
Computer Graphics

11 – Curves

Yoonsang Lee
Spring 2021

기말고사 공지

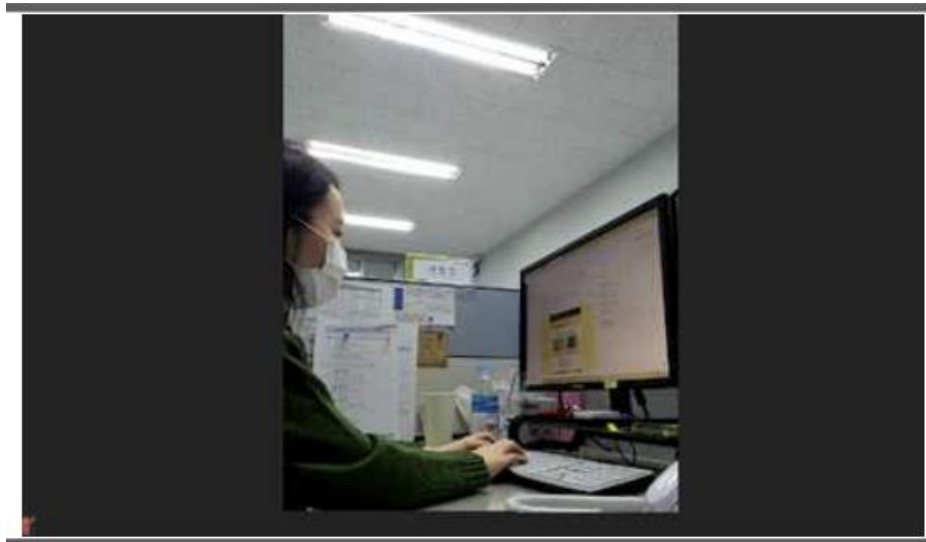
- 일시: 6월 7일 (월) 오전 9시~11시
- 전면 온라인 시험 (LMS의 시험 기능 이용)
 - 5월 25일과 6월 1일 실습 시간에 두 차례에 걸쳐 테스트 시험을 진행할 계획임 (실습 없는 주차는 없음).
 - 두 차례의 실습 및 테스트 시험에 참가하지 않는 경우,
 - 1) 실습 결석으로 체크될 뿐만 아니라,
 - 2) 환경 설정 방법 및 규정을 숙지하지 못해 실제 시험에서 부정행위로 간주될 수 있는 설정 혹은 행동을 하여 불이익을 받는다 하더라도 구제할 수 있는 방법이 없기 때문에,
 - 모두 실습 및 테스트 시험에 성실히 응해서 환경 설정 방법 및 규정을 숙지하여 차질없이 온라인 시험을 치를 수 있기를 바랍니다.

기말고사 공지

- 시험 범위:
 - 3 - Transformation 1
 - 4 - Transformation 2
 - 5 - Rendering Pipeline, Viewing & Projection 1
 - 6 - Viewing & Projection 2, Mesh
 - 7 - Lighting & Shading
 - 8 - Hierarchical Modeling
 - 9 - Orientation & Rotation
 - 10 - Kinematics & Animation
 - 11 - Curves
- 한글로 출제. 넉넉한 양의 문제를 출제할 예정
- **오픈북 시험 아님!**

온라인 시험 주의사항

- 반드시 주변에 다른 사람이 없는, 혼자 있는 공간에서 응시해야 함.
- 본인의 얼굴, 두 손 및 책상, 모니터 (화면 내용이 어느 정도 식별 가능하도록)가 모두 나오도록 스마트폰 카메라를 셋팅
- 그 밖의 세부 사항은 5월 25일 실습시간에 공유할 예정.



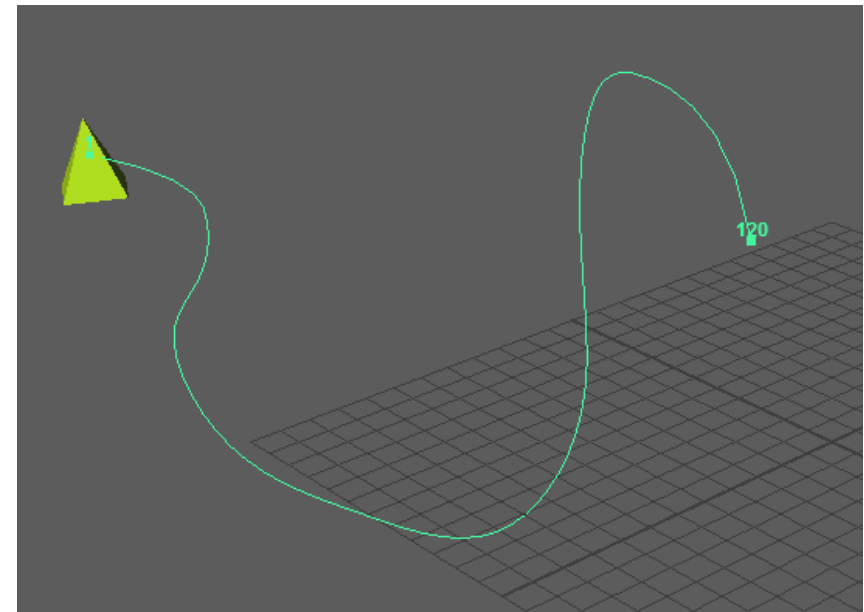
Topics Covered

- Intro: Motivation and Curve Representation
- Polynomial Curve
 - Polynomial Interpolation
 - More Discussion on Polynomials
- Hermite Curve
- Bezier Curve
- (Very short) Intro to Spline

Intro: Motivation and Curve Representation

Motivation: Why Do We Need Curve?

- **Smoothness**
 - no discontinuity
- In many CG applications, we need **smooth shape** and **smooth movement**.

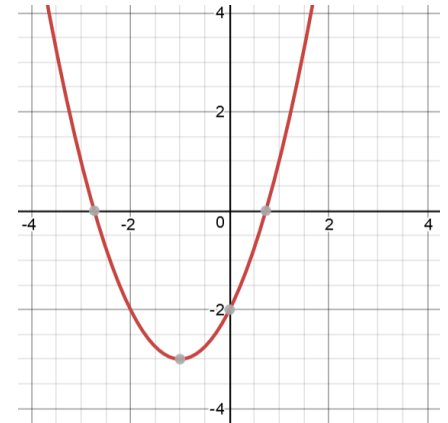


Curve Representations

- Non-parametric

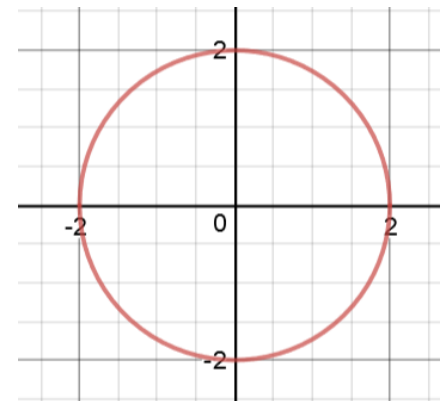
- **Explicit : $y = f(x)$**

- ex) $y = x^2 + 2x - 2$
 - Pros) Easy to generate points
 - Cons) Cannot express vertical lines!



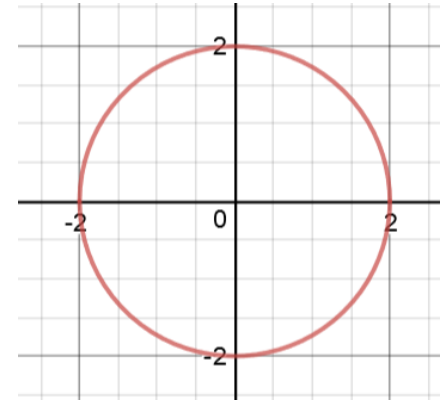
- **Implicit : $f(x, y) = 0$**

- ex) $x^2 + y^2 - 2^2 = 0$
 - Pros) Easy to test if a point is inside or outside
 - Cons) Inconvenient to generate points



Curve Representations

- **Parametric** : $(x, y) = (f(t), g(t))$
 - ex) $(x, y) = (2 \cos(t), 2 \sin(t))$
 - Each point on a curve is expressed as a function of **additional parameter t**
 - Pros) Easy to generate points
 - The parameter **t** acts as a “**local coordinate**” for points on the curve
- For computer graphics, the parametric representation is the most suitable.



Polynomial Curve

Polynomial Curve

- Polynomials are usually used to describe curves in computer graphics
 - Simple
 - Efficient
 - Easy to manipulate
 - Historical reasons
- A polynomial of degree n:

$$x(t) = a_n t^n + a_{n-1} t^{n-1} + \dots + a_1 t + a_0$$

Polynomial Interpolation

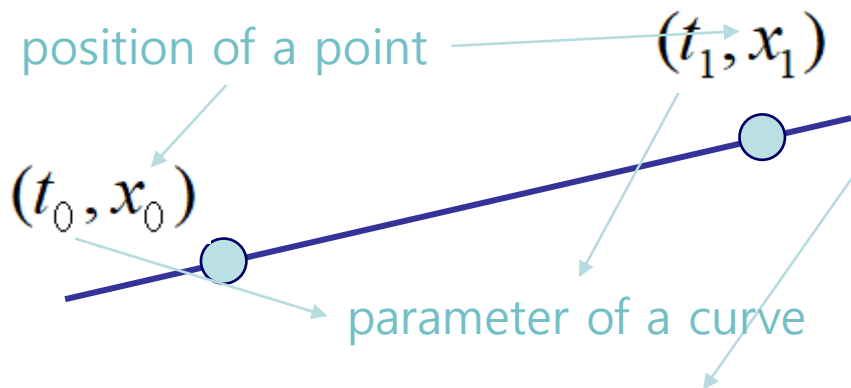
- One way to make a smooth curve is with polynomial interpolation.
- Polynomial interpolation determines a specific smooth polynomial curve **passing through given data points**.

