GRK 7

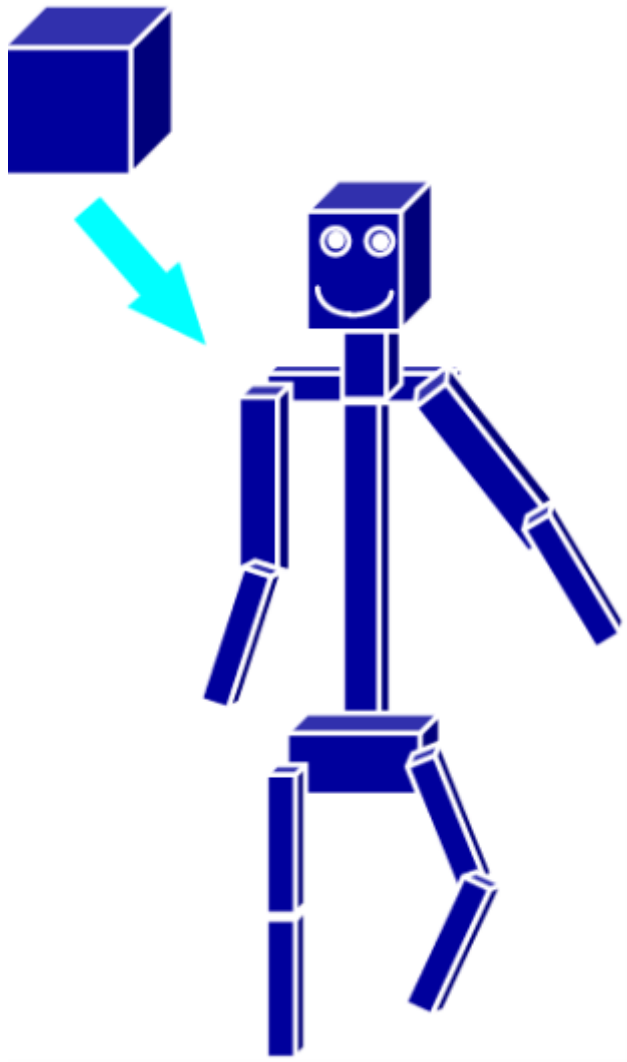
Dr Wojciech Palubicki

Egzamin

- 11.30, 4.7.2025, Aula A
- Prezentacja gupowa (lista tematow dostepna w google docsie)
  - Prezentacja metody grafiki komputerowej (4 minut) – 20%
  - Prezentacja projektu grafiki komputerowej (3 minuty) – 40%
    - Github/Gitlab z krotkim opisem glownych cech i screenshot aplikacji (do 11.30 umieszczony w google docsie jako link)
    - Kod zrodlowy
- Ocena z cwiczen → ocena z wykladu
- Bonusowe punkty doliczamy (max. 10%)

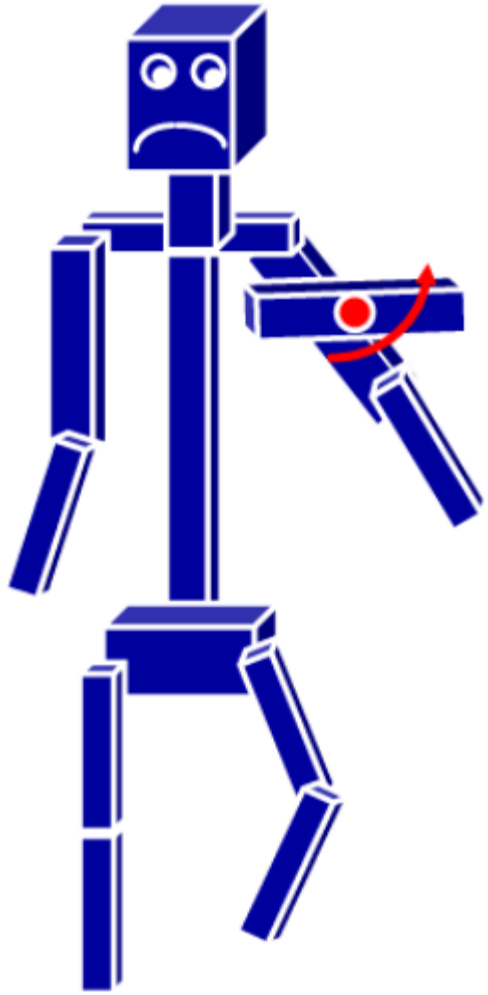
Modeling with Transformations

- Create elementary geometric objects, then rotate, translate and scale them until you define a model

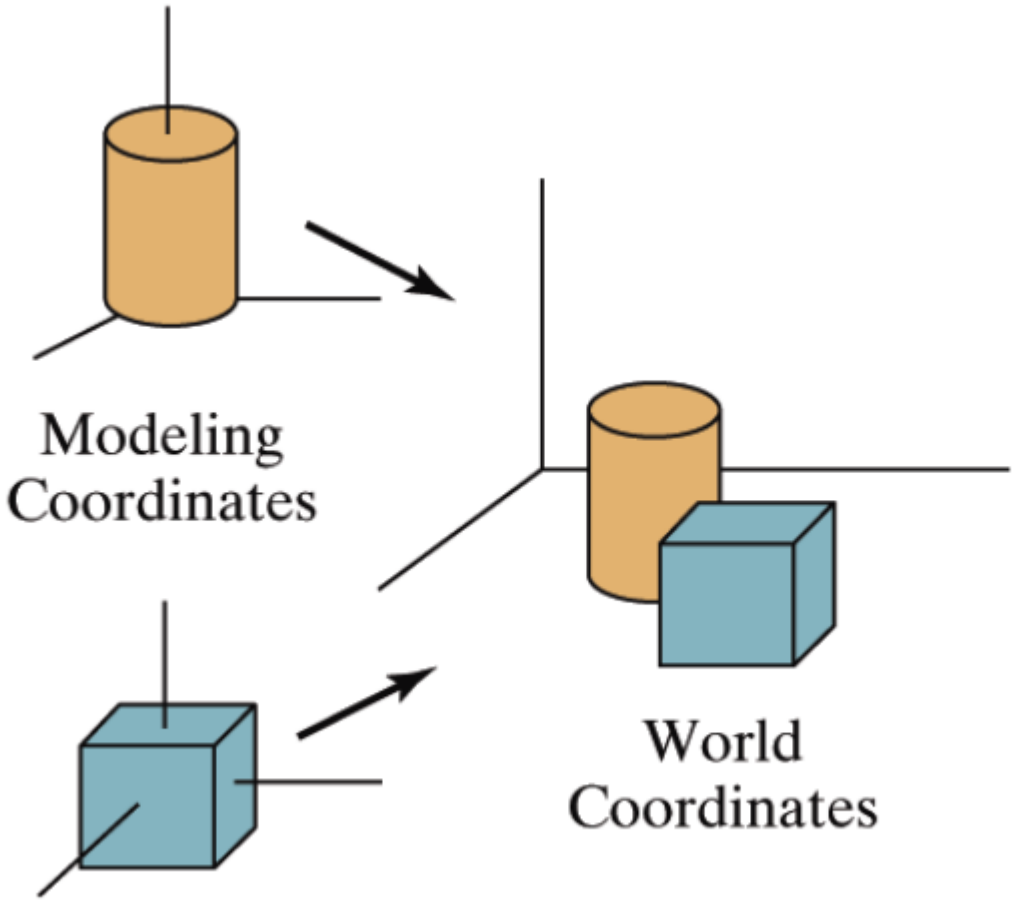


Modeling with Transformations

- But individual parts dont move in a constrained way to each other
- To introduce constraints and express kinematics we need to parametrize our model

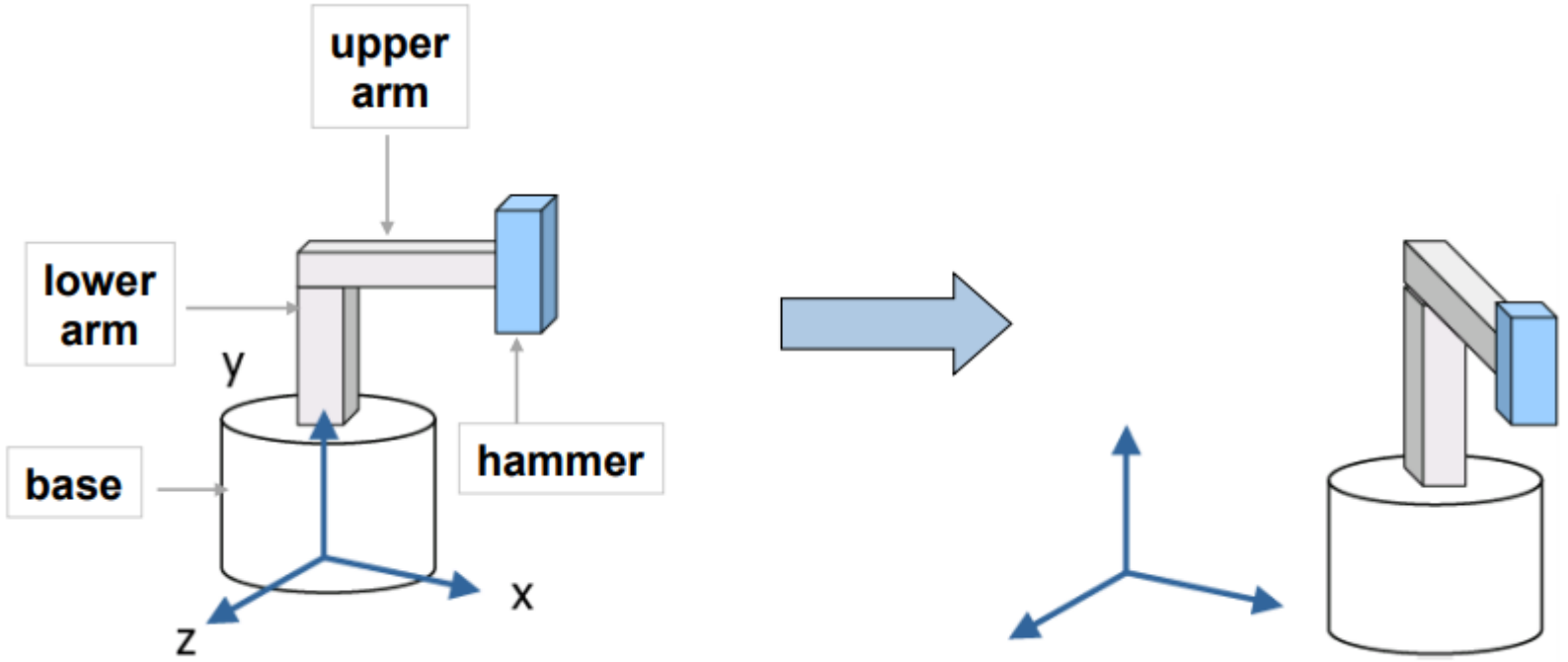


Model to World Space



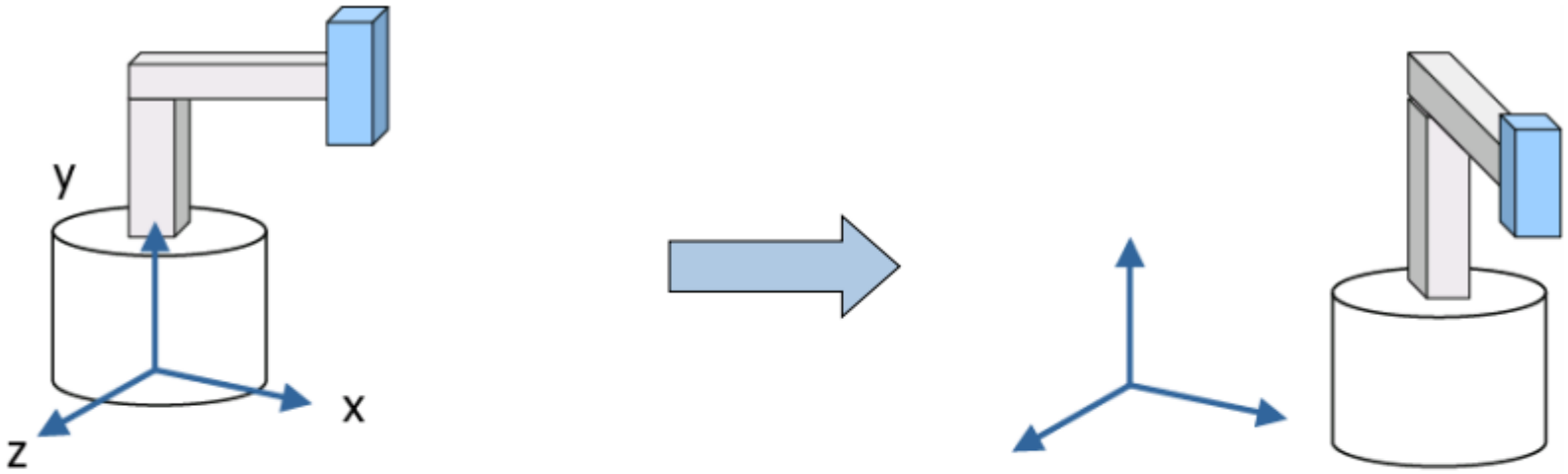
Model → World

- Position and orient the robot hammer in world space



Model → World

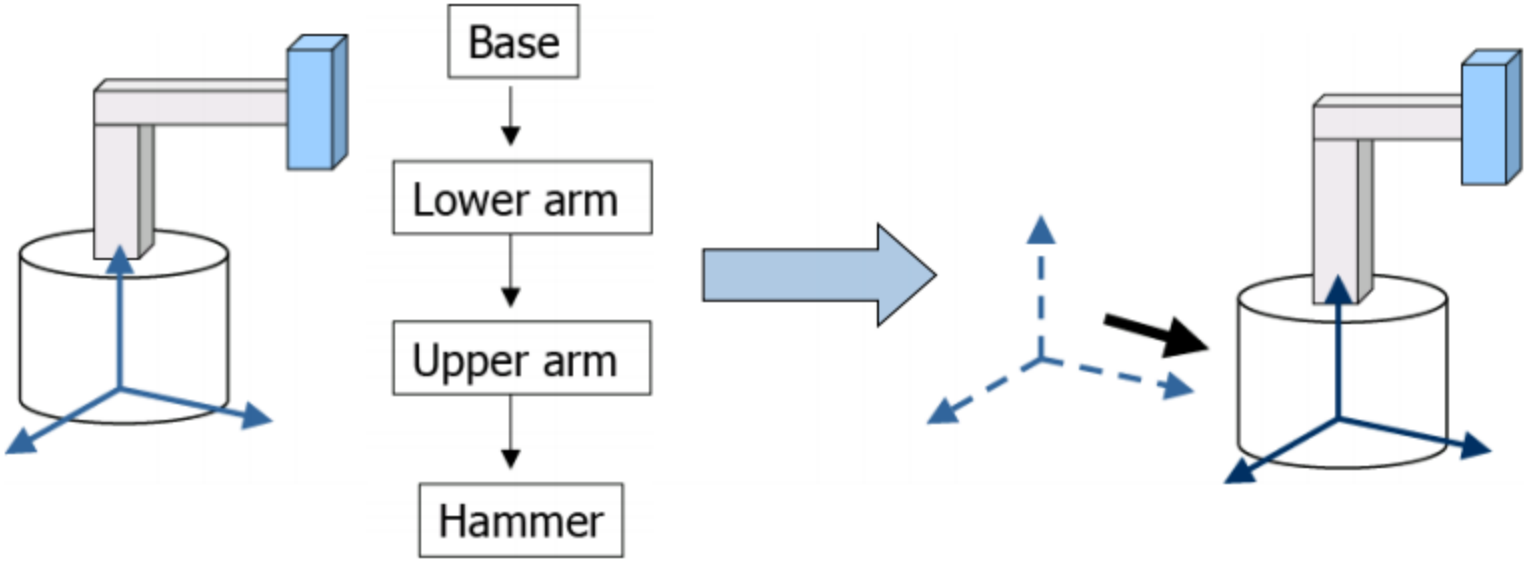
- Each part of the object is transformed independently relative to the origin



Translate base by (5,0,0)
Translate lower arm by (5,0,0)
Translate upper arm by (5,0,0)
Translate hammer by (5,0,0)
…

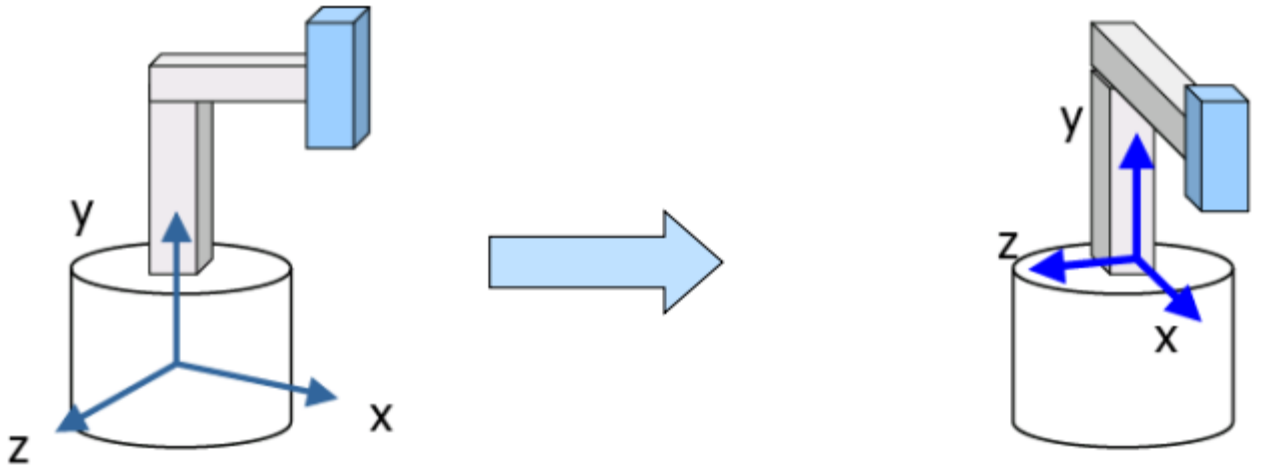
Model → World

- Alternatively, transform every object relative to it's parent



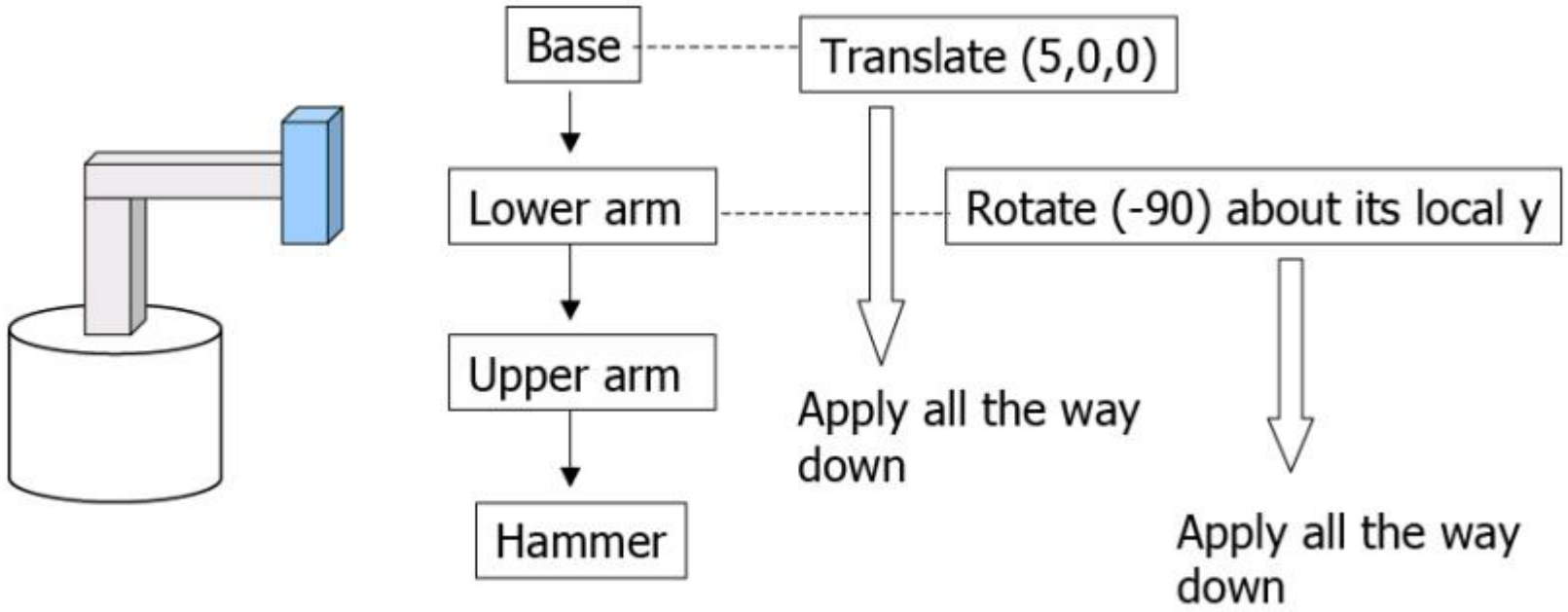
Step 1: Translate *base and its descendants* by (5,0,0)

Relative Transformations



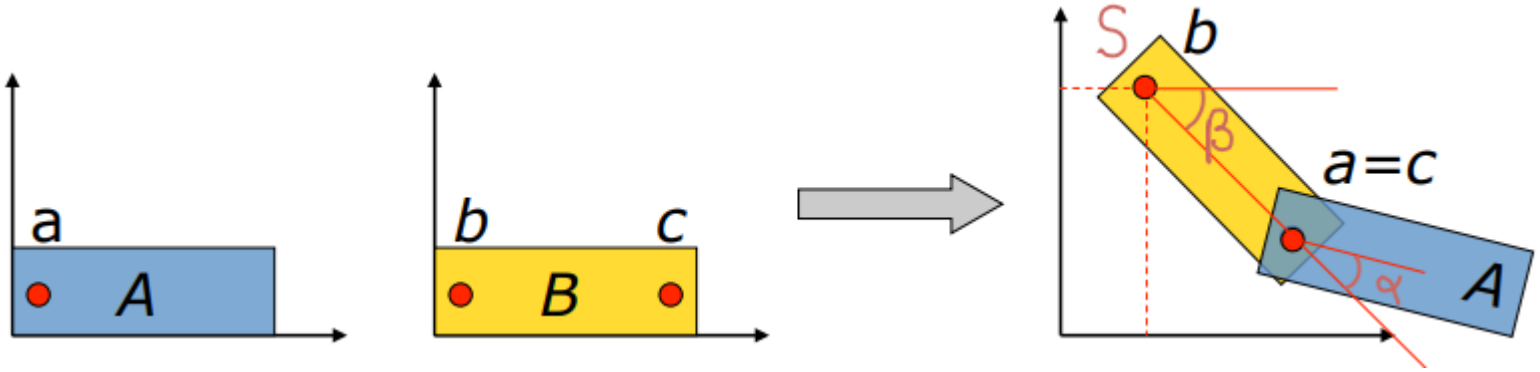
Step 2: Rotate lower arm and its descendants by -90 degrees about local y axis

Hierarchical Transforms



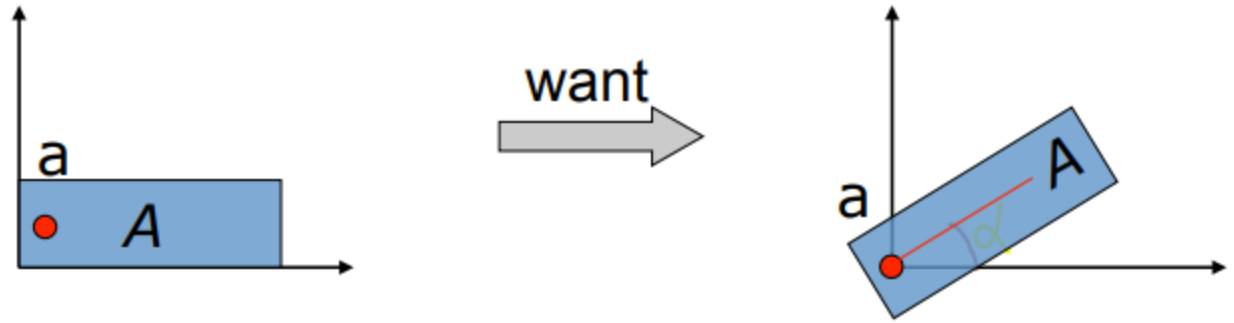
Making an Articulated Arm

- A minimal 2D jointed object:
  - Two pieces, A(”forearm”) and B(”upper arm”)
  - Attach point c on B to point a on A (”elbow”)
- Desired parameters:
  - Shoulder position S (point at which b winds up)
  - Shoulder angle β (A and B rotate together about b)
  - Elbow angle α (A rotates about a = c)



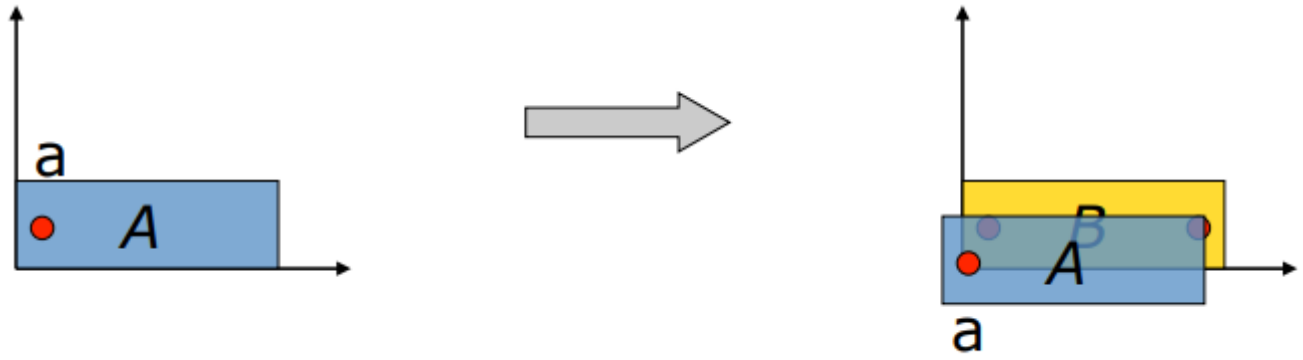
Making an Arm: Step 1

- Start with A and B in their untransformed configurations (B is hiding behind A)
- First apply a series of transformations to A.



