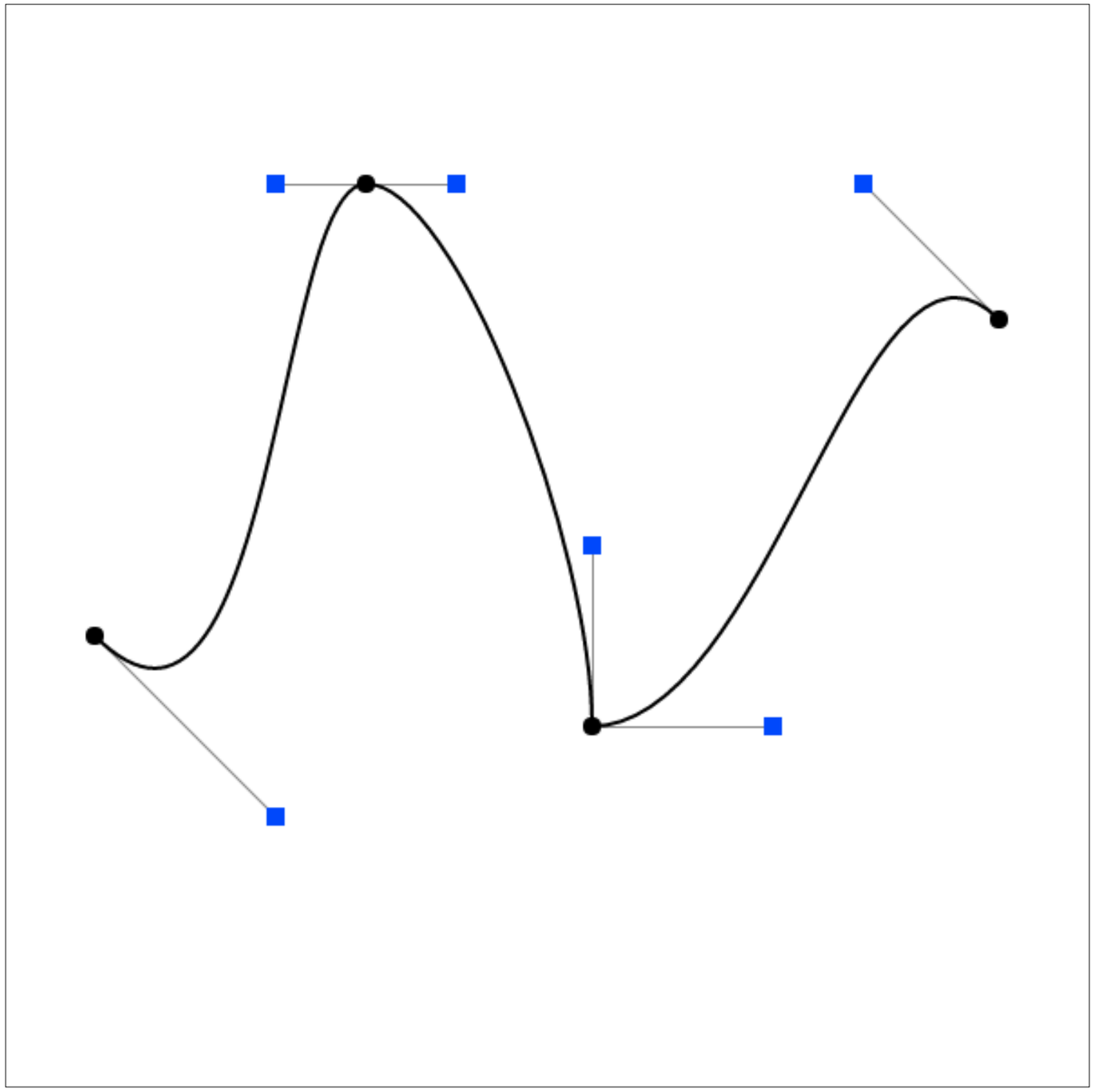
Piecewise cubic Bézier the most common technique



