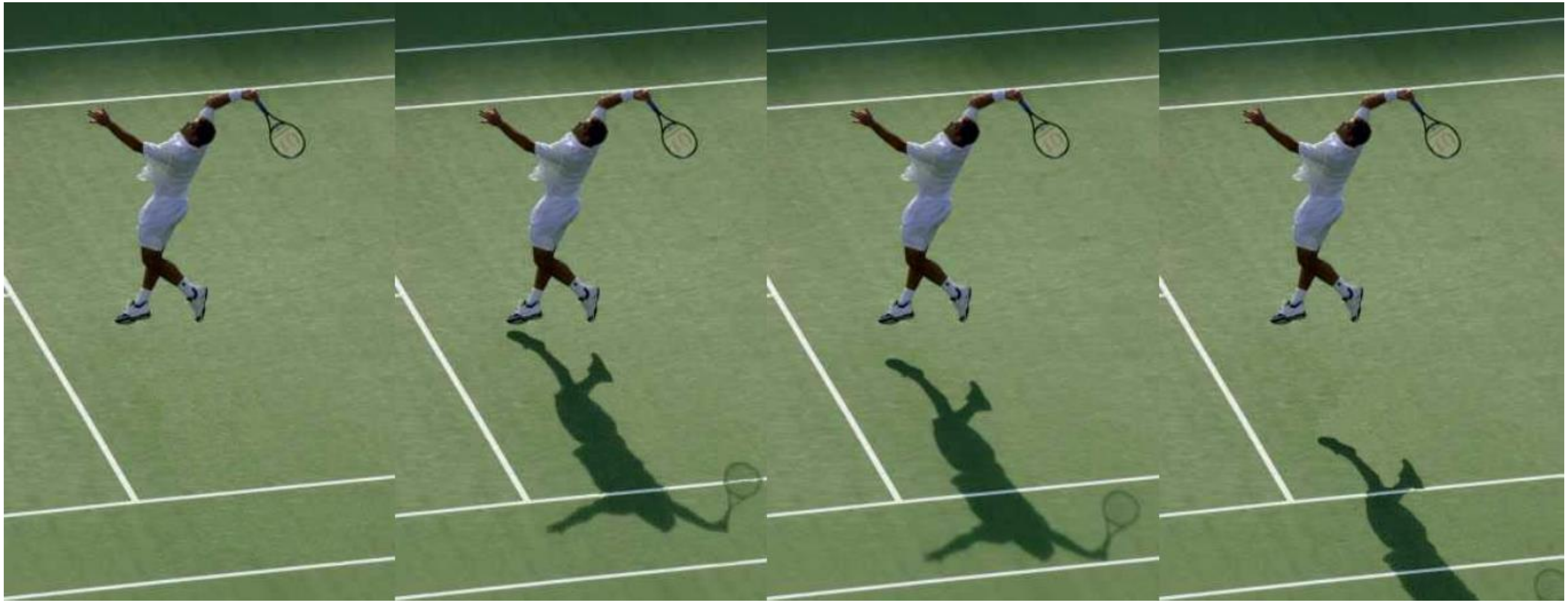
GRK 9

Dr Wojciech Palubicki

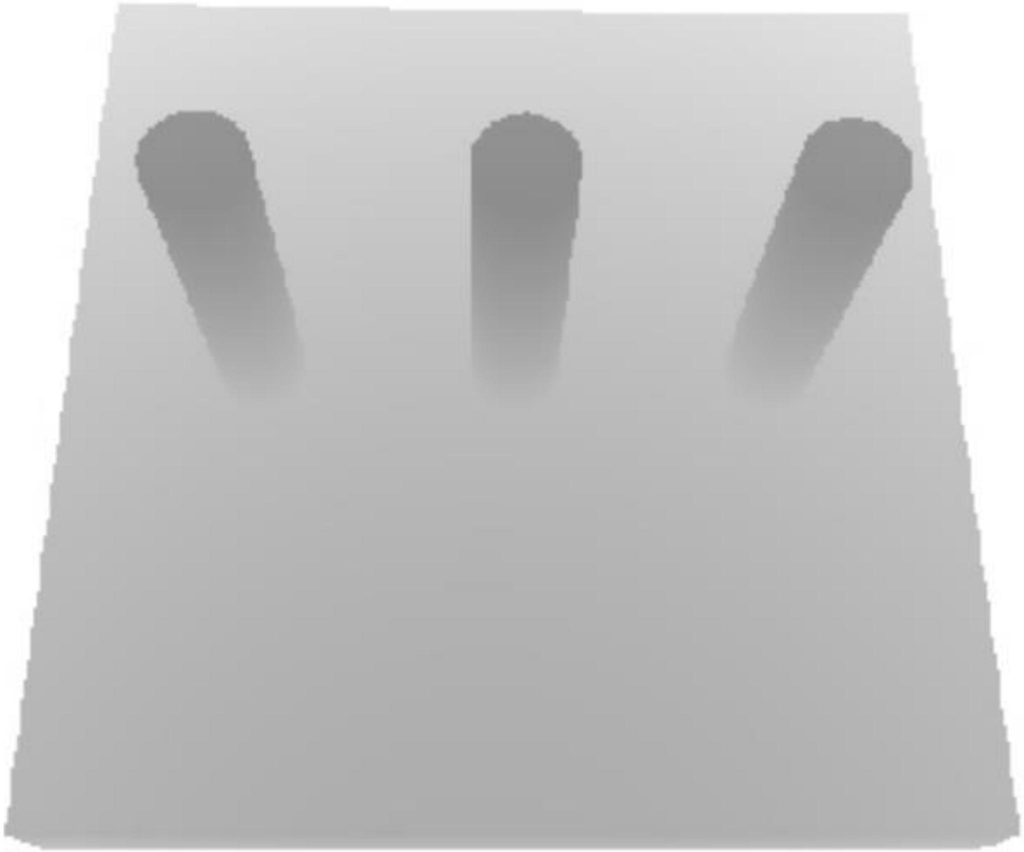
Overview

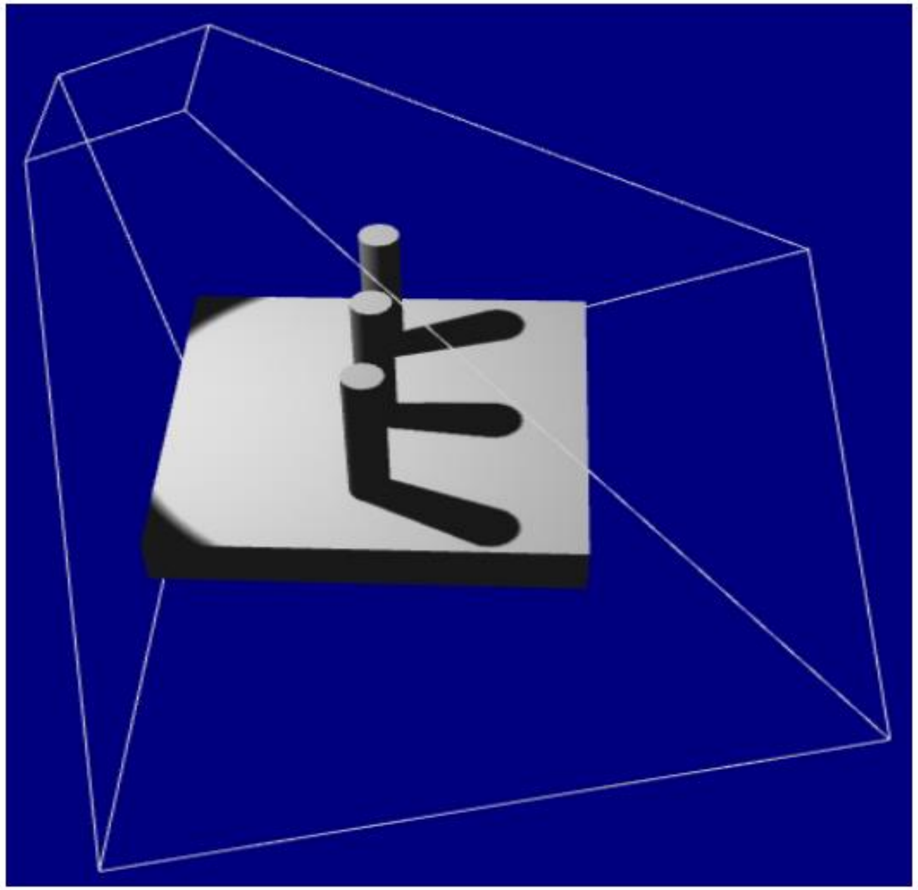
- Shadow mapping
- Animation

Shadows



Shadow mapping



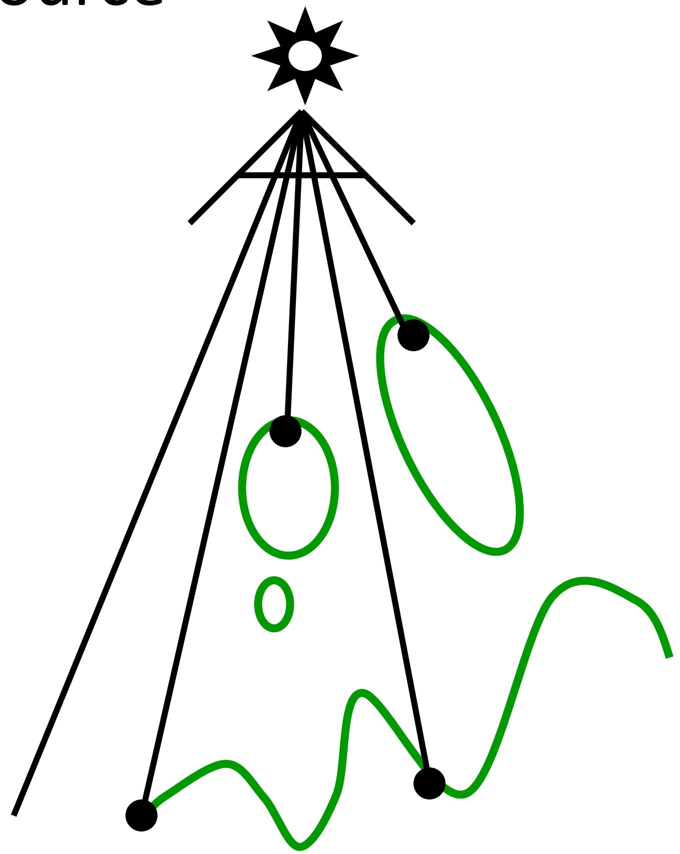


Shadow mappigng

- Image-space shadow determination
  - Lance Williams published the basic idea in 1978
- Completely image-space algorithm
  - means no knowledge of scene’s geometry is required
  - must deal with aliasing artifacts