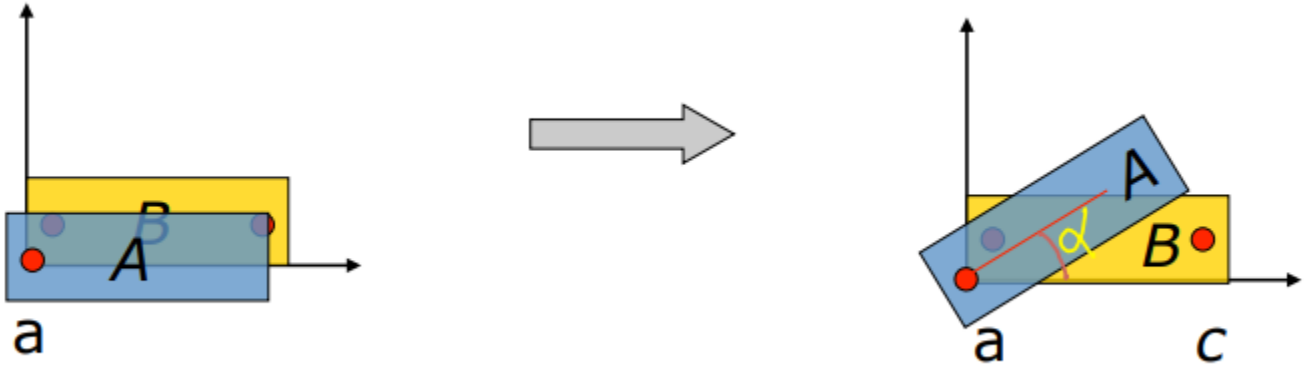
Making an Arm: Step 2

- Translate by –a, bringing a to the origin



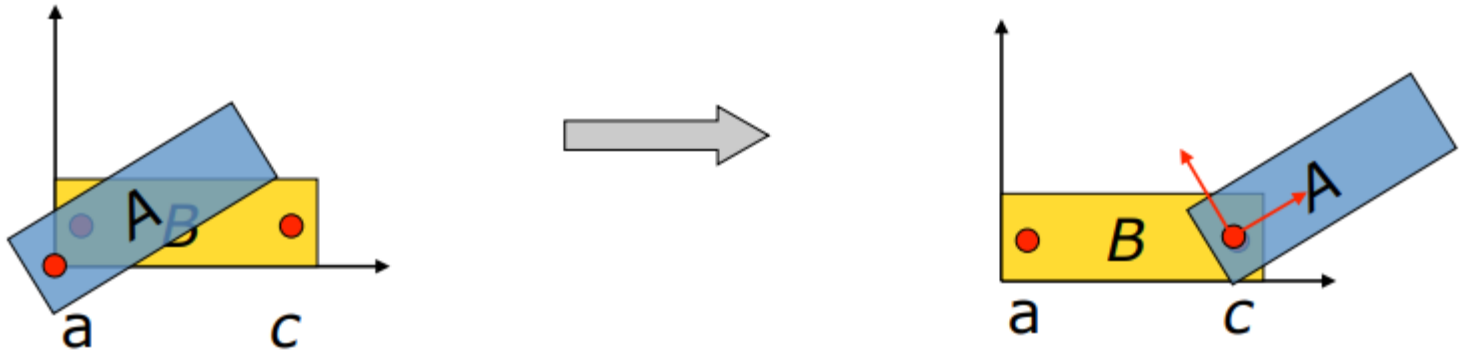
Making an Arm: Step 3

- Next, rotate A by the ”elbow” angle α



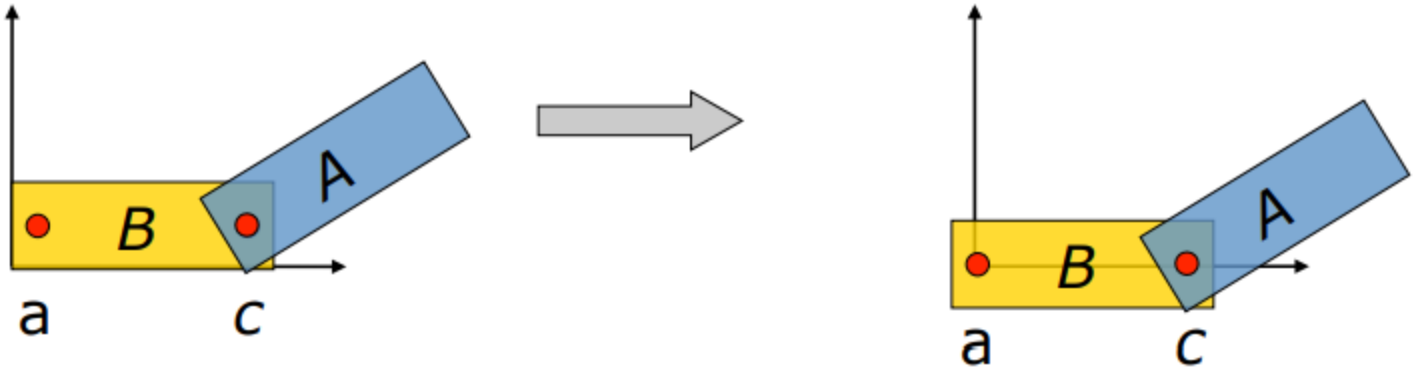
Making an Arm: Step 4

- Translate A to form the elbow joint a c



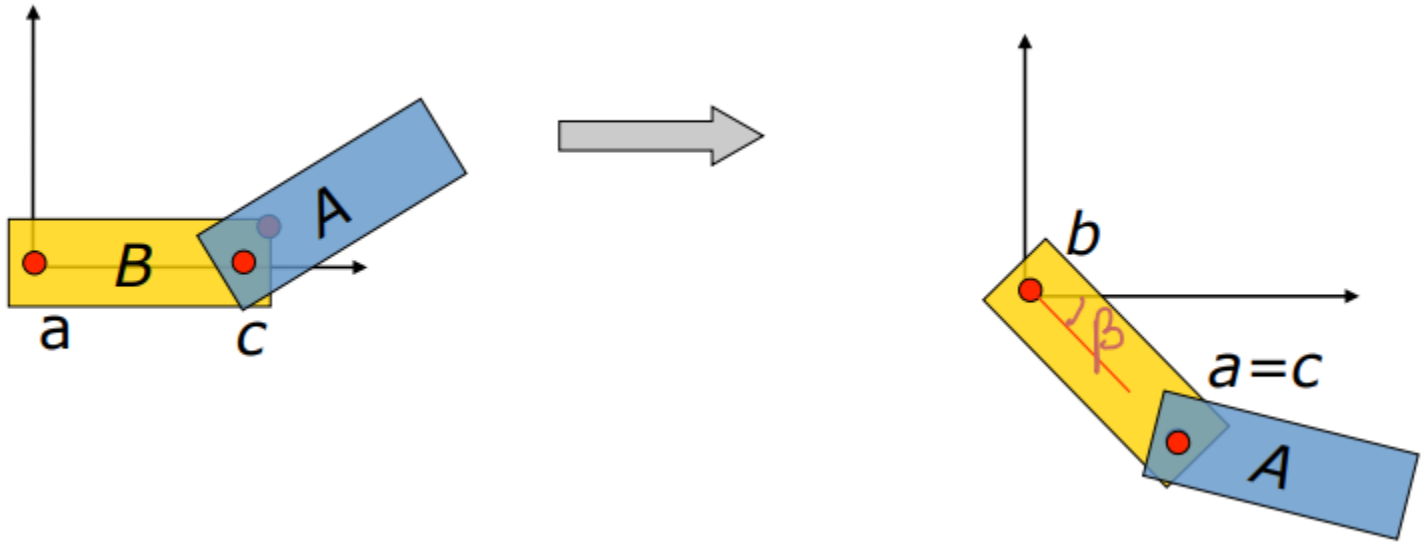
Making an Arm: Step 5

- Translate **both** objects by –a bringing a to the origin (A nd B move together)



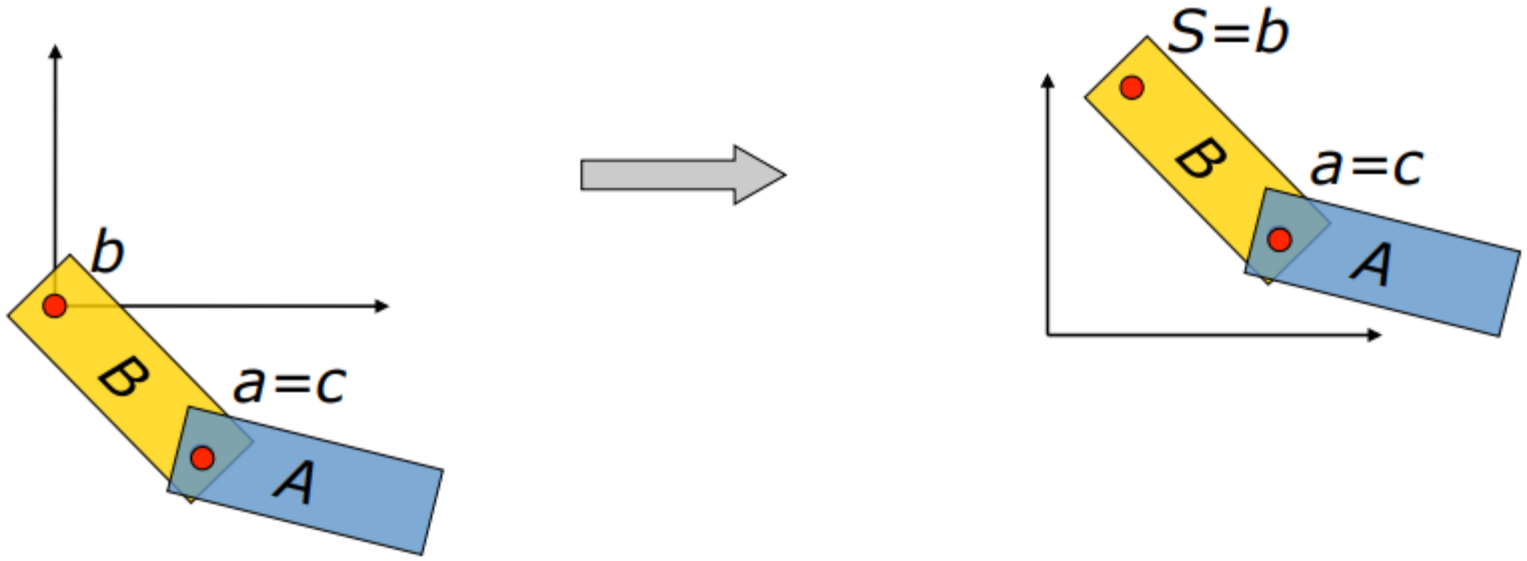
Making an Arm: Step 6

- Next rotate by the shoulder angle $-\beta$



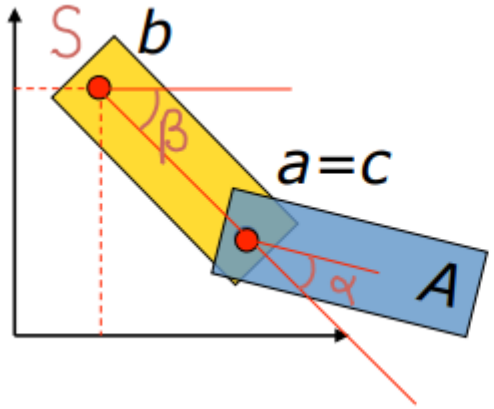
Making an Arm: Last Step

- Finally, translate by the shoulder position S, bringing the arm to its final position

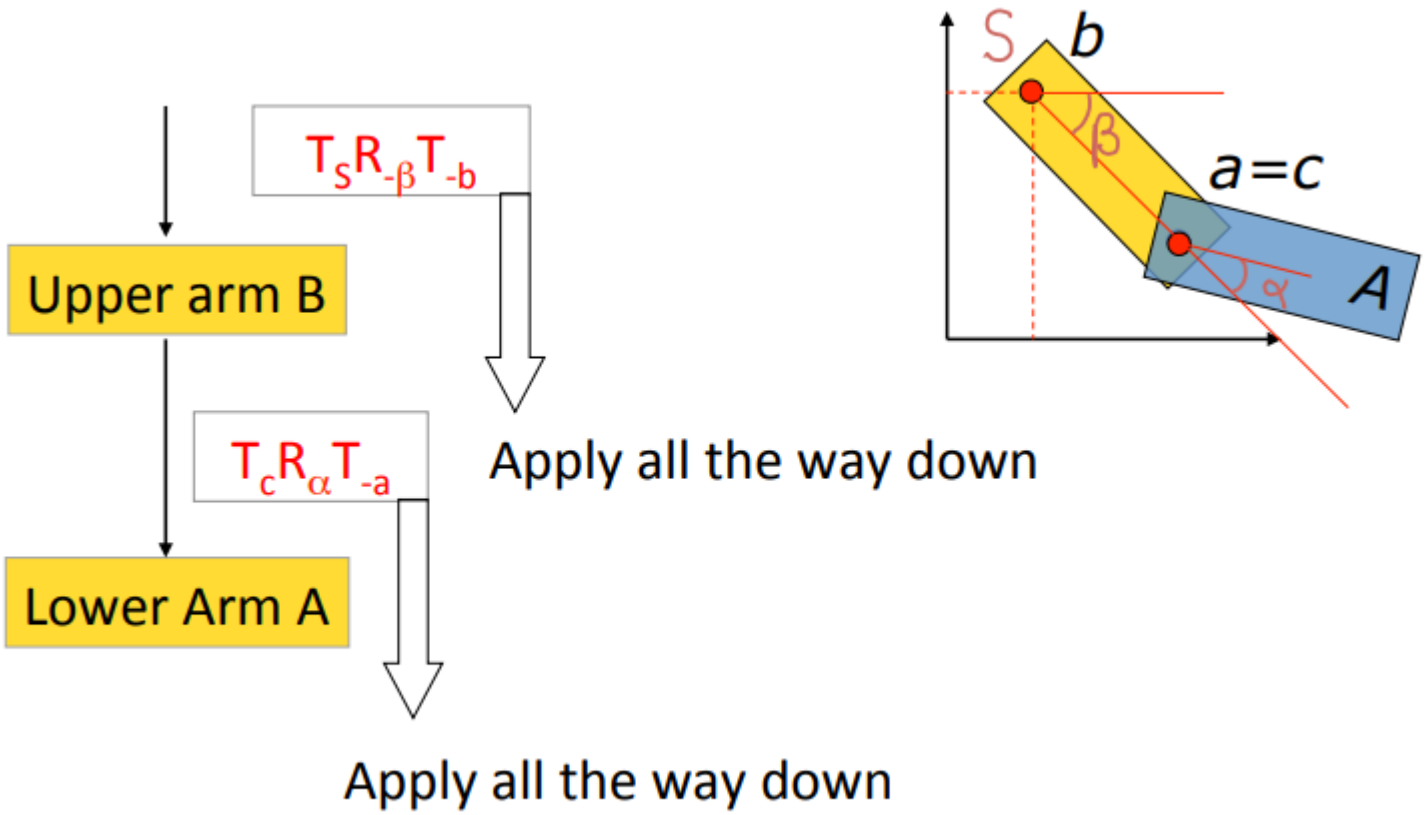


Parametrization

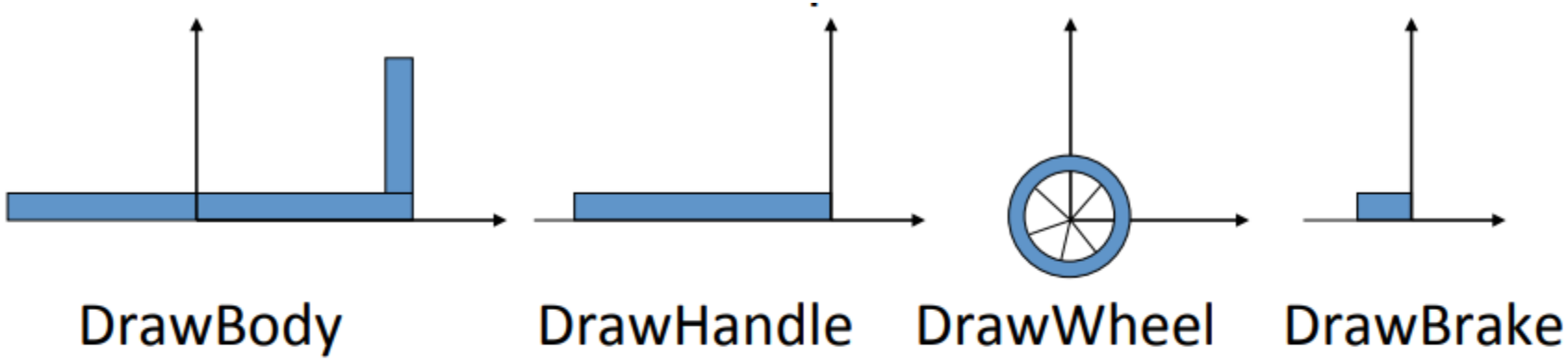
- S,α, β are parameters of the model
- a,b and c are structural constants