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

Demo – Piecewise Cubic Bézier Curve

比如可以就一直用 4 个控制点两条切线来连续定义各段曲线



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

Piecewise Bézier Curve – Continuity

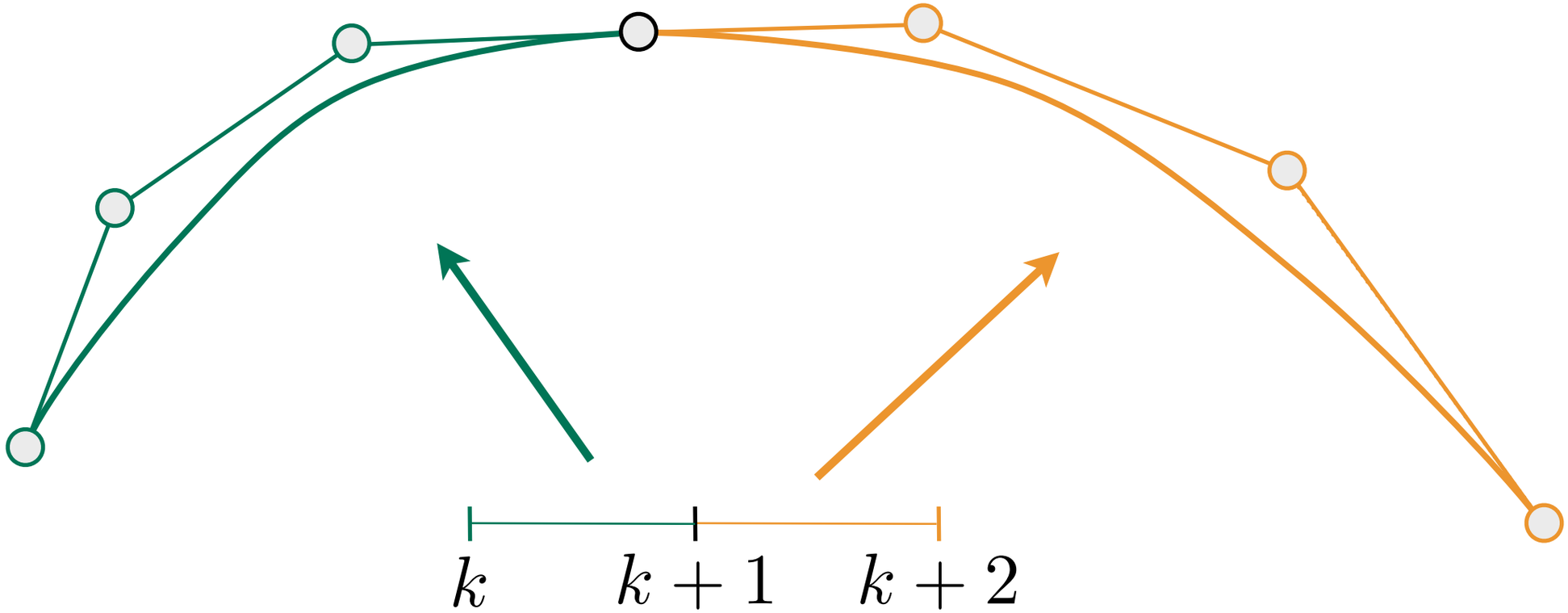
对于 Piecewise Be´zier Curves有连续性指标 C_n来控制其准确

Two Bézier curves

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

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

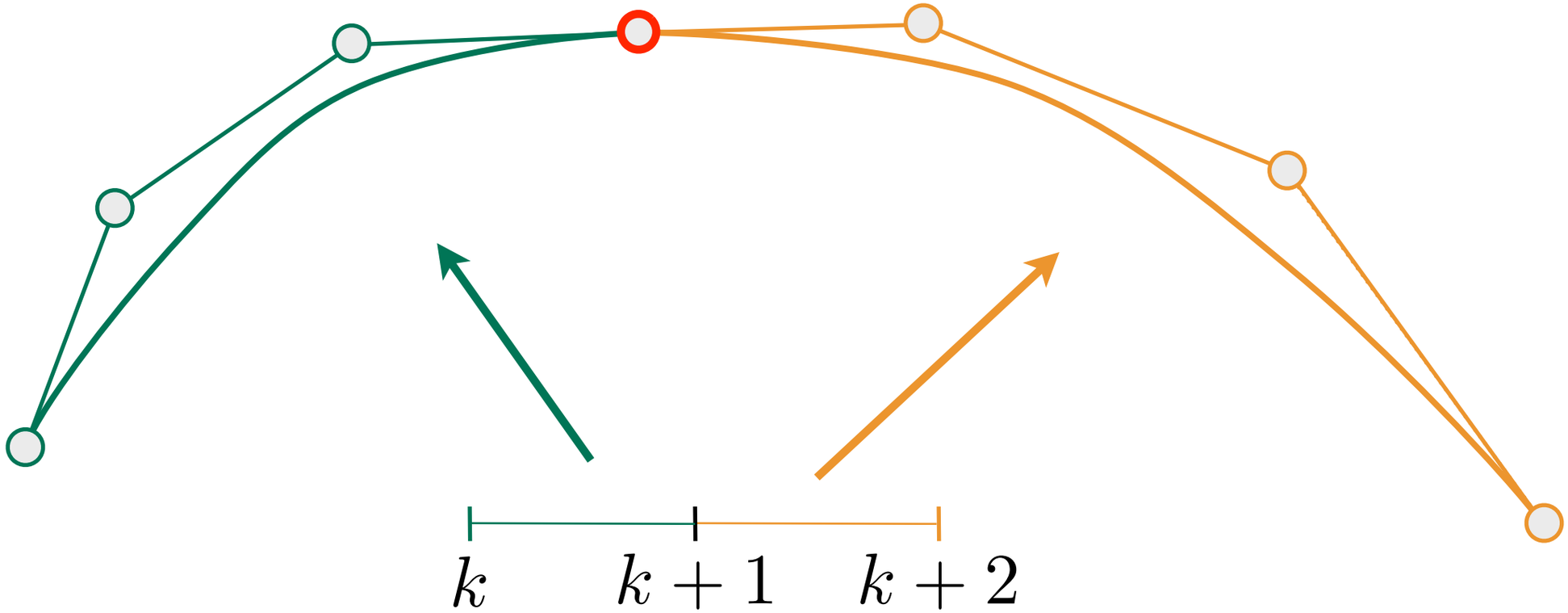
Assuming integer partitions here,
can generalize