Polynomial Interpolation

- Linear interpolation with a polynomial of degree one

- Input: two nodes

- Output: Linear polynomial



$$x(t) = a_1 t + a_0$$

How to find a_0 and a_1 ?



$$a_1 t_0 + a_0 = x_0$$

$$a_1 t_1 + a_0 = x_1$$

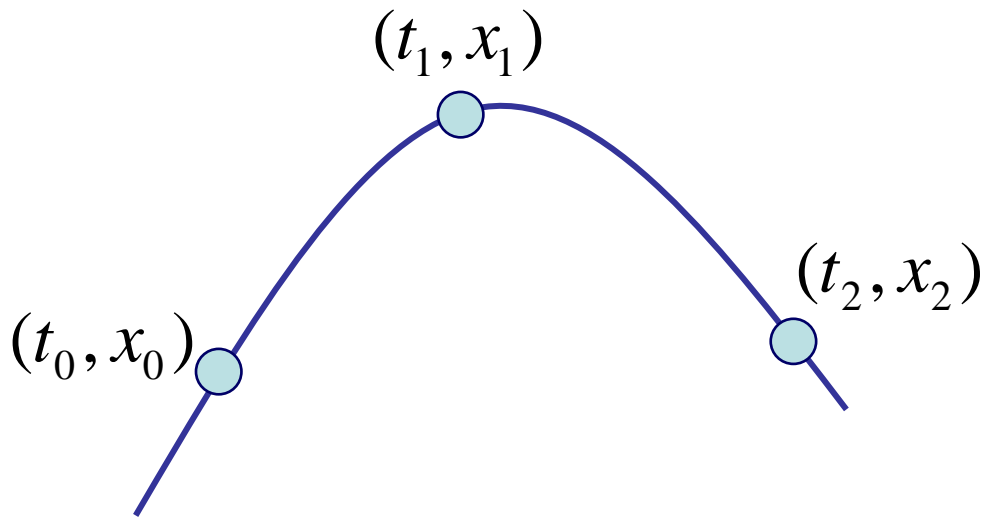
$$\begin{pmatrix} 1 & t_0 \\ 1 & t_1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

We can compute the value of a_0 & a_1 because we have **2 equations** (=2 data points) for **2 unknowns!**

If $t_0=0$ and $t_1=1$, then $a_0=x_0$ and $a_1=x_1-x_0$
 $\rightarrow x(t) = (x_1-x_0)t + x_0 = (1-t)x_0 + tx_1$

Polynomial Interpolation

- Quadratic interpolation with a polynomial of degree two



$$x(t) = a_2 t^2 + a_1 t + a_0$$

(we need **3 points** to get the value of **3 unknowns**)

$$a_2 t_0^2 + a_1 t_0 + a_0 = x_0$$

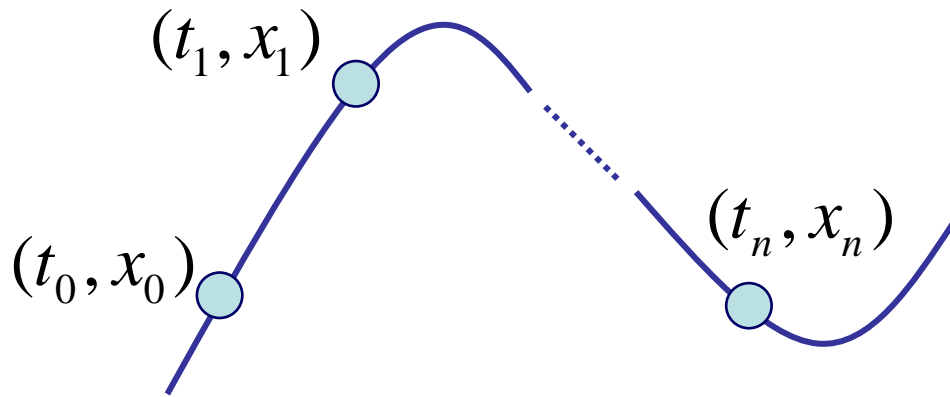
$$a_2 t_1^2 + a_1 t_1 + a_0 = x_1$$

$$a_2 t_2^2 + a_1 t_2 + a_0 = x_2$$

$$\begin{pmatrix} 1 & t_0 & t_0^2 \\ 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

Polynomial Interpolation

- Polynomial interpolation of degree n



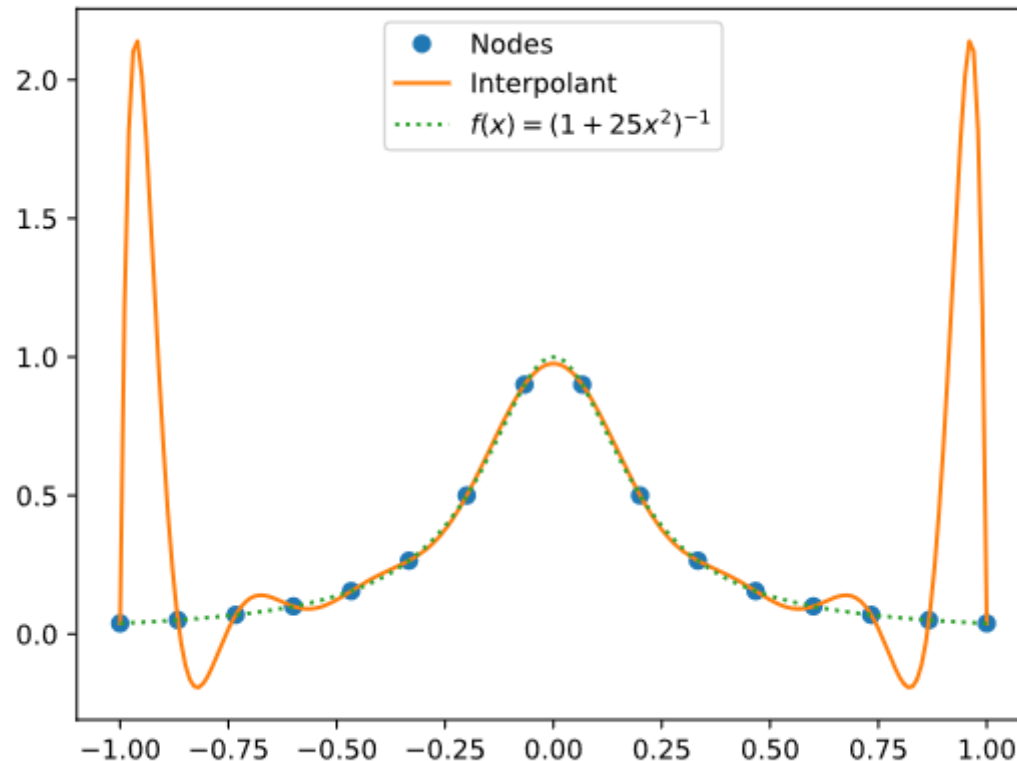
$$x(t) = a_n t^n + \dots + a_1 t + a_0$$

$$\begin{pmatrix} 1 & \dots & t_0^{n-1} & t_0^n \\ 1 & \dots & t_1^{n-1} & t_1^n \\ \vdots & \ddots & \vdots & \vdots \\ 1 & \dots & t_n^{n-1} & t_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix}$$

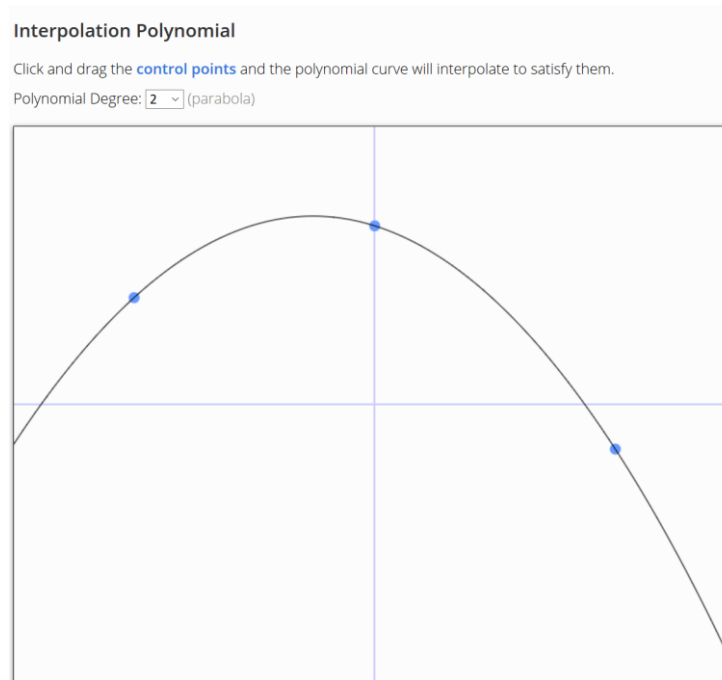
- How to find the value of unknowns a_n, \dots, a_0 ?
- Several methods:
 - Solving linear system, Lagrange's, Newton's method, ...

Problem of Higher Degree Polynomial Interpolation

- Oscillations at the ends – Runge's Phenomenon
 - Nobody uses higher degree polynomial interpolation now



[Practice] Polynomial Interpolation



<https://www.benjoffe.com/code/demos/interpolate>

- Drag points and observe changes of the curve.
- Increase polynomial degree and drag points.