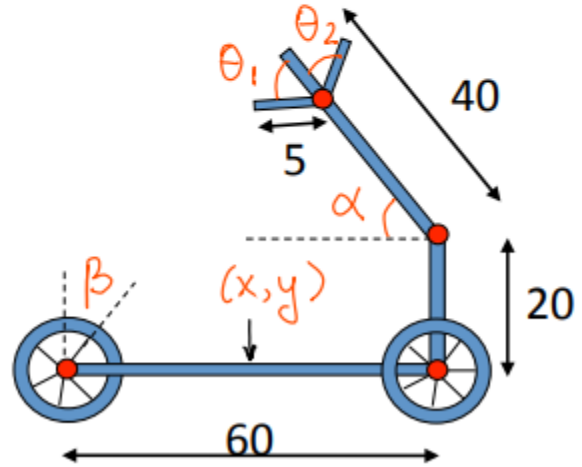


Hierarchical Transforms

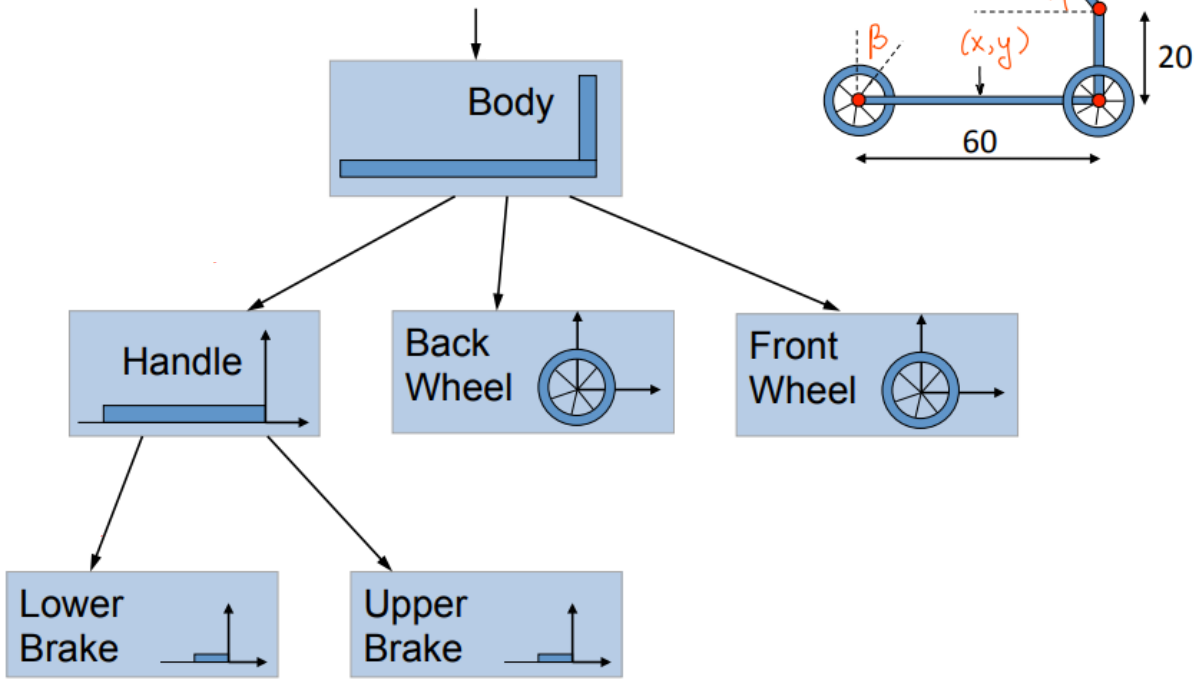


Model Construction

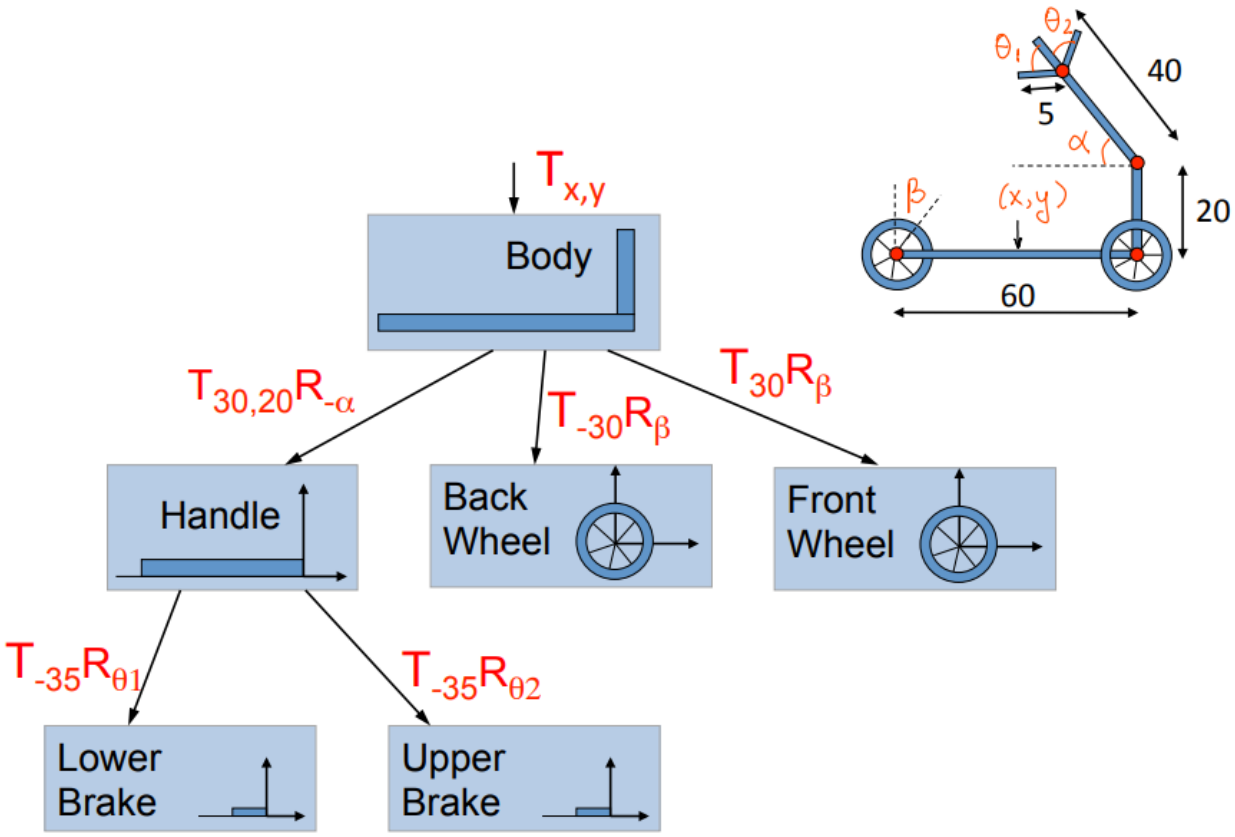




Scene Graph

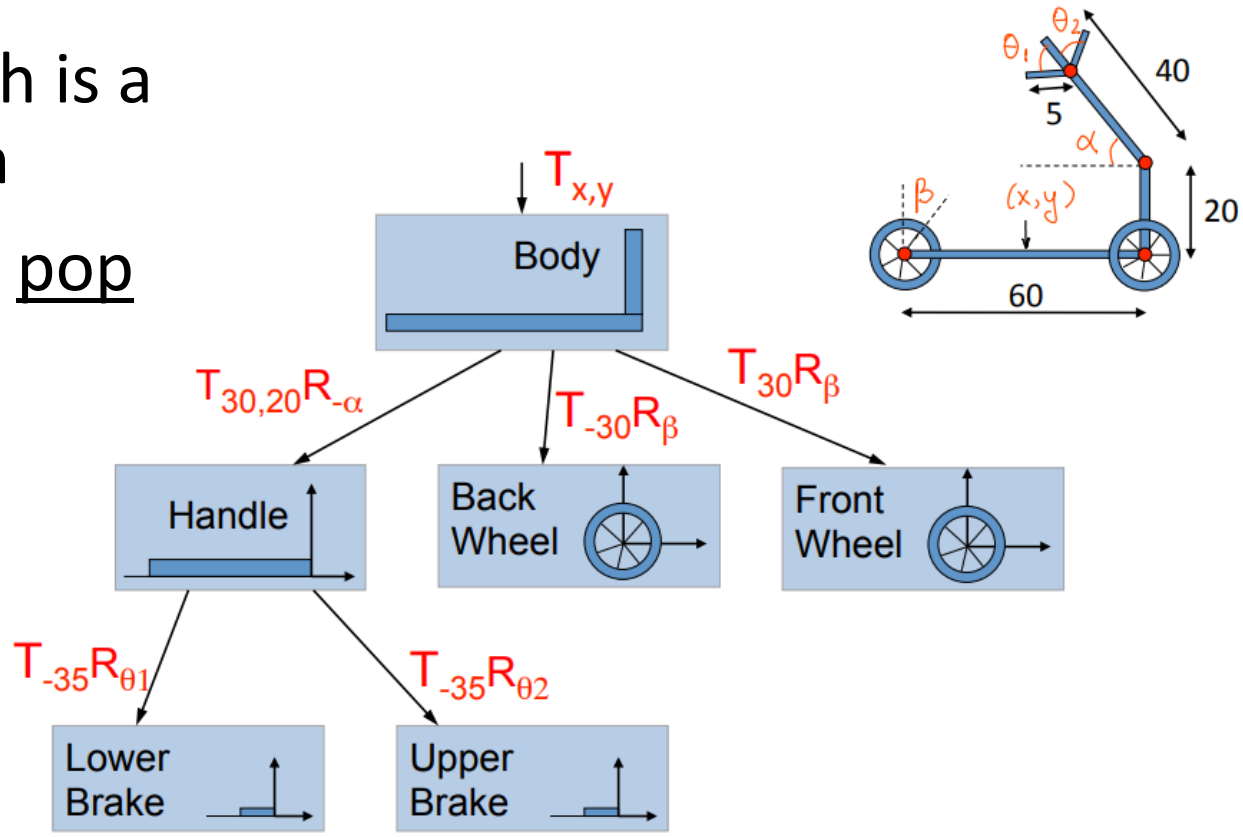

Body
Handle
Back Wheel
Front Wheel
Lower Brake
Upper Brake
θ2
θ1
40
5
α
β
(x,y)
20
60

Scene Graph


$T_{x,y}$
Body
$T_{30,20}R_{-\alpha}$
$T_{-30}R_{\beta}$
$T_{30}R_{\beta}$
Handle
Back Wheel
Front Wheel
$T_{-35}R_{\theta 1}$
$T_{-35}R_{\theta 2}$
Lower Brake
Upper Brake
θ2
θ1
40
5
α
β
(x,y)
20
60

Scene Graph

- Model e.g. as a stack
- Down the graph is a <u>push</u> operation
- Up the graph a <u>pop</u> operation



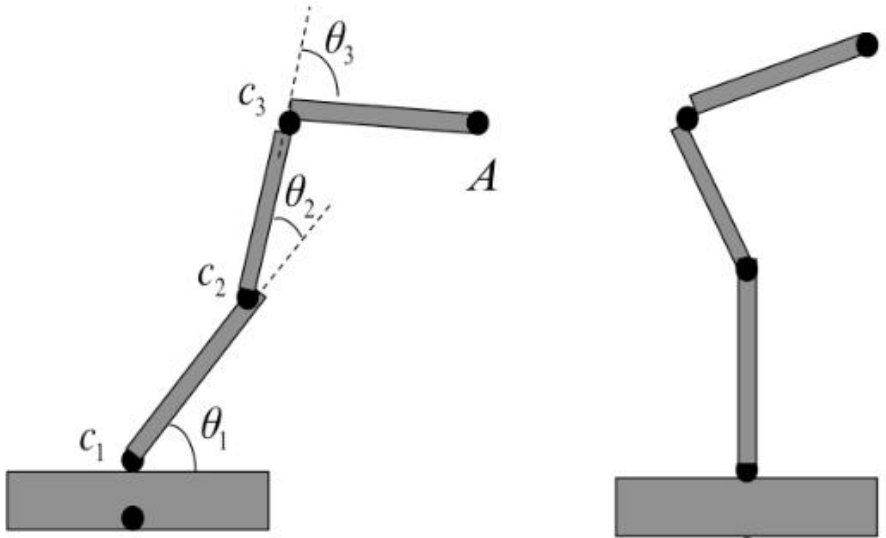
Scene Graph OpenGL 3.0+

```
void renderMesh(Matrix transform, Mesh mesh)
{
    // here call glDrawElements/glDrawArrays and send transform matrix to MVP uniform
    mesh->draw(transform);

    // now render all the sub-meshes, then will be transformed relative to current mesh
    for (int i=0; i<mesh->subMeshCount(); i++)
    {
        Matrix subMeshTransform = mesh->getSubMeshTransform(i);
        Mesh subMesh = mesh->getSubMesh();

        renderMesh(subMeshTransform * transform, subMesh);
    }
}
```

Keyframing

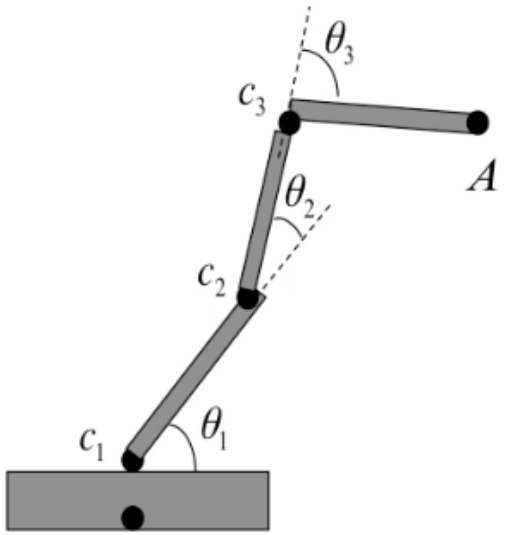


t=0

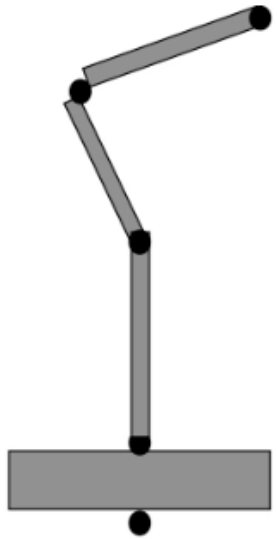
t=50ms

What's the inbetween motion?

Keyframing



t=0

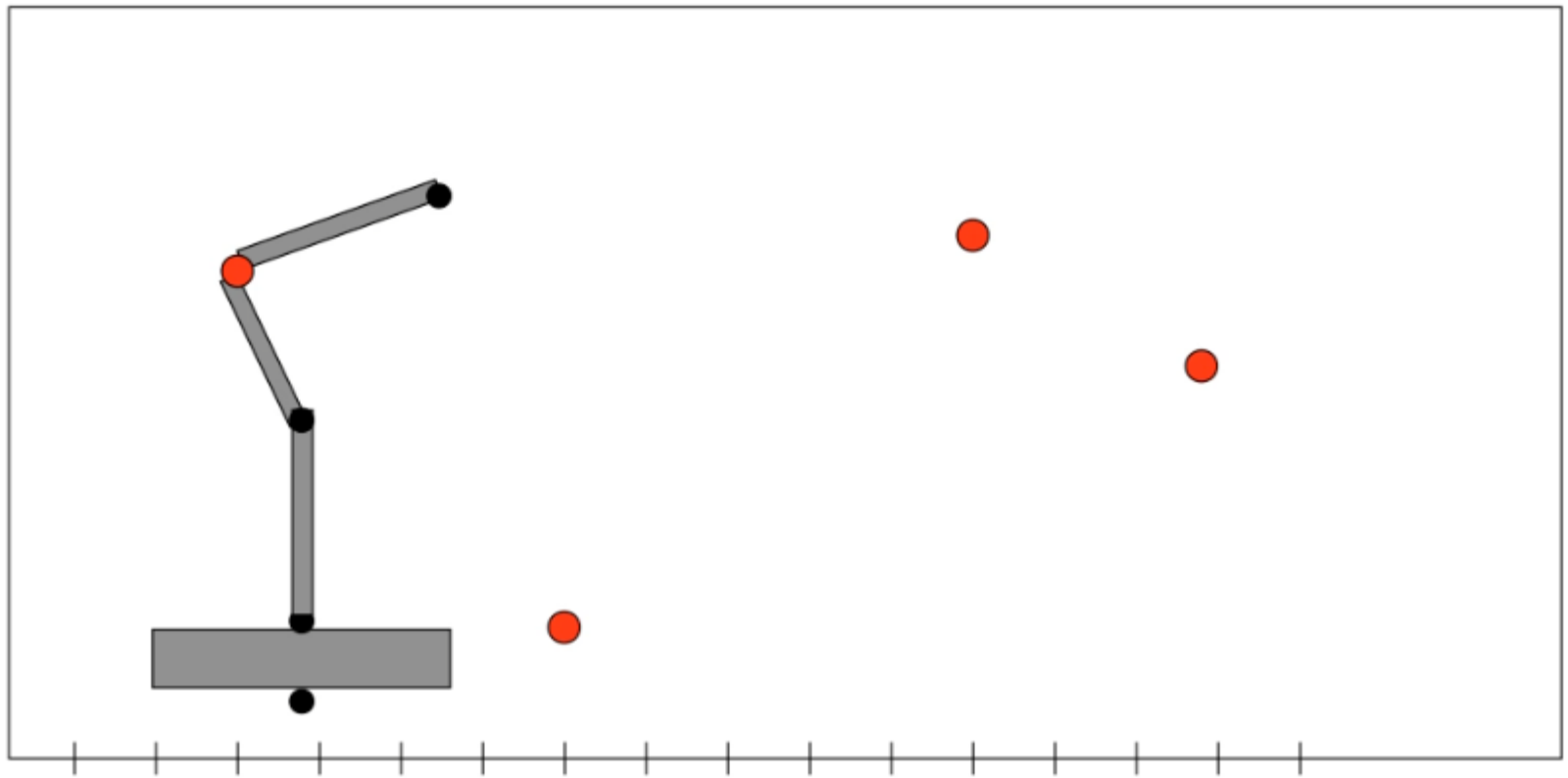


t=50ms

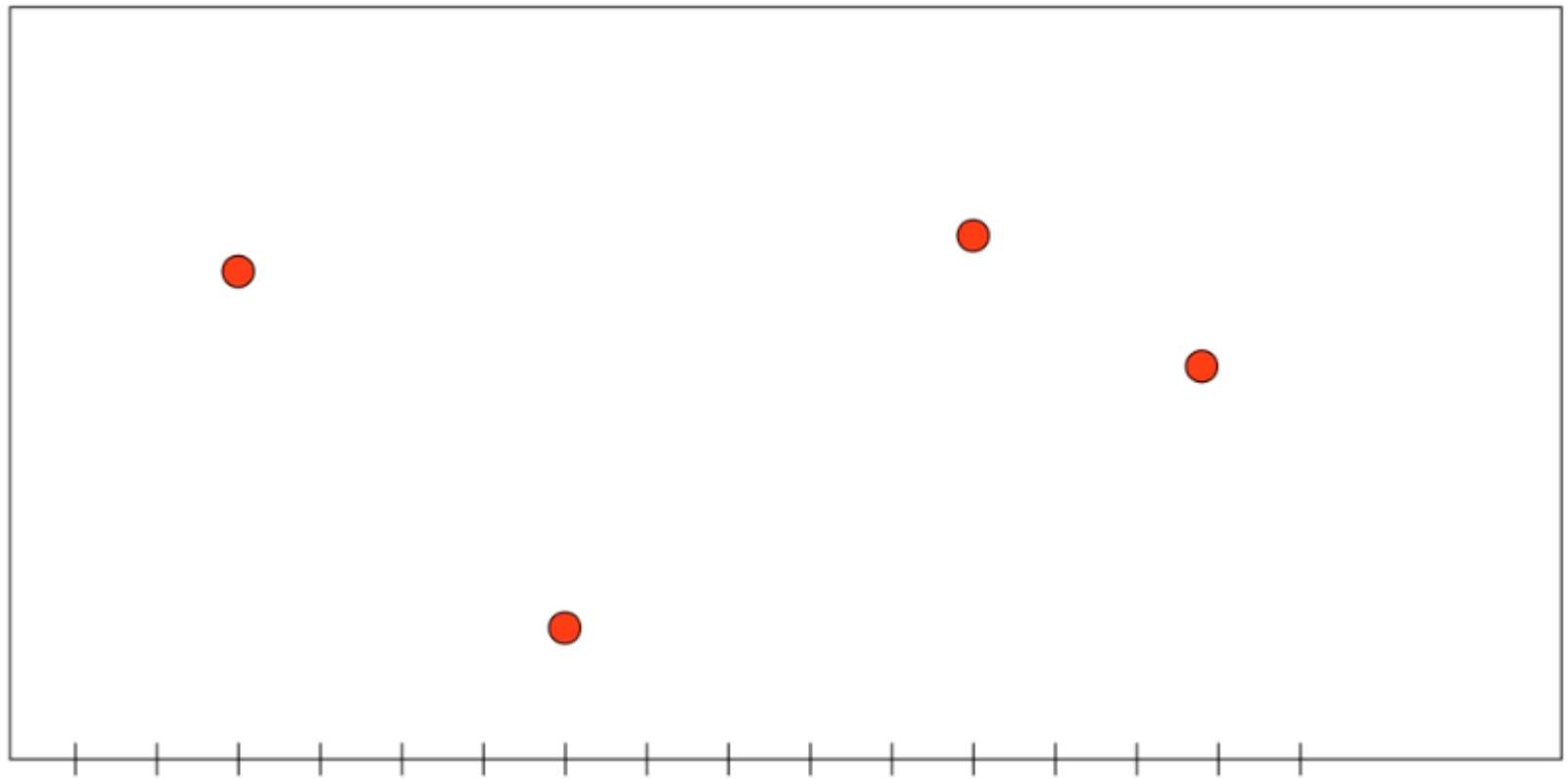
60 sec animation

- Video: 30 frames / sec
  = 1800 frames
- We have 6 key frames
- Keyframe system must generate 1794 frames

What's the inbetween motion?

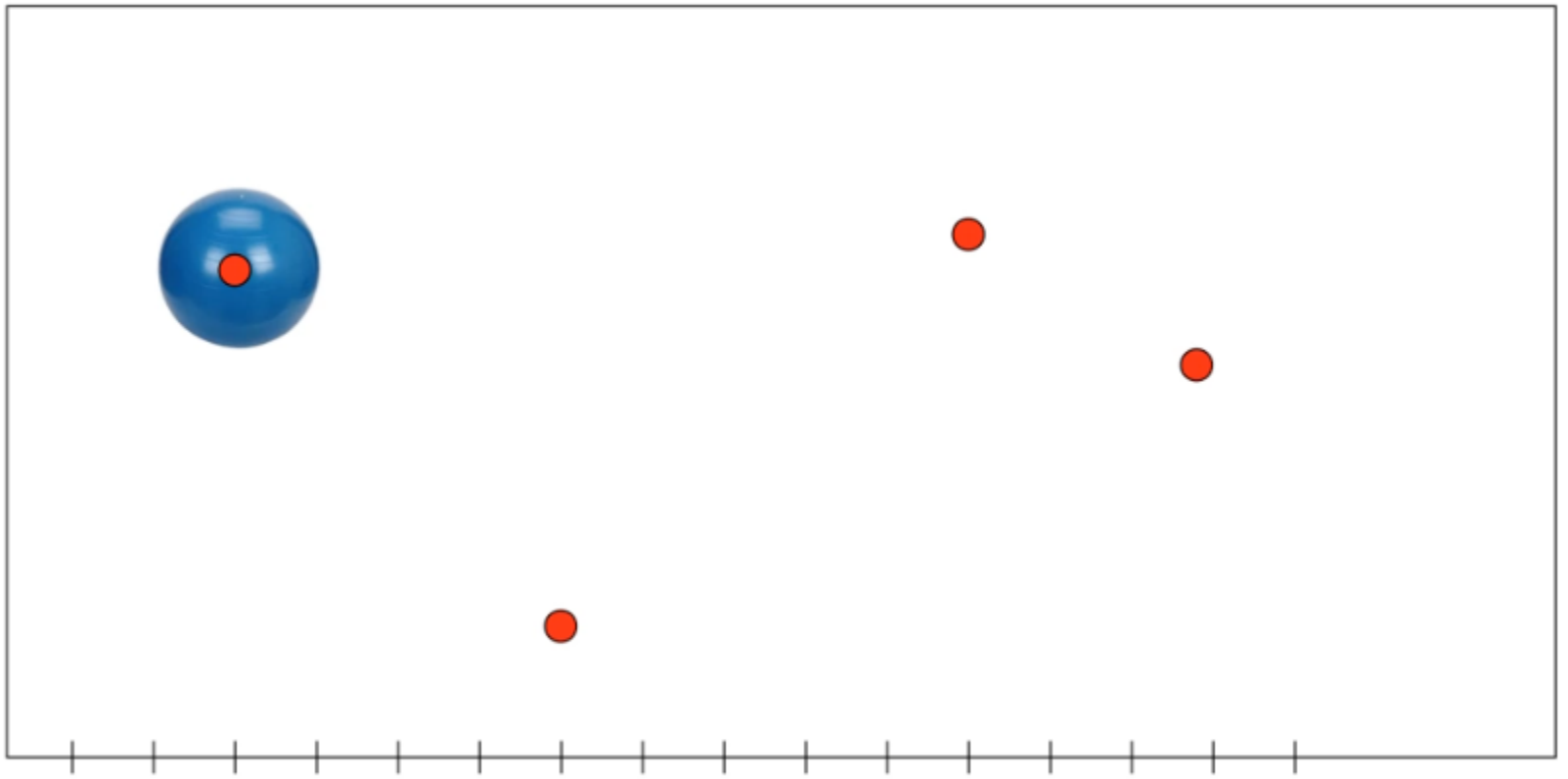


Mathematical problem: Given a set of points,
what are the most reasonable points *in between*?