Piecewise Bézier Curve – Continuity

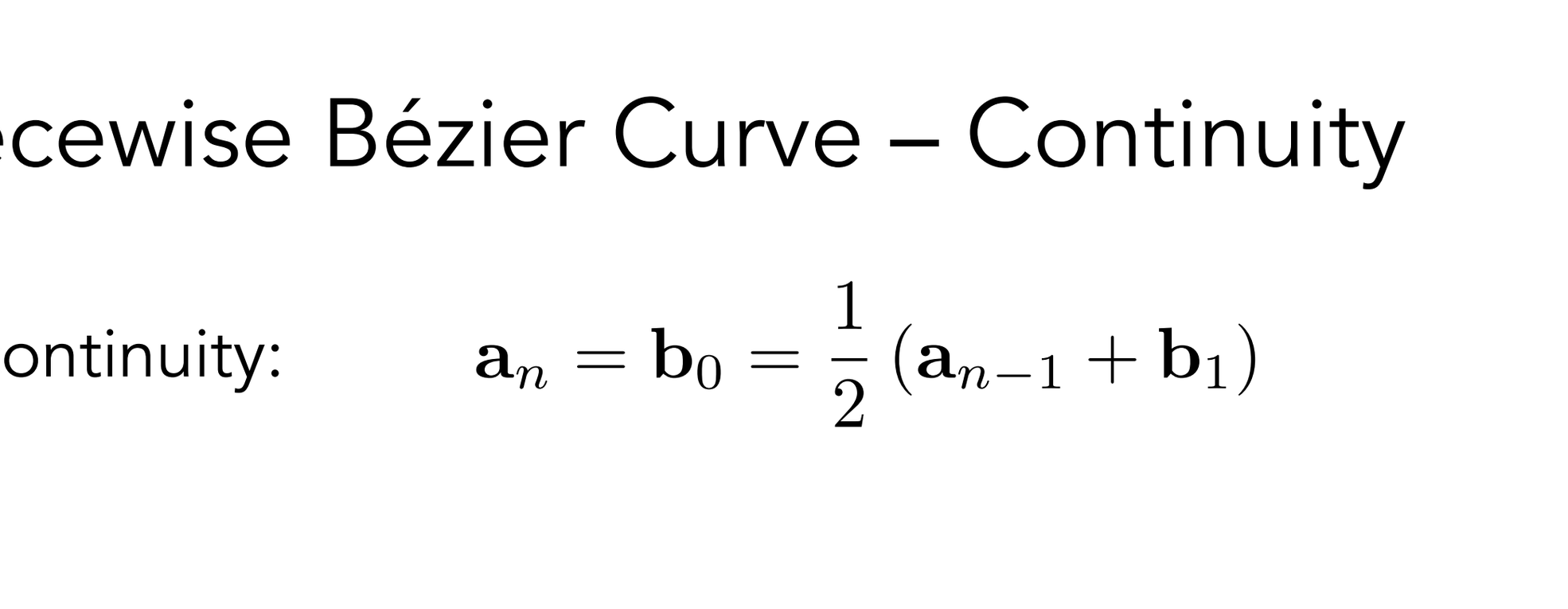
C^0 continuity: $\mathbf{a}_n = \mathbf{b}_0$