Cubic Polynomials

- Cubic (degree of 3) polynomials are commonly used in computer graphics because...
- The lowest-degree polynomials representing a 3D space curve..
- Unwanted wiggles of higher-degree polynomials (Runge's Phenomenon)

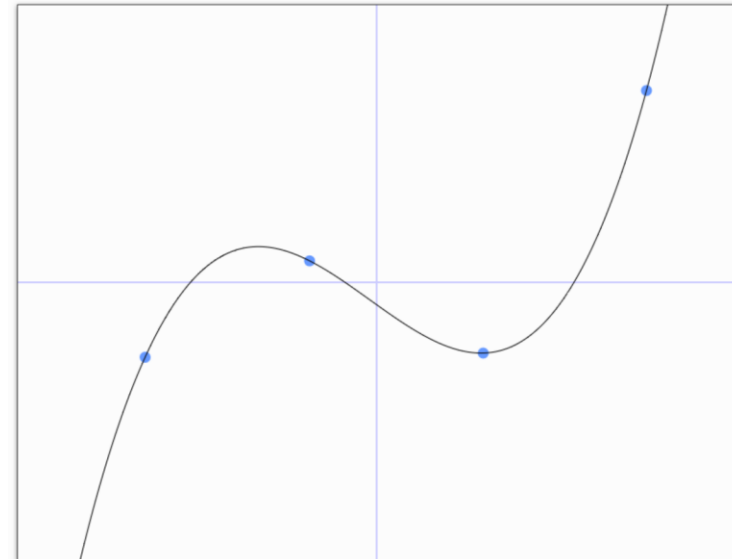
$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

or

$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

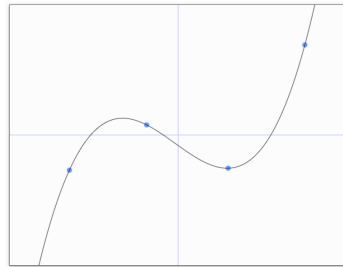


Then, how can we make complex curves using such a low degree polynomial?

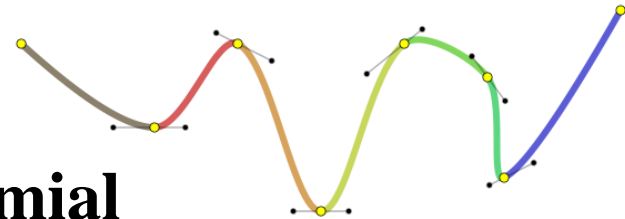
- How to make



- using



- Answer → **Spline: *piecewise* polynomial**

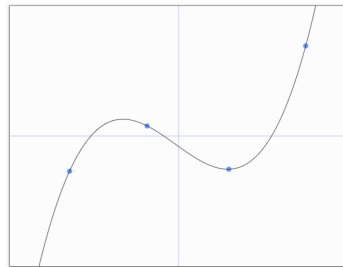


- At this moment, let's just think about a single piece of polynomial.

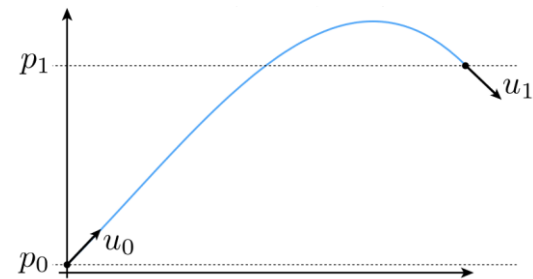
Defining a Single Piece of Cubic Polynomial

$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

- Goal: Defining a specific curve (finding \mathbf{a} , \mathbf{b} , \mathbf{c} , \mathbf{d}) as we want (using data points or *conditions* given by you)
- **4 unknowns**, so we need **4 equations (conditions or constraints)**. For example,
 - 4 data points



- position and derivative of 2 end points
- ...



Formulation of a Single Piece of Polynomial

- A polynomial can be formulated in two ways:

- With **coefficients** and **variable**:

$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

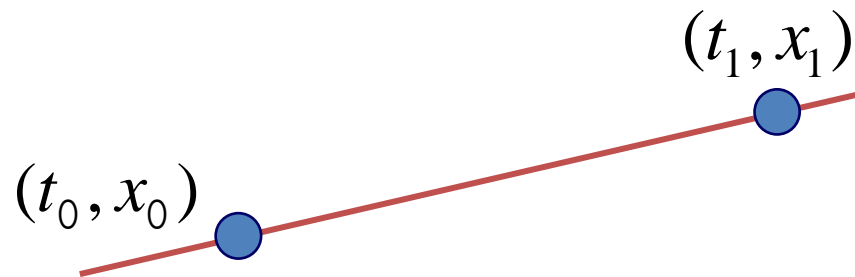
- coefficients: **a, b, c, d**
- variable: **t**

- With *basis functions* and **points**:

$$\mathbf{p}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2 + b_3(t)\mathbf{p}_3$$

- *basis functions*: $b_0(t), b_1(t), b_2(t), b_3(t)$
- points: **p₀, p₁, p₂, p₃**

Trivial Example: Linear Polynomial



$$x(t) = a_1 t + a_0$$

Trivial Example: Linear Polynomial

- Formulation with coefficients and variable:

$$x(t) = (x_1 - x_0)t + x_0$$

$$y(t) = (y_1 - y_0)t + y_0$$

$$\mathbf{p}(t) = (\mathbf{p}_1 - \mathbf{p}_0)t + \mathbf{p}_0$$

- Matrix formulation

$$\mathbf{p}(t) = \begin{bmatrix} t & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}$$

$$\mathbf{p}(t) = \begin{bmatrix} x(t) & y(t) \end{bmatrix}$$


basis matrix

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \end{bmatrix}$$

Trivial Example: Linear Polynomial

- Formulation with basis functions and points:
 - regroup expression by \mathbf{p} rather than t

$$\begin{aligned}\mathbf{p}(t) &= (\mathbf{p}_1 - \mathbf{p}_0)t + \mathbf{p}_0 \\ &= \underbrace{(1 - t)}_{\text{basis functions}} \mathbf{p}_0 + \underbrace{t}_{\text{basis functions}} \mathbf{p}_1\end{aligned}$$

basis functions

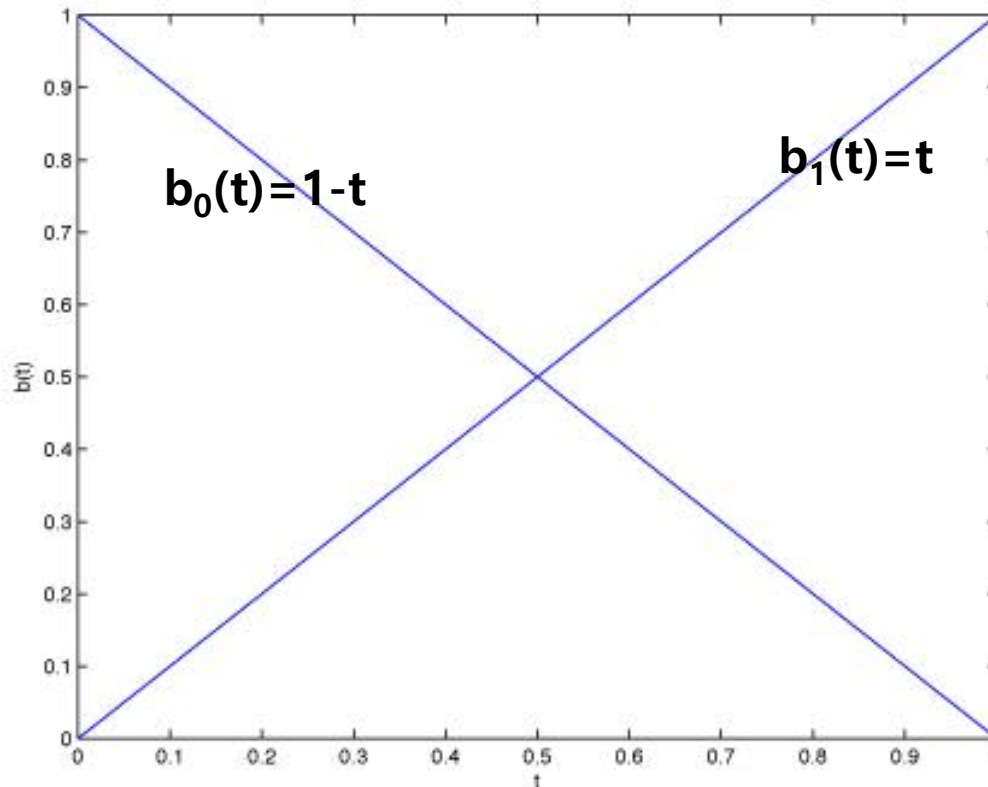
- interpretation in matrix viewpoint

$$\mathbf{p}(t) = \begin{pmatrix} [t & 1] & \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \end{pmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}$$

Meaning of Basis Functions

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

- Contribution of each point as t changes



Quiz #1

- Go to <https://www.slido.com/>
- Join #cg-ys
- Click “Polls”

- Submit your answer in the following format:
 - **Student ID: Your answer**
 - e.g. **2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