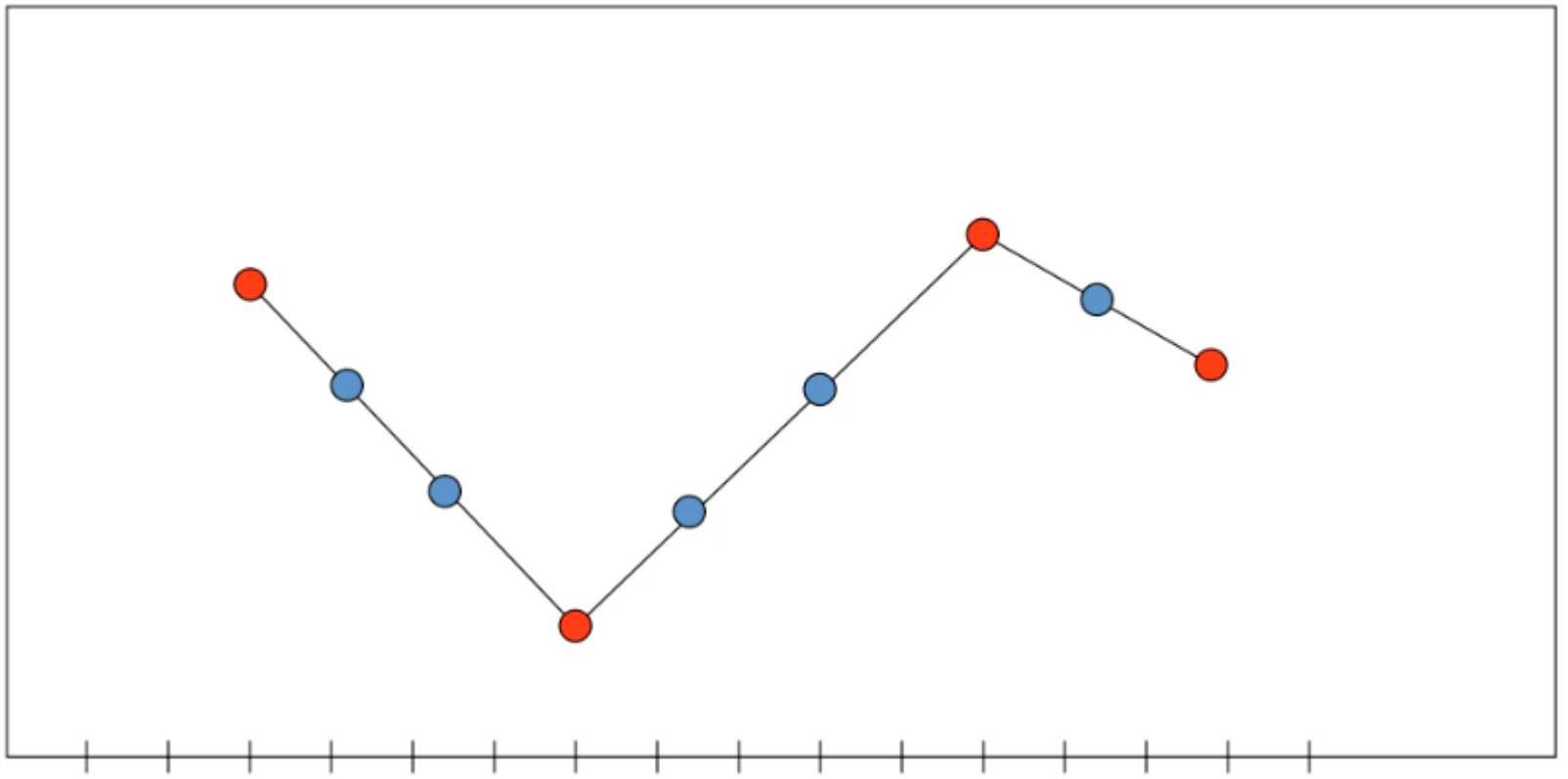
Mathematical problem: Given a set of points,
what are the most reasonable points *in between*?

Interpolation

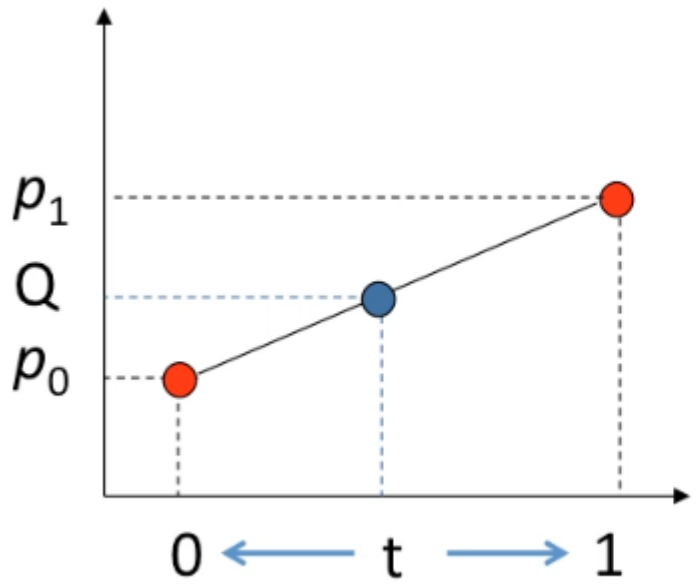


Mathematical problem: Given a set of points,
what are the most reasonable points *in between*?

Linear Interpolation



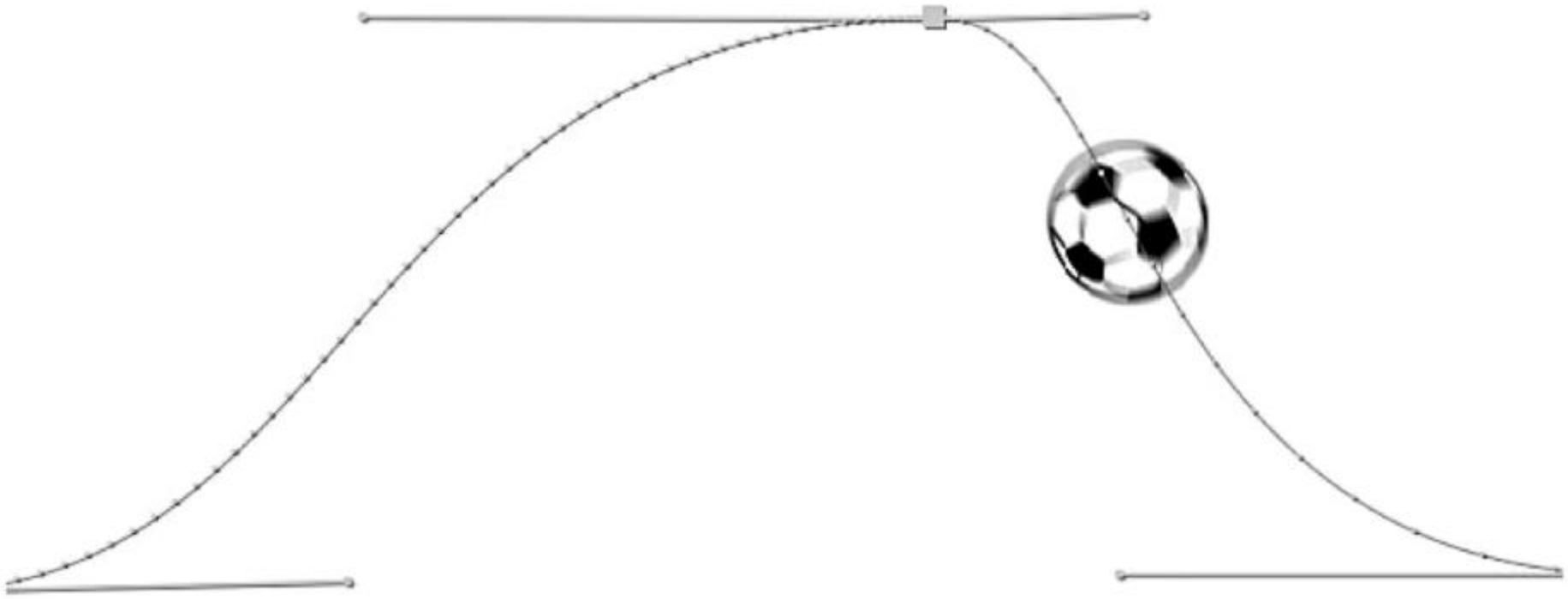
Linear Interpolation



$$Q(t) = (1-t)p_0 + tp_1$$

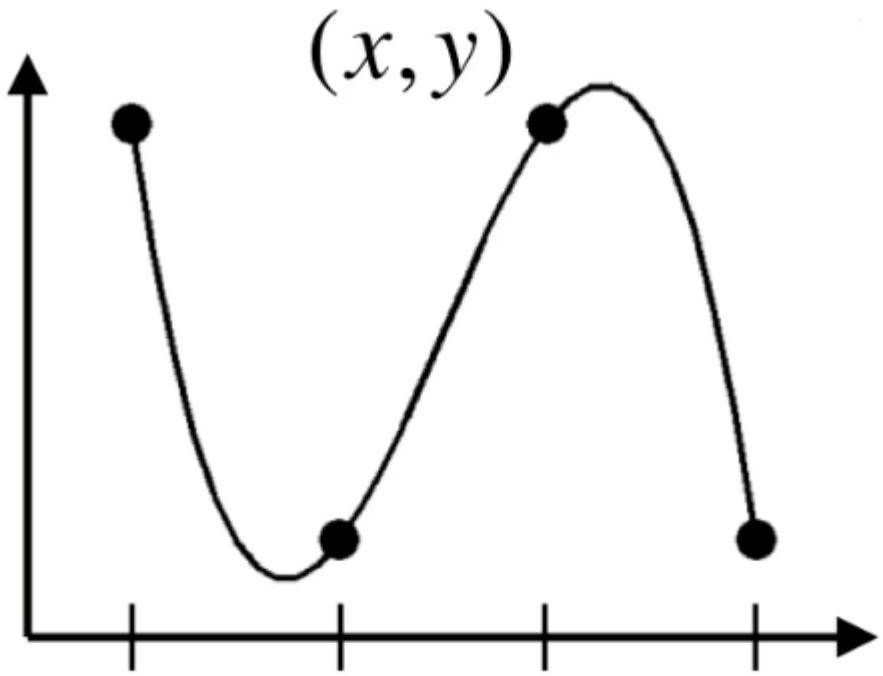
Linear Interpolation: Limitations

- May need a large numer of keyframes if motion is non-linear



Parametric Curves

- Define a continuous smooth curve f passing through the data points
- Explicit form y = f(x)
- Implicit form f(x, y) = 0
- Parametric form x = f(t), y = g(t)



Parametric Curve Example

- What curve does this represent?

$$x = \cos(t)$$

$$y = \sin(t)$$

Cubic Curves

- We can use a cubic function to represent a smooth curve in 3D

$$Q_x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \qquad 0 \le t \le 1$$

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

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

Cubic Curves

- We can use a cubic function to represent a smooth curve in 3D

$$Q_x(t) = a_x t^3 + b_x t^2 + c_x t + d_x \qquad 0 \le t \le 1$$

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

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

- Vector Form:

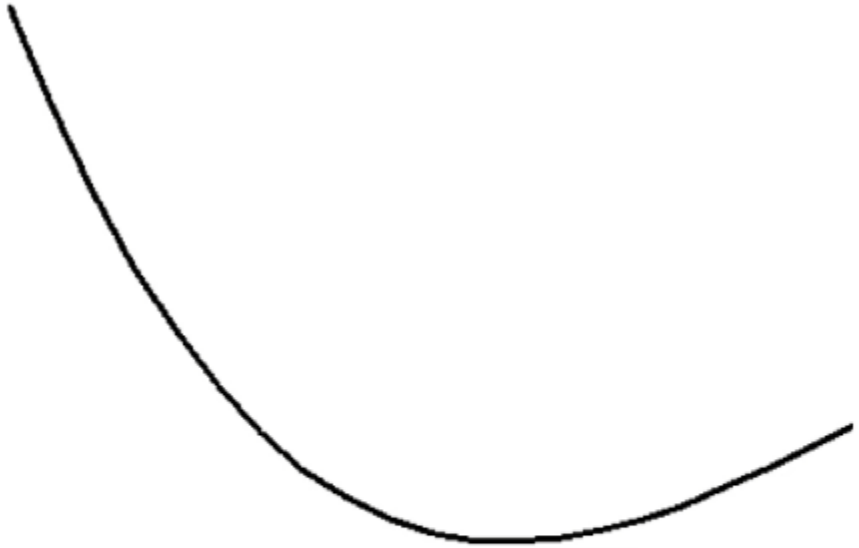
$$\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}$$

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

Smooth Curves

- Controlling the shape of the curve

$$Q_x(t) = 1 - t + t^2 - t^3$$



$$Q_x(t) = 1 - t + 3t^2 - t^3$$

Constraints on the Cubics

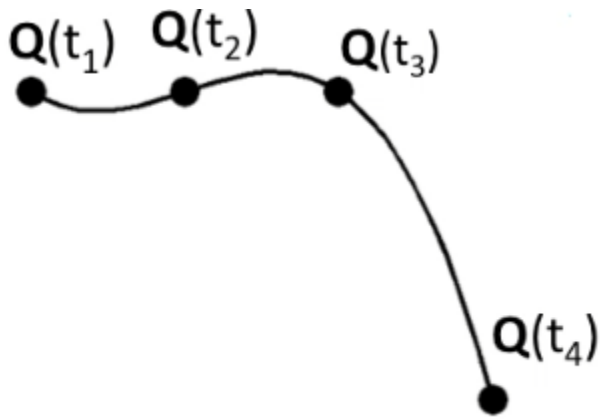
- How many constraints do we need to determine a cubic curve?

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

Constraints on the Cubics

- How many constraints do we need to determine a cubic curve?

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

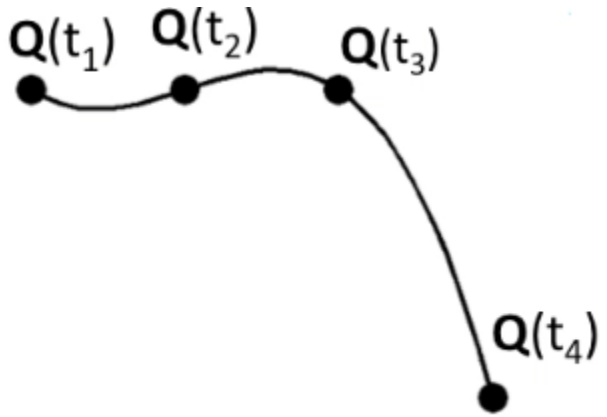


Constraints on the Cubics

- How many constraints do we need to determine a cubic curve?

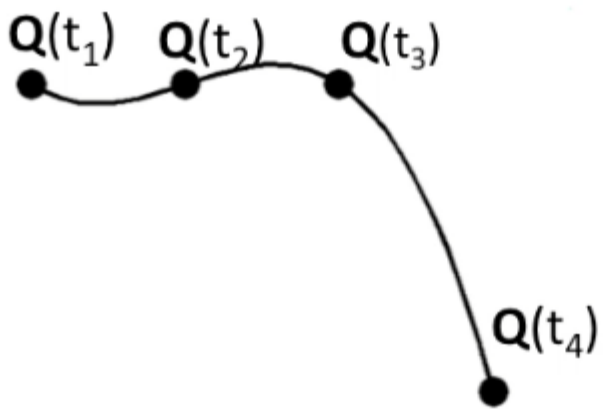
$$\mathbf{Q}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

$$\begin{bmatrix} \mathbf{Q}(t_1) \\ \mathbf{Q}(t_2) \\ \mathbf{Q}(t_3) \\ \mathbf{Q}(t_4) \end{bmatrix} = \begin{bmatrix} t_1^3 & t_1^2 & t_1 & 1 \\ t_2^3 & t_2^2 & t_2 & 1 \\ t_3^3 & t_3^2 & t_3 & 1 \\ t_4^3 & t_4^2 & t_4 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$



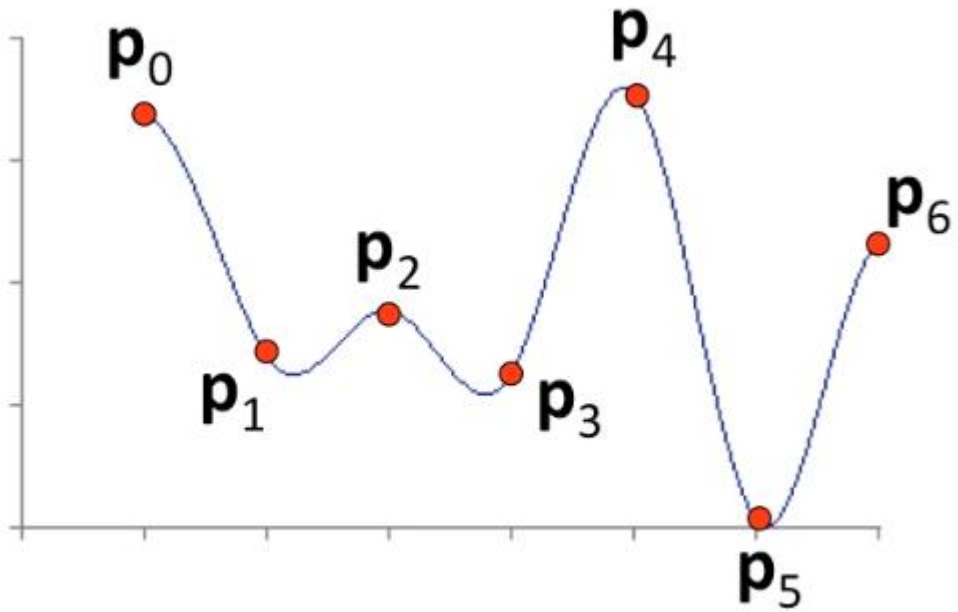
Natural Cubic Curves

$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} t_1^3 & t_1^2 & t_1 & 1 \\ t_2^3 & t_2^2 & t_2 & 1 \\ t_3^3 & t_3^2 & t_3 & 1 \\ t_4^3 & t_4^2 & t_4 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{Q}(t_1) \\ \mathbf{Q}(t_2) \\ \mathbf{Q}(t_3) \\ \mathbf{Q}(t_4) \end{bmatrix}$$

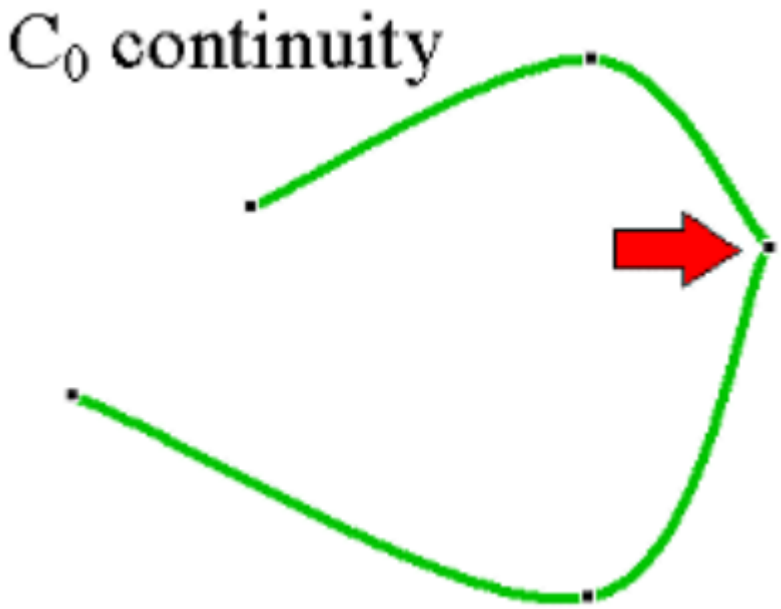


Natural Cubic Spline

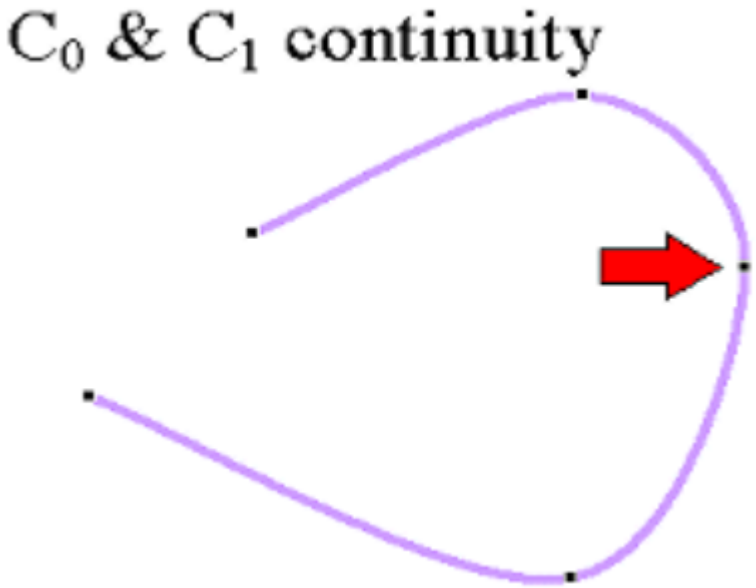
- A **spline** is a curve that is piecewise-defined and is smooth at the places where the pieces connect