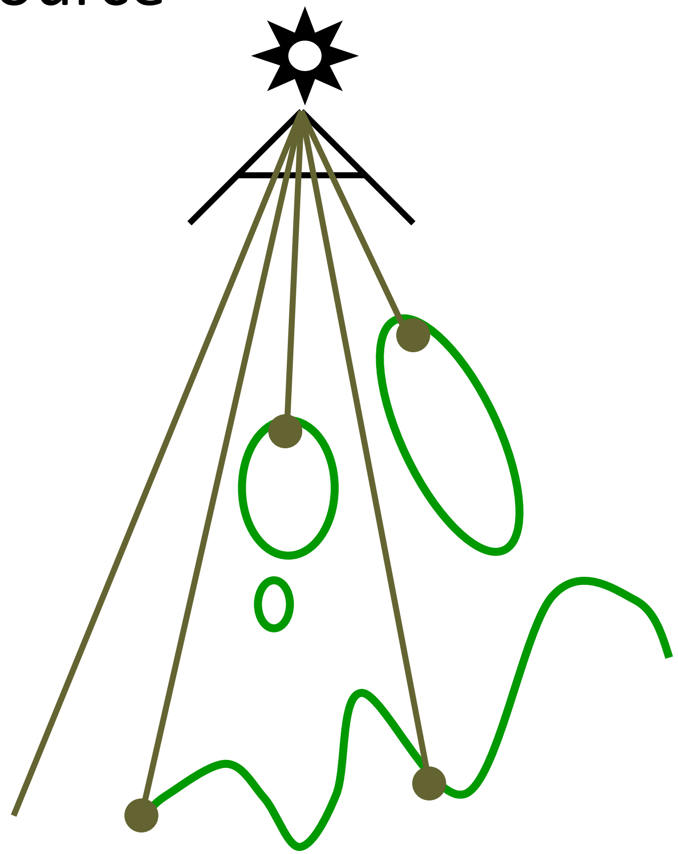
Phase 1: Render from Light Position

- Depth image from light source



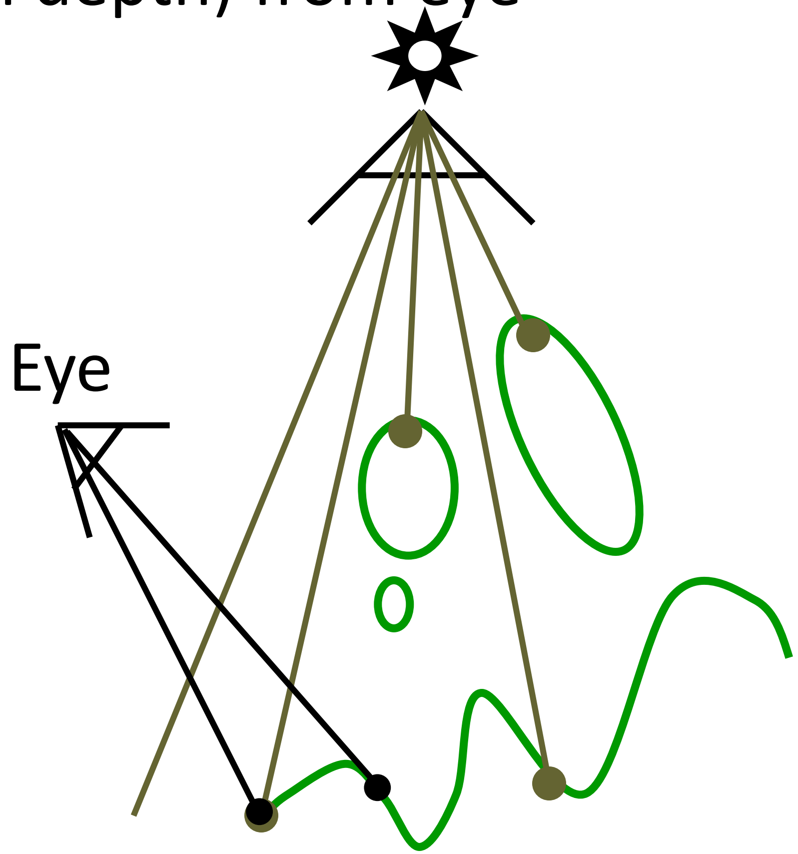
Phase 1: Render from Light Position

- Depth image from light source



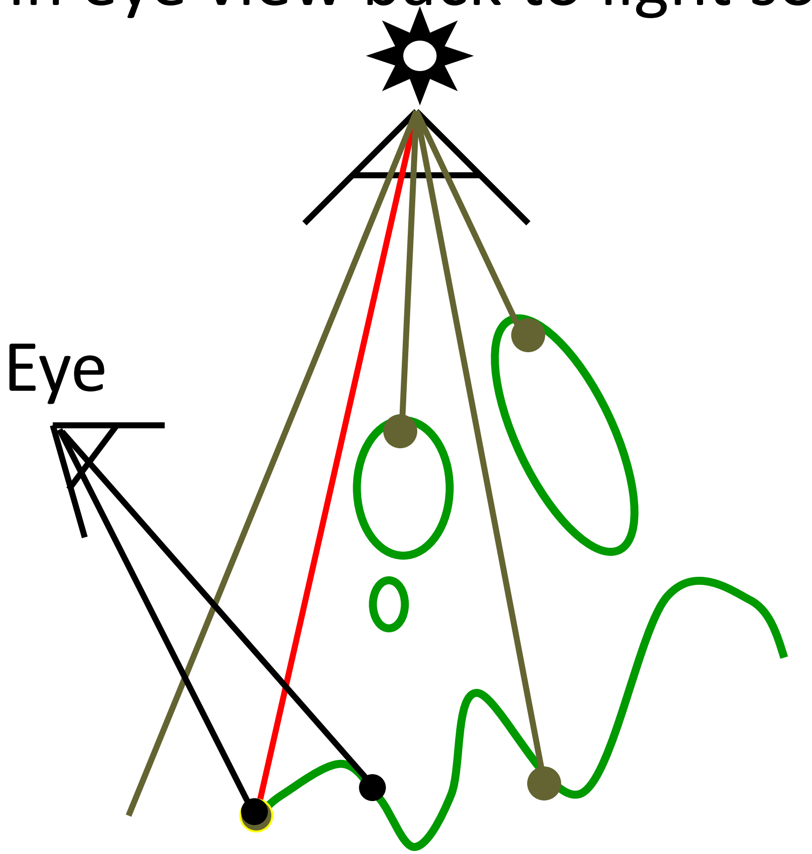
Phase 2: Render from Eye Position

- Standard image (with depth) from eye



Phase 2: Project to light for shadows

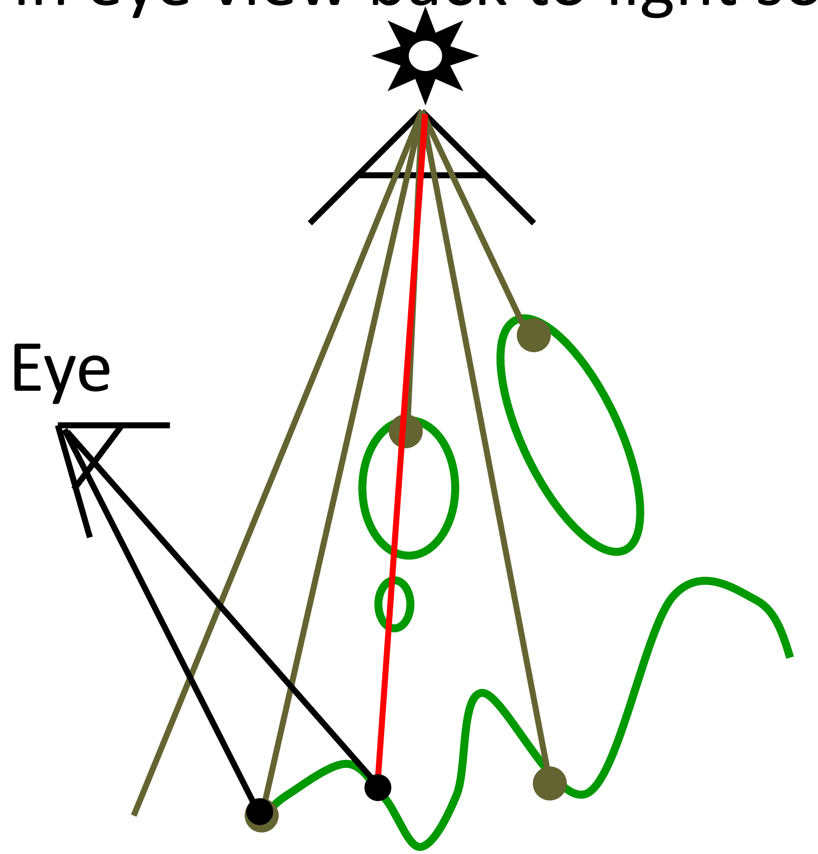
- Project visible points in eye view back to light source