即a的最后一个点接b的起始点



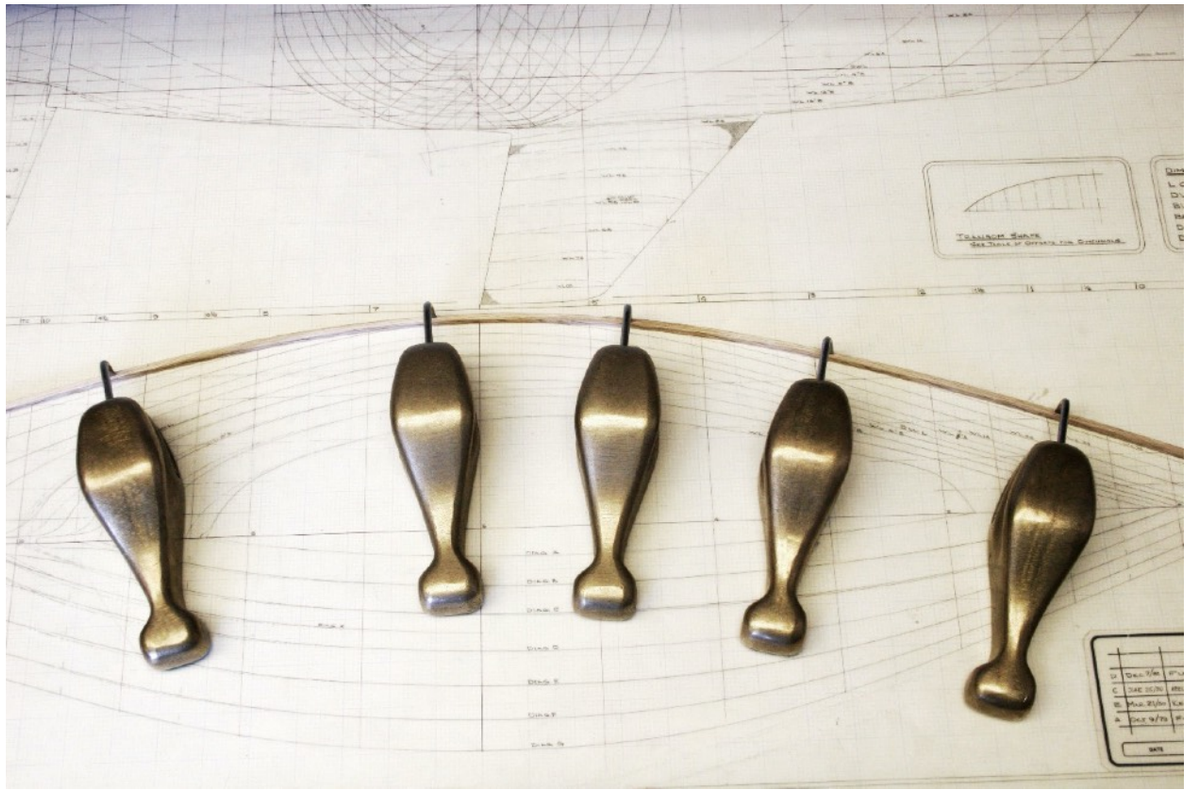
Piecewise Bézier Curve – Continuity

C¹ continuity: $\mathbf{a}_n = \mathbf{b}_0 = \frac{1}{2}\left(\mathbf{a}_{n-1} + \mathbf{b}_1\right)$



Other types of splines

- Spline 连续的曲线，由一系列控制点控制，满足一定的连续性，即可控的曲线
  - 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



A Real Draftsman’s Spline

http://www.alatown.com/spline-history-architecture/

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

Important Note

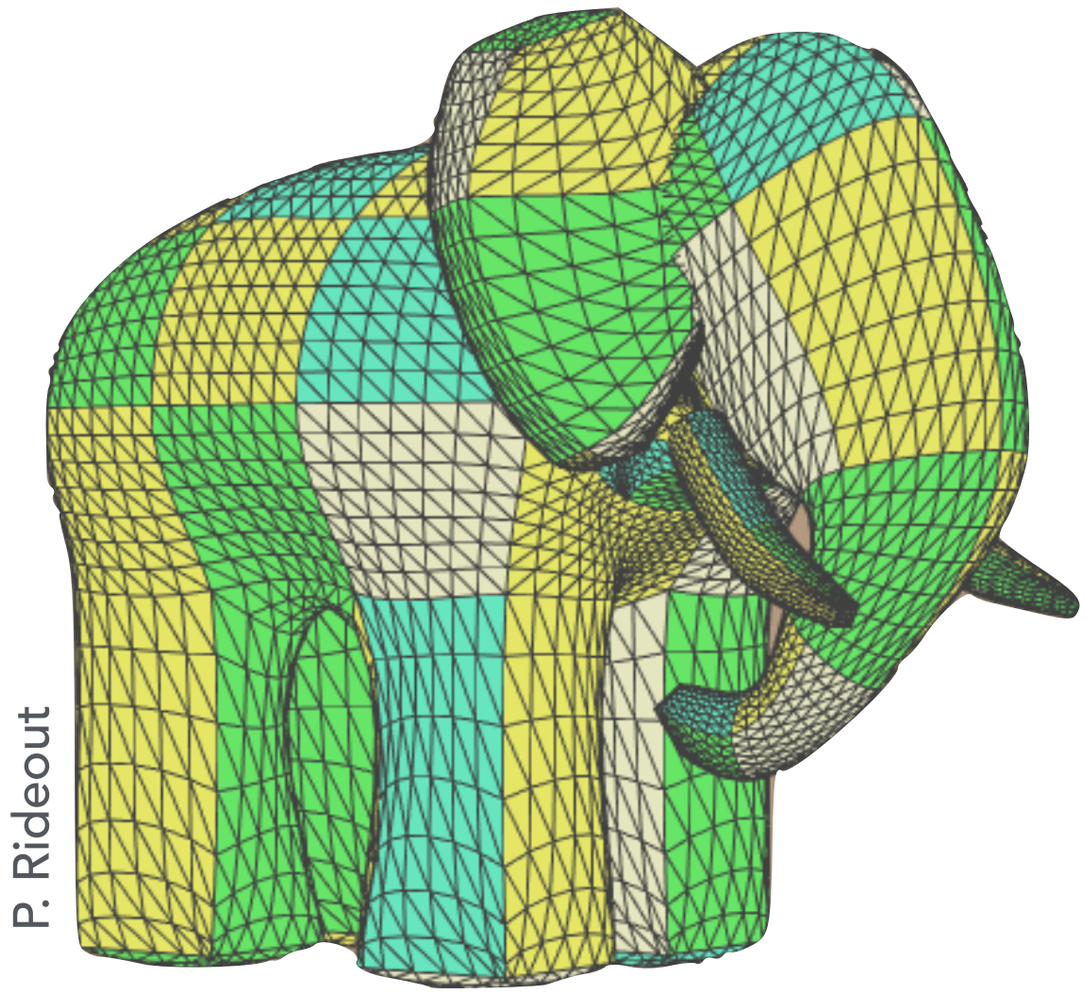
- In this course
  - We do not cover B-splines and NURBS
  - We also do not cover operations on curves (e.g. increasing/decreasing orders, etc.)
  - To learn more / deeper, you are welcome to refer to Prof. Shi-Min Hu's course: https://www.bilibili.com/video/av66548502?from=search&seid=6525680587613148 5