Continuity



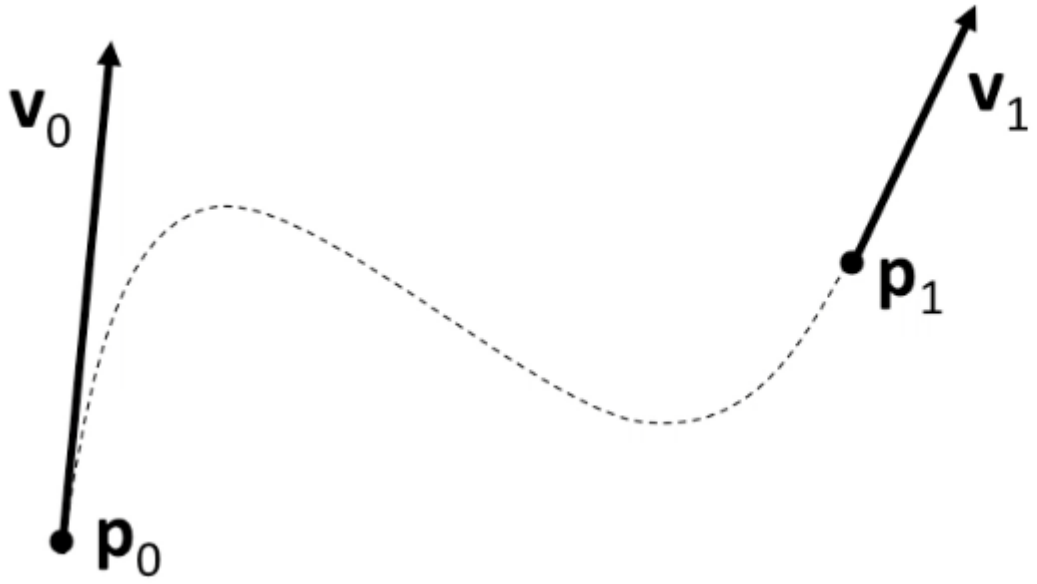
Positions of splines align



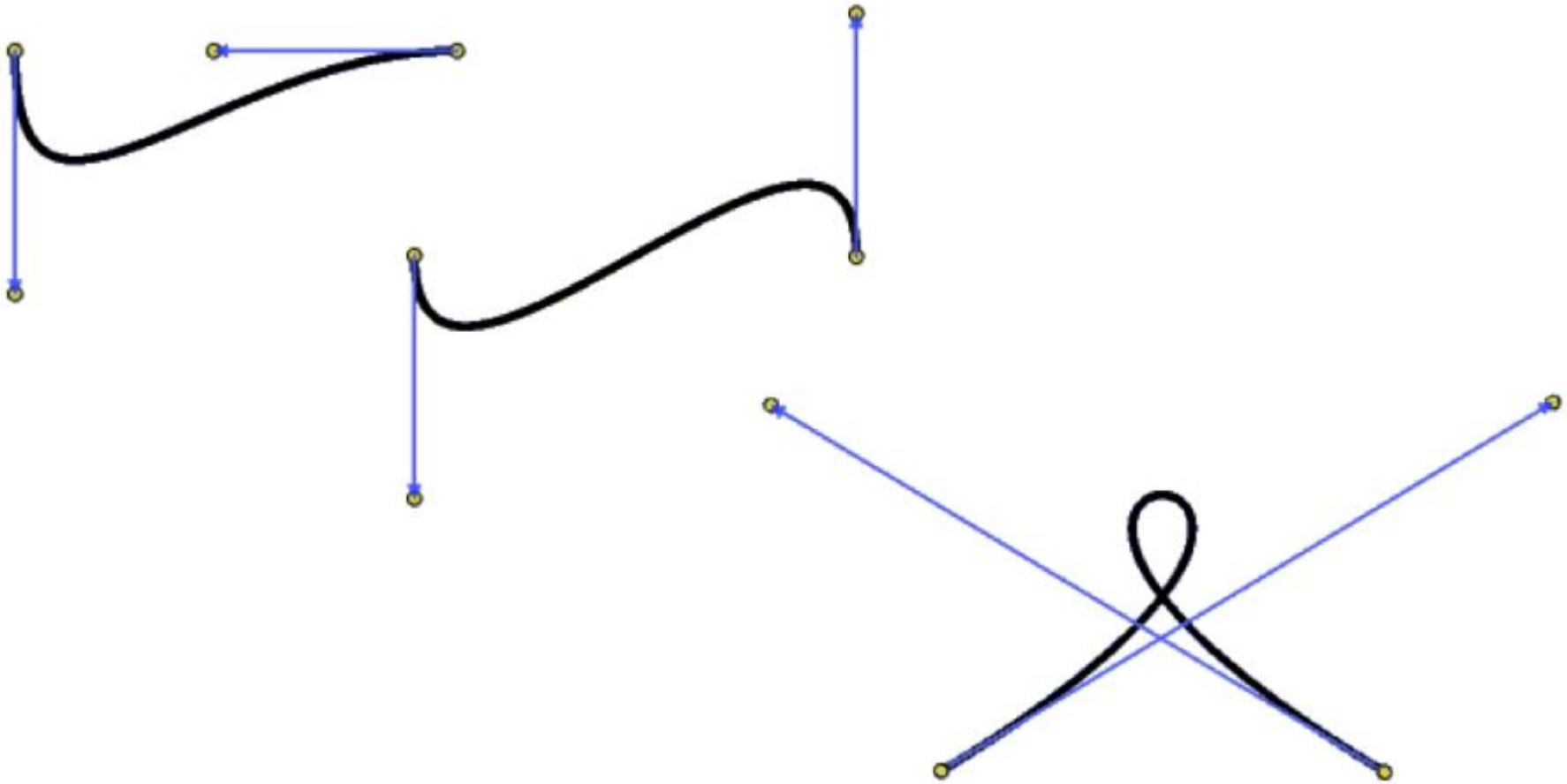
Positions and tangents of splines align

Hermite Curves

- A Hermite curve is a cubic curve determined by
  - Endpoints $\mathbf{p_0}$ and $\mathbf{p_1}$
  - Tangent vectors (velocities) $\mathbf{v_0}$ and $\mathbf{v_1}$ at endpoints



Example of Hermite Curves



Tangents (Derivatives)

$$\mathbf{Q} = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

Tangents (Derivatives)

$$\mathbf{Q} = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d} \qquad \frac{d\mathbf{Q}}{dt} = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}$$

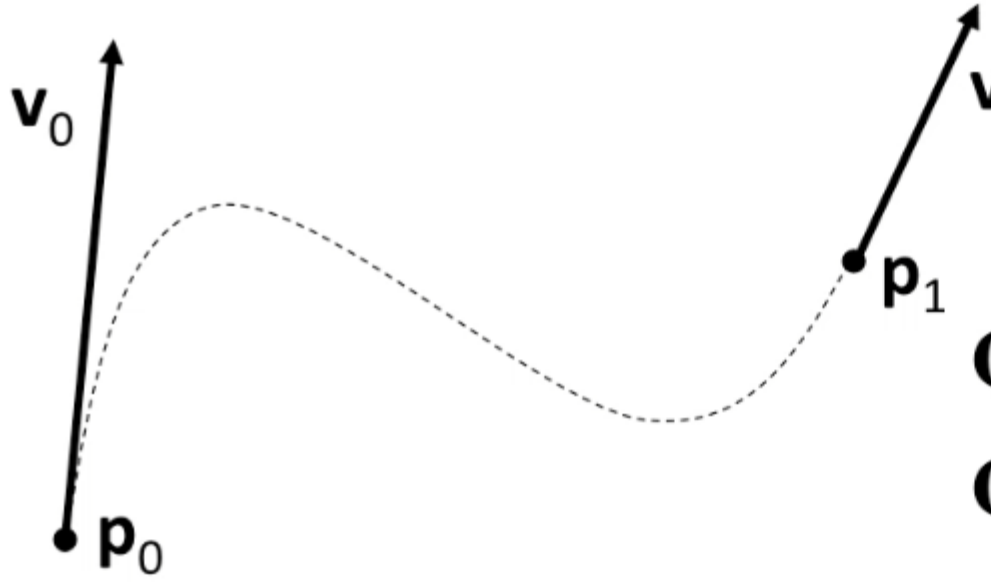
Tangents (Derivatives)

$$\mathbf{Q} = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d} \qquad \frac{d\mathbf{Q}}{dt} = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}$$

$$\mathbf{Q} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} \qquad \frac{d\mathbf{Q}}{dt} = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

Hermite Curves

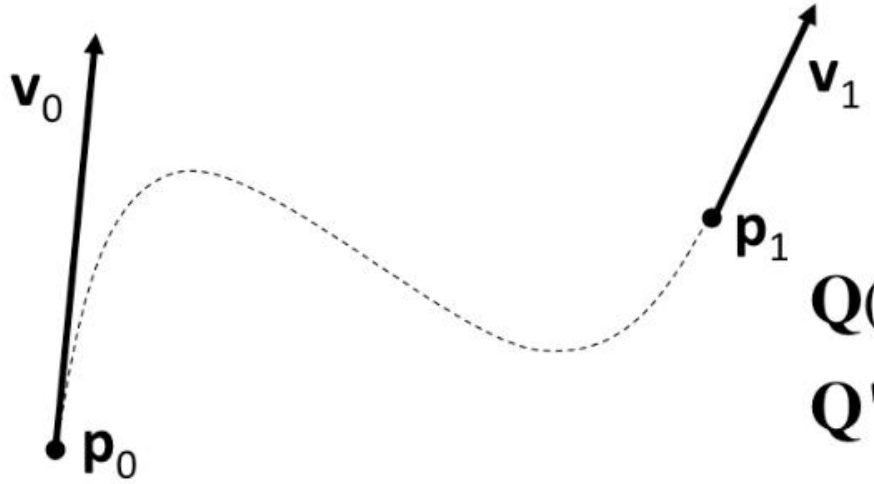
- The value of the curve is $\mathbf{Q}(0)=p_0$ at t=0 and $\mathbf{Q}(1)=p_1$ at t=1
- The derivative of the curve to be v_0 at t=0 and v_1 at t=1



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

$$\mathbf{Q}'(t) = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}$$

Hermite Curves



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

$$\mathbf{Q}'(t) = 3\mathbf{a}t^2 + 2\mathbf{b}t + \mathbf{c}$$

$$\begin{aligned}
\mathbf{Q}(0) &= \mathbf{p}_0 &&= \mathbf{a}\cdot 0^3 + \mathbf{b}\cdot 0^2 + \mathbf{c}\cdot 0 + \mathbf{d} &&= \mathbf{d} \\
\mathbf{Q}(1) &= \mathbf{p}_1 &&= \mathbf{a}\cdot 1^3 + \mathbf{b}\cdot 1^2 + \mathbf{c}\cdot 1 + \mathbf{d} &&= \mathbf{a}+\mathbf{b}+\mathbf{c}+\mathbf{d} \\
\mathbf{Q}'(0) &= \mathbf{v}_0 &&= 3\mathbf{a}\cdot 0^2 + 2\mathbf{b}\cdot 0 + \mathbf{c} &&= \mathbf{c} \\
\mathbf{Q}'(1) &= \mathbf{v}_1 &&= 3\mathbf{a}\cdot 1^2 + 2\mathbf{b}\cdot 1 + \mathbf{c} &&= 3\mathbf{a}+2\mathbf{b}+\mathbf{c}
\end{aligned}$$

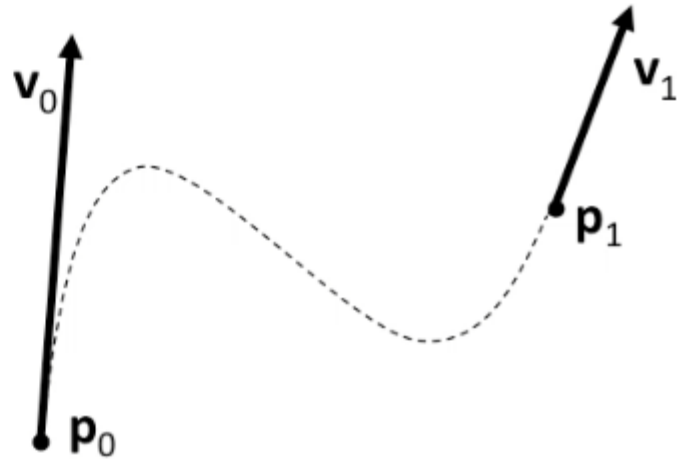
Hermite Curves

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

$$\mathbf{p}_1 = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$

$$\mathbf{v}_0 = \mathbf{c}$$

$$\mathbf{v}_1 = 3\mathbf{a} + 2\mathbf{b} + \mathbf{c}$$



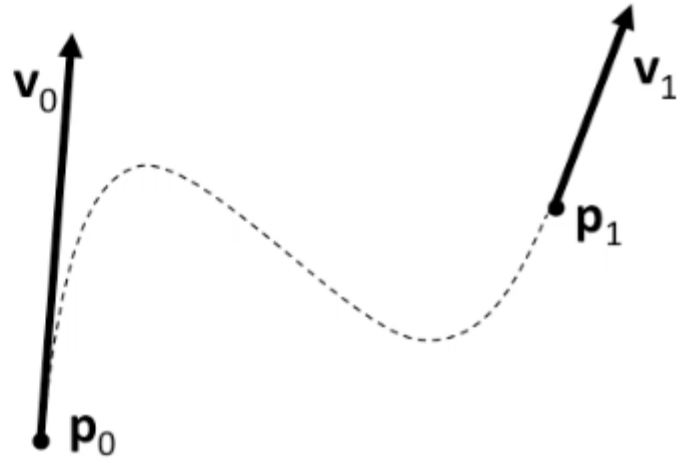
Hermite Curves

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

$$\mathbf{p}_1 = \mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d}$$

$$\mathbf{v}_0 = \mathbf{c}$$

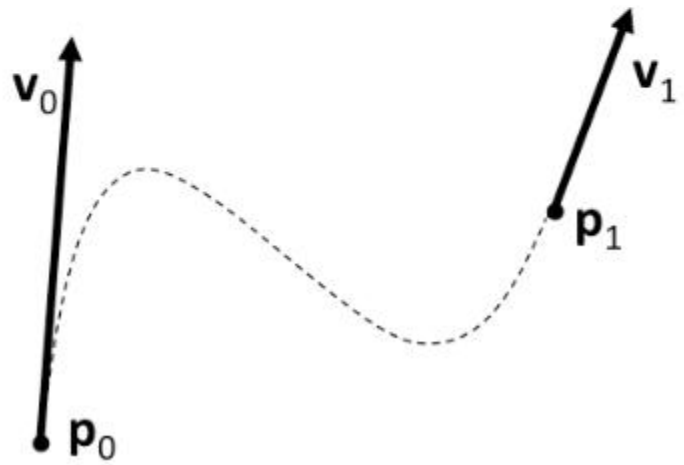
$$\mathbf{v}_1 = 3\mathbf{a} + 2\mathbf{b} + \mathbf{c}$$



$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

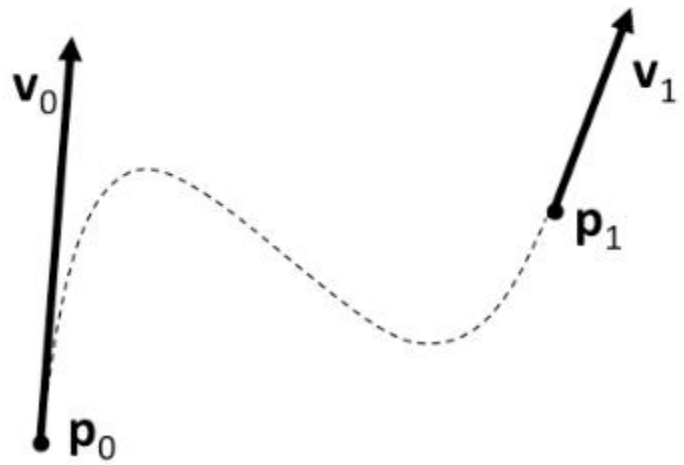
Hermite Interpolation

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



Hermite Interpolation

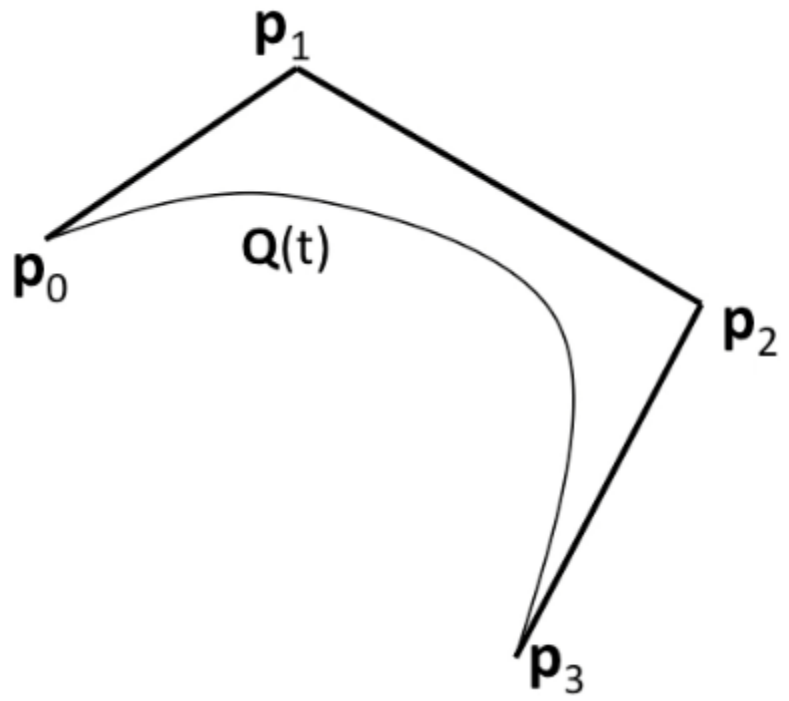
$$\mathbf{Q} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix}$$



$$\begin{bmatrix} p_{0x} & p_{0y} & p_{0z} \\ p_{1x} & p_{1y} & p_{1z} \\ v_{0x} & v_{0y} & v_{0z} \\ v_{1x} & v_{1y} & v_{1z} \end{bmatrix}$$

Bezier Curves

- Variations of Hermite curves
  - Indirectly specify tangent vectors, by specifying two intermediate points



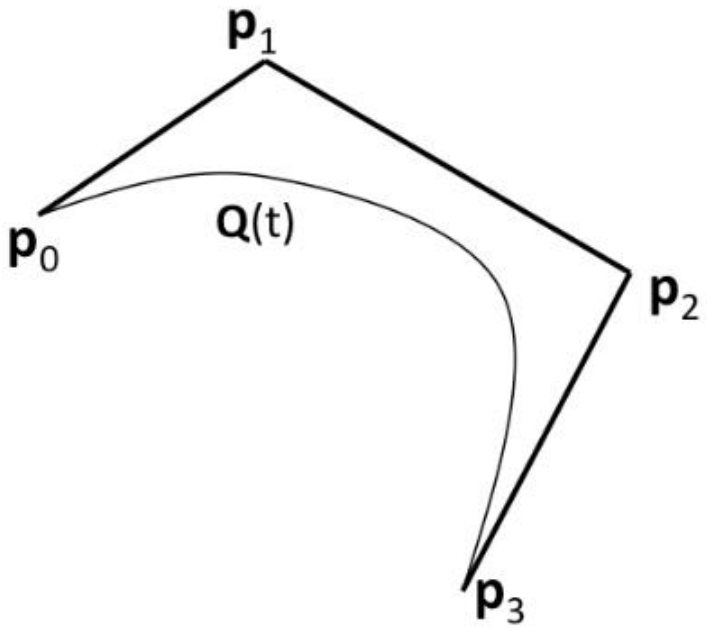
Bezier Curve Formulation

- There exist a number of ways to formulate Bezier curves
  - De Casteljau (recursive linear interpolations)
  - Bernstein polynomials
  - Cubic equations
  - Matrix form → most useful for CG purposes

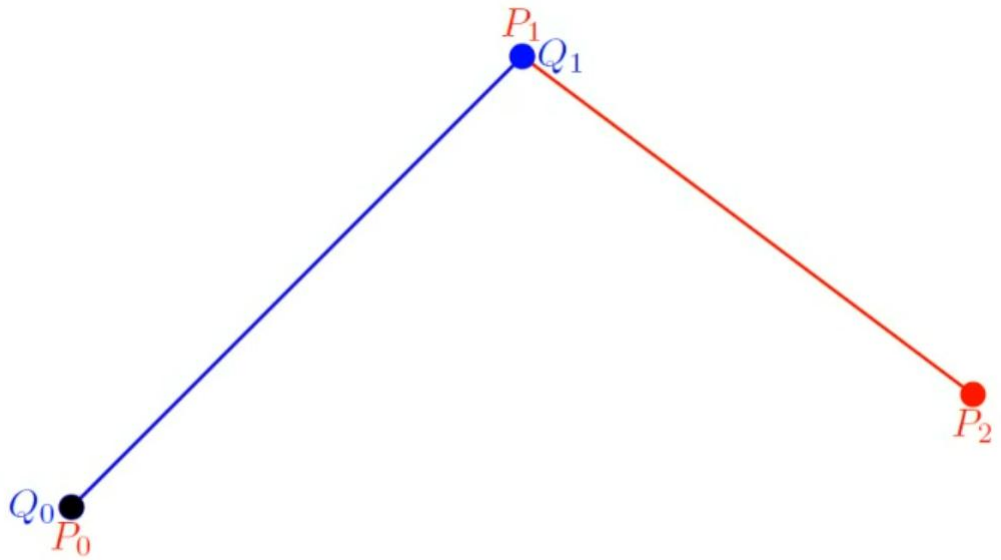
Bezier Curves

- Find the curve Q(t) as a function of parameter t
  - Endpoints $\mathbf{p_0}$ and $\mathbf{p_3}$
  - Tangents aligned with $\mathbf{p_0p_1}$ and $\mathbf{p_3p_2}$

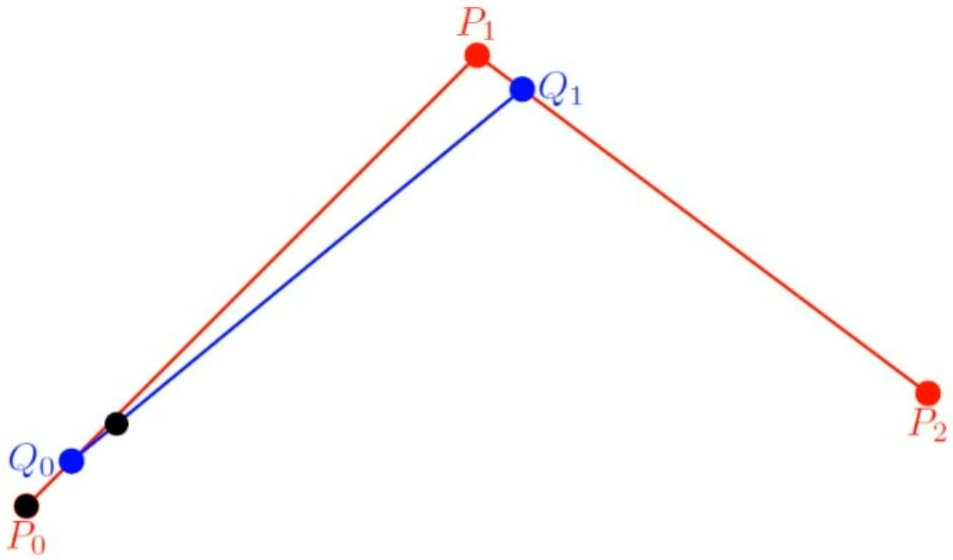
$$\mathbf{v}0=\alpha(\mathbf{p}1-\mathbf{p}0)$$
$$\mathbf{v}1=\alpha(\mathbf{p}3-\mathbf{p}2)$$