Projected depths match for light and eye. VISIBLE

Phase 2: Project to light for shadows

- Project visible points in eye view back to light source

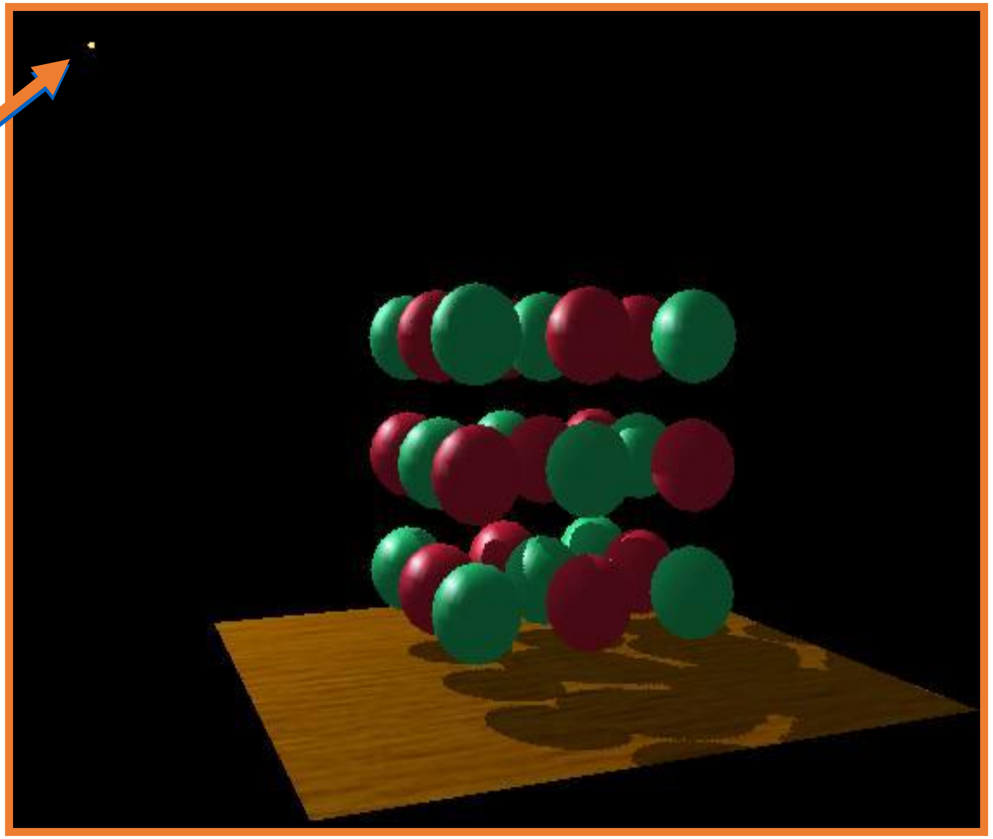


Projected depths from light, eye not the same. BLOCKED

Visualizing Shadow Mapping

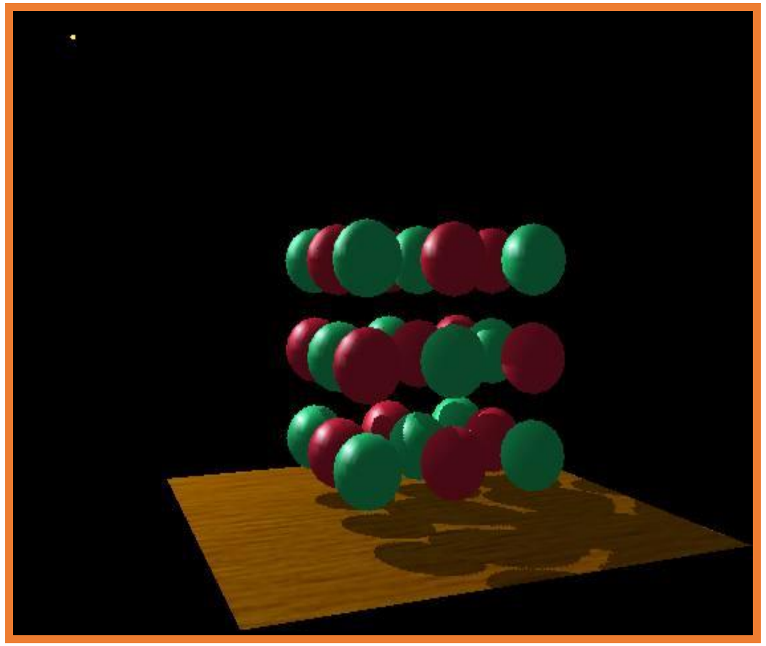
- A fairly complex scene with shadows

the point
light source

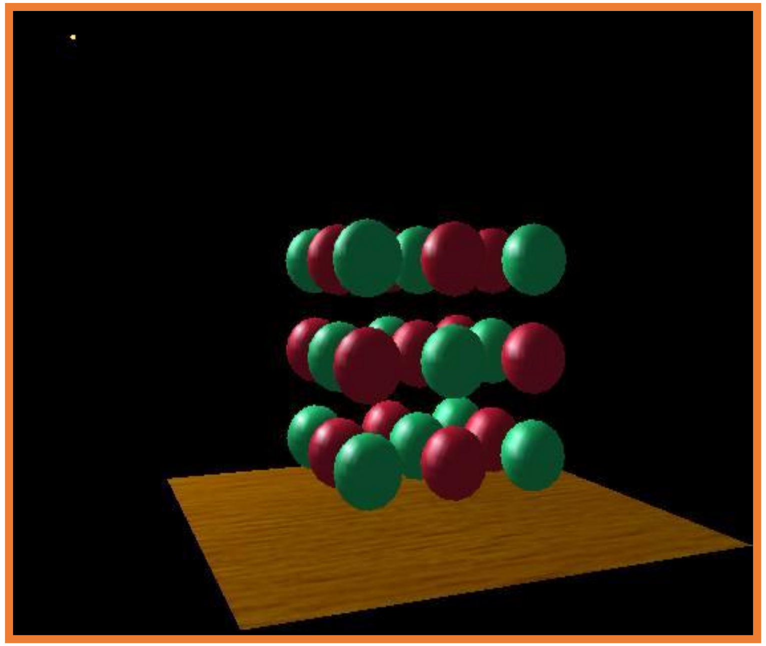


Visualizing Shadow Mapping

- Compare with and without shadows



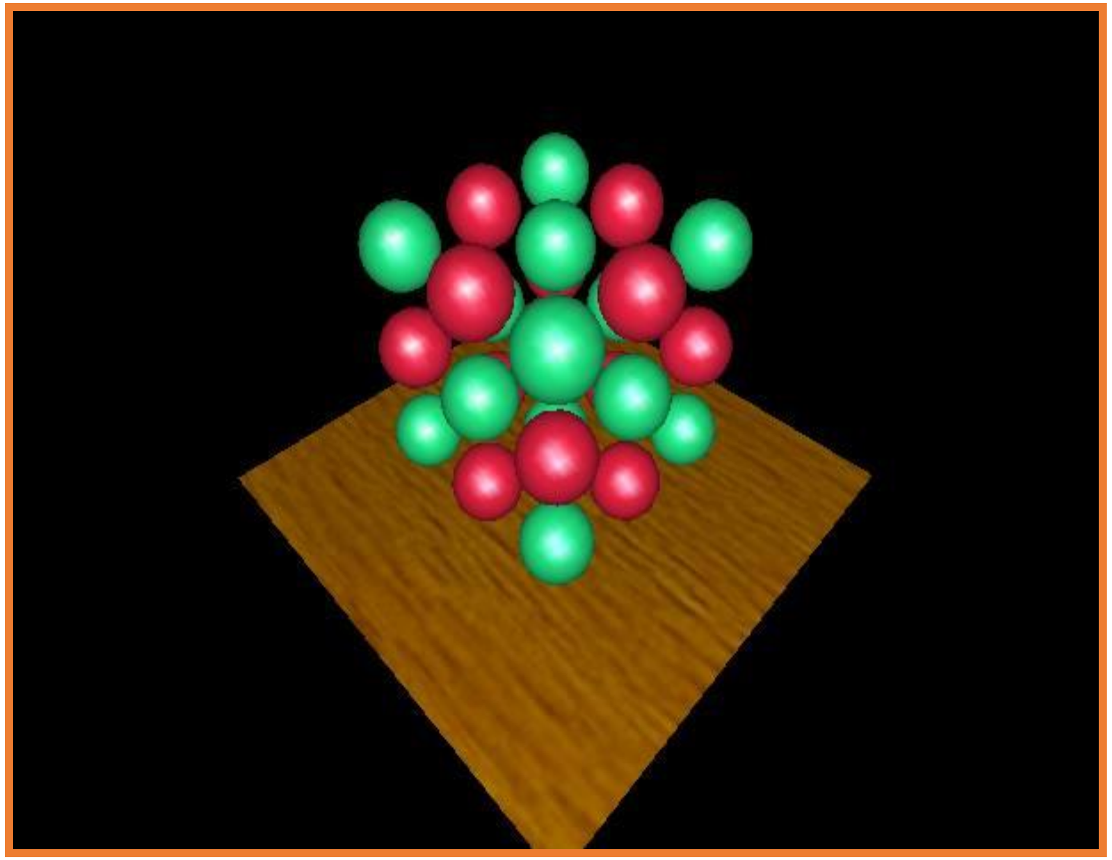
with shadows

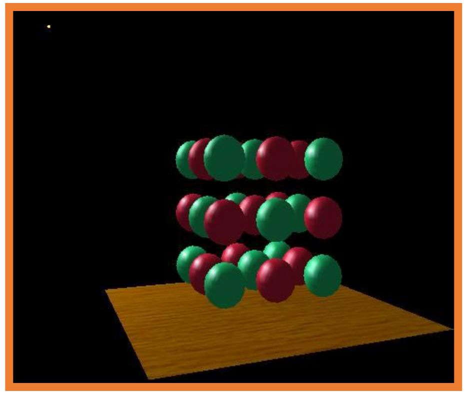


without shadows

Visualizing Shadow Mapping

- The scene from the light's point-of-view

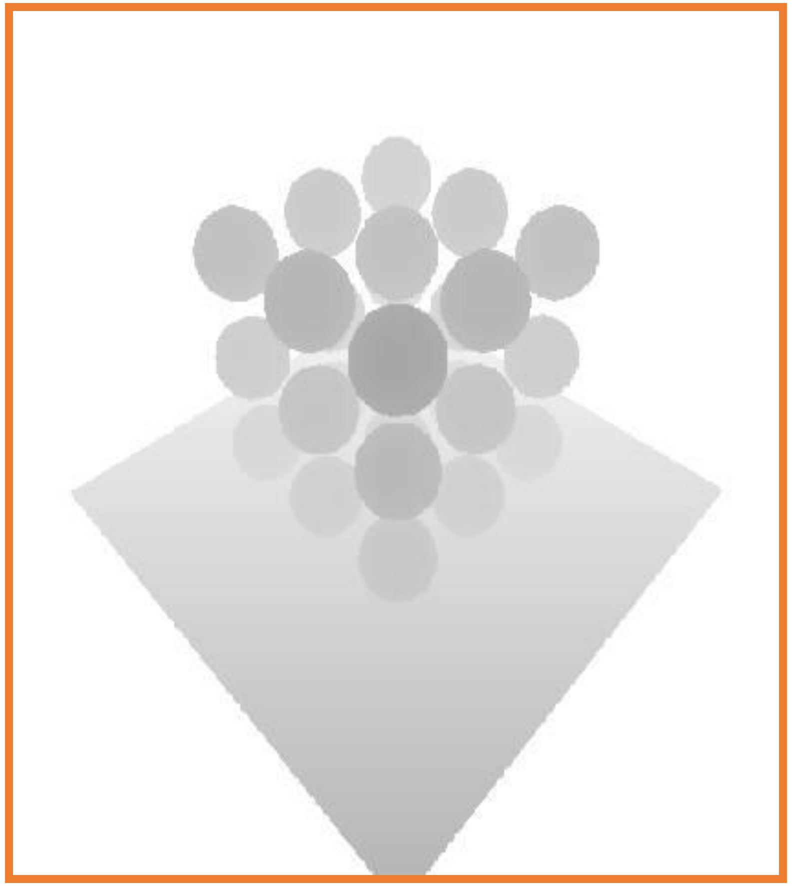


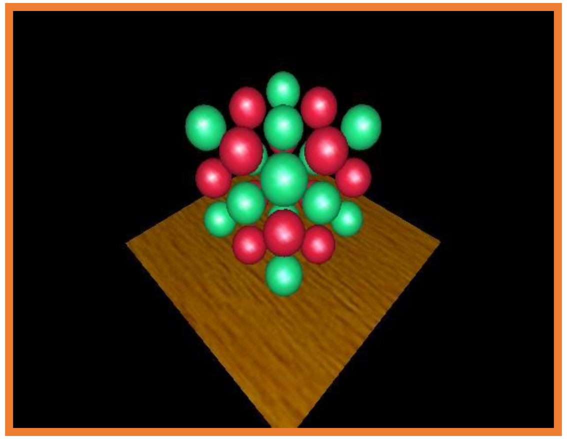


from the
eye's point-of-view
again

Visualizing Shadow Mapping

- The depth buffer from the light's point-of-view

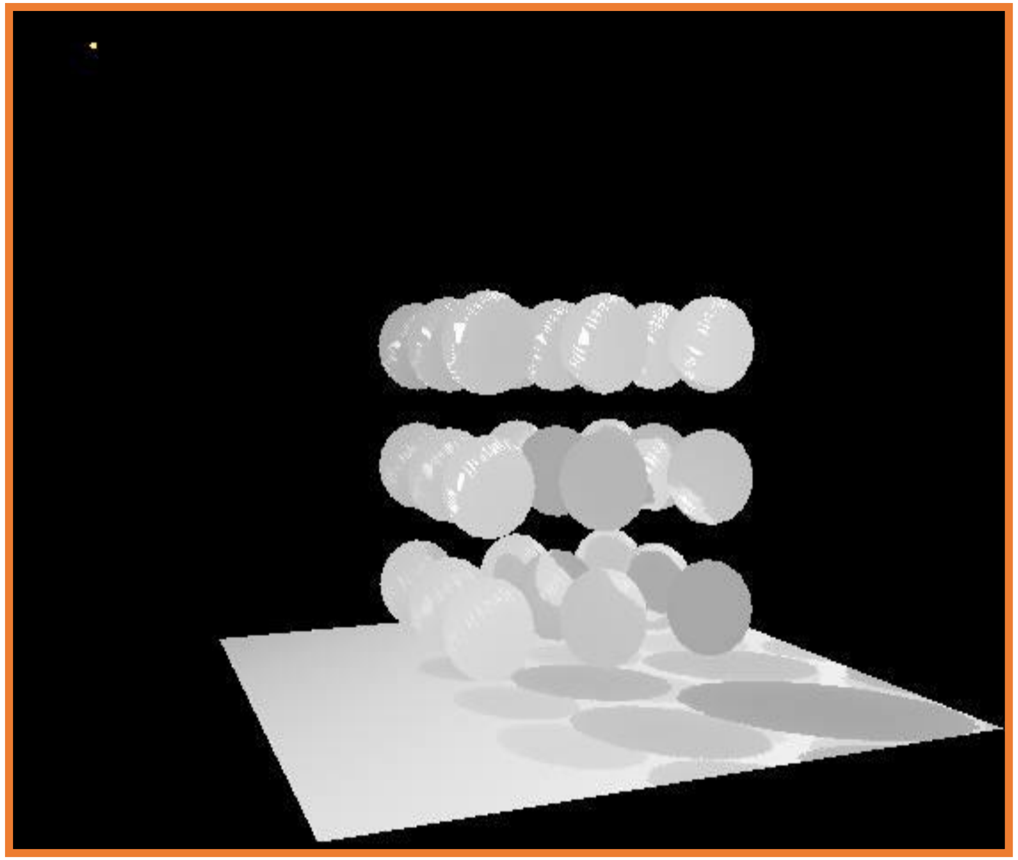


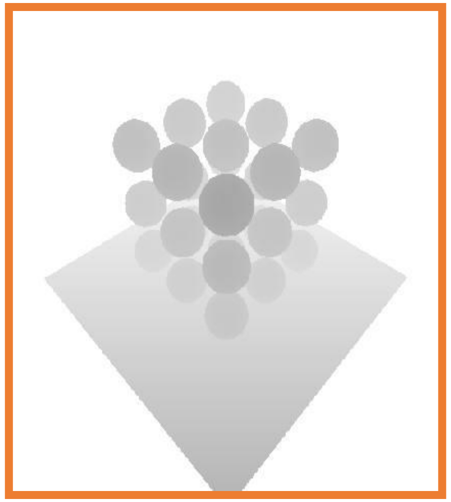


from the
light's point-of-view
again

Visualizing Shadow Mapping

- Projecting the depth map onto the eye's view



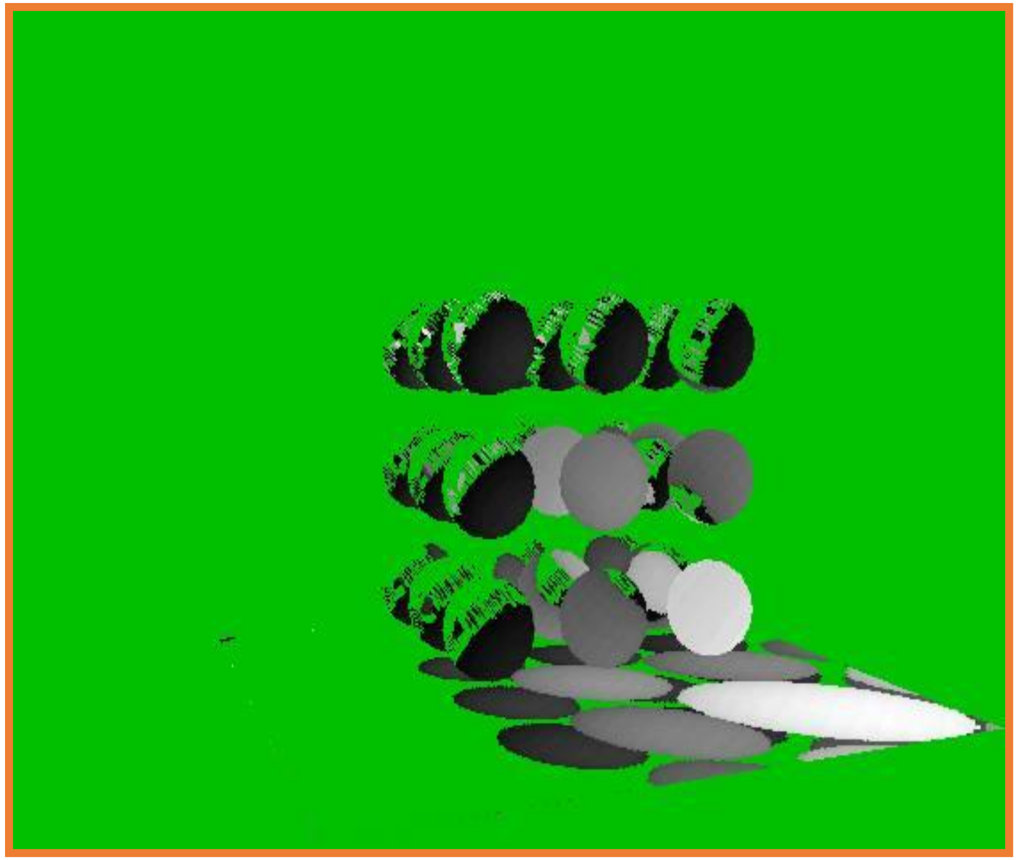


depth map for light's point-of-view again

Visualizing Shadow Mapping

- Comparing light distance to light depth map

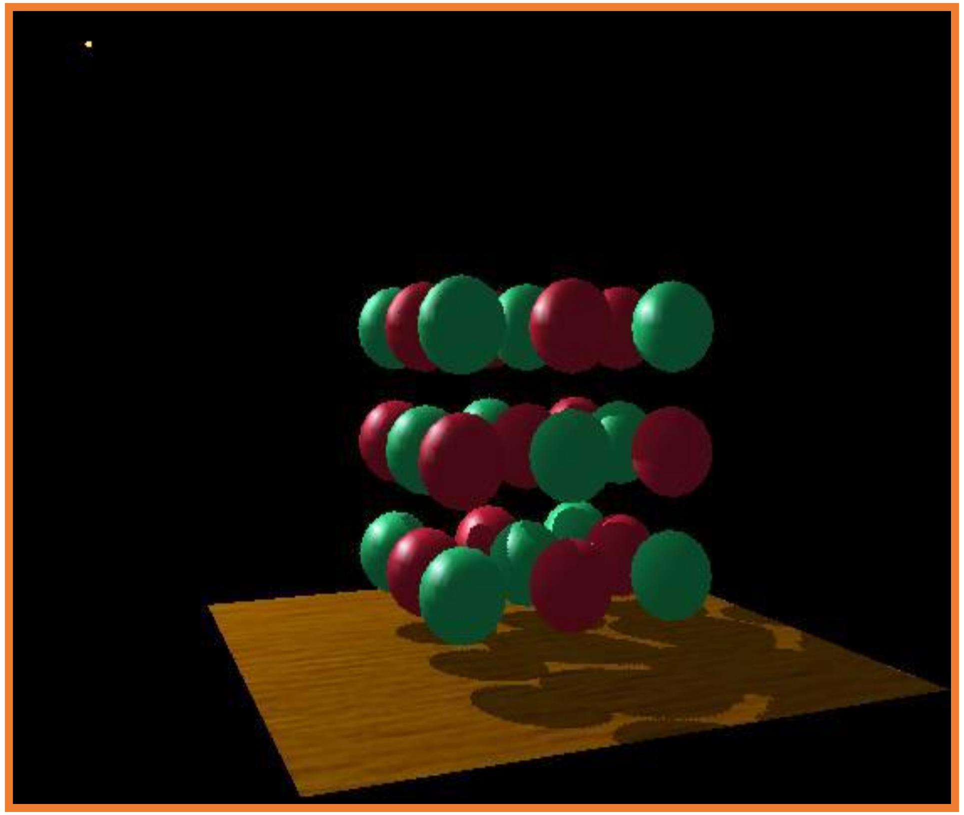
Green is where the light planar distance and the light depth map are approximately equal