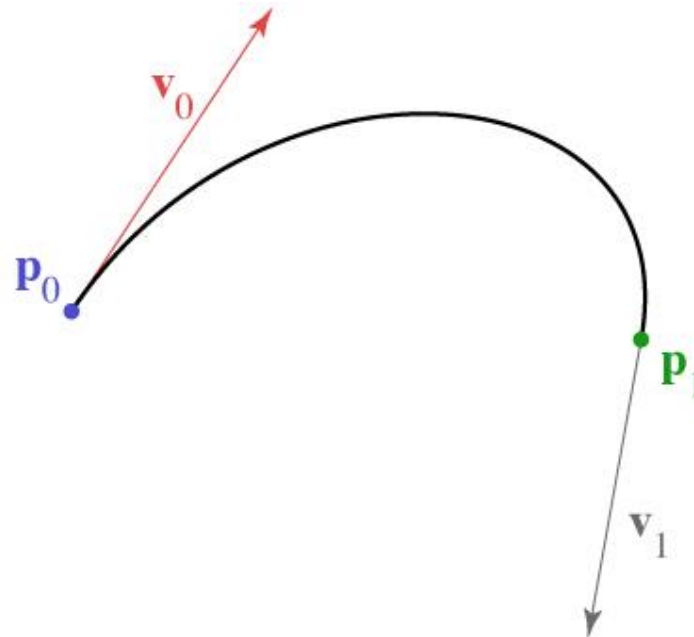
Hermite Curve

Hermite curve

- Less trivial example
- Form of curve: piecewise cubic
- Constraints: endpoints and tangents (derivatives)



Charles Hermite
(1822-1901)



Hermite curve

- Solve constraints to find coefficients

$$x(t) = at^3 + bt^2 + ct + d$$

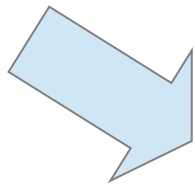
$$x'(t) = 3at^2 + 2bt + c$$

$$x(0) = x_0 = d$$

$$x(1) = x_1 = a + b + c + d$$

$$x'(0) = x'_0 = c$$

$$x'(1) = x'_1 = 3a + 2b + c$$



$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x'_0 \\ x'_1 \end{bmatrix}$$

Hermite curve

- Solve constraints to find coefficients

$$x(t) = at^3 + bt^2 + ct + d$$

$$x'(t) = 3at^2 + 2bt + c$$

$$x(0) = x_0 = d$$

$$x(1) = x_1 = a + b + c + d$$

$$x'(0) = x'_0 = c$$

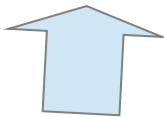
$$x'(1) = x'_1 = 3a + 2b + c$$

$$d = x_0$$

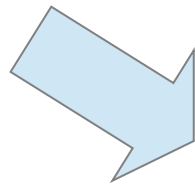
$$c = x'_0$$

$$a = 2x_0 - 2x_1 + x'_0 + x'_1$$

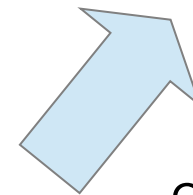
$$b = -3x_0 + 3x_1 - 2x'_0 - x'_1$$



$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x'_0 \\ x'_1 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x'_0 \\ x'_1 \end{bmatrix}$$



calculate A^{-1}

Hermite curve

- Matrix form is much simpler

$$\mathbf{p}(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix}$$

- coefficients = rows
- basis functions = columns

Hermite basis matrix

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x'_0 & y'_0 \\ x'_1 & y'_1 \end{bmatrix}$$

Coefficients = rows

$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

$$\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{p}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2 + b_3(t)\mathbf{p}_3$$

Basis functions = columns

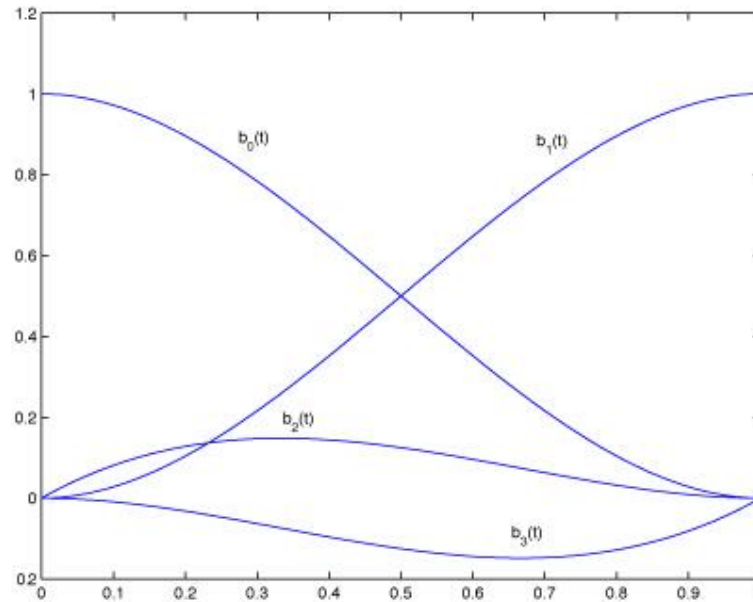
$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

$$\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

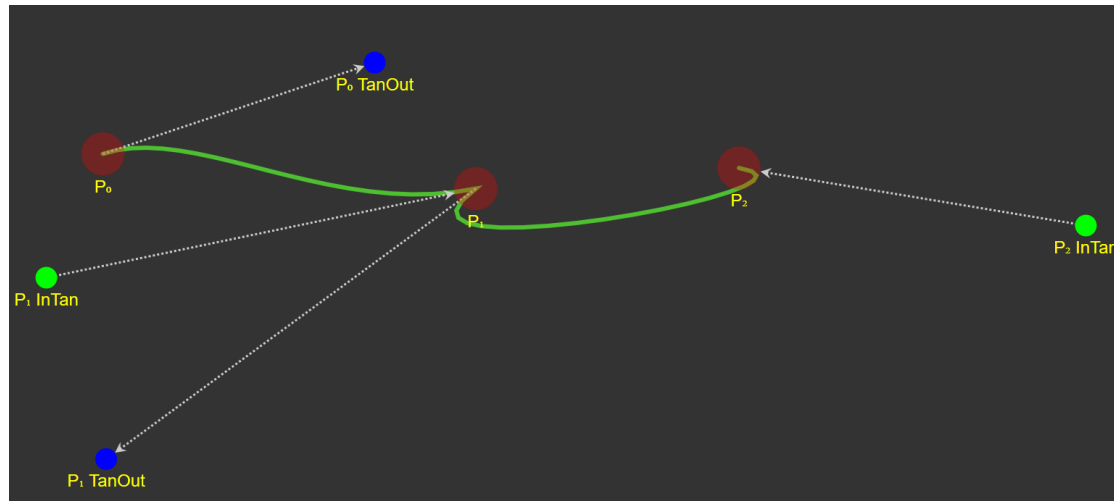
$$\mathbf{p}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2 + b_3(t)\mathbf{p}_3$$

Hermite curve

- Hermite basis functions



[Practice] Hermite Curve Online Demo



<https://codepen.io/liorda/pen/KrvBwr>

- Change the position of end points and their derivatives by dragging

Quiz #2

- Go to <https://www.slido.com/>
- Join #cg-ys
- Click “Polls”

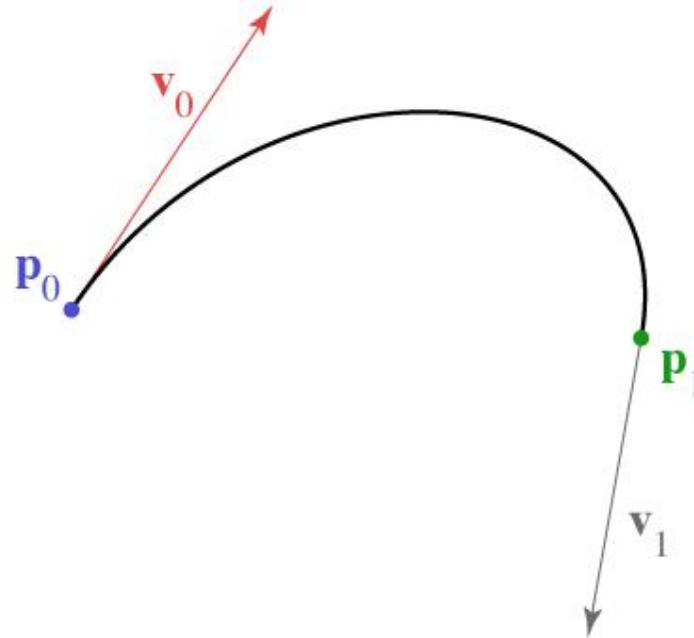
- Submit your answer in the following format:
 - **Student ID: Your answer**
 - e.g. **2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

Bezier Curve

Recall: Hermite curve

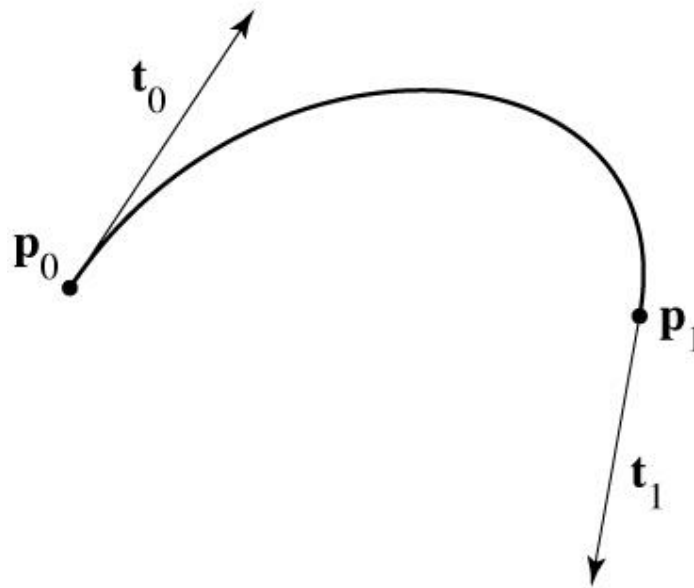
- Constraints: endpoints and tangents (derivatives)



$$\mathbf{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix}$$

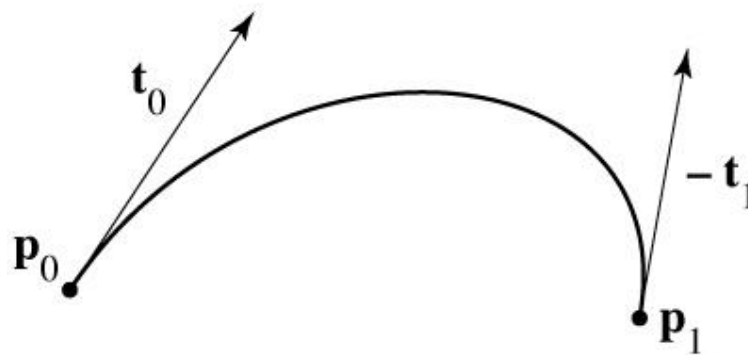
Hermite to Bézier

- Mixture of points and vectors is awkward
- Specify tangents as differences of points