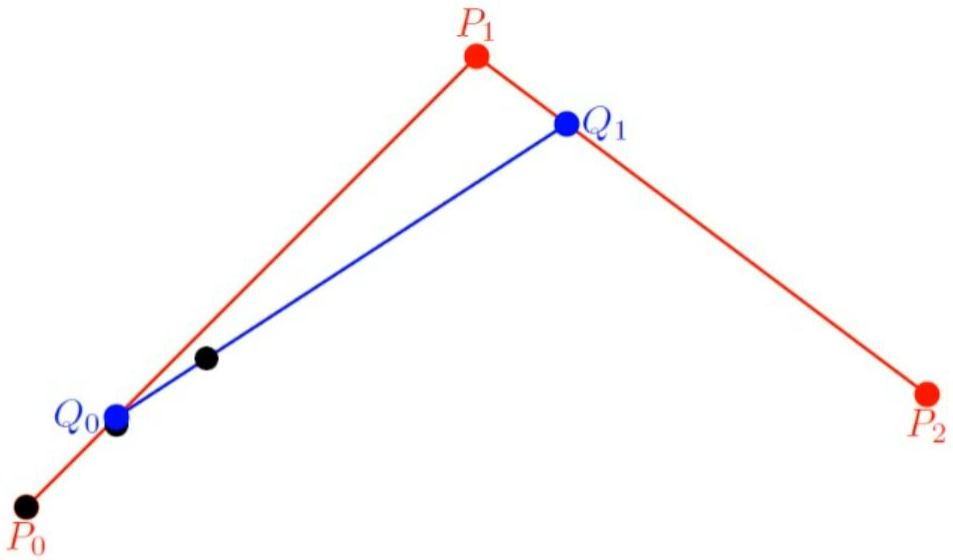
De Casteljau



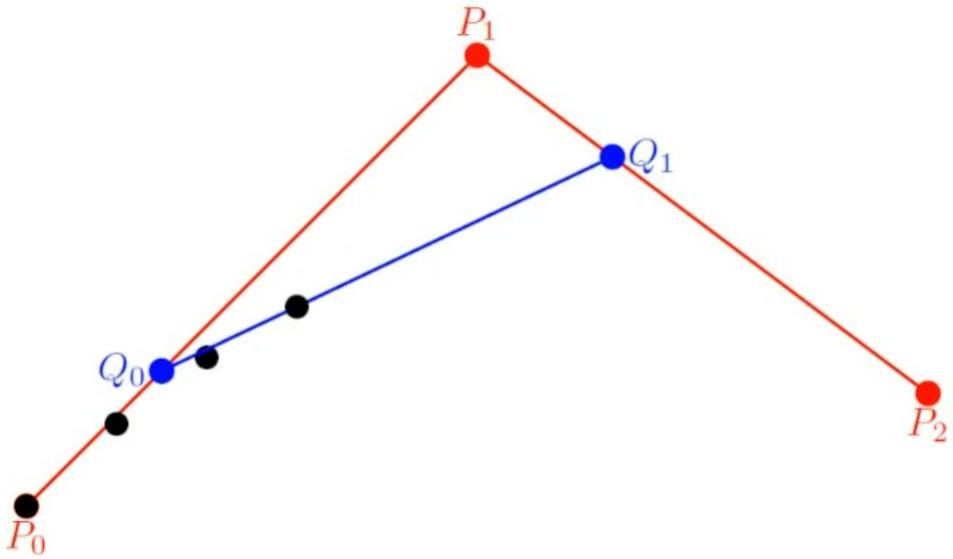
De Casteljau



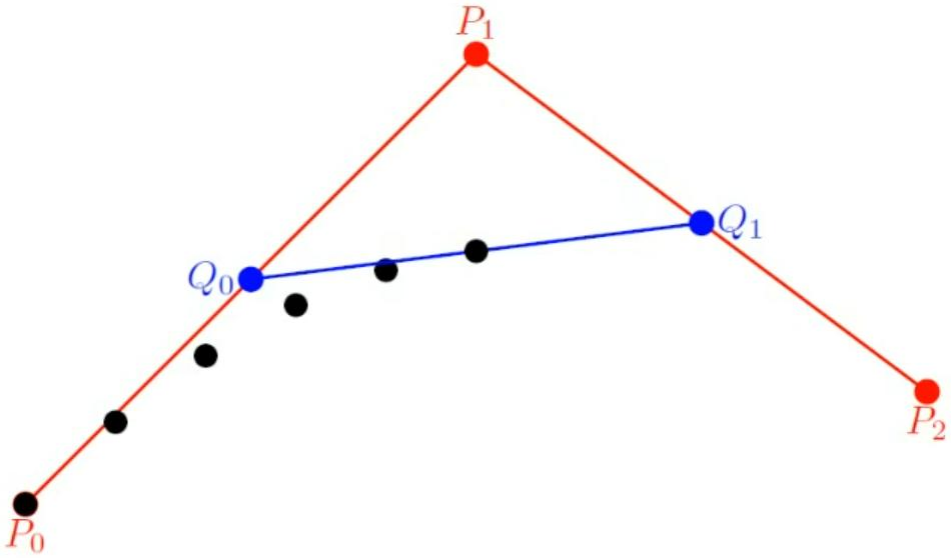
De Casteljau



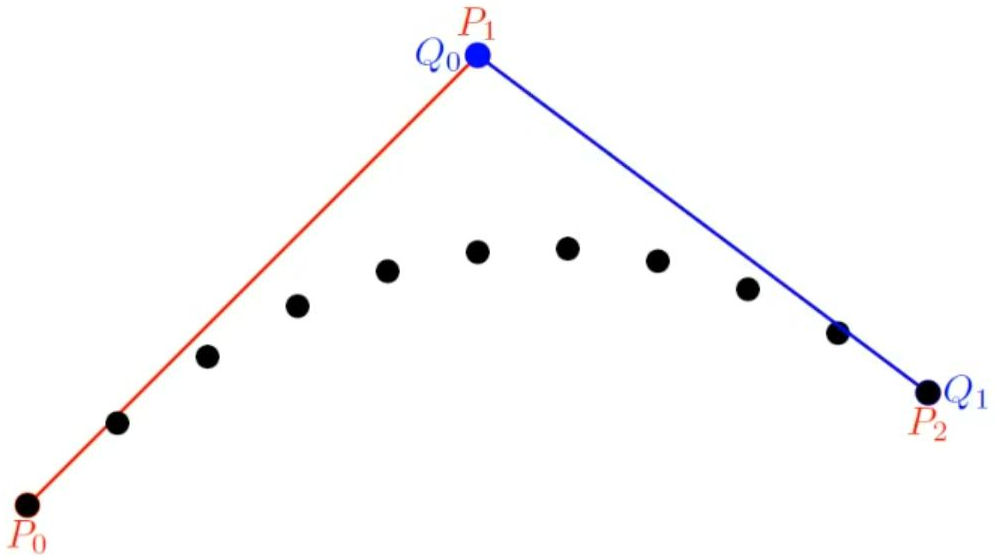
De Casteljau



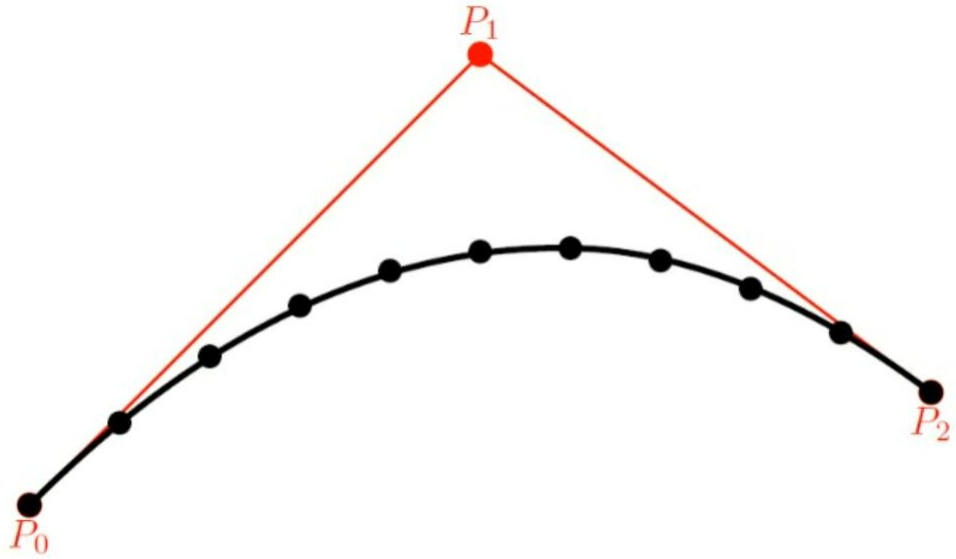
De Casteljau



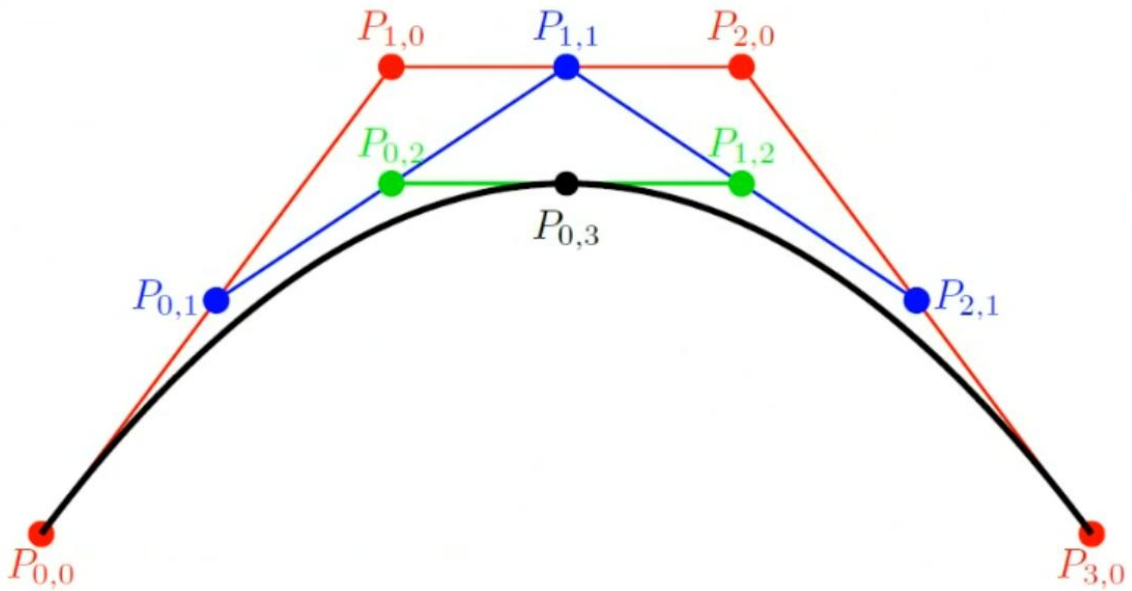
De Casteljau



De Casteljau



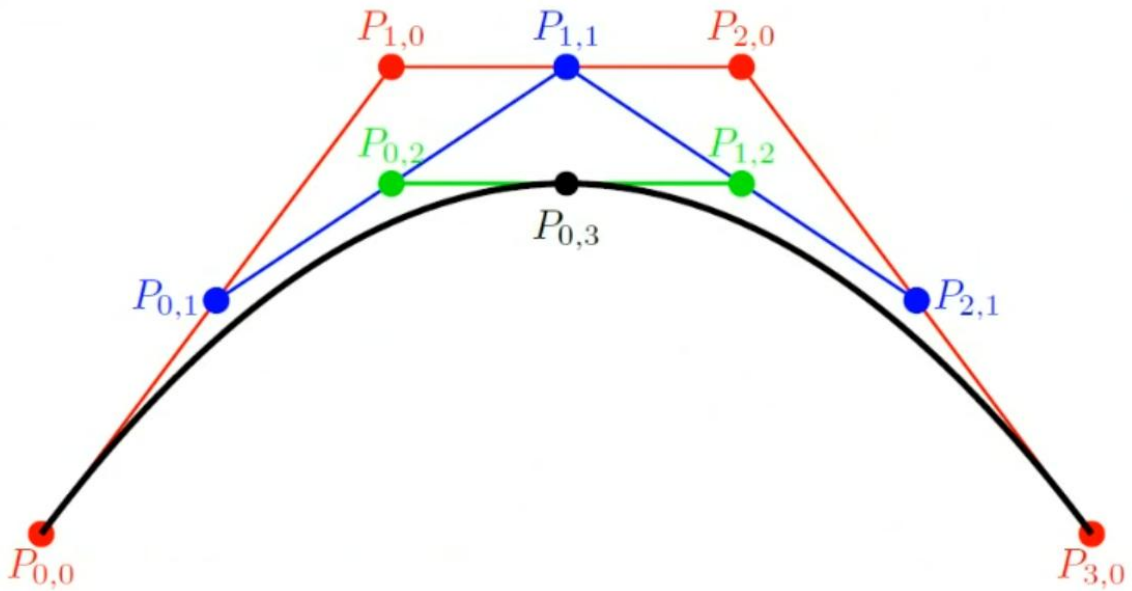
De Casteljau



Bezier – Cubic Form

- For four control points $P_{0,0}$, $P_{1,0}$, $P_{2,0}$ and $P_{3,0}$:

$$P_{0,3}(t) = (1-t)^3 P_{0,0} + 3t(1-t)^2 P_{1,0} + 3t^2(1-t)P_{2,0} + t^3 P_{3,0}$$



Bezier – Matrix Form

$$\mathbf{Q} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Hermite vs. Bezier Curves

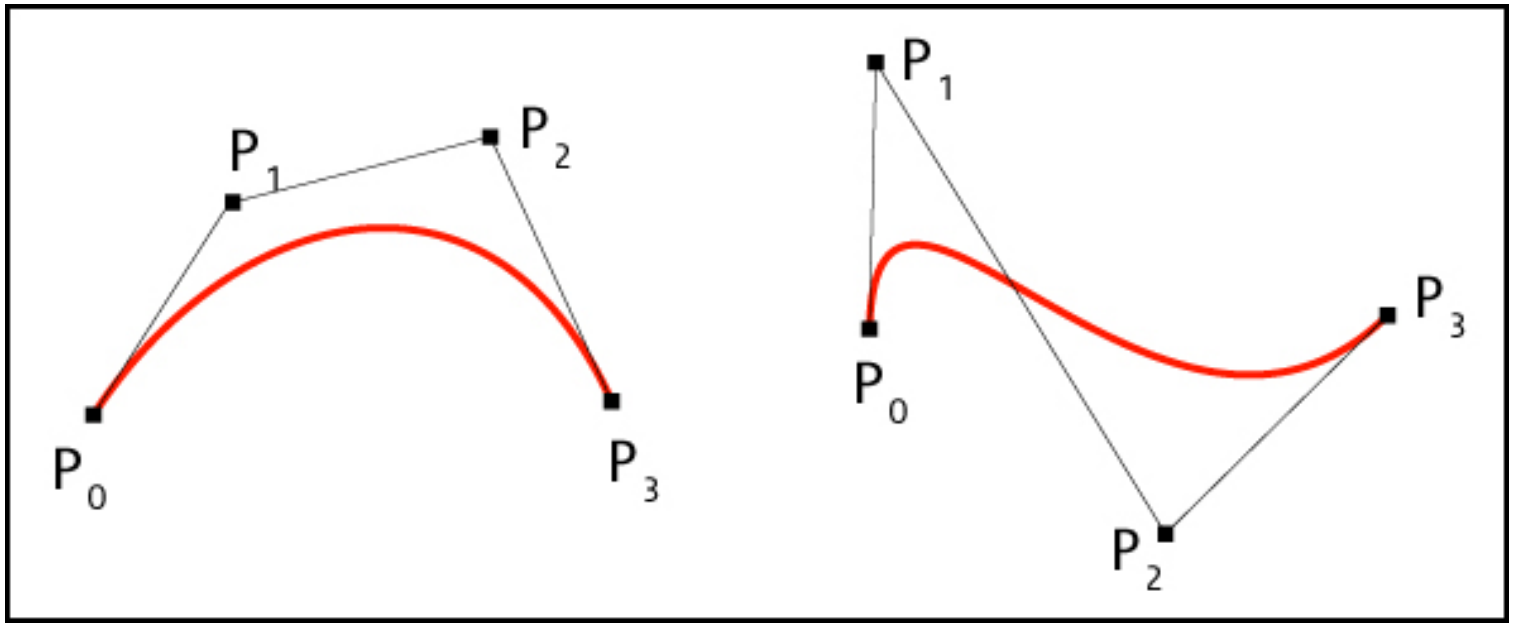
- Hermite curves:

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

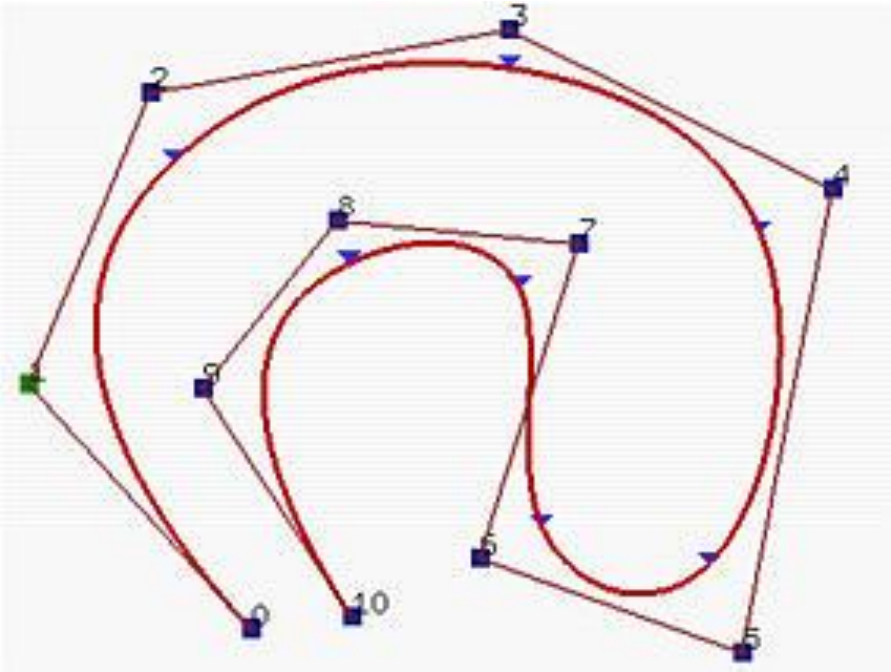
- Bezier curves:

$$\mathbf{Q} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

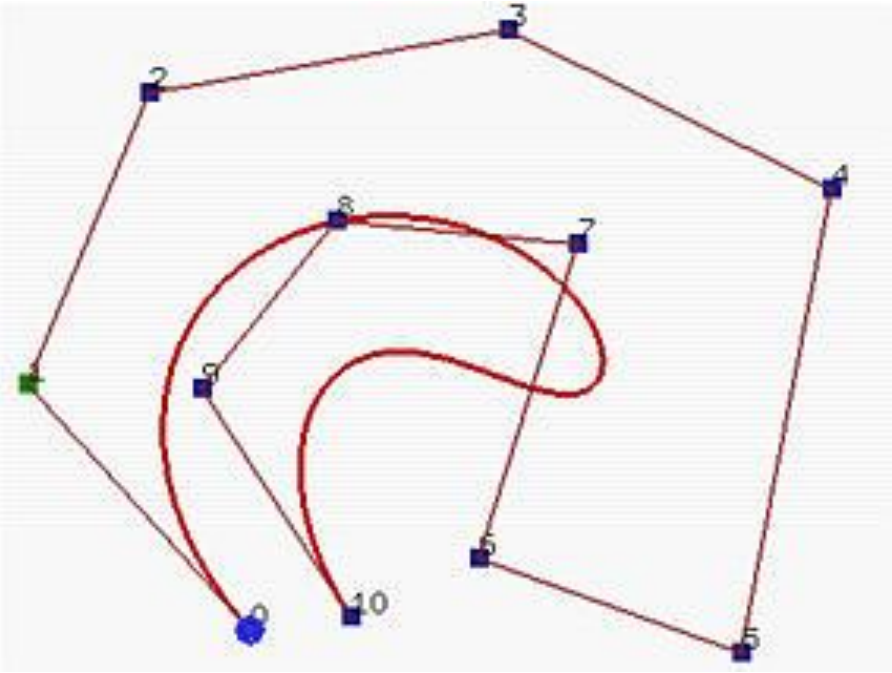
Bezier Curve Examples



B-splines

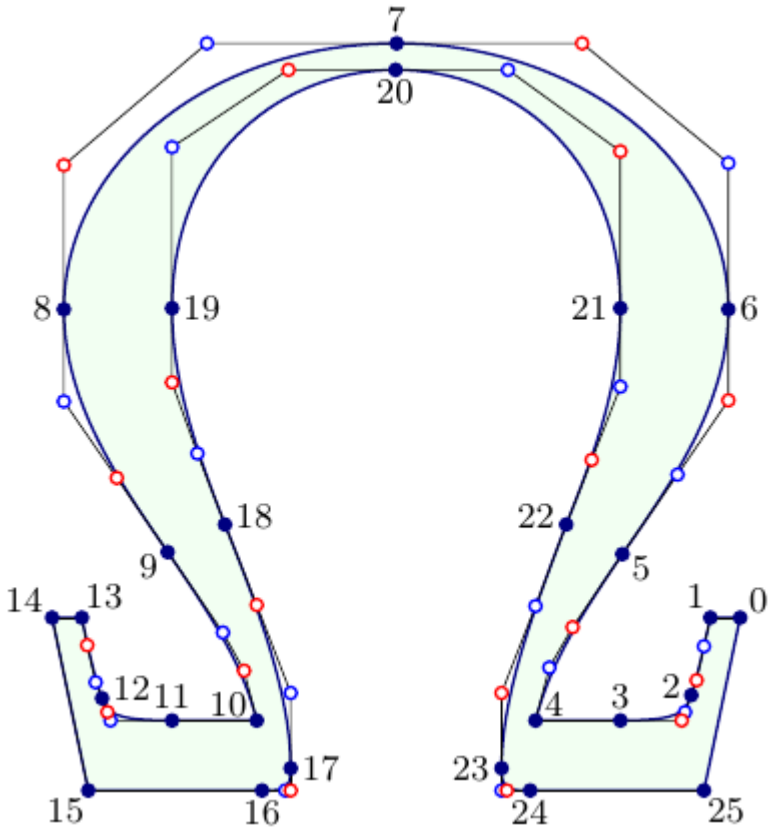


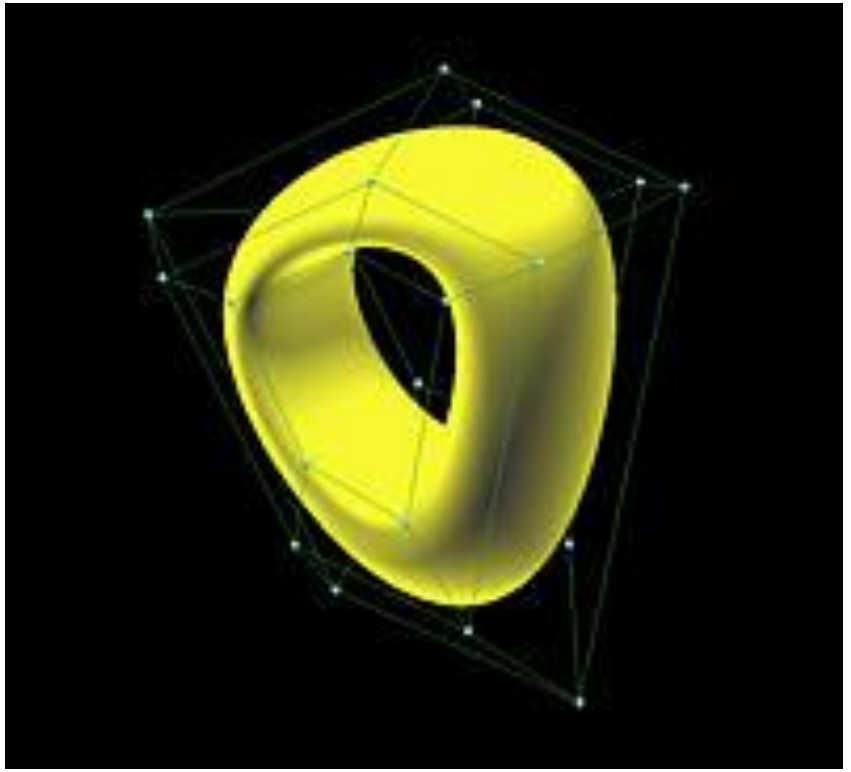
B-spline



Bézier 10th degree

B-spline curves and surfaces



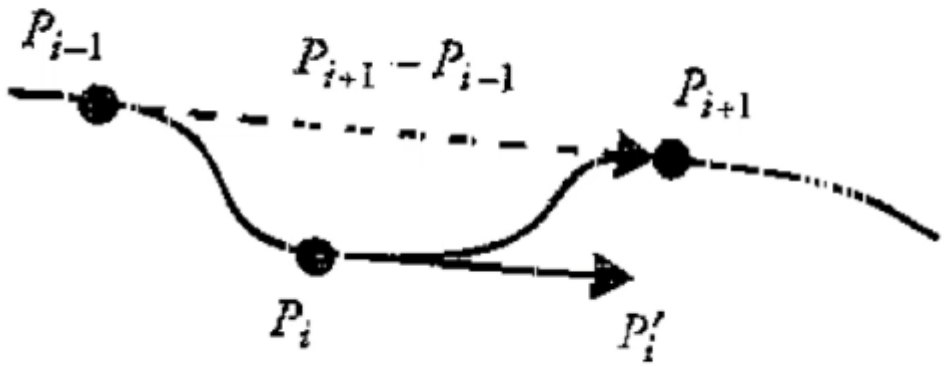


Problems with Splines?