Grey is where shadows should be

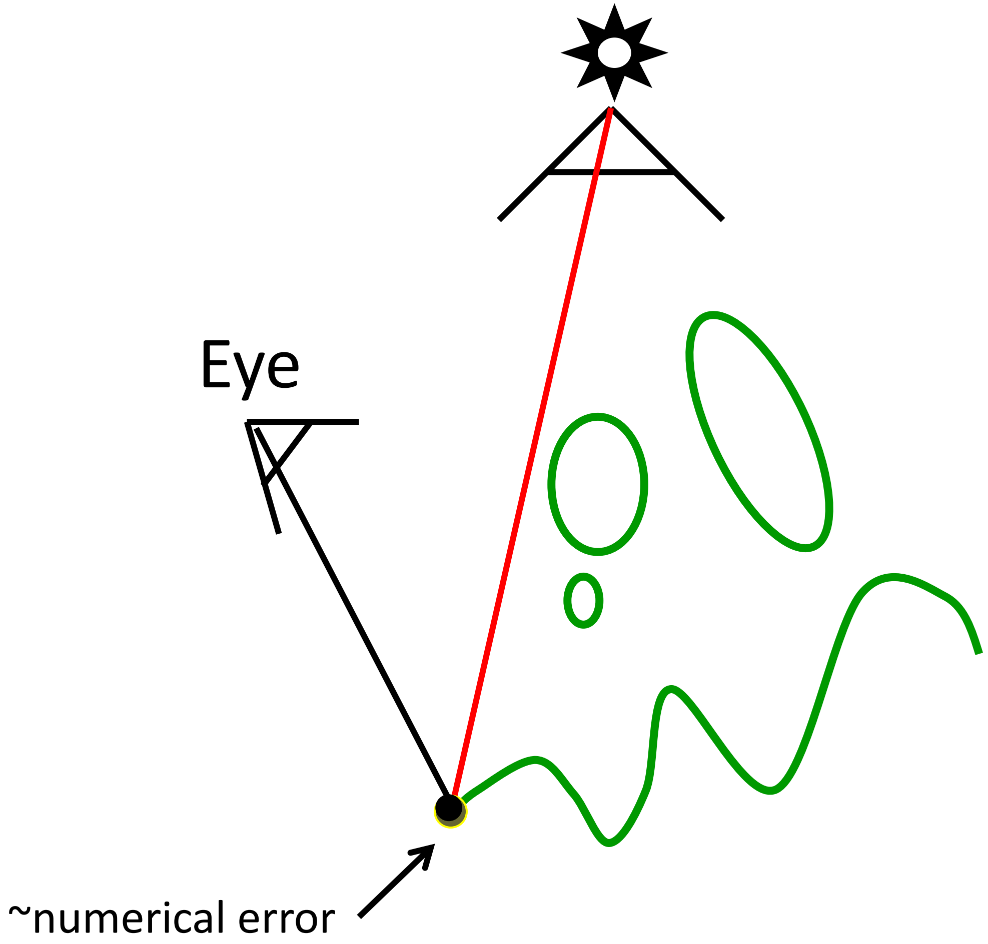
Visualizing Shadow Mapping

Notice how specular highlights never appear in shadows

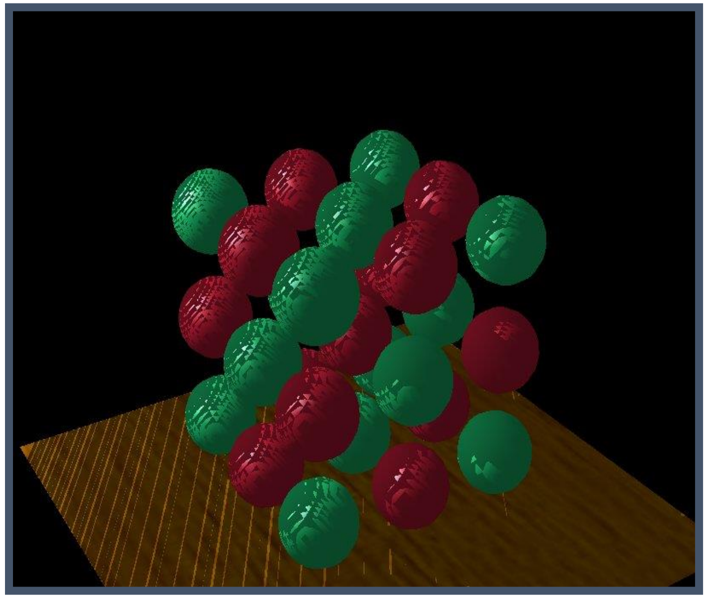


Notice how curved surfaces cast shadows on each other

Depth Map Bias

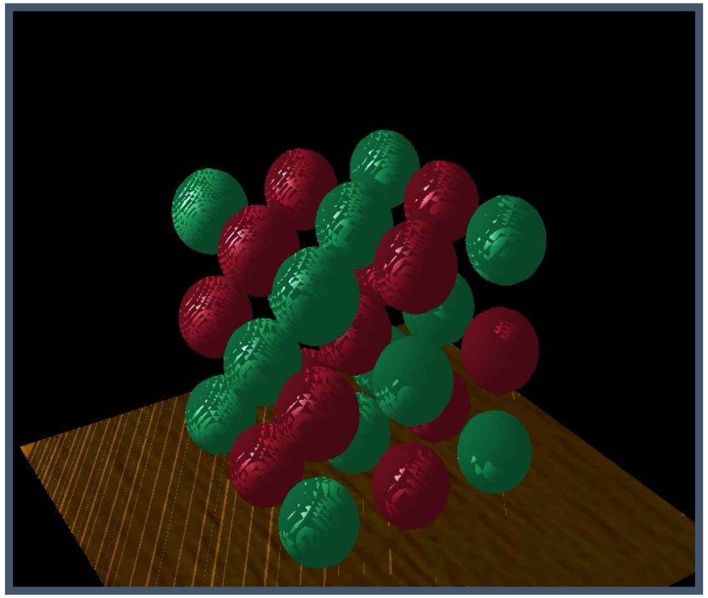


Depth Map Bias

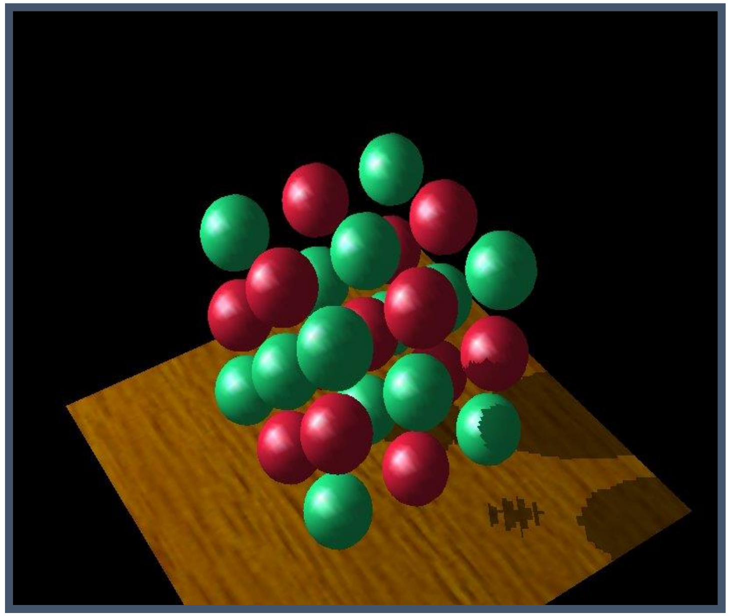


Too little bias,
everything begins to
shadow

Depth Map Bias

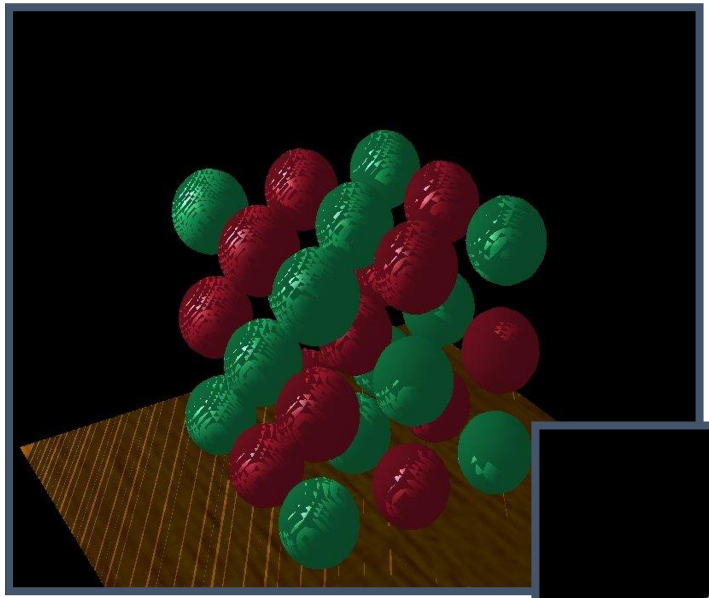


Too little bias, everything begins to shadow

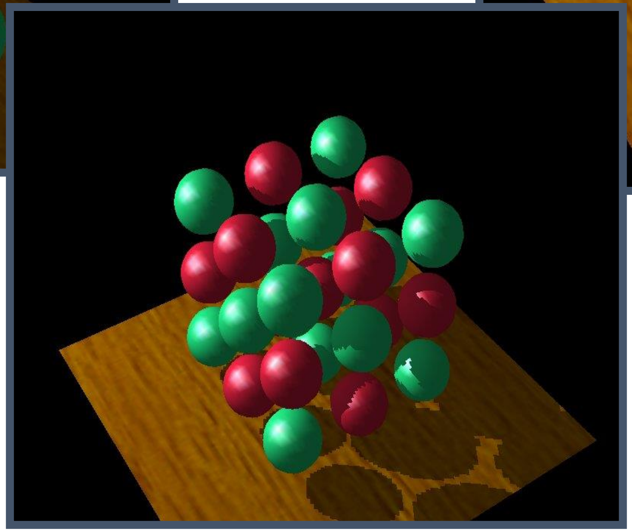


Too much bias, shadow starts too far back

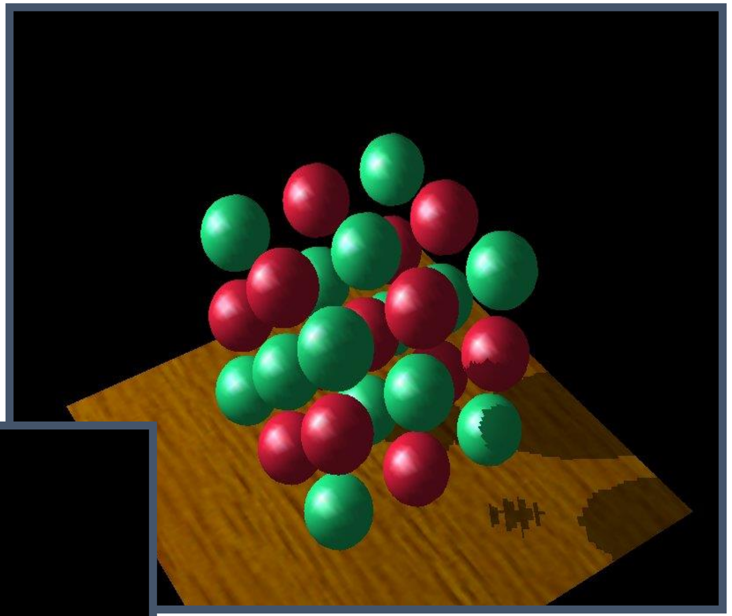
Depth Map Bias



Too little bias, everything begins to shadow



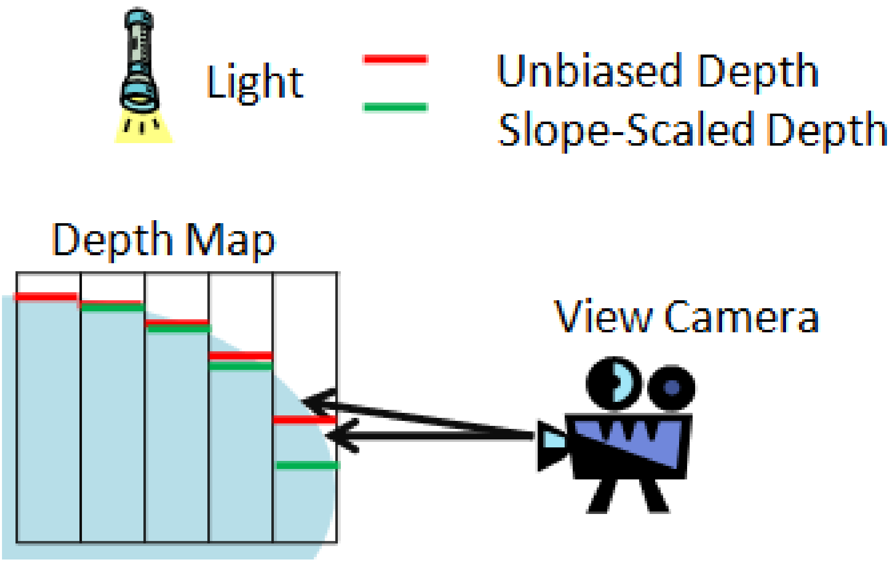
Right amount of bias



Too much bias, shadow starts too far back

Slope Scaled Bias

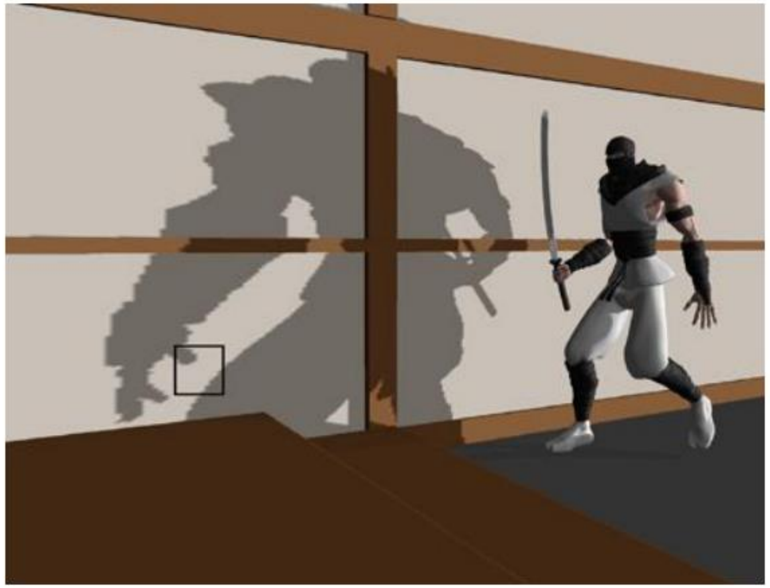
```
float bias = max(0.05 * (1.0 - dot(normal, light)), 0.005);
```



Percentage closer filtering (PCF)

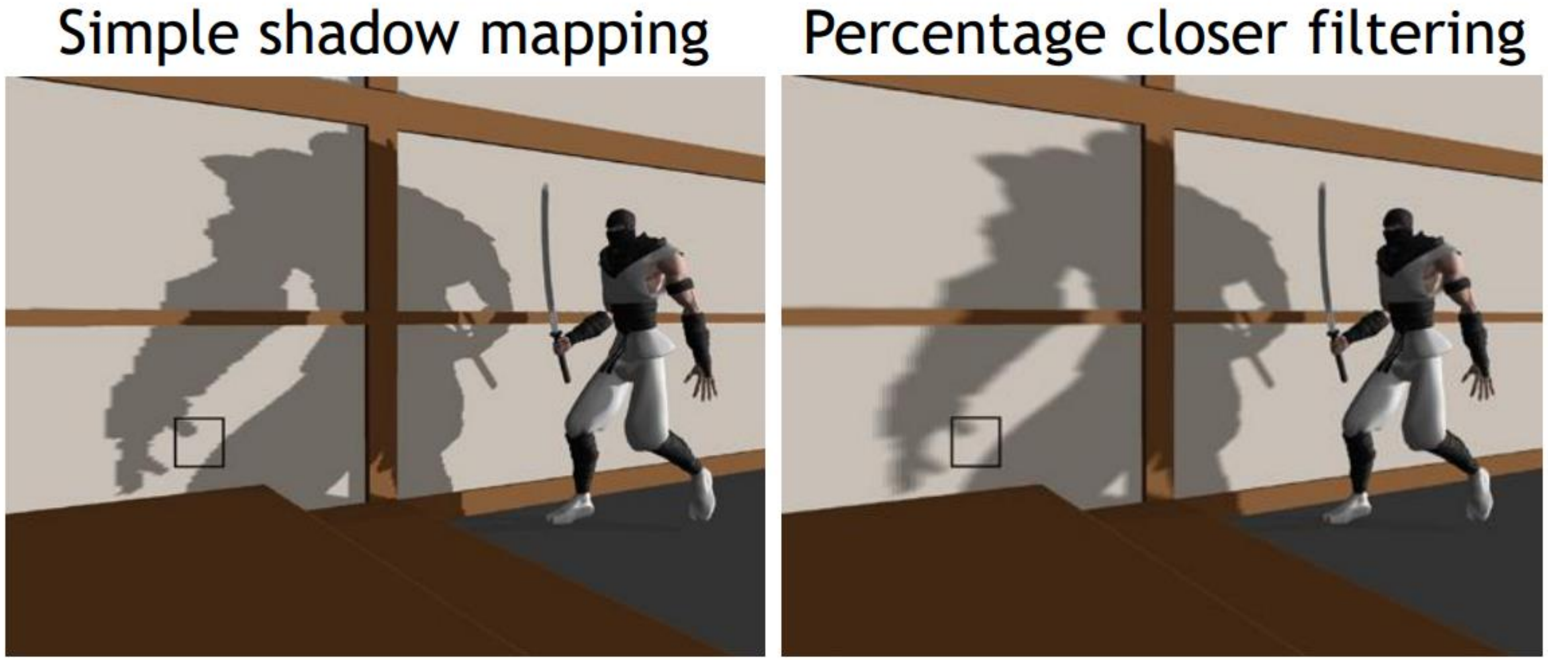
- Goal: avoid stair-stepping artifacts
- Similar to texture filtering