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

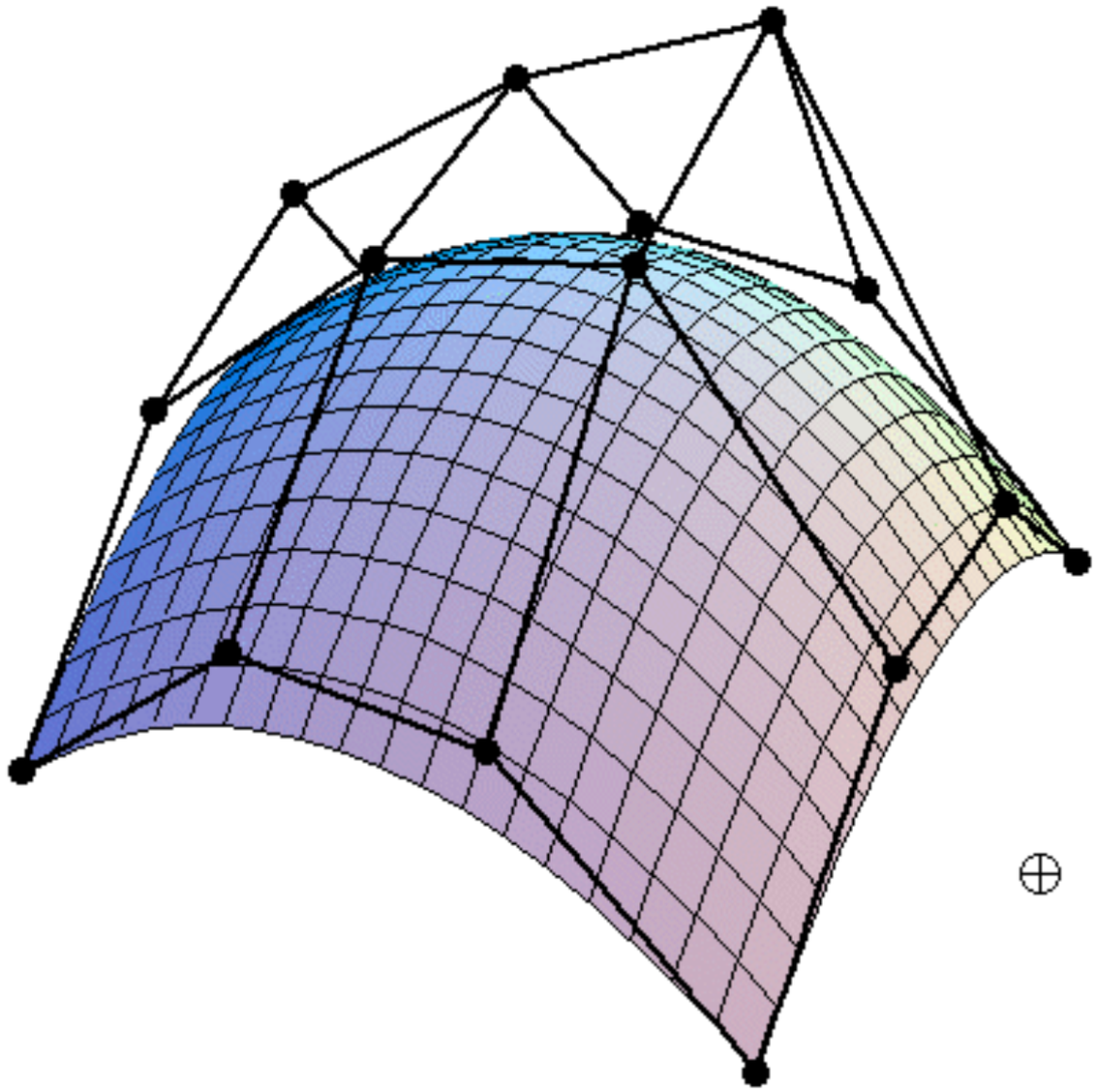


Ed Catmull's "Gumbo" model

Utah Teapot

Bicubic Bézier Surface Patch

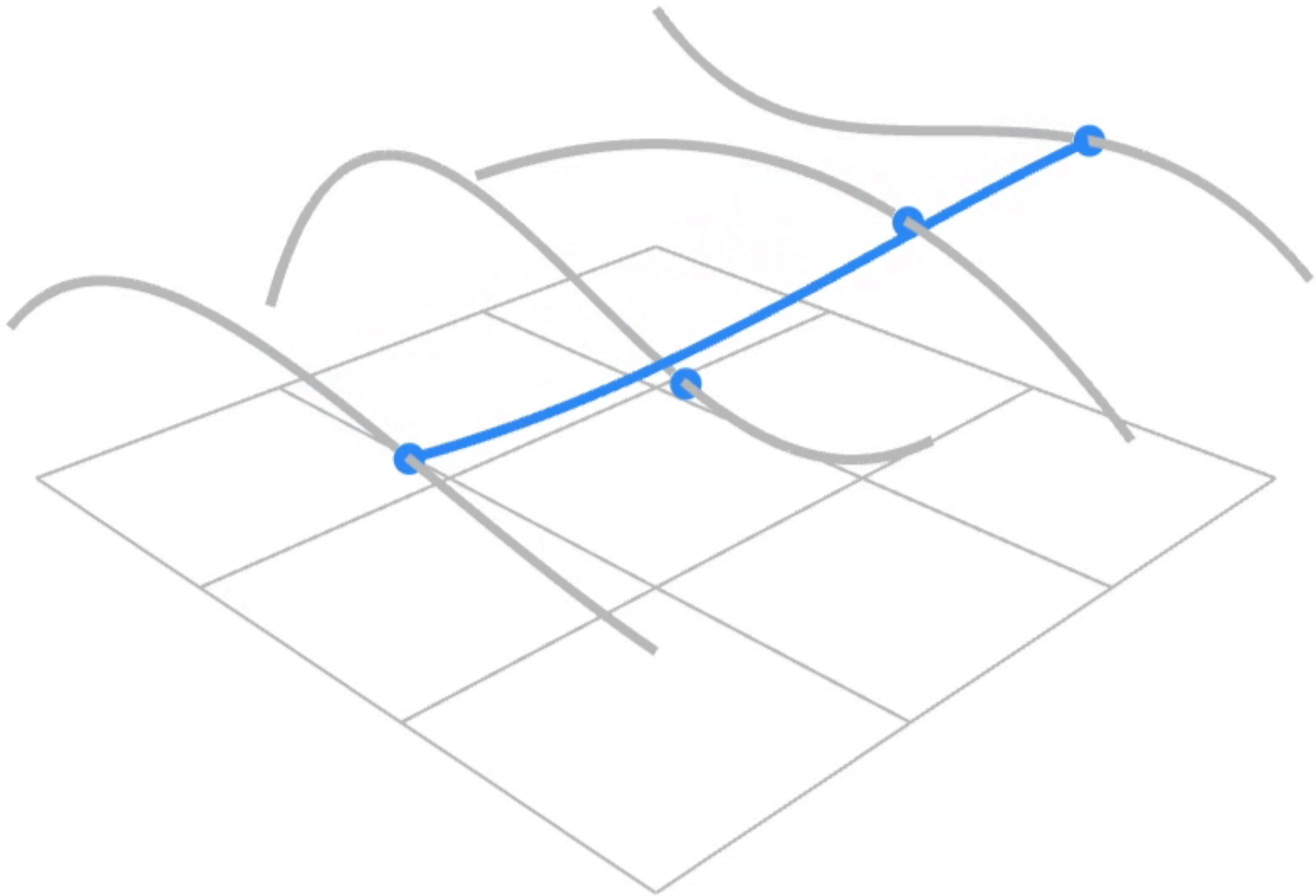
其实曲面也就是曲线的延伸，也就是在两个方向上分别应用贝塞尔曲线就可以得到曲面



Bezier surface and 4 x 4 array of control points

Visualizing Bicubic Bézier Surface Patch

首先先在每行定义四个控制点，然后在每行都得到一条曲线，再将曲线上每个时间t得到的点又看做新曲线的4个控制点，此时就得到了新曲线，这就在每个时间t都有一条曲线，把这些曲线组合起来就得到了贝塞尔曲面