- Require explicit definition of derivatives at each point.

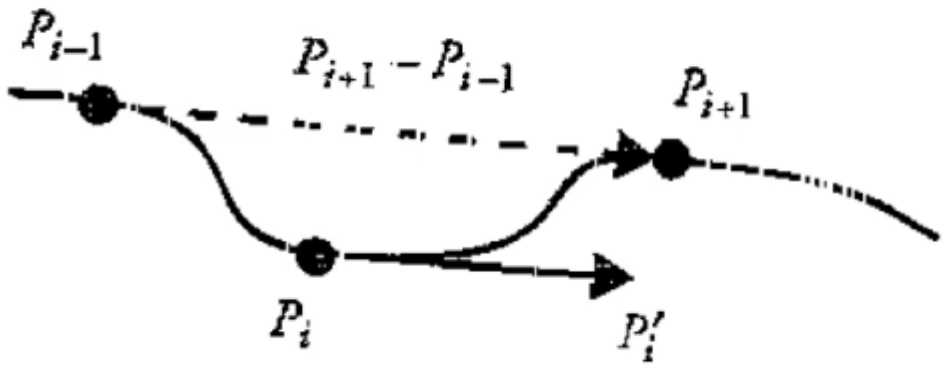
Catmull-Rom spline

- Tangent at point P_i is half of the vector connecting P_{i-1} to P_{i+1}



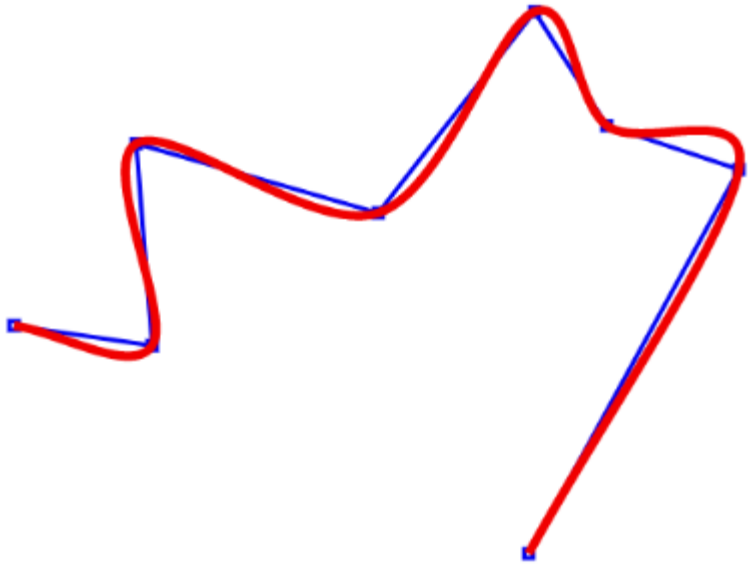
Catmull-Rom spline

- Tangent at point P_i is half of the vector connecting P_{i-1} to P_{i+1}



$$\mathbf{p}'_i = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2}$$

Catmull-Rom Example



https://www.ibiblio.org/e-notes/Splines/cardinal.html

Catmull-Rom Spline

- Hermite curve

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

Catmull-Rom Spline

- Hermite curve

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

- For two control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$

$$\mathbf{Q} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2} \\ \frac{\mathbf{p}_{i+2} - \mathbf{p}_i}{2} \end{bmatrix}$$

Catmull-Rom Spline

$$\mathbf{Q} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2} \\ \frac{\mathbf{p}_{i+2} - \mathbf{p}_i}{2} \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix}$$

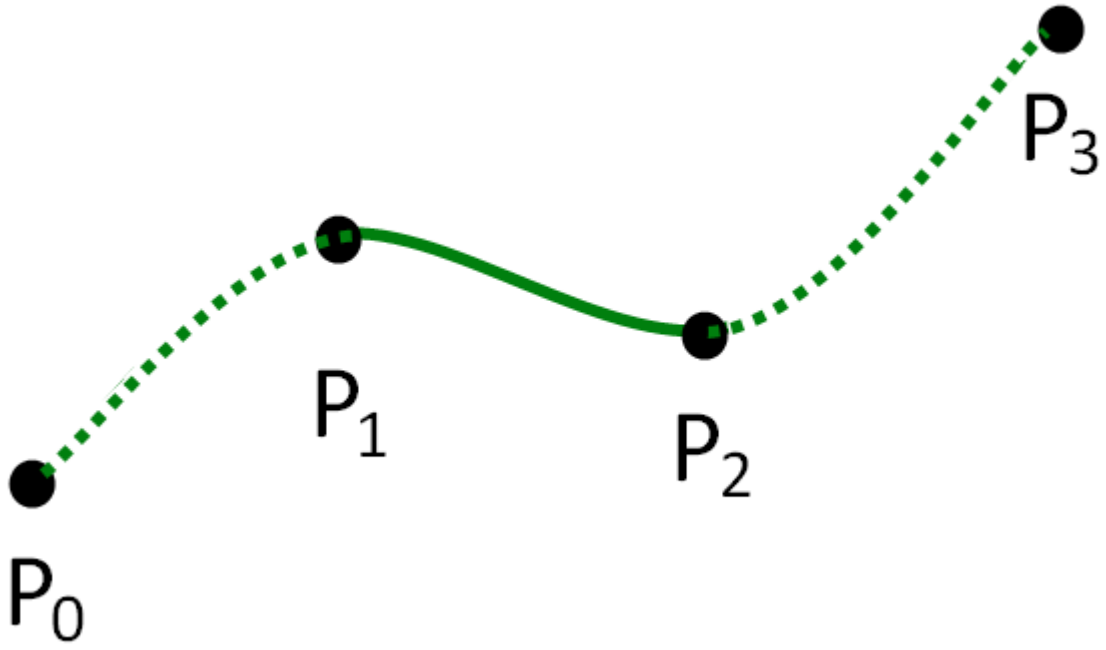
Catmull-Rom Spline

- Curve piece between control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$
- First and last curve segments are special cases

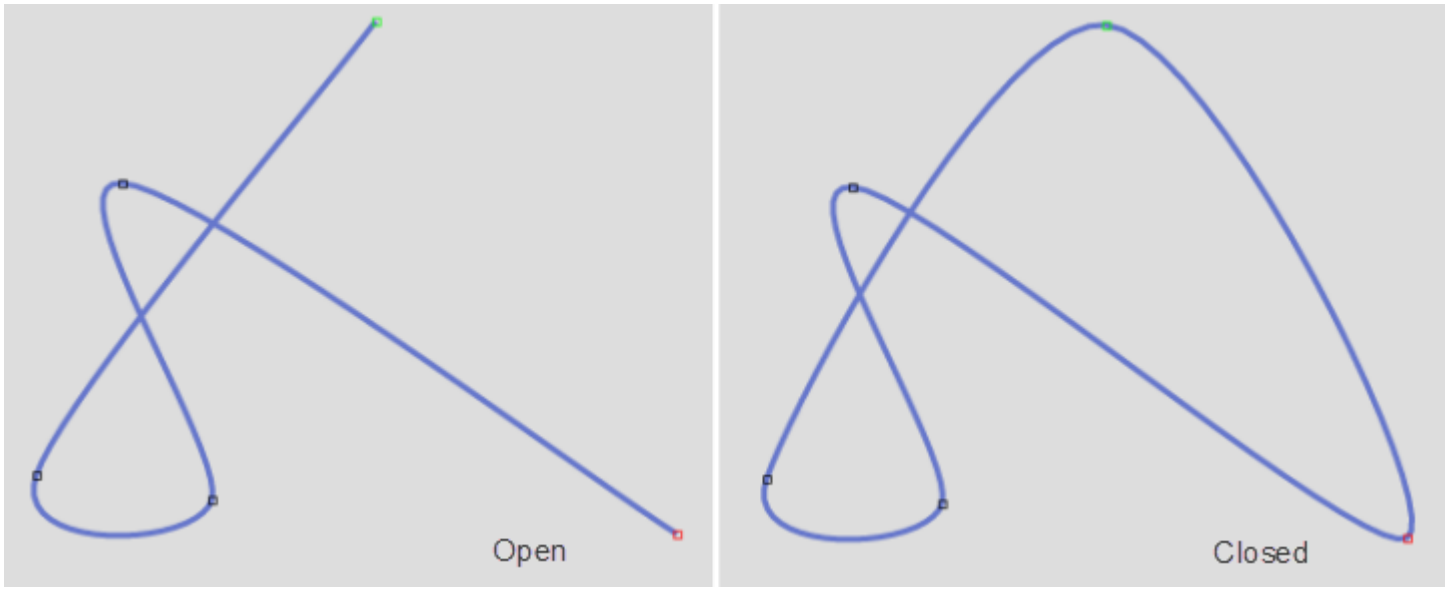
$$\mathbf{Q} = \frac{1}{2}\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix}$$

Catmull-Rom Spline

- Curve piece between control points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$
- First and last curve segments are special cases



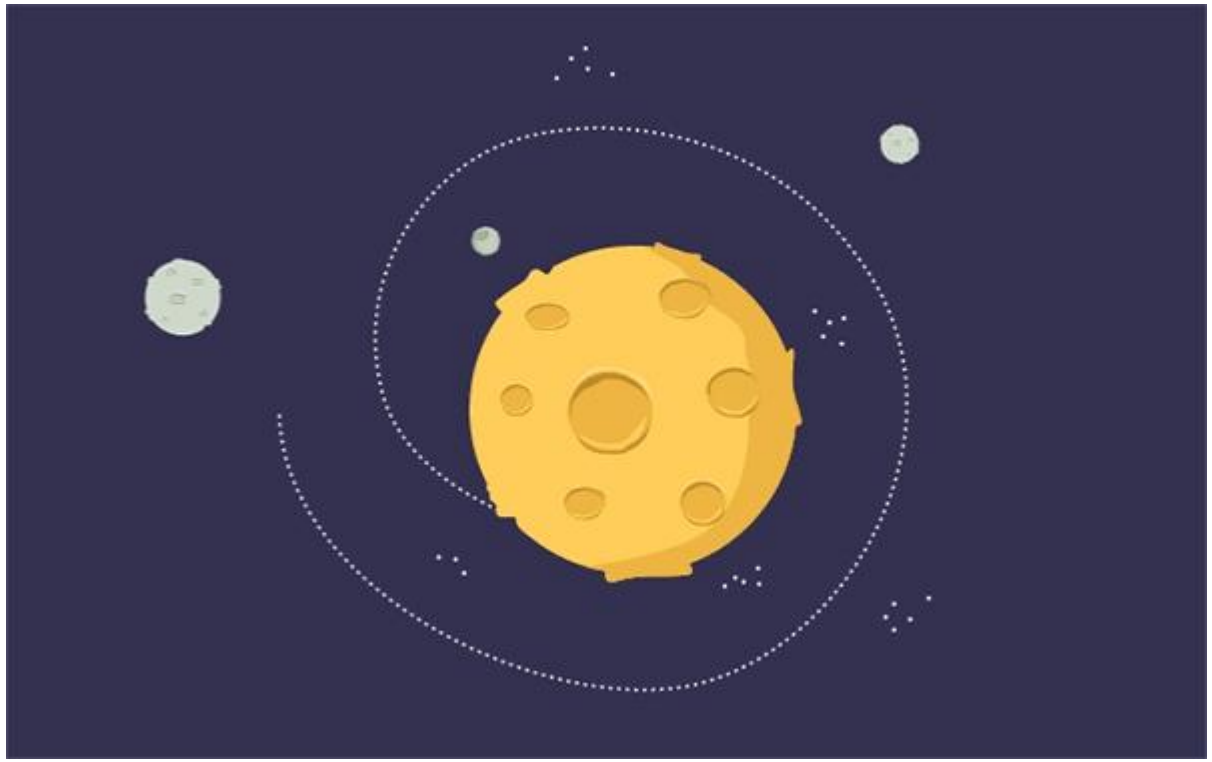
Closed-loop Catmull-Rom



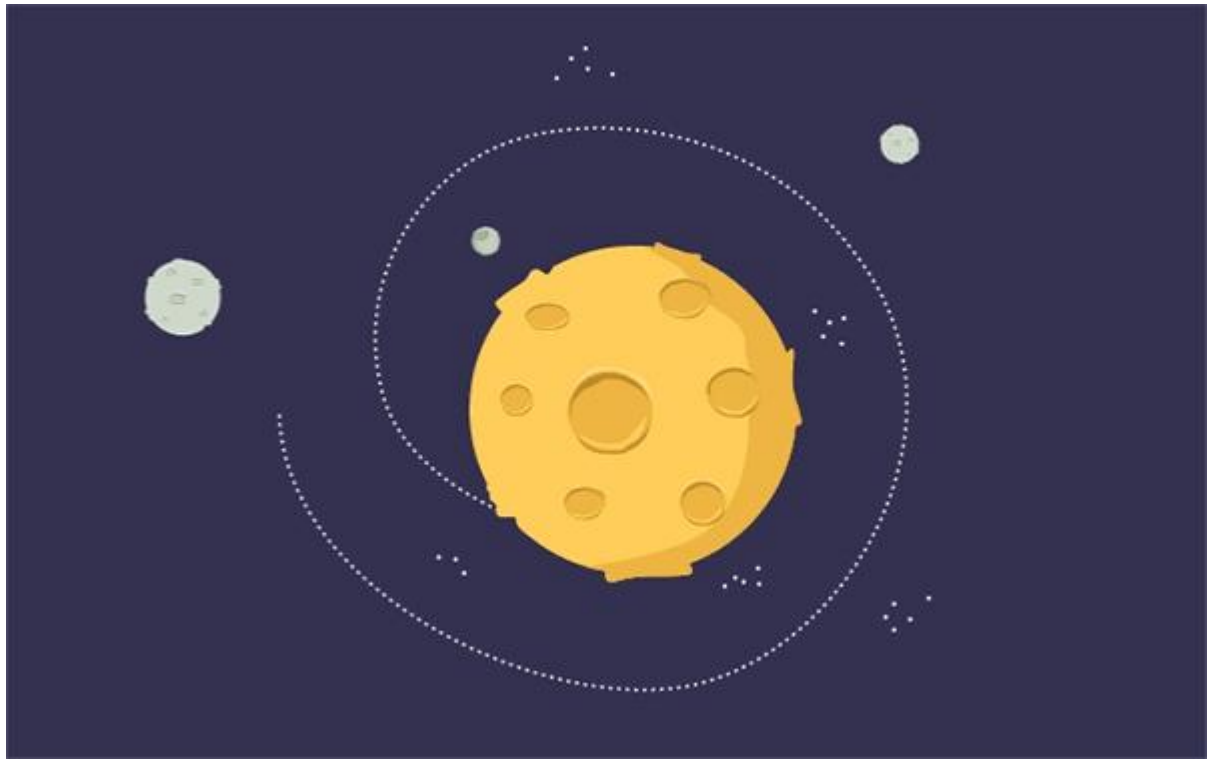
Parametric Cubic Curves

- Natural
- Hermite
- Bezier
- Catmull-Rom

Parametric Curves and Animation of Objects

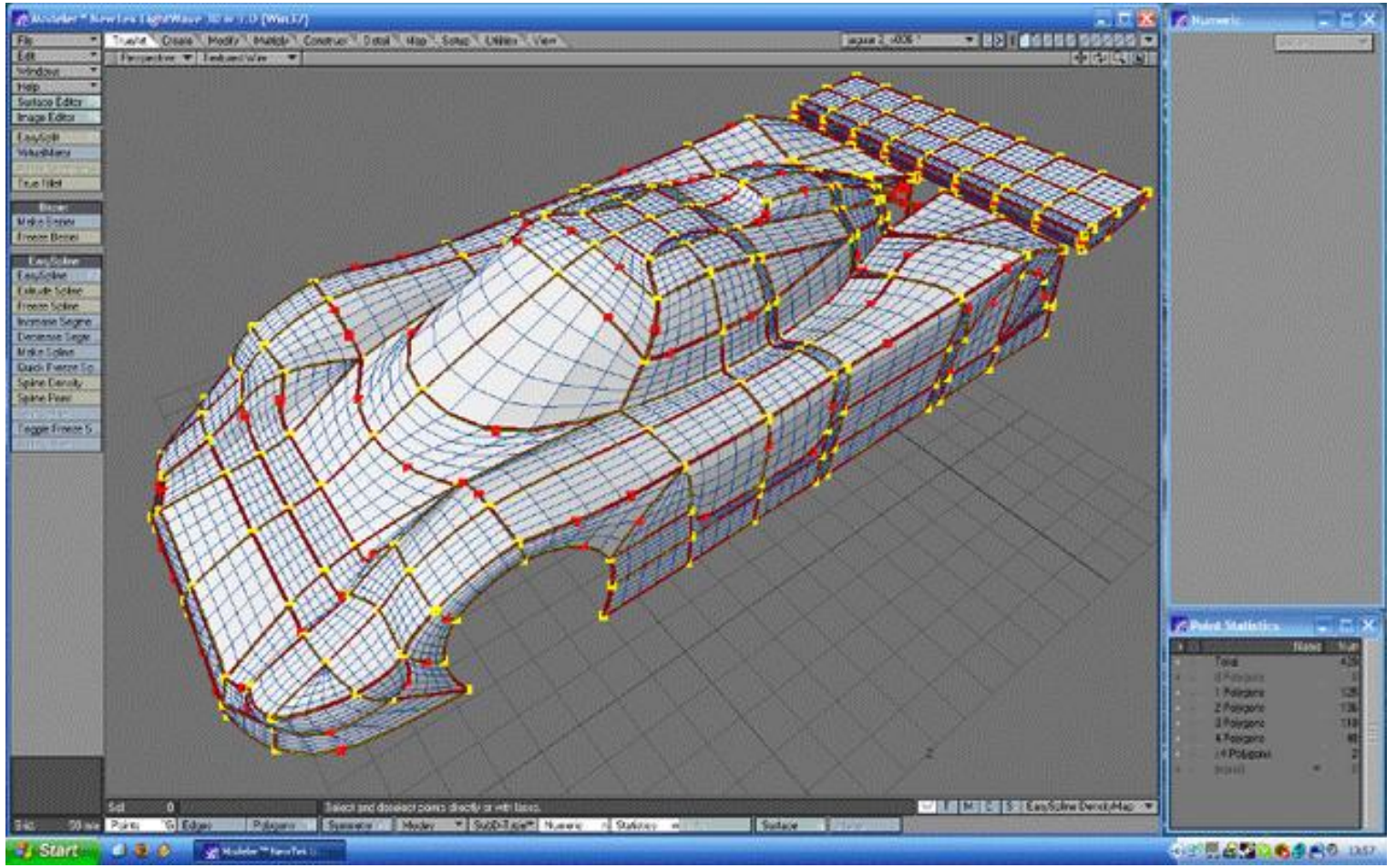


Parametric Curves and Animation of Objects



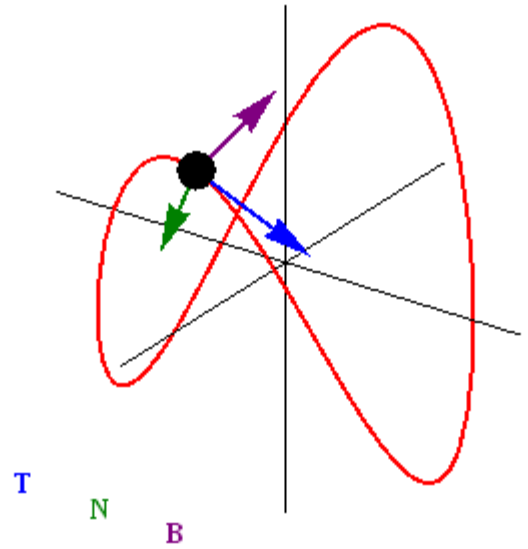
- Use parameter t of a curve **Q**(t) to represent **time** in animation

3D modeling with spline surfaces



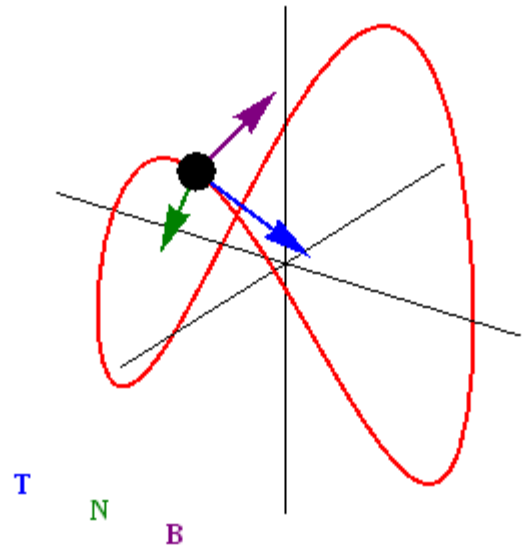
Curve Framing

- Defining orientation for every point of the curve
- Useful if curve is trajectory of a camera
- Or clone objects onto the curve and have their orientations vary smoothly