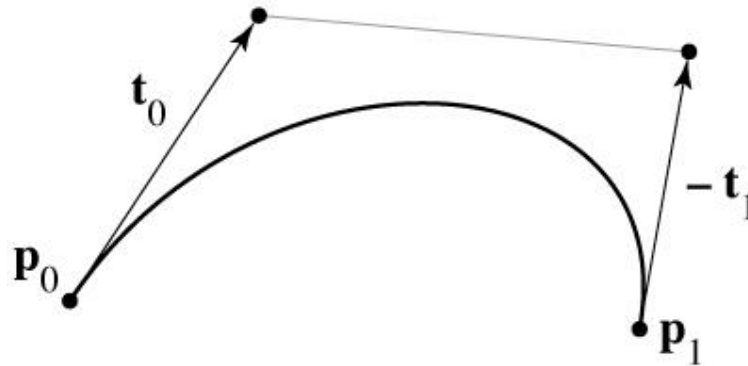
Hermite to Bézier

- Mixture of points and vectors is awkward
- Specify tangents as differences of points



Hermite to Bézier

- Mixture of points and vectors is awkward
- Specify tangents as differences of points

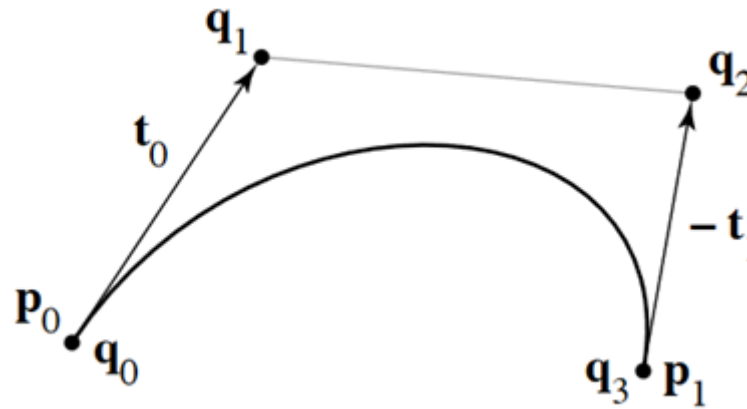


Hermite to Bézier



Pierre Bézier (1910-1999)
widely published
research on this curve
while working at Renault

- Mixture of points and vectors is awkward
- Specify tangents as differences of points



– note derivative is defined as 3 times offset t

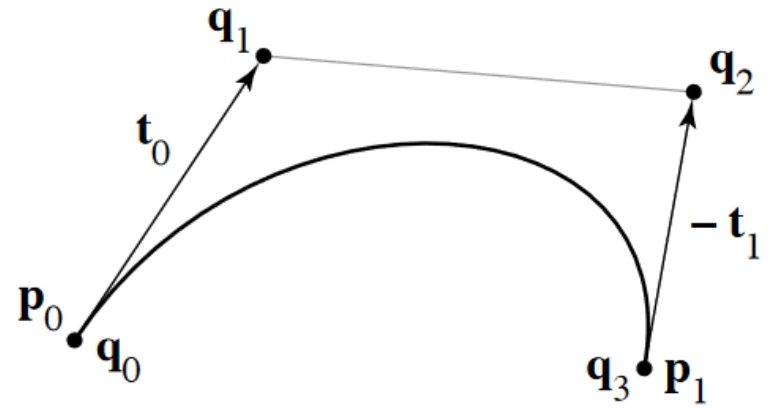
Hermite to Bézier

$$\mathbf{p}_0 = \mathbf{q}_0$$

$$\mathbf{p}_1 = \mathbf{q}_3$$

$$\mathbf{v}_0 = 3(\mathbf{q}_1 - \mathbf{q}_0)$$

$$\mathbf{v}_1 = 3(\mathbf{q}_3 - \mathbf{q}_2)$$



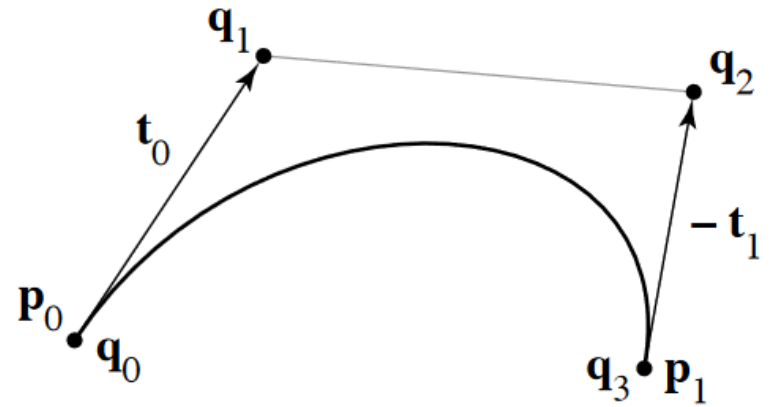
Hermite to Bézier

$$\mathbf{p}_0 = \mathbf{q}_0$$

$$\mathbf{p}_1 = \mathbf{q}_3$$

$$\mathbf{v}_0 = 3(\mathbf{q}_1 - \mathbf{q}_0)$$

$$\mathbf{v}_1 = 3(\mathbf{q}_3 - \mathbf{q}_2)$$



$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \end{bmatrix}$$

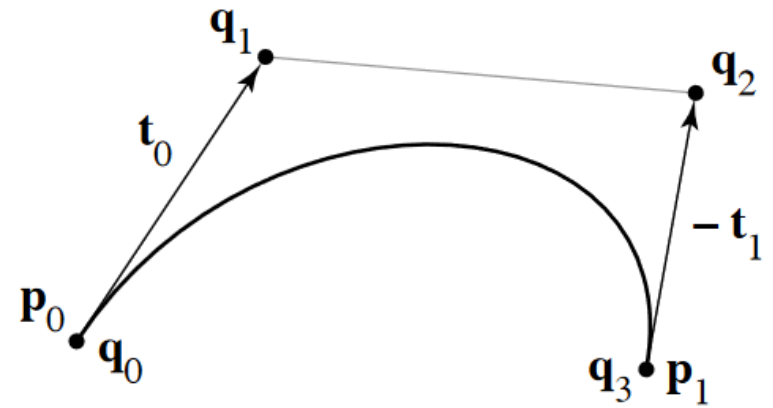
Hermite to Bézier

$$\mathbf{p}_0 = \mathbf{q}_0$$

$$\mathbf{p}_1 = \mathbf{q}_3$$

$$\mathbf{v}_0 = 3(\mathbf{q}_1 - \mathbf{q}_0)$$

$$\mathbf{v}_1 = 3(\mathbf{q}_3 - \mathbf{q}_2)$$



$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \end{bmatrix}$$

Hermite matrix

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \end{bmatrix}$$

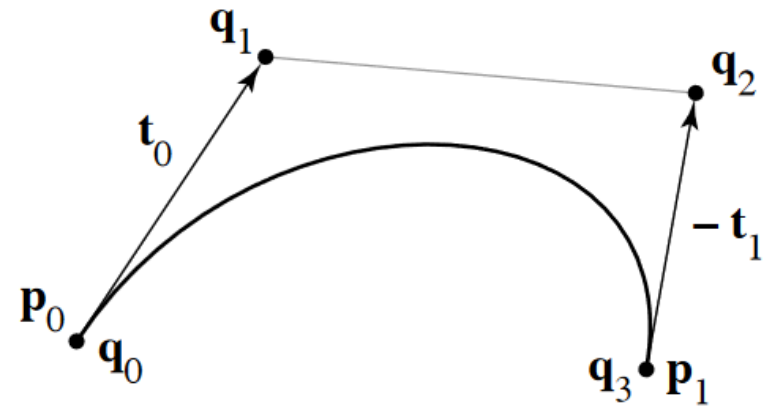
$$\mathbf{p}_0 = \mathbf{q}_0$$

$$\mathbf{p}_1 = \mathbf{q}_3$$

$$\mathbf{v}_0 = 3(\mathbf{q}_1 - \mathbf{q}_0)$$

$$\mathbf{v}_1 = 3(\mathbf{q}_3 - \mathbf{q}_2)$$

Hermite to Bézier



$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_3 \end{bmatrix}$$

Bézier matrix

Bezier basis matrix

use notation '**p**' instead of '**q**'

$$\mathbf{p}(t) = [t^3 \quad t^2 \quad t \quad 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