Simple shadow mapping



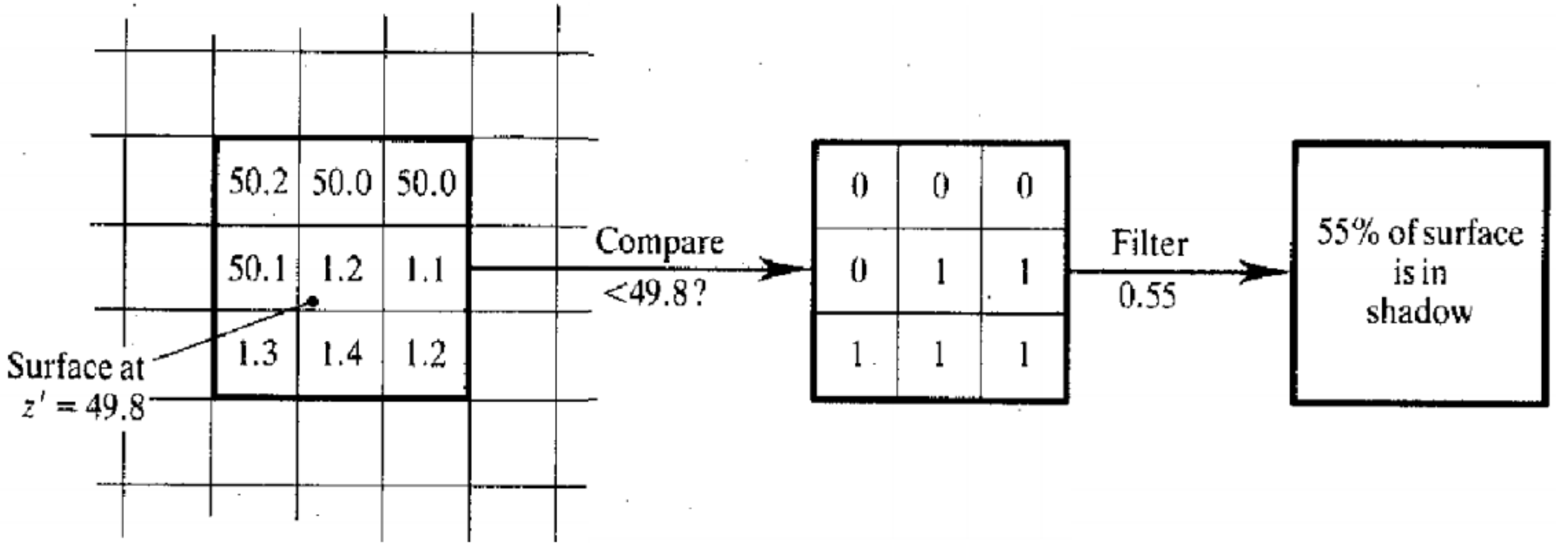
Percentage closer filtering (PCF)

- Goal: avoid stair-stepping artifacts
- Similar to texture filtering



Percentage closer filtering (PCF)

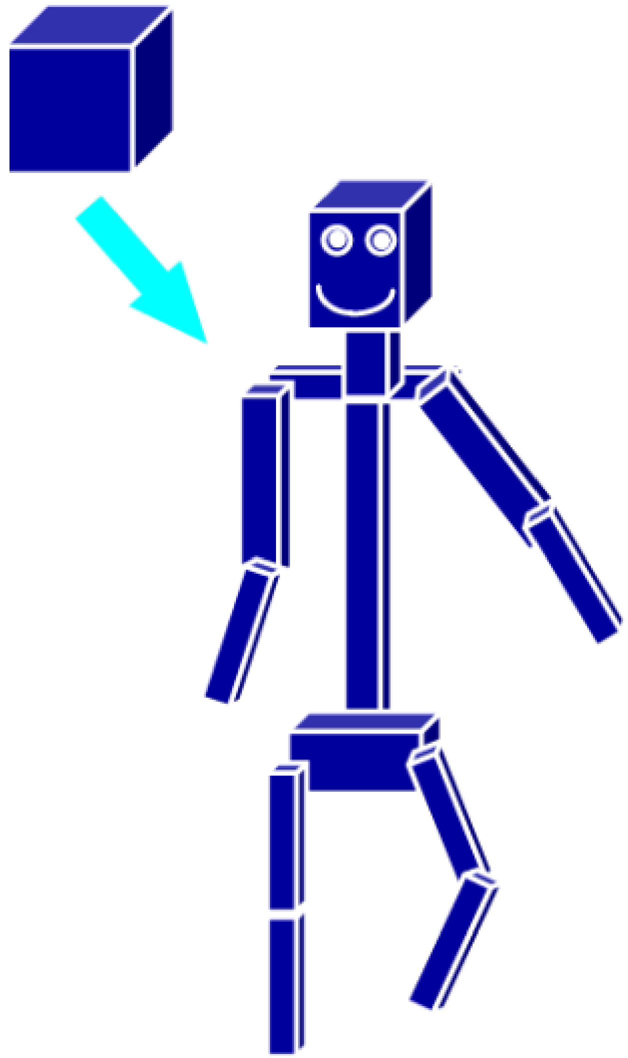
- Instead of looking up one shadow map pixel, look up several
- Perform depth test for each shadow map pixel
- Compute percentage of lit shadow map pixels