Curve Framing

- Have only tangent vector T



Parallel Transport Frames

Hansen et al. 1995

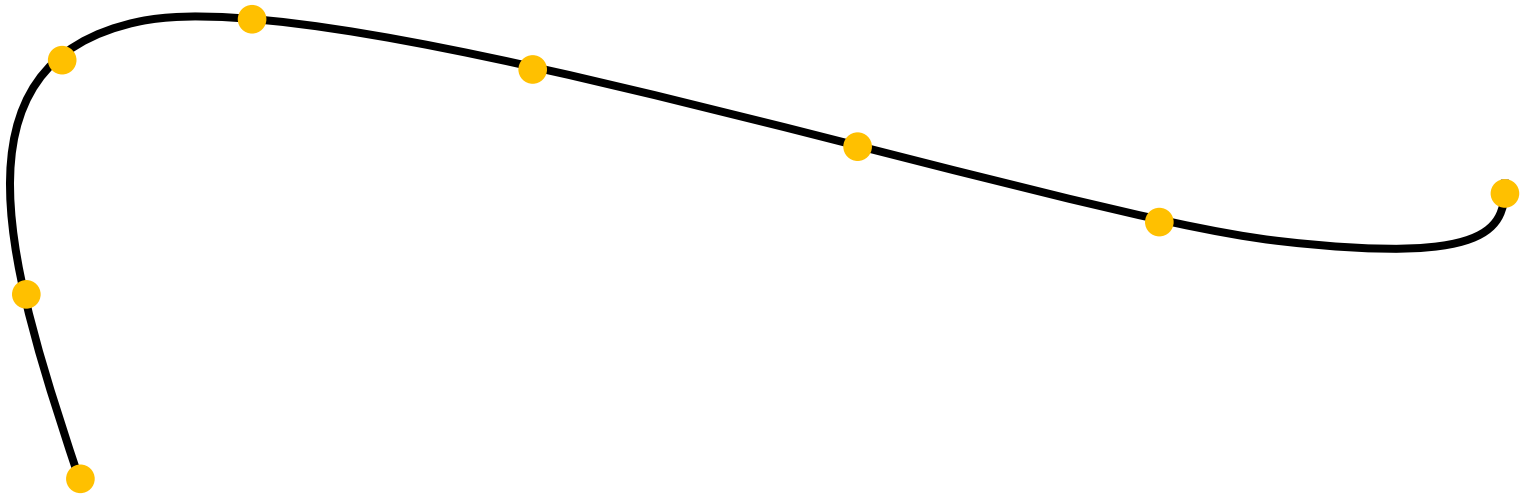
INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

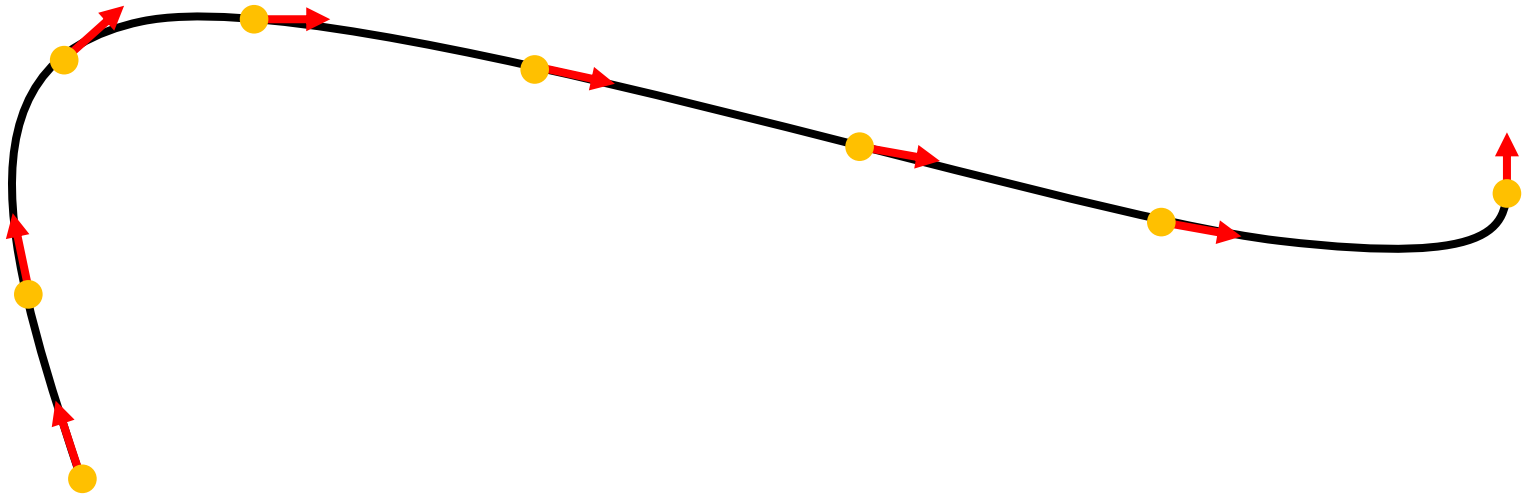
ALGORITHM:

for $i \leftarrow 0$ **to** $N - 1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \leq \theta \leq \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

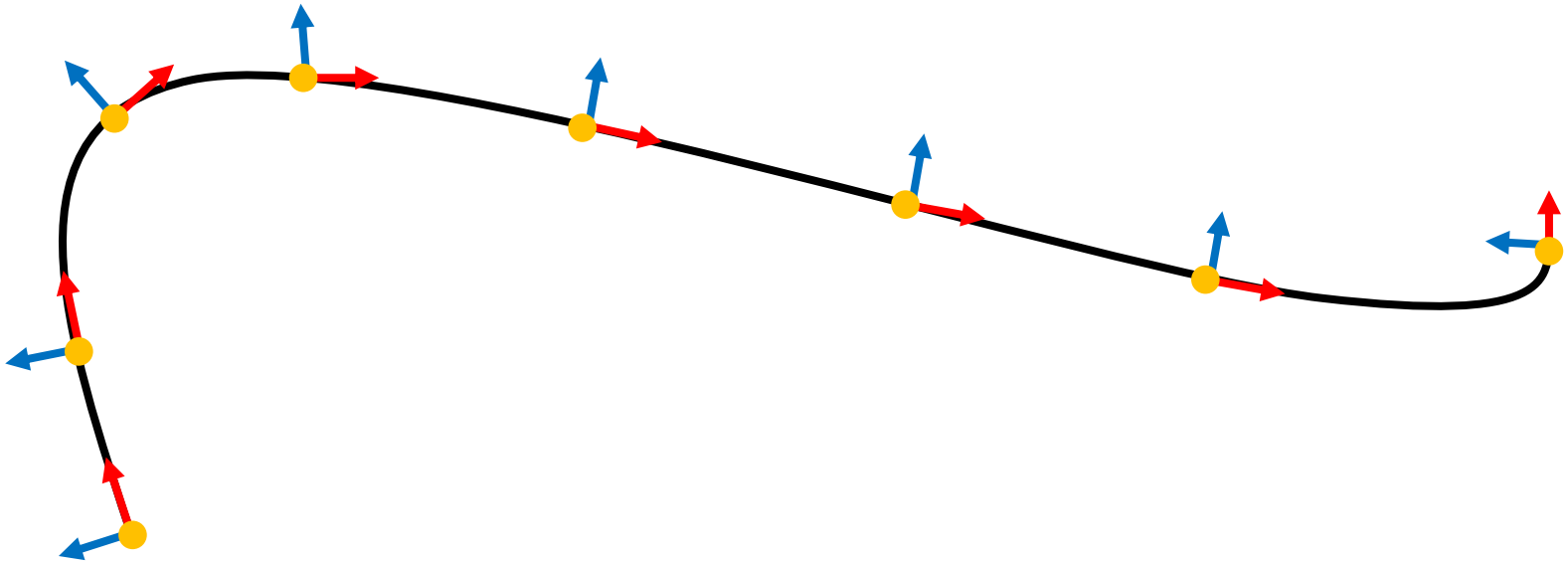
Parametric Curve



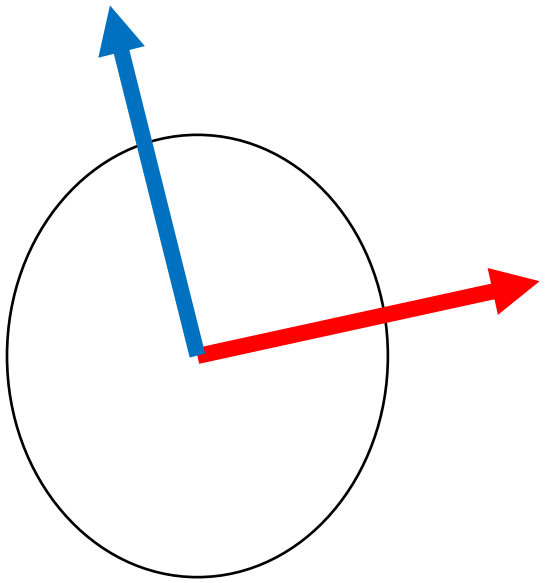
Tangents



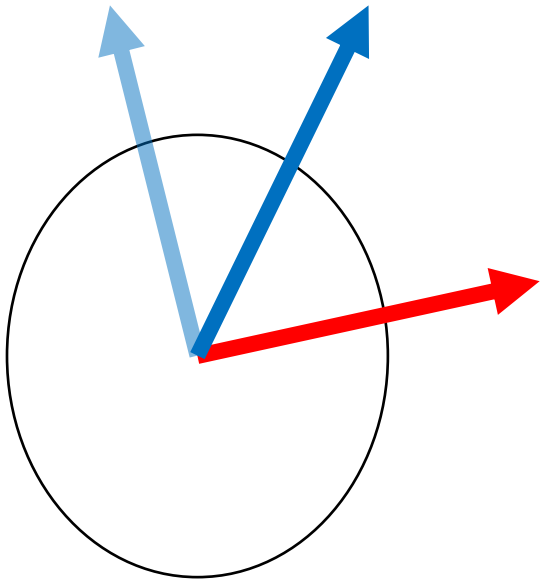
Torsion of a Curve



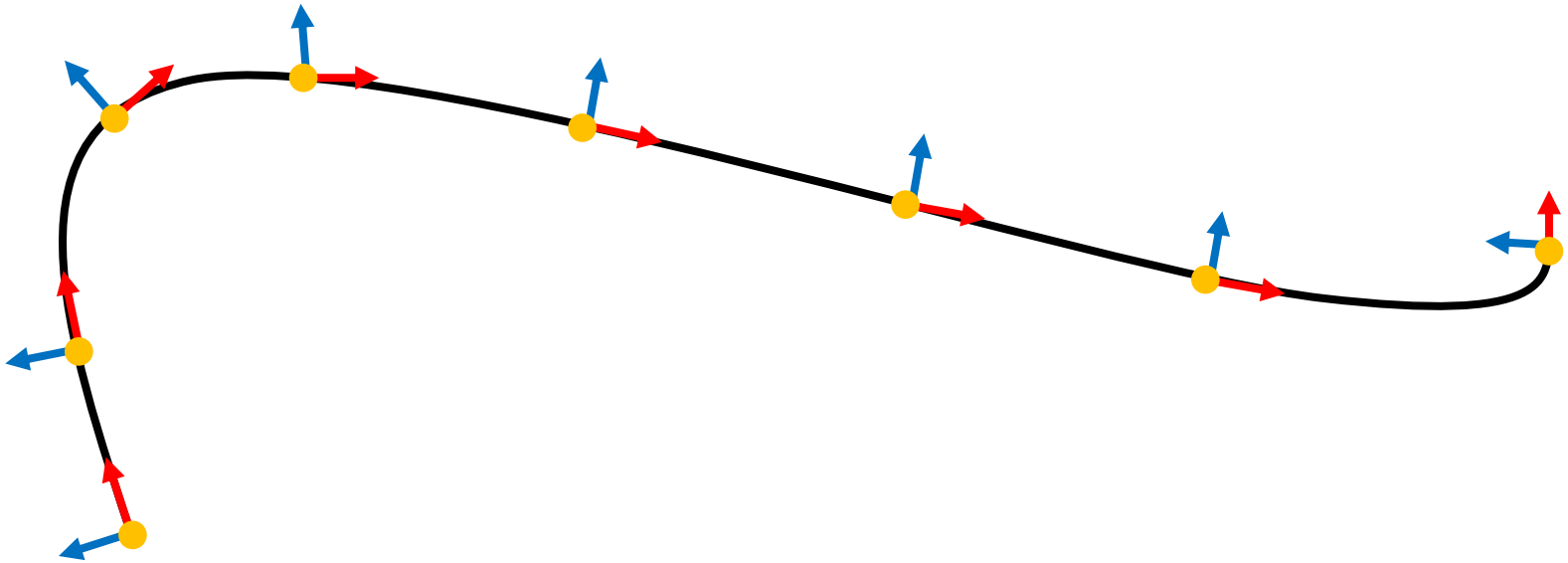
Torsion of a Curve



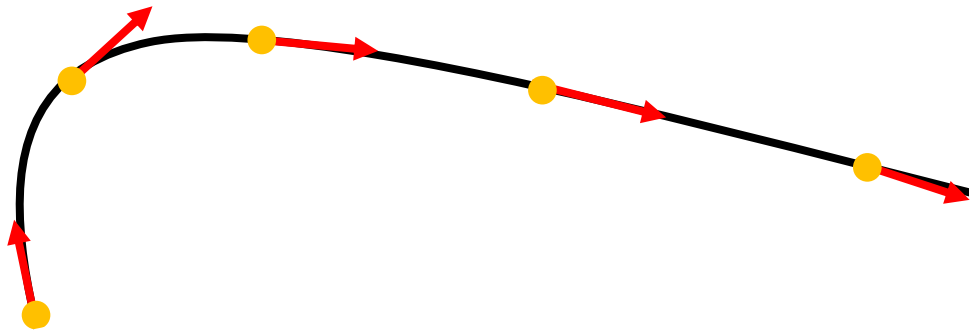
Torsion of a Curve



Smooth Variation of Torsion



Parallel Transport



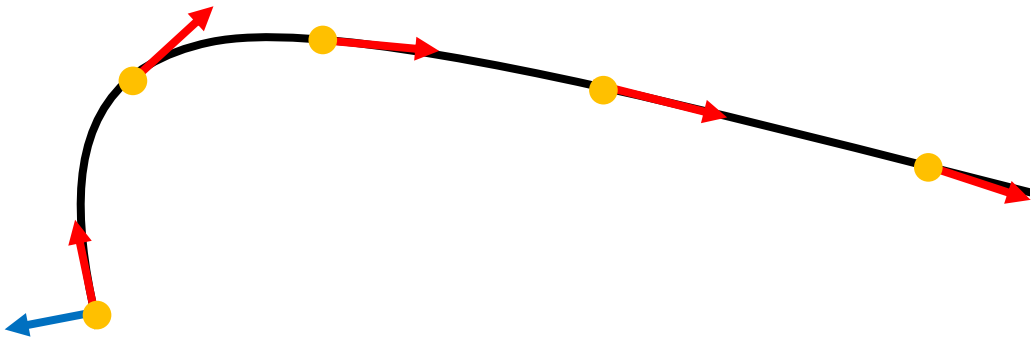
INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

ALGORITHM:

for $i \leftarrow 0$ **to** $N-1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \le \theta \le \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

Parallel Transport



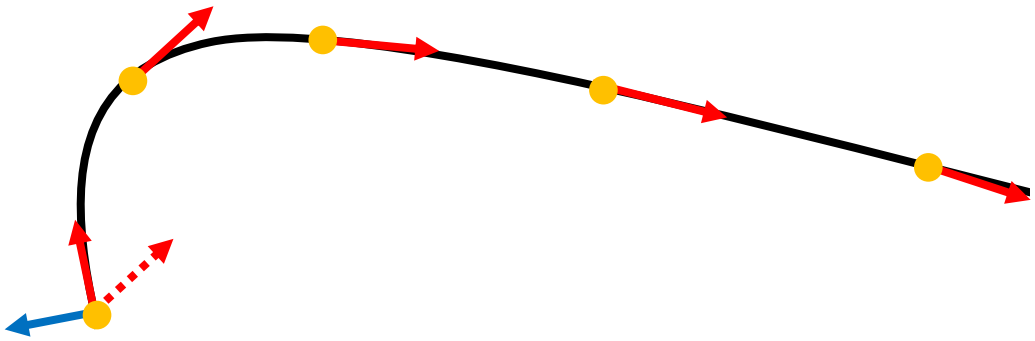
INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

ALGORITHM:

for $i \leftarrow 0$ **to** $N-1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \le \theta \le \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

Parallel Transport



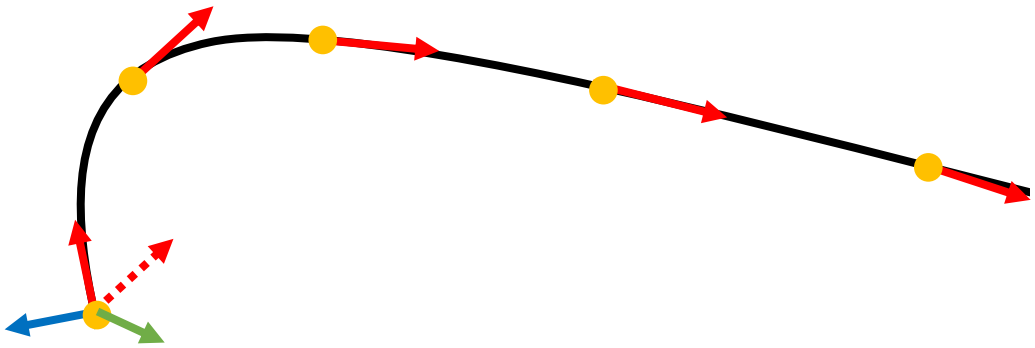
INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

ALGORITHM:

for $i \leftarrow 0$ **to** $N-1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \le \theta \le \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

Parallel Transport



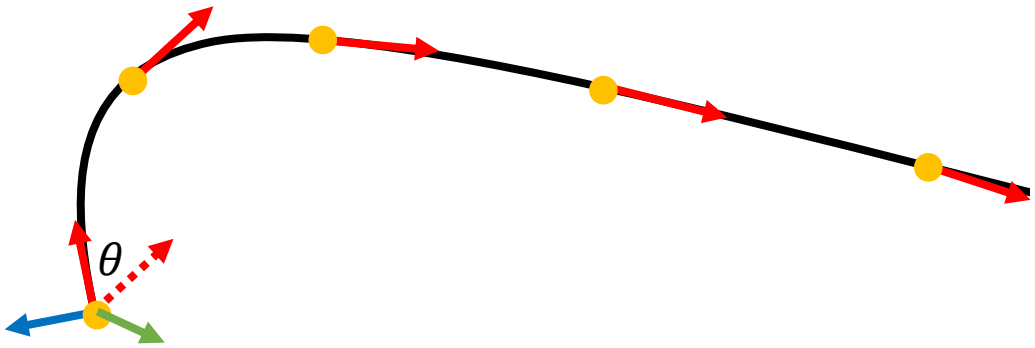
INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

ALGORITHM:

for $i \leftarrow 0$ **to** $N-1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \le \theta \le \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

Parallel Transport



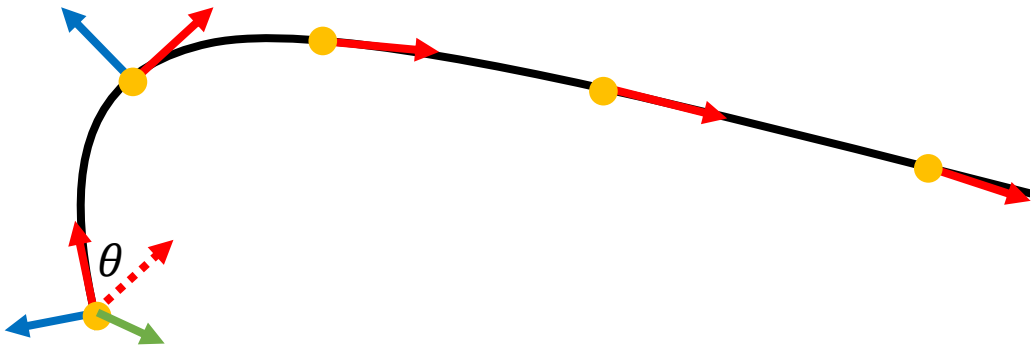
INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

ALGORITHM:

for $i \leftarrow 0$ **to** $N - 1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \leq \theta \leq \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

Parallel Transport



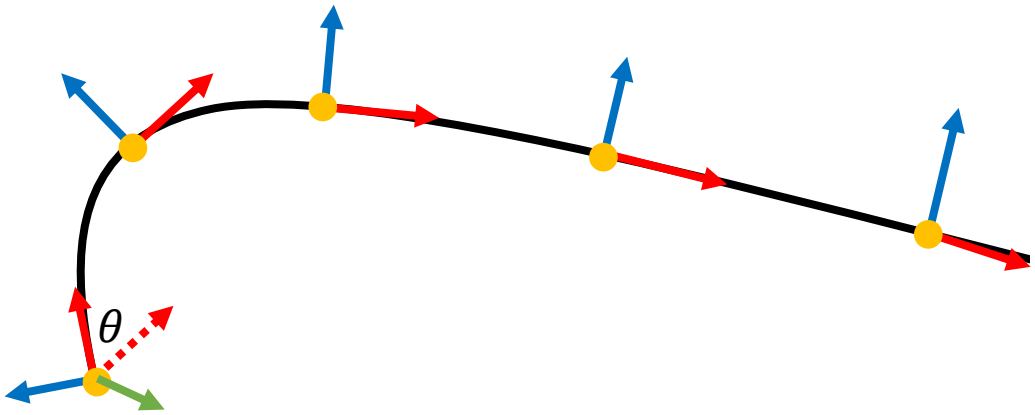
INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

ALGORITHM:

for $i \leftarrow 0$ **to** $N - 1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \leq \theta \leq \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

Parallel Transport



INPUT: (1) A list of unit tangent vectors $\{\hat{\mathbf{T}}_i\}$, $i = 0, \ldots, N$;
(2) An initial normal vector $\vec{\mathbf{V}}_0$, $\vec{\mathbf{V}}_0 \perp \hat{\mathbf{T}}_0$.

OUTPUT: A list of parallel-transported normal vectors $\{\vec{\mathbf{V}}_i\}$, $i = 1, \ldots, N$, $\vec{\mathbf{V}}_i \perp \hat{\mathbf{T}}_i$.

ALGORITHM:

for $i \leftarrow 0$ **to** $N - 1$ **step** 1
  $\vec{\mathbf{B}} \leftarrow \hat{\mathbf{T}}_i \times \hat{\mathbf{T}}_{i+1}$;
  if $\|\vec{\mathbf{B}}\| = 0$ **then**
    $\vec{\mathbf{V}}_{i+1} \leftarrow \vec{\mathbf{V}}_i$;
  else
    $\hat{\mathbf{B}} \leftarrow \vec{\mathbf{B}}/\|\vec{\mathbf{B}}\|$;
    $\theta \leftarrow \arccos(\hat{\mathbf{T}}_i \cdot \hat{\mathbf{T}}_{i+1})$; // $0 \leq \theta \leq \pi$
    $\vec{\mathbf{V}}_{i+1} \leftarrow \mathbf{R}(\hat{\mathbf{B}}, \theta) * \vec{\mathbf{V}}_i$;
    // Rotate by angle θ about $\hat{\mathbf{B}}$ (see Appendix A)
  end if
end for

Animation

- Forward Kinematics
  - Scene Graph
- Keyframing
  - Interpolation with parametric curves
  - Curve Framing