- note that these are the Bernstein polynomials

$$b_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k}$$

and that defines Bézier curves for any degree

Bezier Curve

- Bernstein basis functions

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

$$B_0^3(t) = (1-t)^3$$

$$B_1^3(t) = 3t(1-t)^2$$

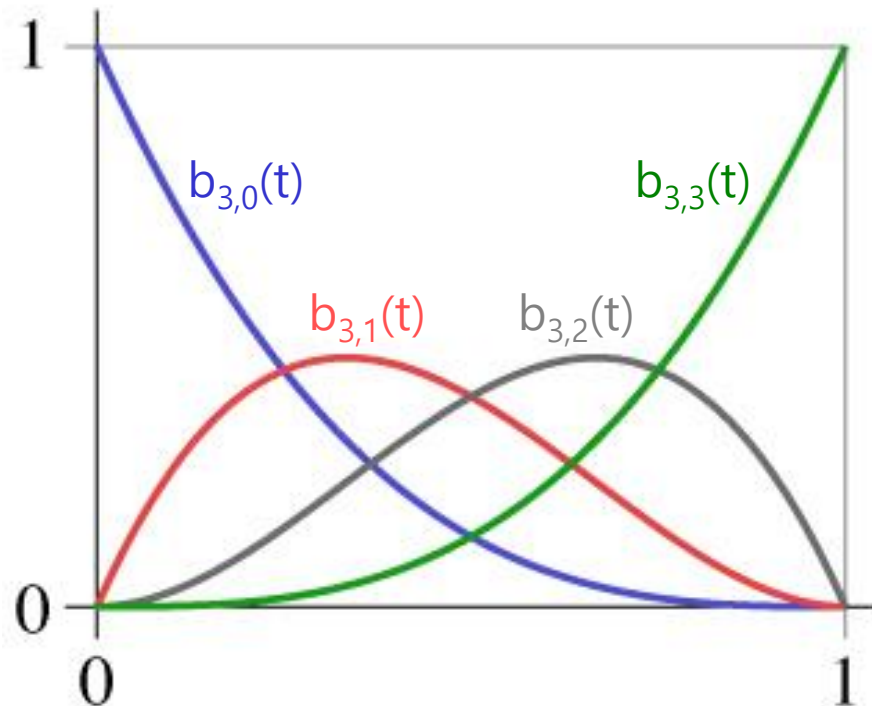
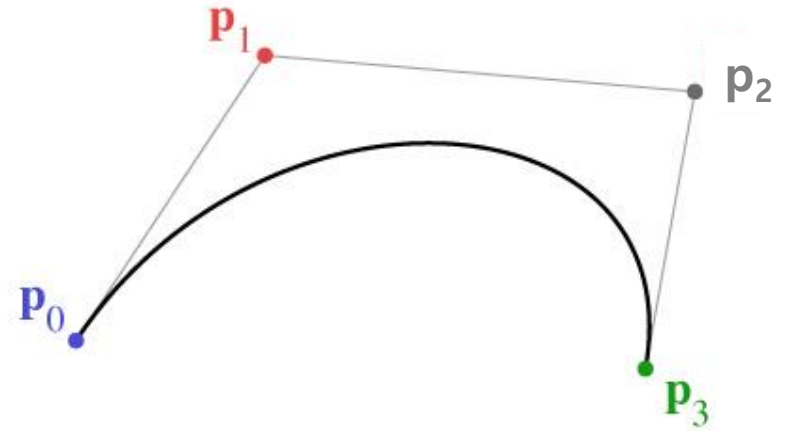
$$B_2^3(t) = 3t^2(1-t)$$

$$B_3^3(t) = t^3$$

- Cubic Bezier curve: Cubic polynomial in Bernstein bases

$$\begin{aligned} \mathbf{p}(t) &= B_0^3(t)\mathbf{p}_0 + B_1^3(t)\mathbf{p}_1 + B_2^3(t)\mathbf{p}_2 + B_3^3(t)\mathbf{p}_3 \\ &= (1-t)^3 \mathbf{p}_0 + 3t(1-t)^2 \mathbf{p}_1 + 3t^2(1-t) \mathbf{p}_2 + t^3 \mathbf{p}_3 \end{aligned}$$

Bézier basis



de Casteljau's Algorithm



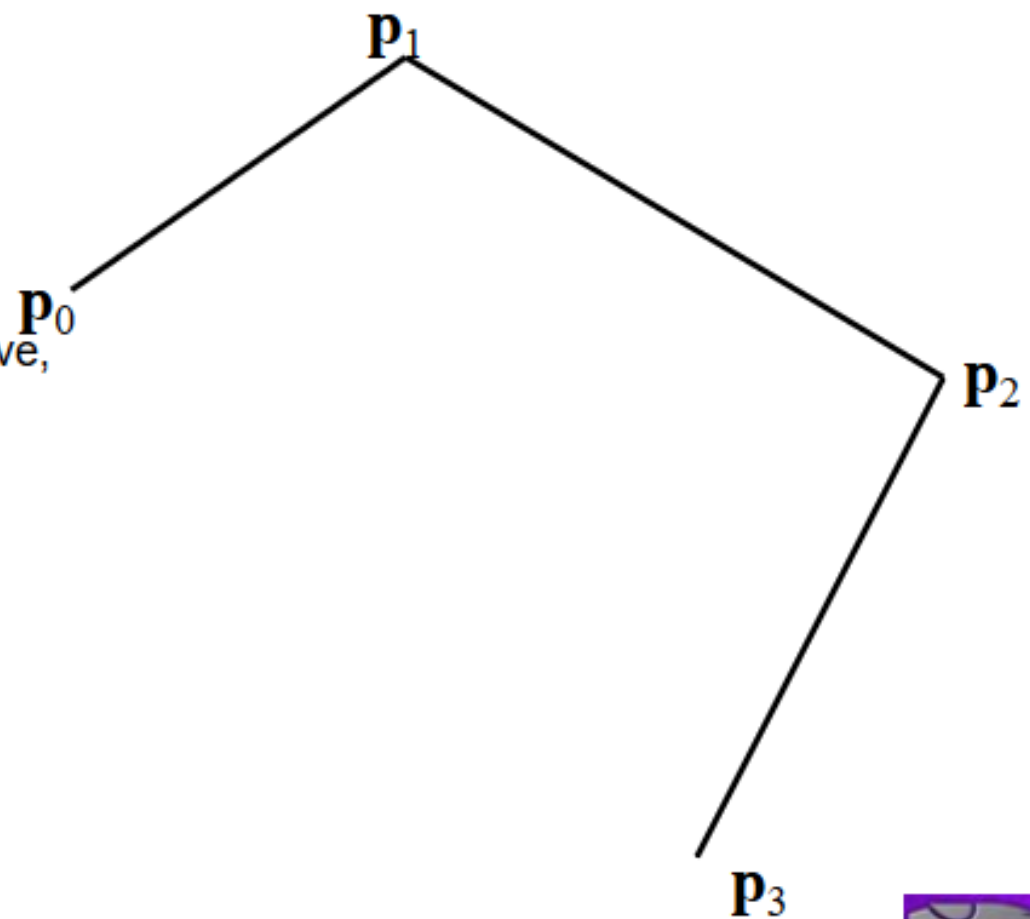
Paul de Casteljau (1930-) first developed the 'Bezier' curve using this algorithm in 1959 while working at Citroën, but was not able to publish them due to company policy

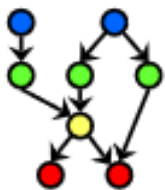
- Another method to compute Bezier curve



DE CASTELJAU ALGORITHM

- We start with our original set of points
- In the case of a cubic Bezier curve, we start with four points