Animation

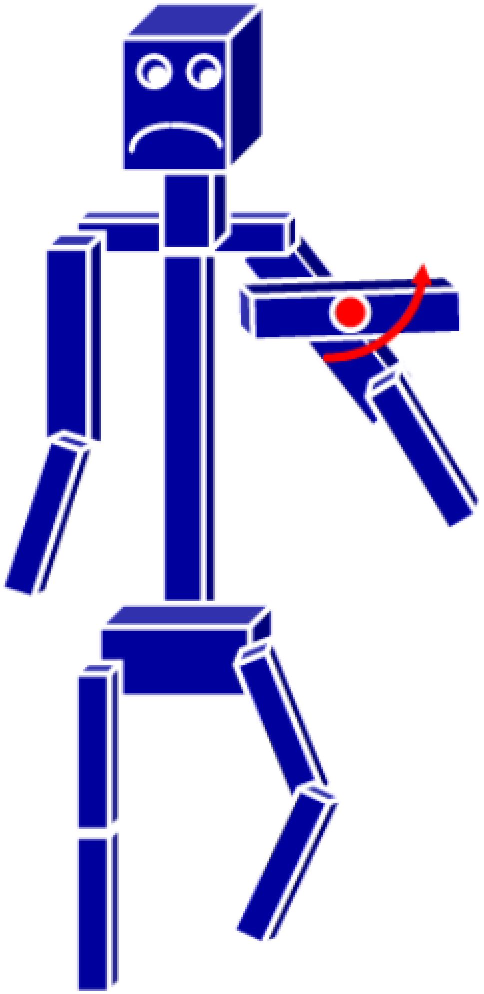
Modeling with Transformations

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

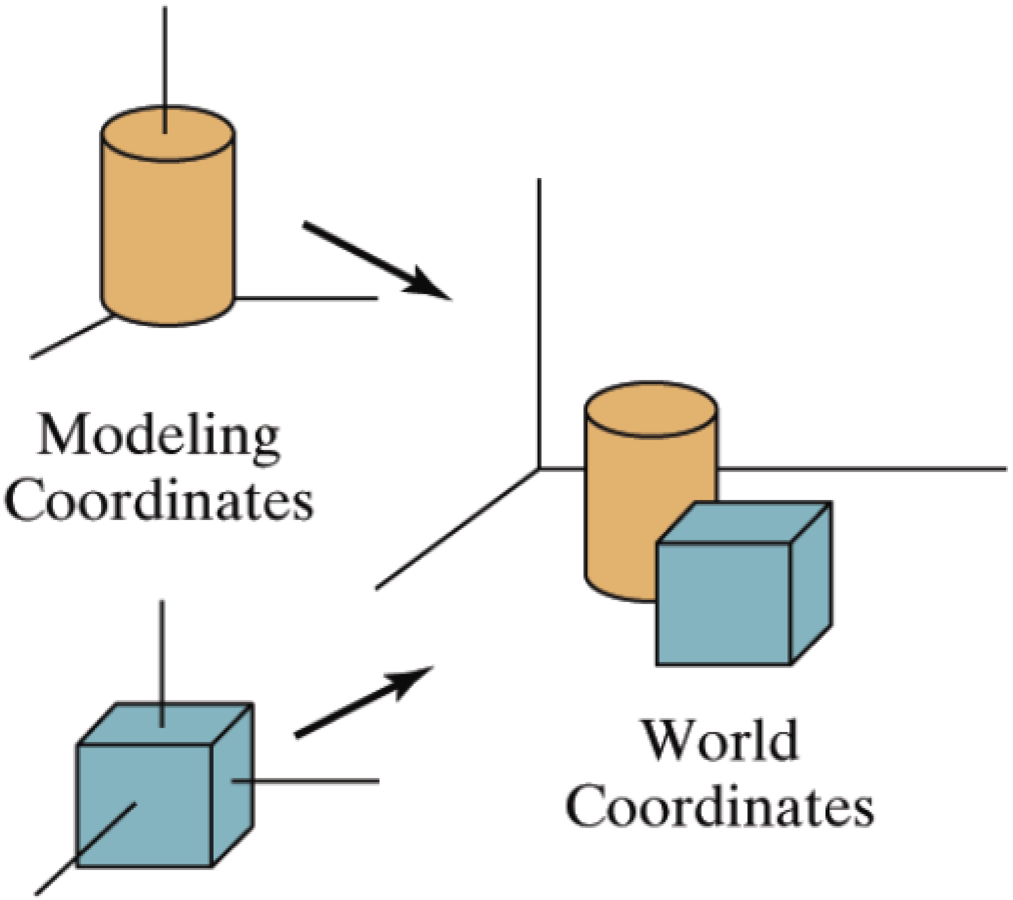


Modeling with Transformations

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

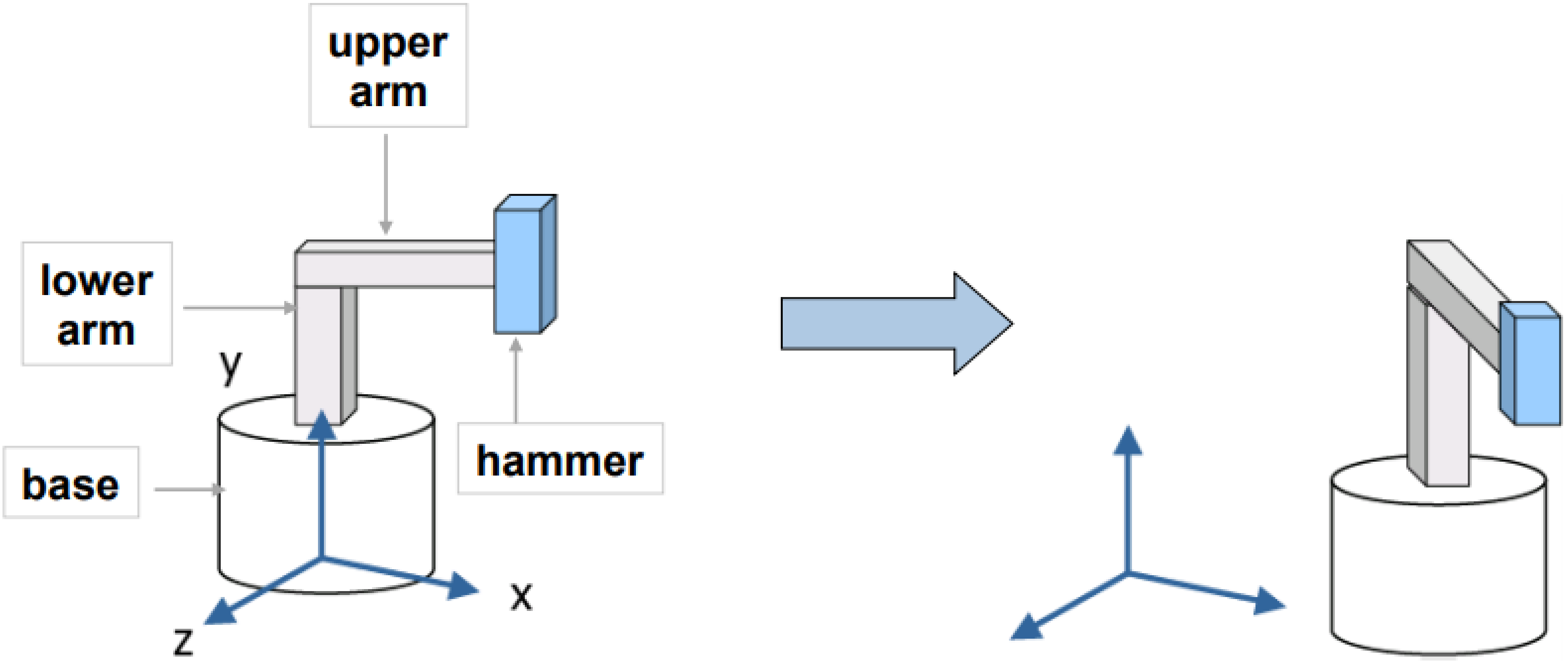


Model to World Space



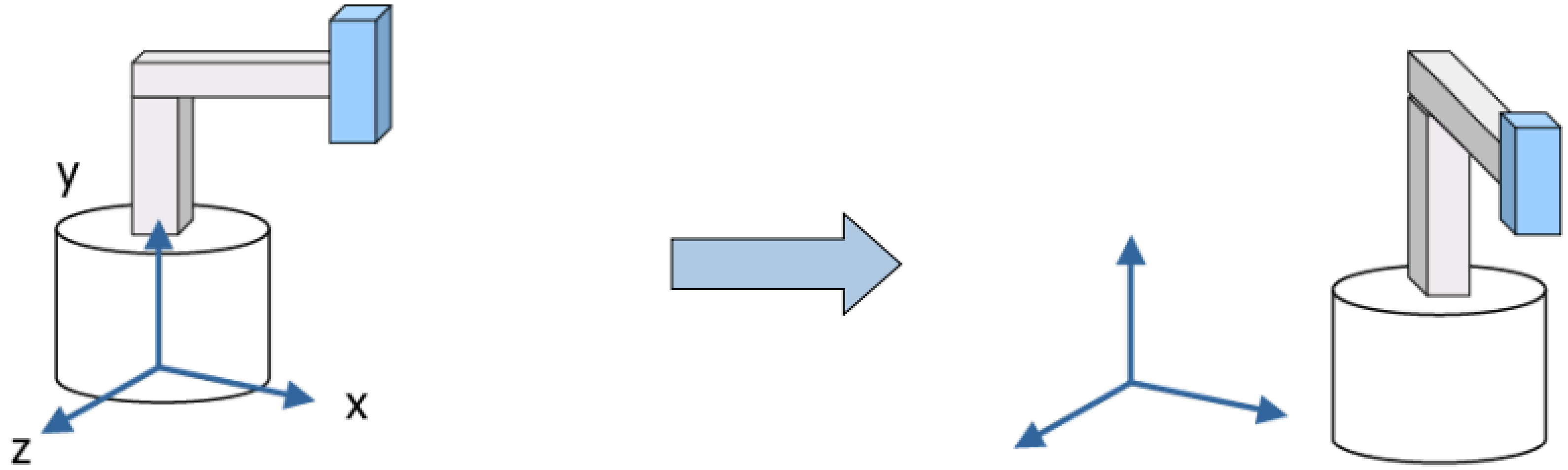
Model → World

- Position and orient the robot hammer in world space



Model → World

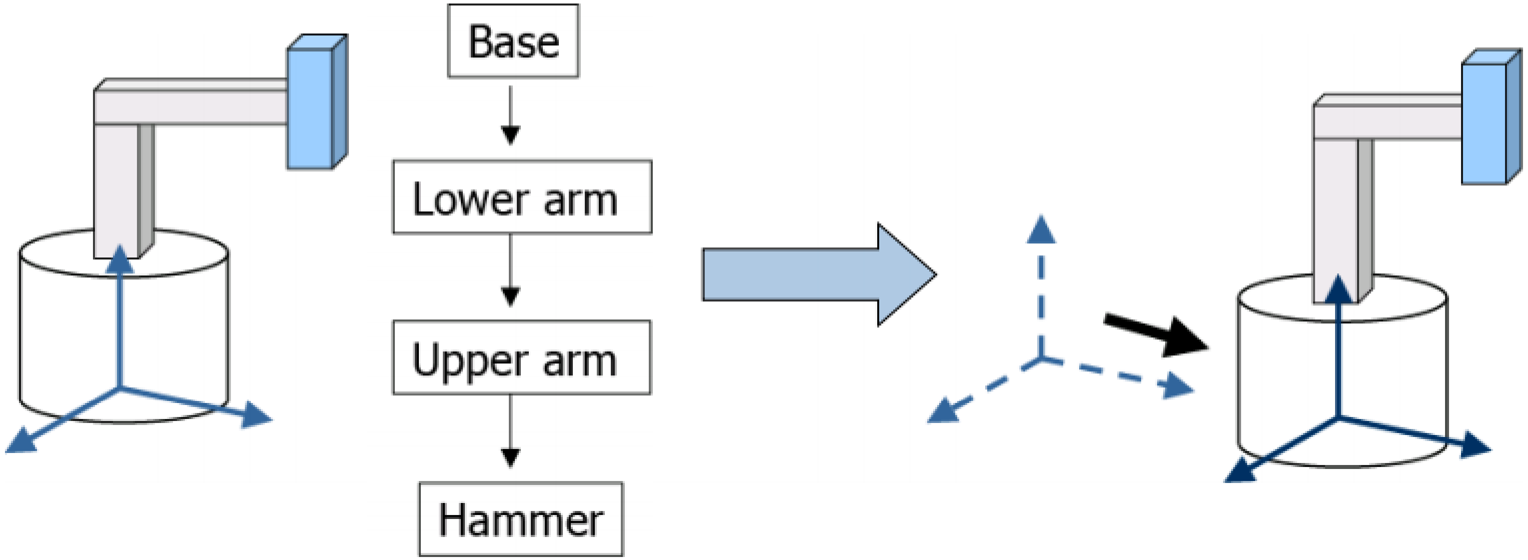
- Each part of the object is transformed independently relative to the origin



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)
...

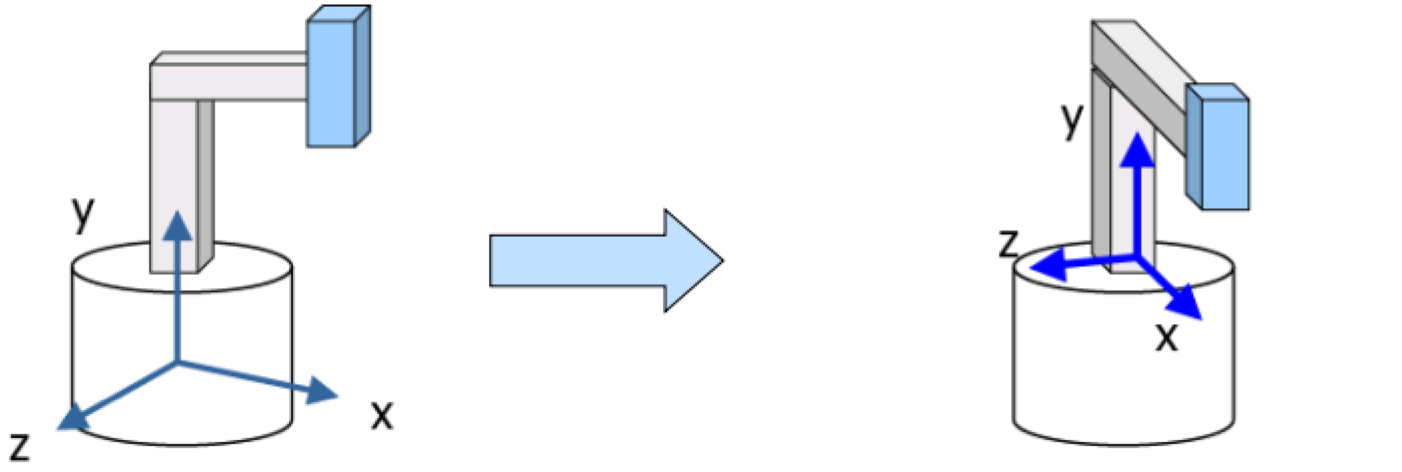
Model → World

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



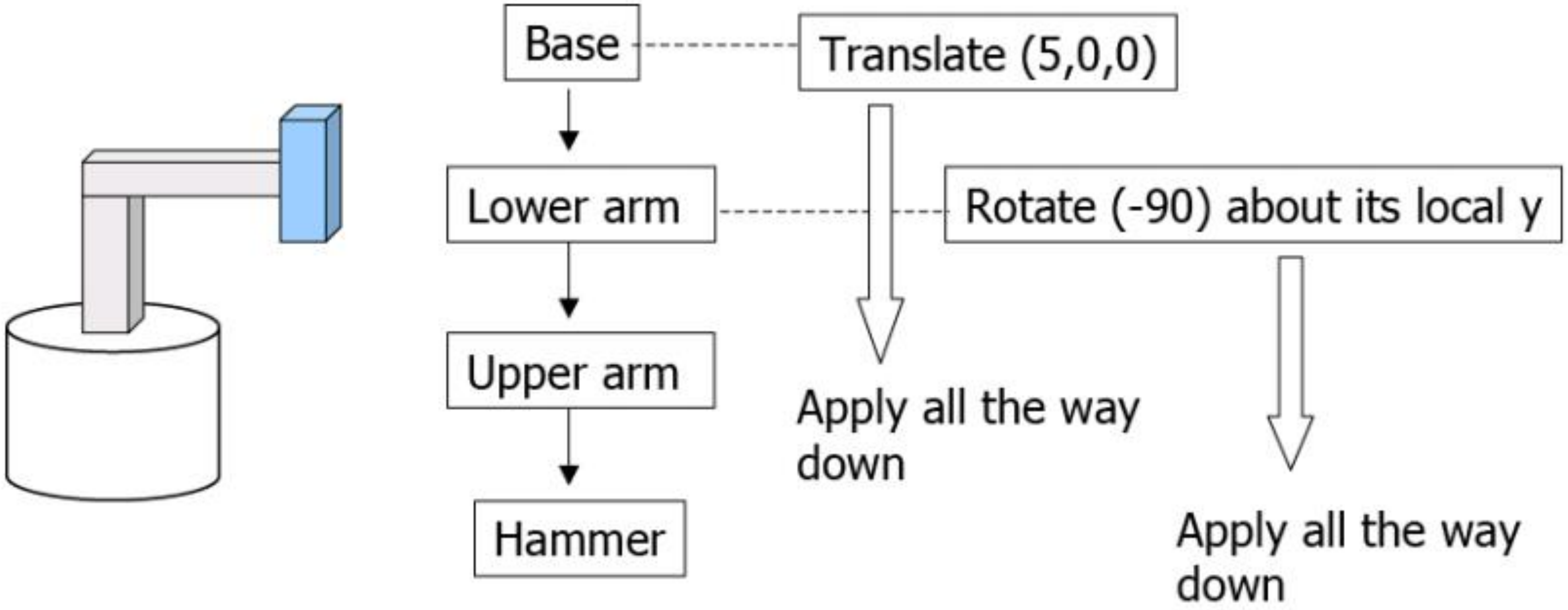
Step 1: Translate *base and its descendants* by (5,0,0)

Relative Transformations



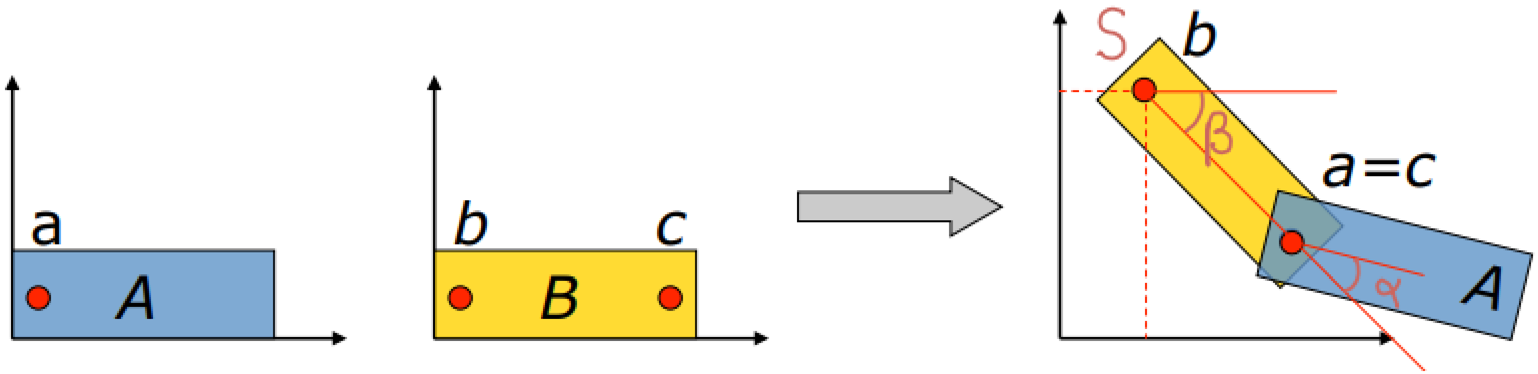
Step 2: Rotate lower arm and its descendants
by -90 degrees about local y axis

Hierarchical Transforms



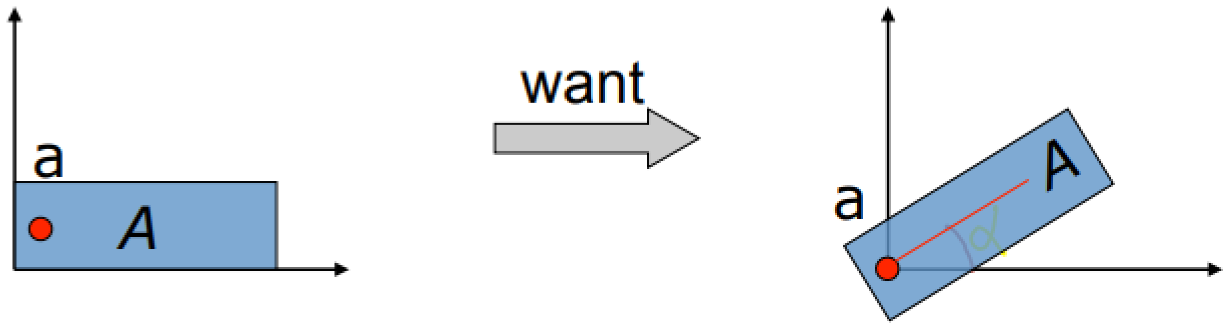
Making an Articulated Arm

- 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)