注意，这里的时间t扩展到了二维，也就是有两个时间 u，v in [0, 1] ,其中u控制横向点（基础4条曲线上的t时刻点），v控制纵向点 (新曲线上的t)。

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

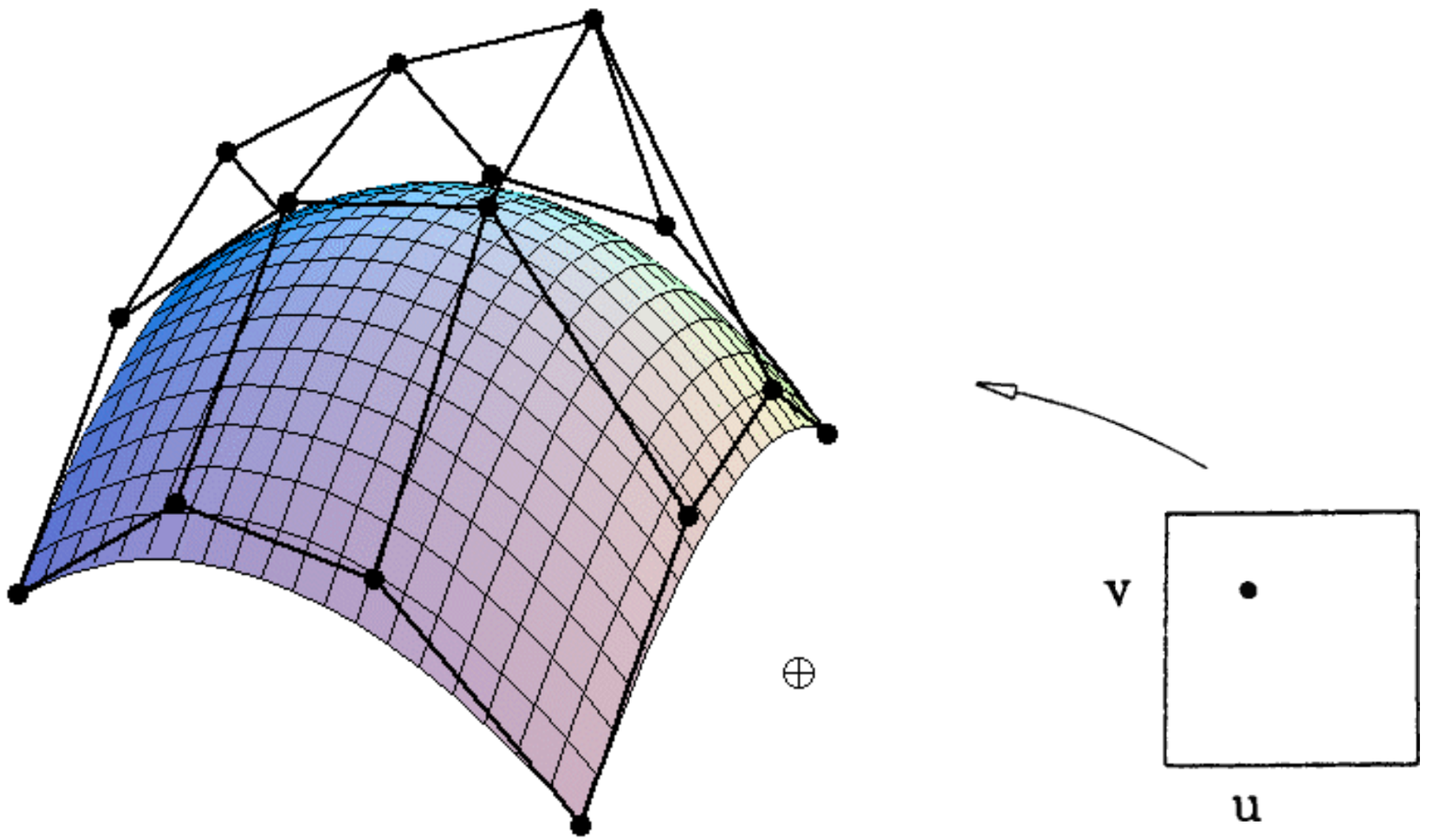
Evaluating Bézier Surfaces

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 $[0,1]^2$

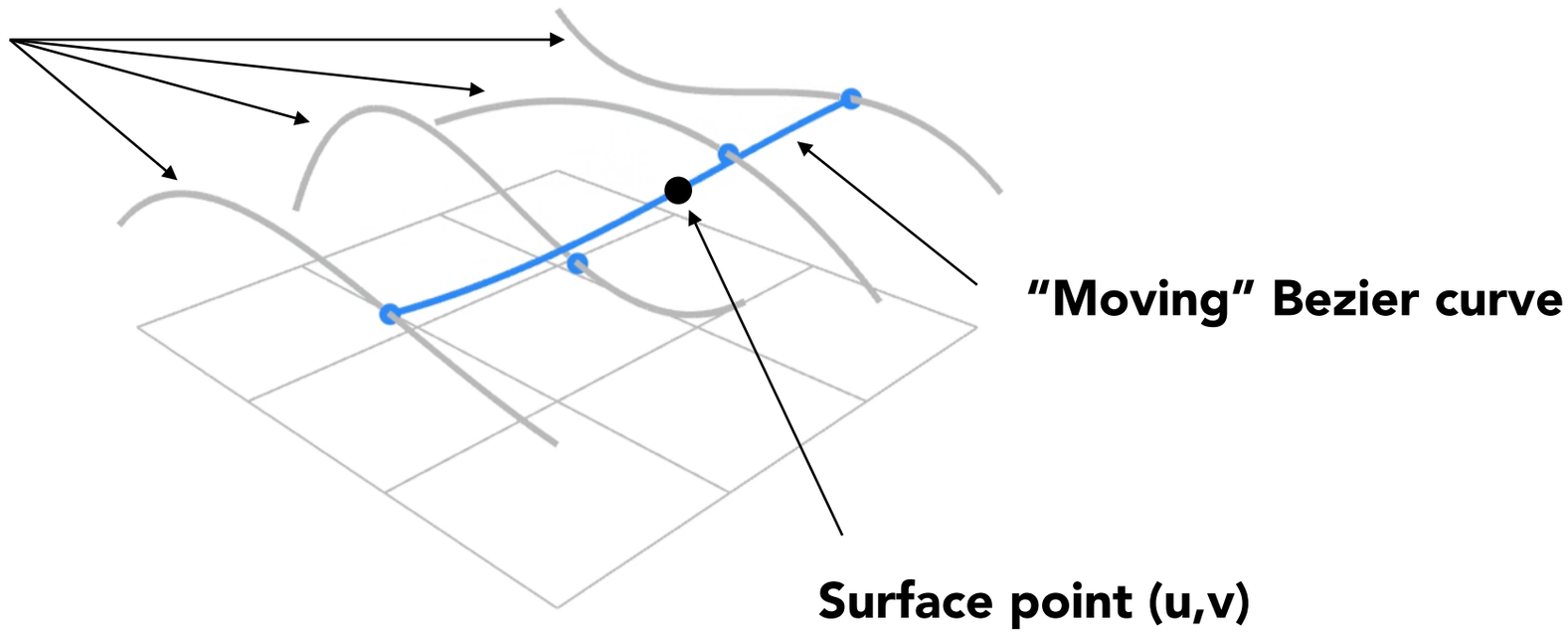


Method: Separable 1D de Casteljau Algorithm

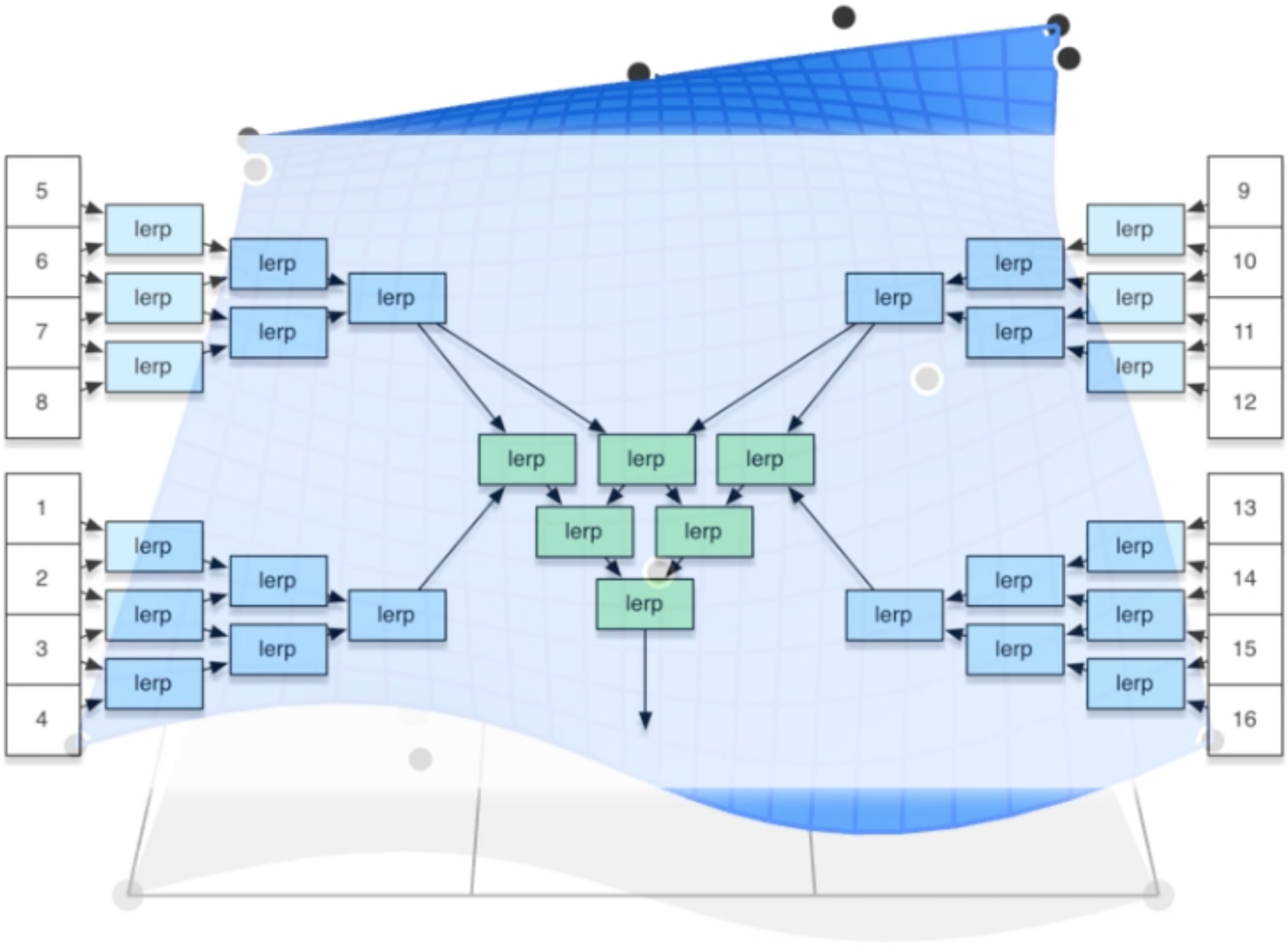