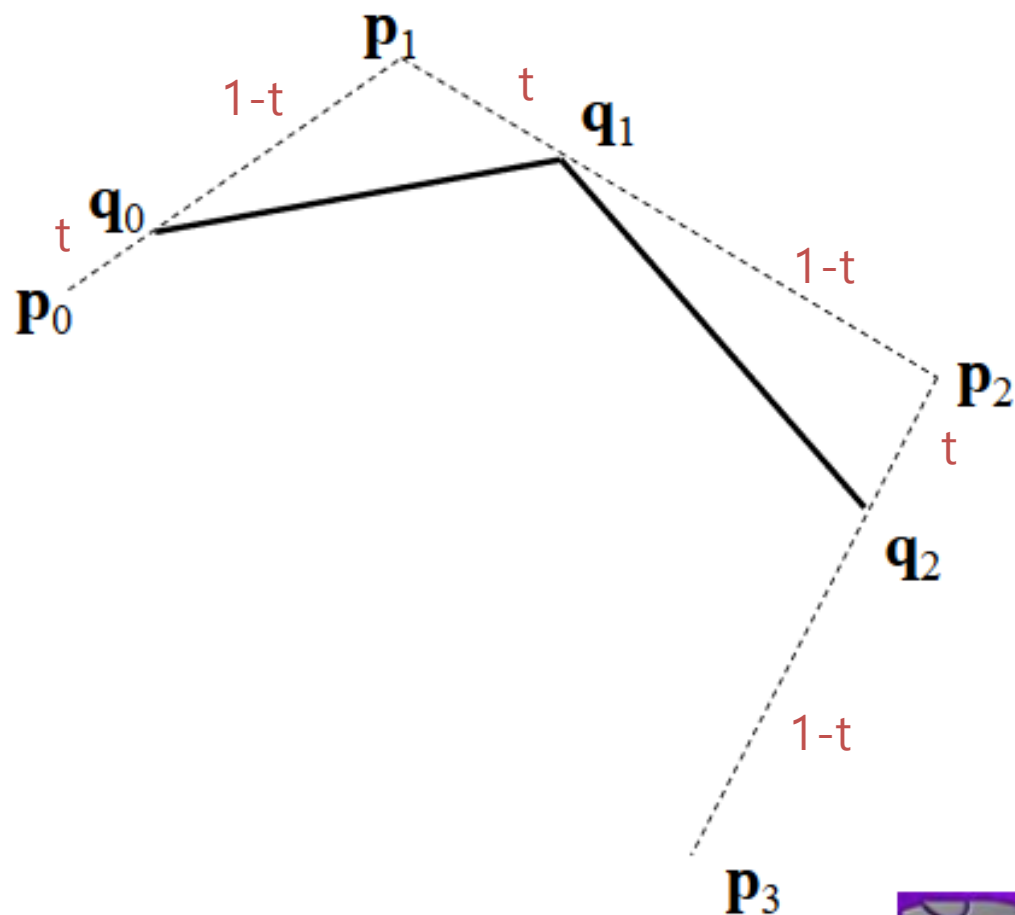


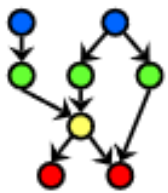
DE CASTELJAU ALGORITHM

$$\mathbf{q}_0 = \text{Lerp}(t, \mathbf{p}_0, \mathbf{p}_1)$$

$$\mathbf{q}_1 = \text{Lerp}(t, \mathbf{p}_1, \mathbf{p}_2)$$

$$\mathbf{q}_2 = \text{Lerp}(t, \mathbf{p}_2, \mathbf{p}_3)$$

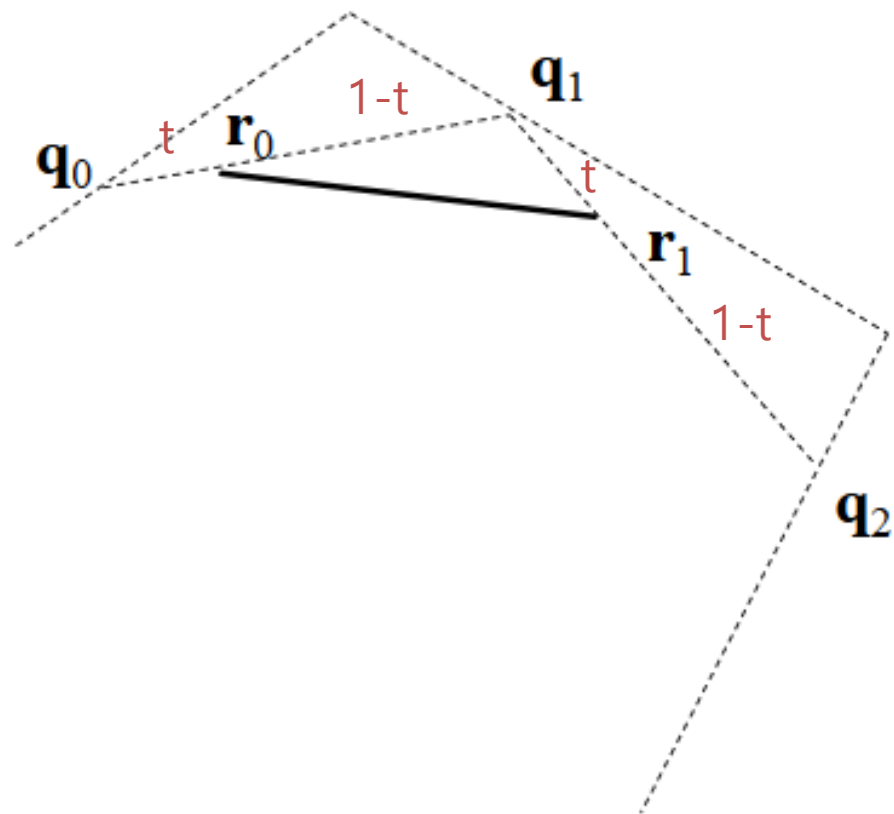


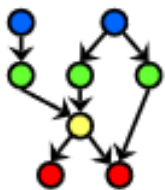


DE CASTELJAU ALGORITHM

$$\mathbf{r}_0 = \text{Lerp}(t, \mathbf{q}_0, \mathbf{q}_1)$$

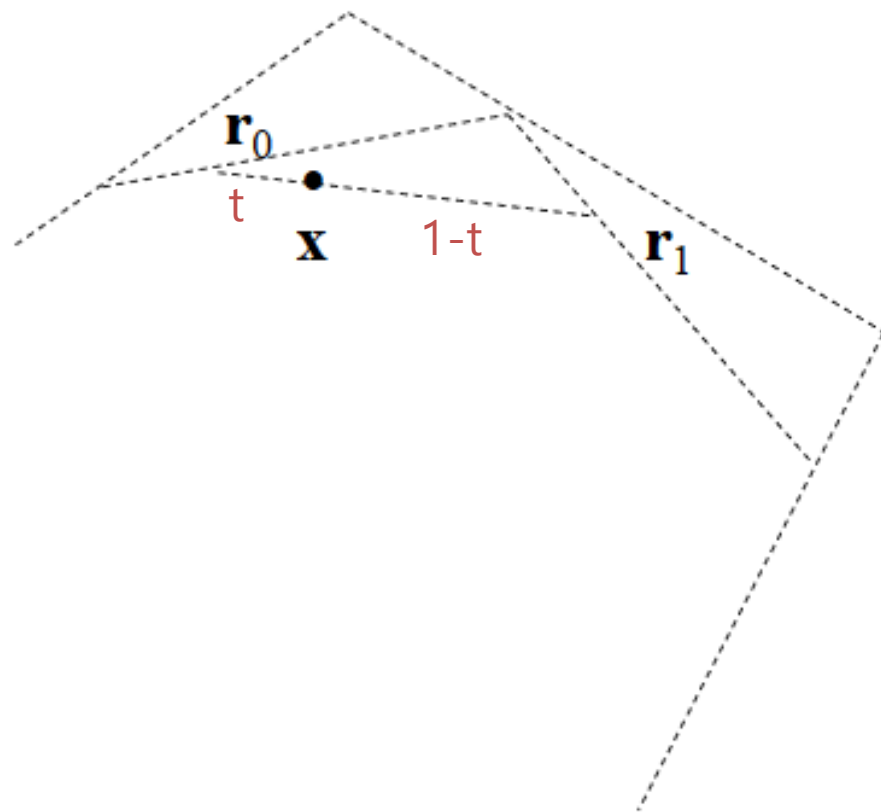
$$\mathbf{r}_1 = \text{Lerp}(t, \mathbf{q}_1, \mathbf{q}_2)$$