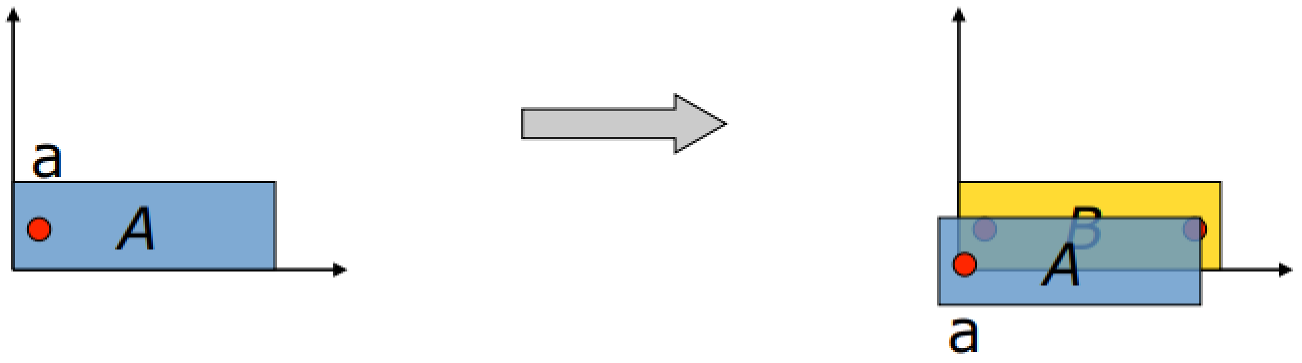
Making an Arm: Step 1

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



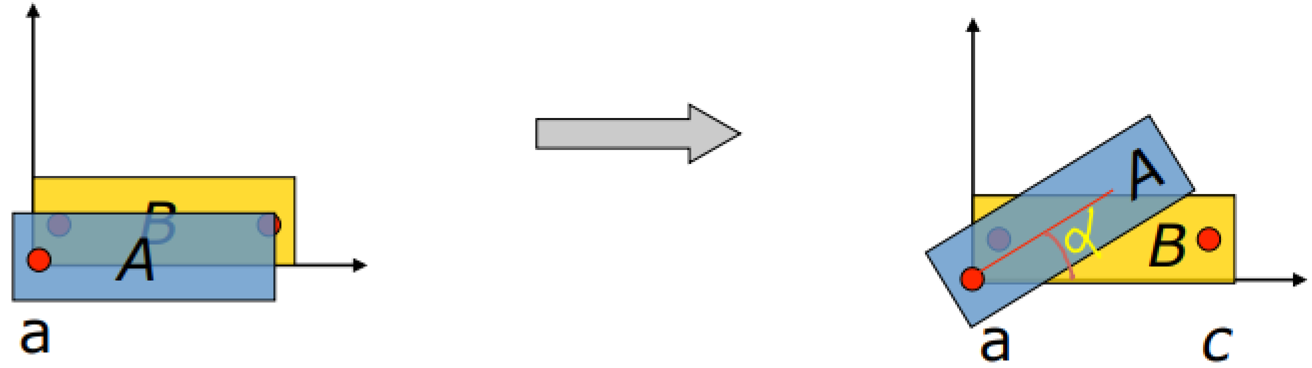
Making an Arm: Step 2

- Translate by –a, bringing a to the origin



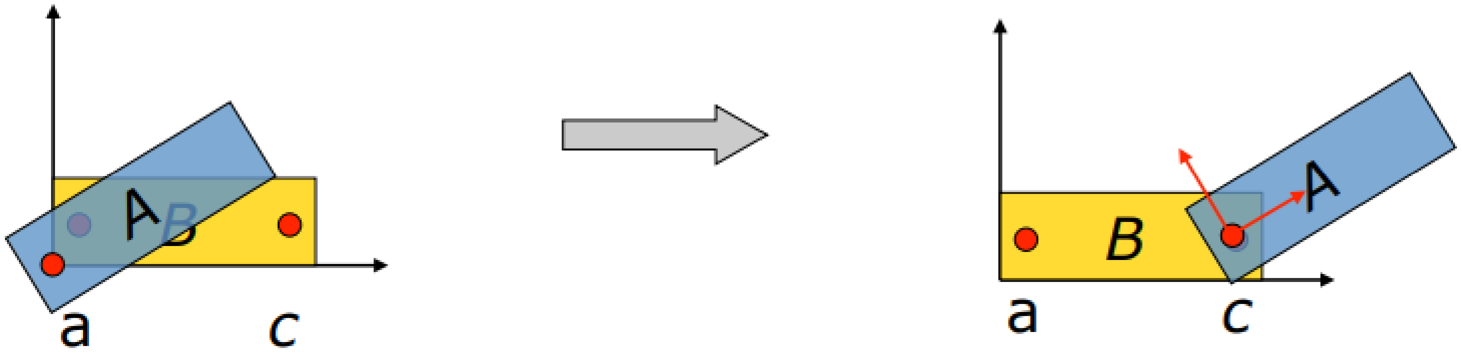
Making an Arm: Step 3

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



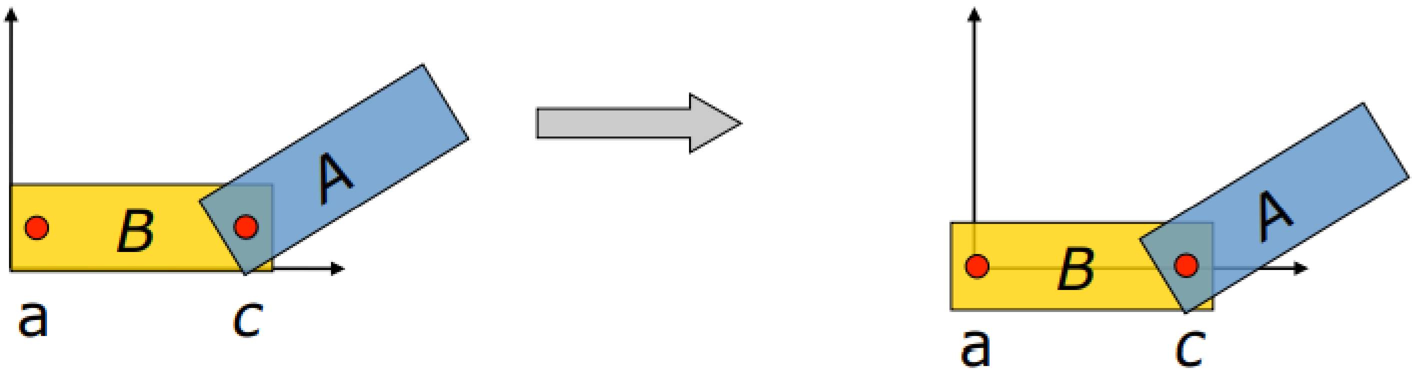
Making an Arm: Step 4

- Translate A to form the elbow joint a c



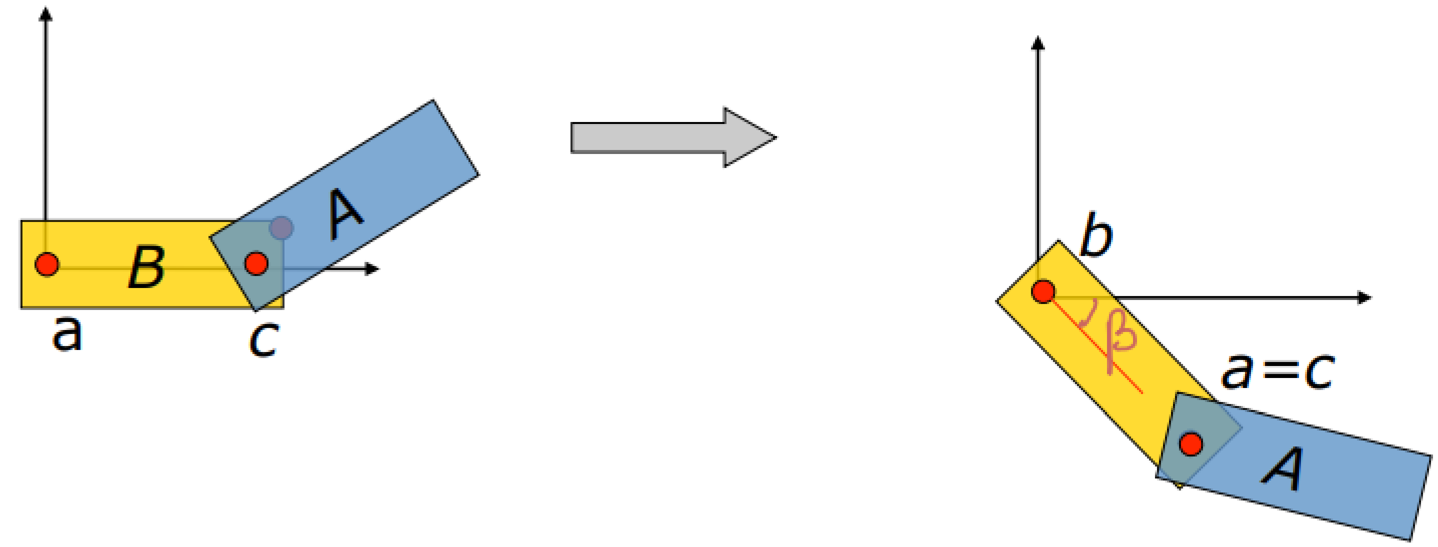
Making an Arm: Step 5

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



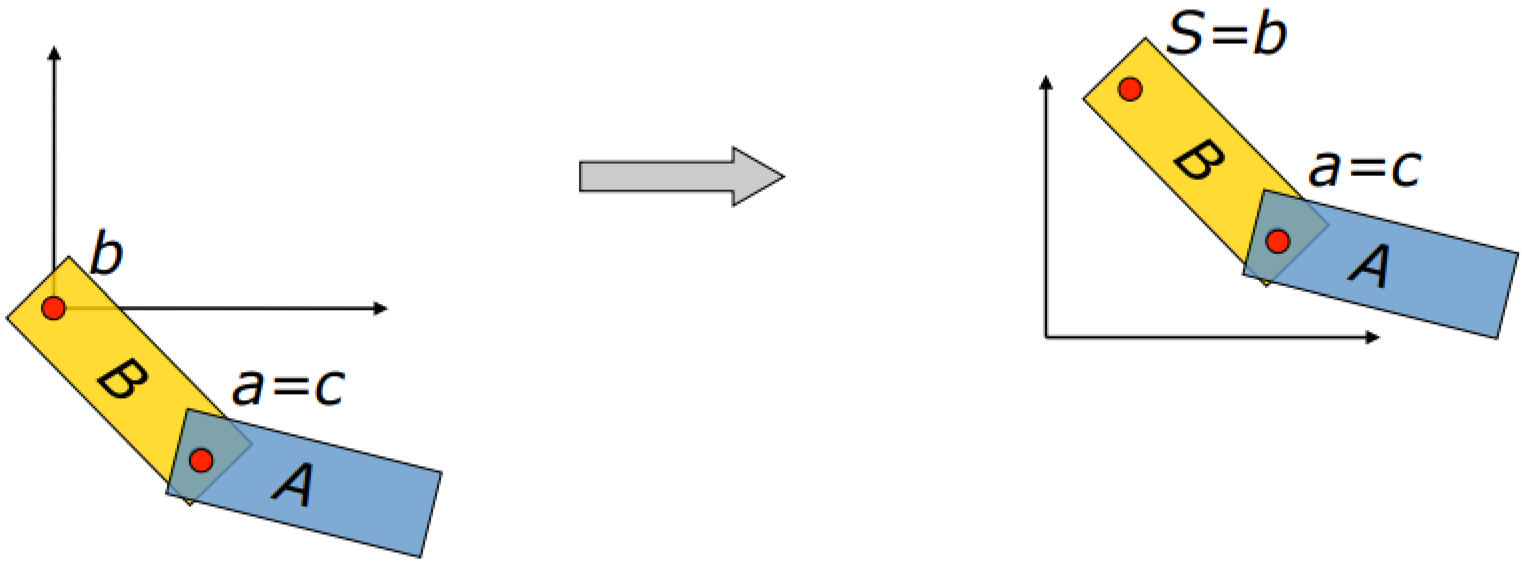
Making an Arm: Step 6

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



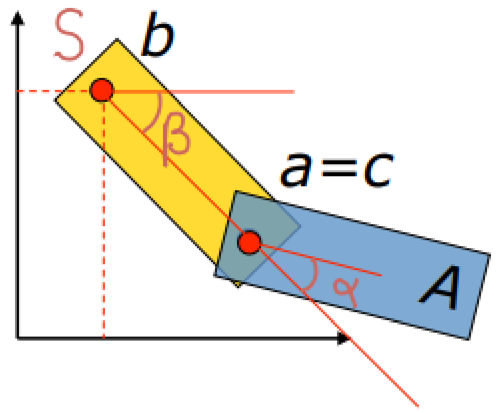
Making an Arm: Last Step

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

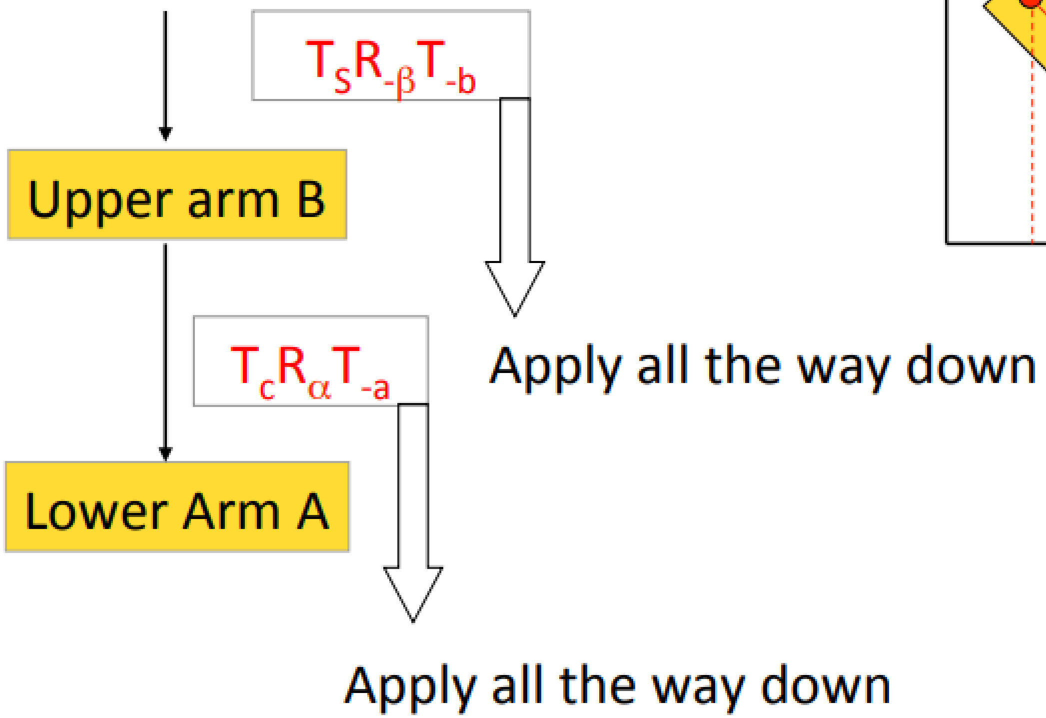


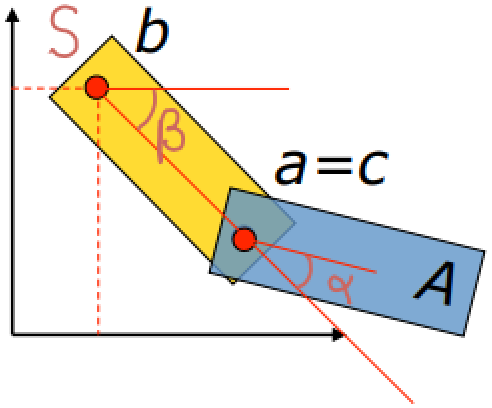
Parametrization

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