Goal: Evaluate surface position corresponding to (u,v)

(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

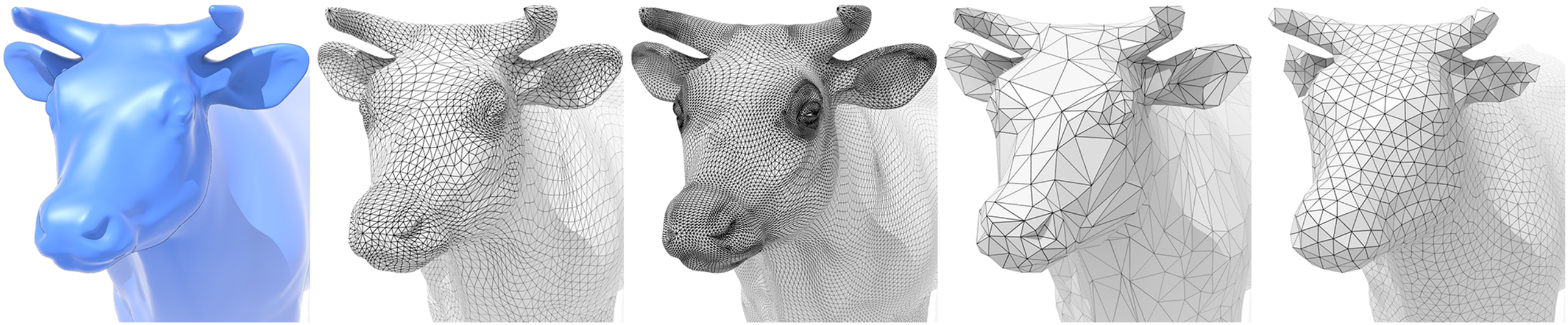


Method: Separable 1D de Casteljau Algorithm



Mesh Operations: Geometry Processing

- Mesh subdivision
- Mesh simplification
- Mesh regularization



Thank you!

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