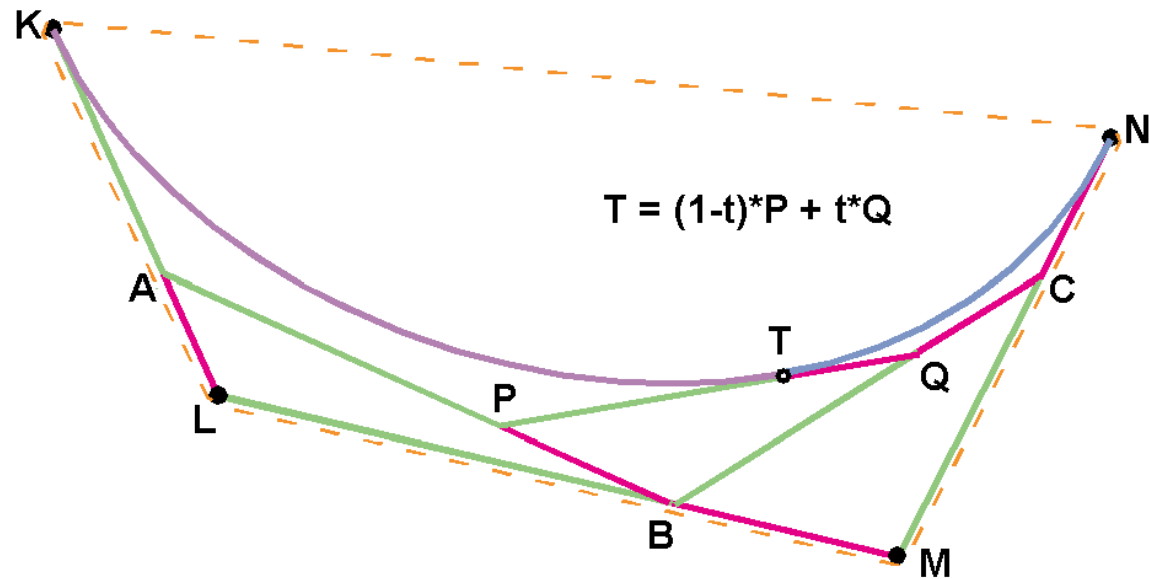


DE CASTELJAU ALGORITHM

$$\mathbf{x} = \text{Lerp}(t, \mathbf{r}_0, \mathbf{r}_1)$$



de Casteljau's Algorithm



$$T = (1-t)P + tQ$$

$$P = (1-t)A + tB$$

$$Q = (1-t)B + tC$$

$$T = (1-t)(1-t)A + (1-t)tB + t(1-t)B + t^2C$$

$$A = (1-t)K + tL$$

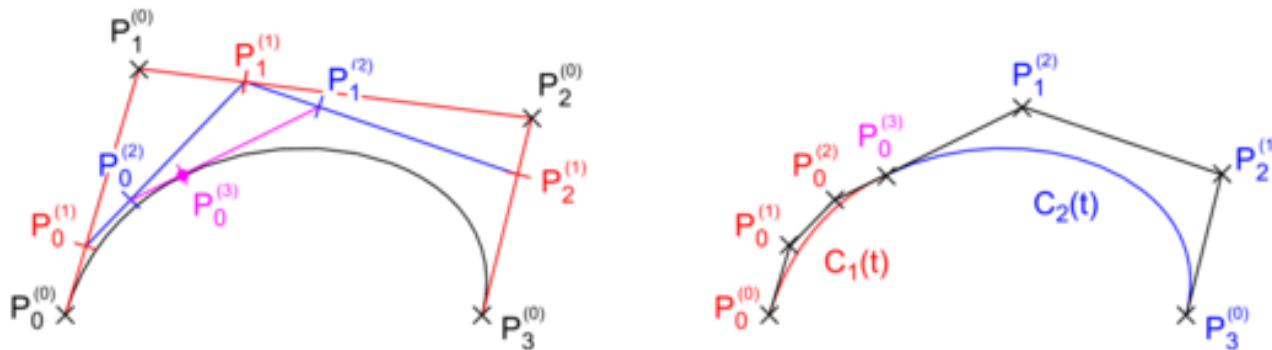
$$B = (1-t)L + tM$$

$$C = (1-t)M + tN$$

$$T = (1-t)^3K + (1-t)^2tL + 2(1-t)t^2L + 2(1-t)t^2M + (1-t)t^2M + t^3N$$

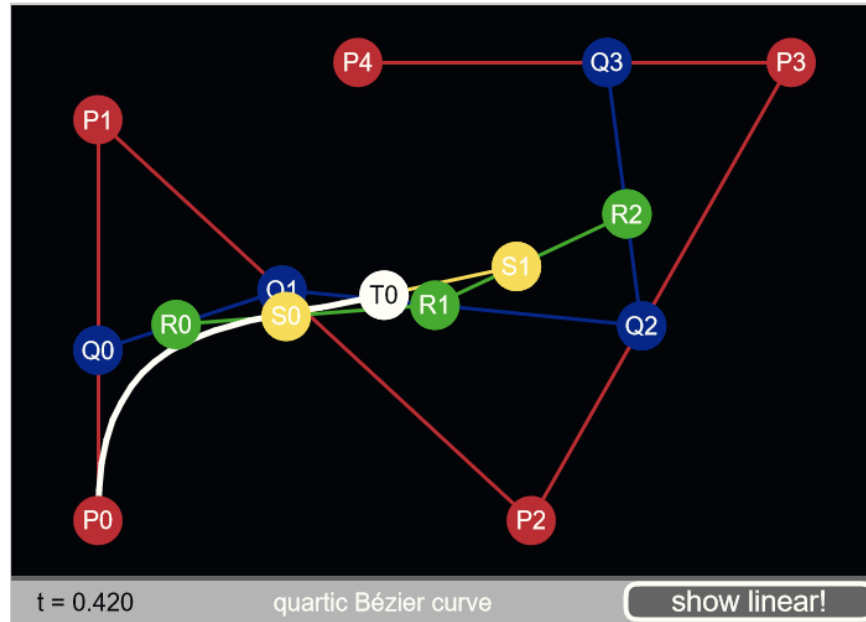
de Casteljau's Algorithm

- Nice recursive algorithm to compute a point on a Bezier curve
- Additionally, it subdivides a Bezier curve into two segments



- You can draw a curve with a sufficient number of subdivided control points
 - "Subdivision" method for displaying curves

[Practice] de Casteljau's Algorithm



<http://www.malinc.se/m/DeCasteljauAndBezier.php>

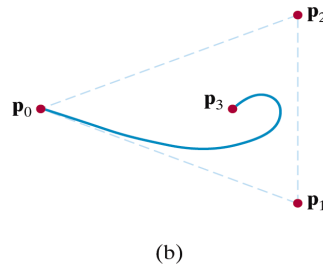
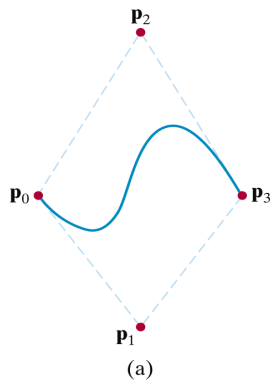
- Move red points
- Also check the subdivision demo

Displaying Curves

- Need to generate a list of line segments to draw
 - What we can compute is a set of points on a curve
 - Connecting them with line segments would be good approximation for the curve
- Brute-force
 - Evaluate $\mathbf{p}(t)$ for incrementally spaced values of t
- Finite difference
 - The same idea, but much more efficient
 - See <http://www.drdoobbs.com/forward-difference-calculation-of-bezier/184403417>
- Subdivision
 - Use de Casteljau's algorithm

Properties of Bezier Curve

- Intuitively controlled by control points
- Contained in the *convex hull* of control points



Convex hull: Minimal-sized convex polygon containing all points

- End point interpolation

Quiz #3

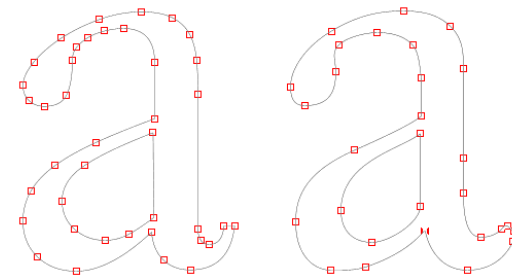
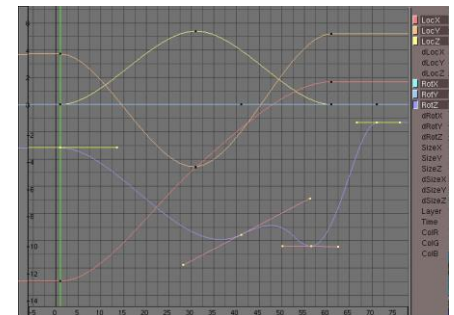
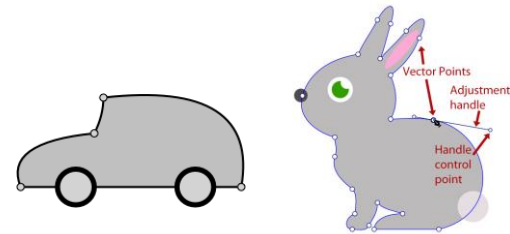
- Go to <https://www.slido.com/>
- Join #cg-ys
- Click “Polls”

- Submit your answer in the following format:
 - **Student ID: Your answer**
 - e.g. **2017123456: 4)**

- Note that you must submit all quiz answers in the above format to be checked for “attendance”.

Bezier Spline

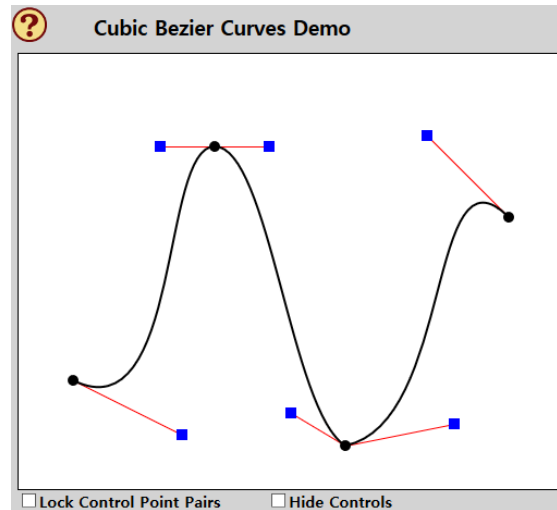
- A combination of piecewise Bezier curves, Bezier spline, is very widely used. For example,
- To draw shapes in graphic tools such as Adobe Illustrator
- To define animation paths in 3D authoring tools such as Blender and Maya
- TrueType fonts use quadratic Bezier spline, PostScript fonts use cubic Bezier spline



True Type Font

Postscript Font

[Practice] Bezier Spline

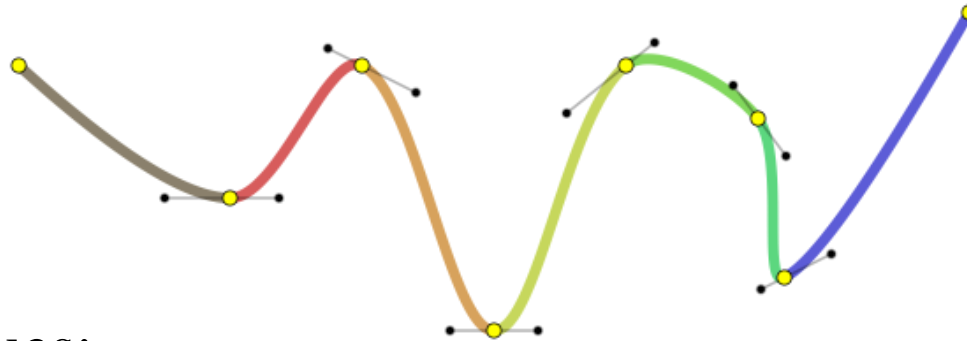


<http://math.hws.edu/graphicsbook/demos/c2/cubic-bezier.html>

- How to “smooth” the spline?

Spline

- Spline: *piecewise polynomial*



- Three issues:
 - How to connect these pieces *continuously*?
 - How easy is it to "*control*" the shape of a spline?
 - Does a spline have to *pass through* specific points?
- For details, see *11-reference-splines.pdf*

Next Time

- Lab in this week:
 - Lab assignment 11

- Next lecture:
 - 12 - More Lighting, Texture

- Acknowledgement: Some materials come from the lecture slides of
 - Prof. Jehee Lee, SNU, http://mrl.snu.ac.kr/courses/CourseGraphics/index_2017spring.html
 - Prof. Taesoo Kwon, Hanyang Univ., <http://calab.hanyang.ac.kr/cgi-bin/cg.cgi>
 - Prof. Steve Marschner, Cornell Univ., <http://www.cs.cornell.edu/courses/cs4620/2014fa/index.shtml>
 - Prof. William H. Hsu, Kansas State Univ. <http://slideplayer.com/slide/4635444/>