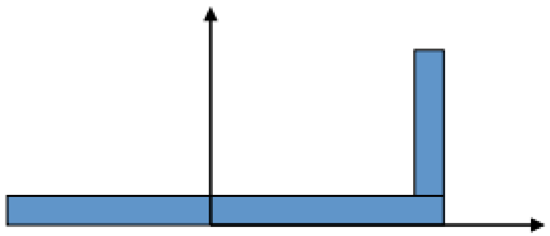


Hierarchical Transforms

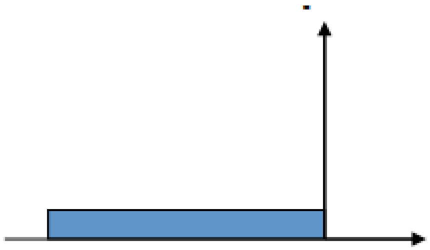




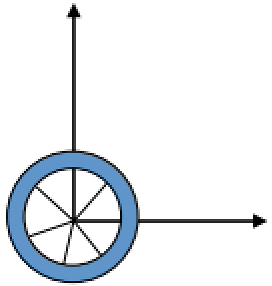
Model Construction



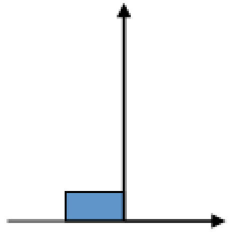
DrawBody



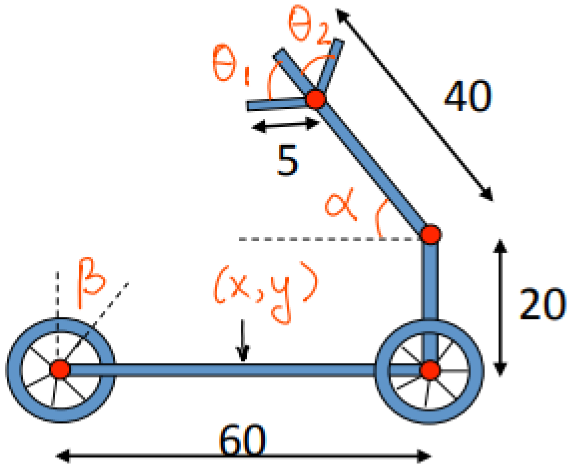
DrawHandle



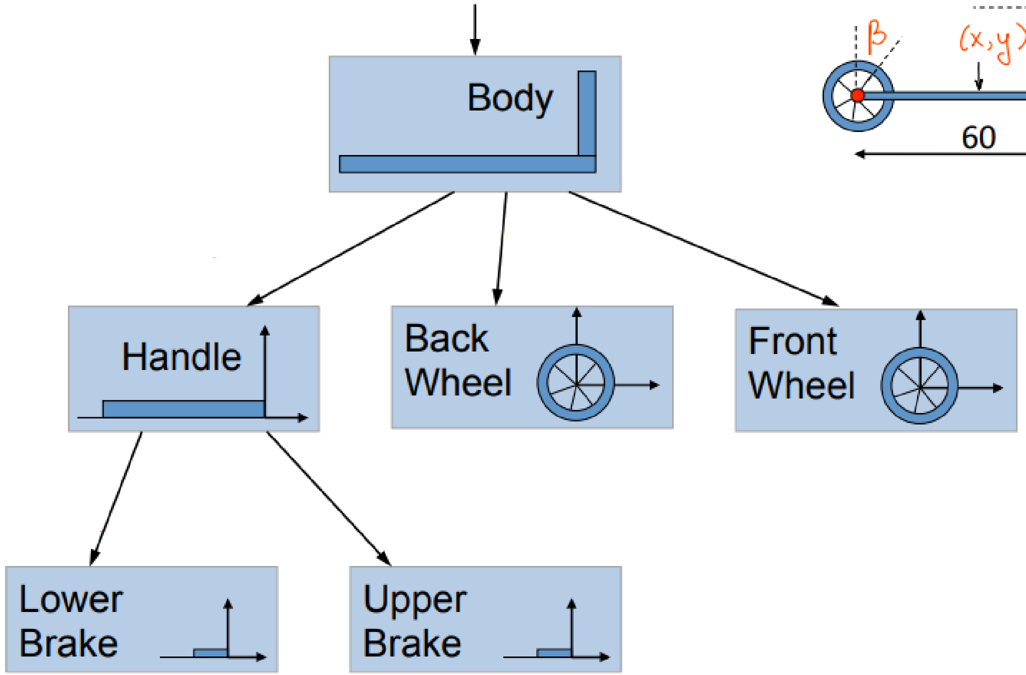
DrawWheel



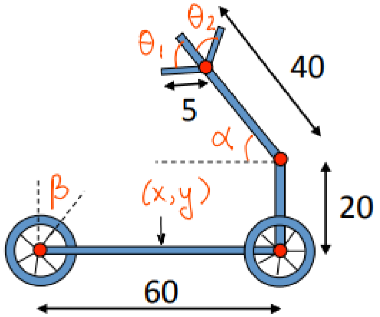
DrawBrake



Scene Graph

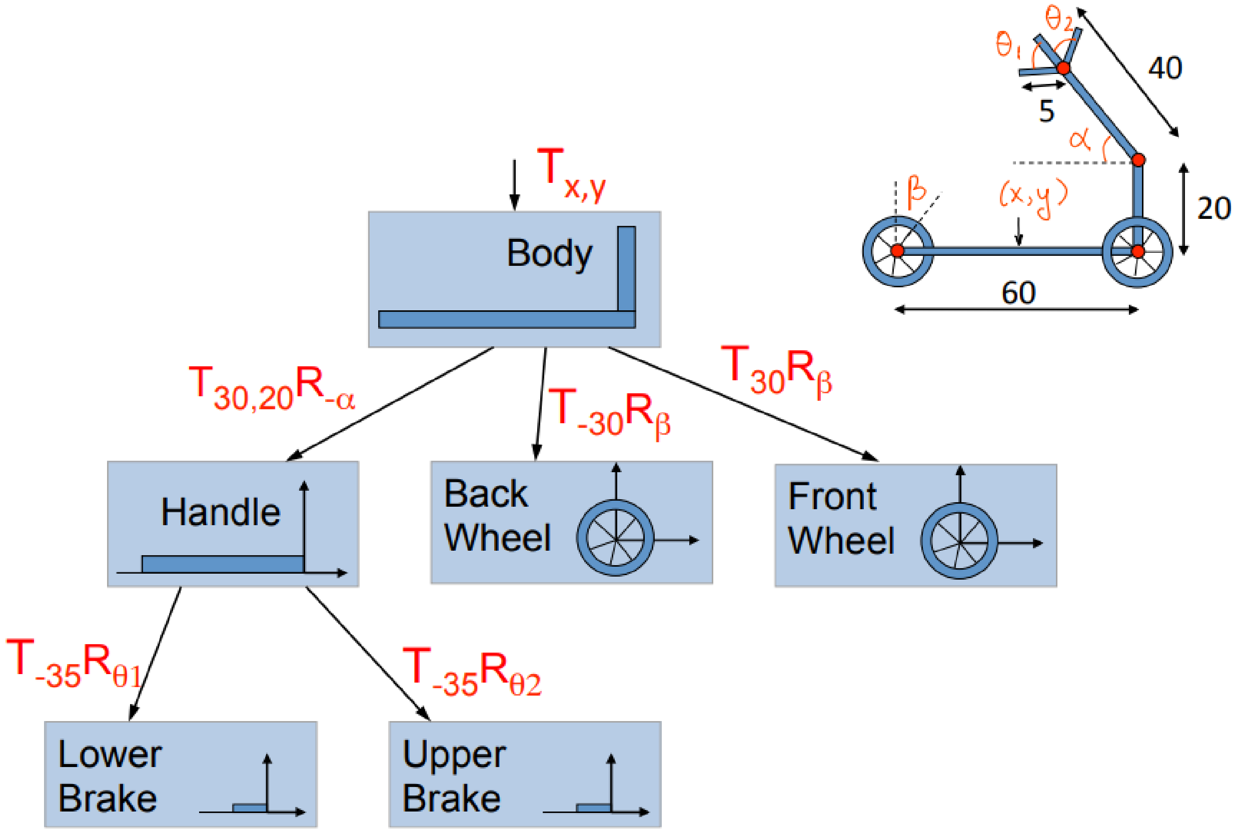


Body
Handle
Back Wheel
Front Wheel
Lower Brake
Upper Brake



40
5
20
60
(x,y)

Scene Graph



T x,y
Body
T 30,20 R -α
T -30 R β
T 30 R β
Handle
Back Wheel
Front Wheel
T -35 R θ1
T -35 R θ2
Lower Brake
Upper Brake
θ1
θ2
40
5
α
β
(x,y)
20
60

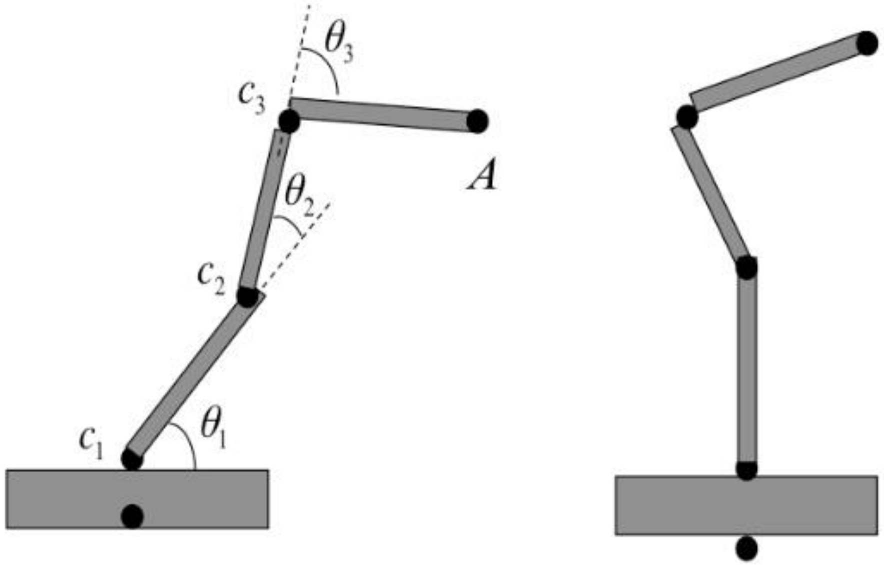
Scene Graph OpenGL 3.0+

```
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

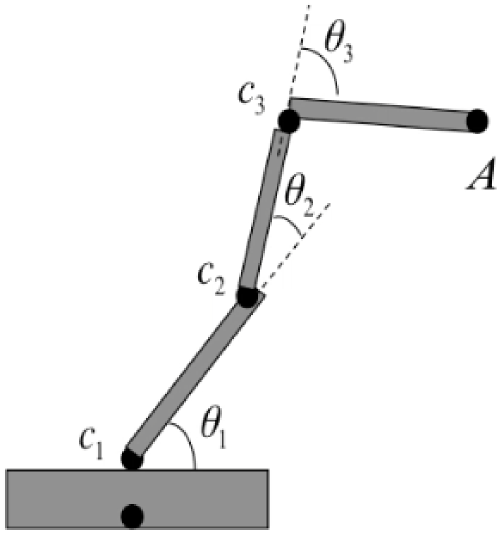


t=0

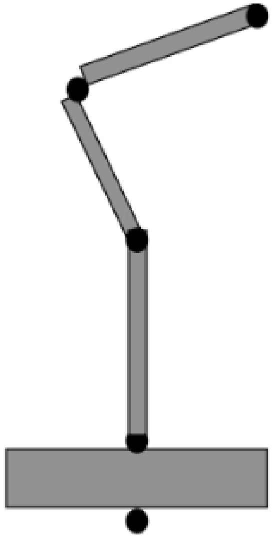
t=50ms

What’s the inbetween motion?

Keyframing



t=0

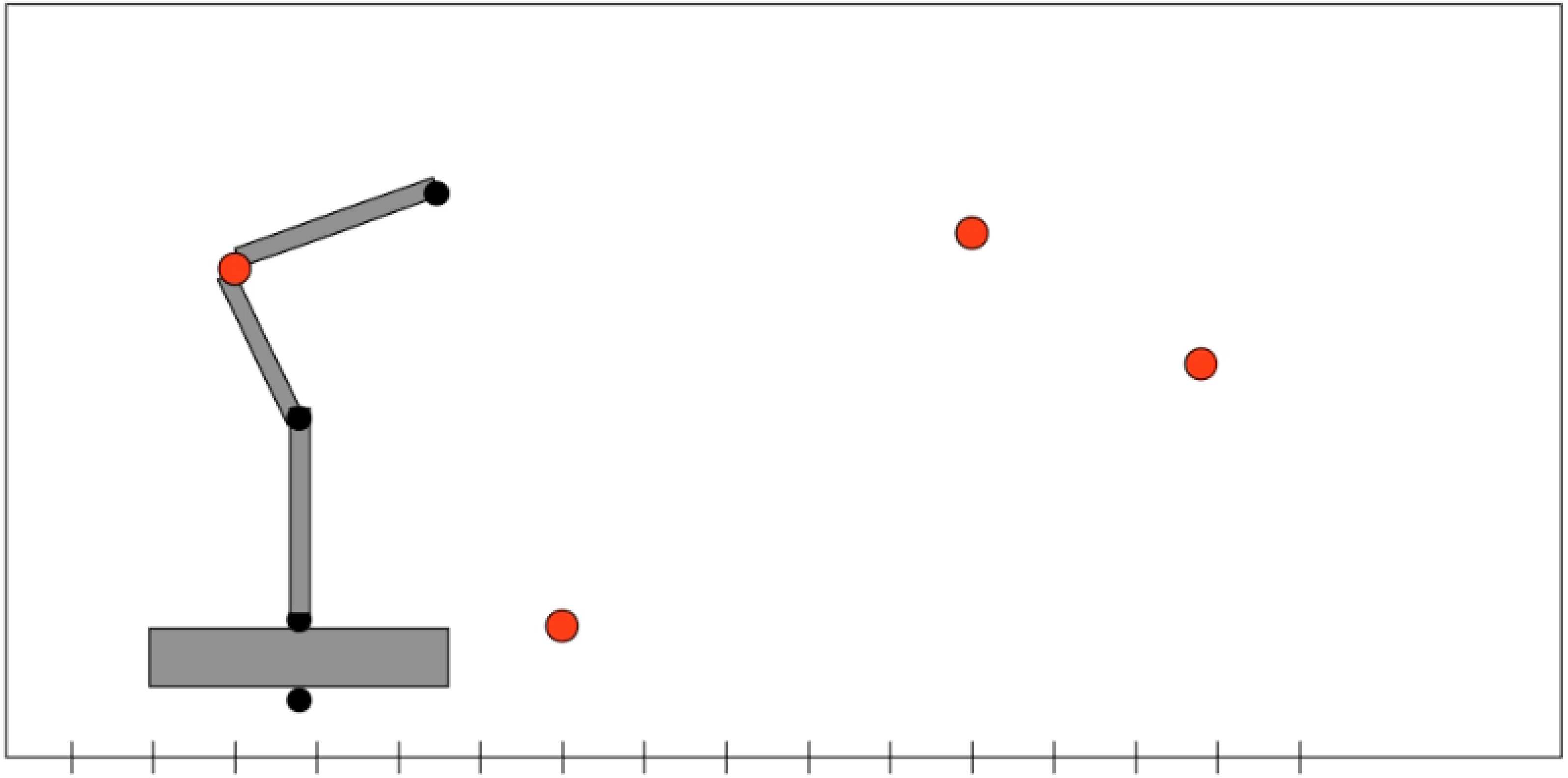


t=50ms

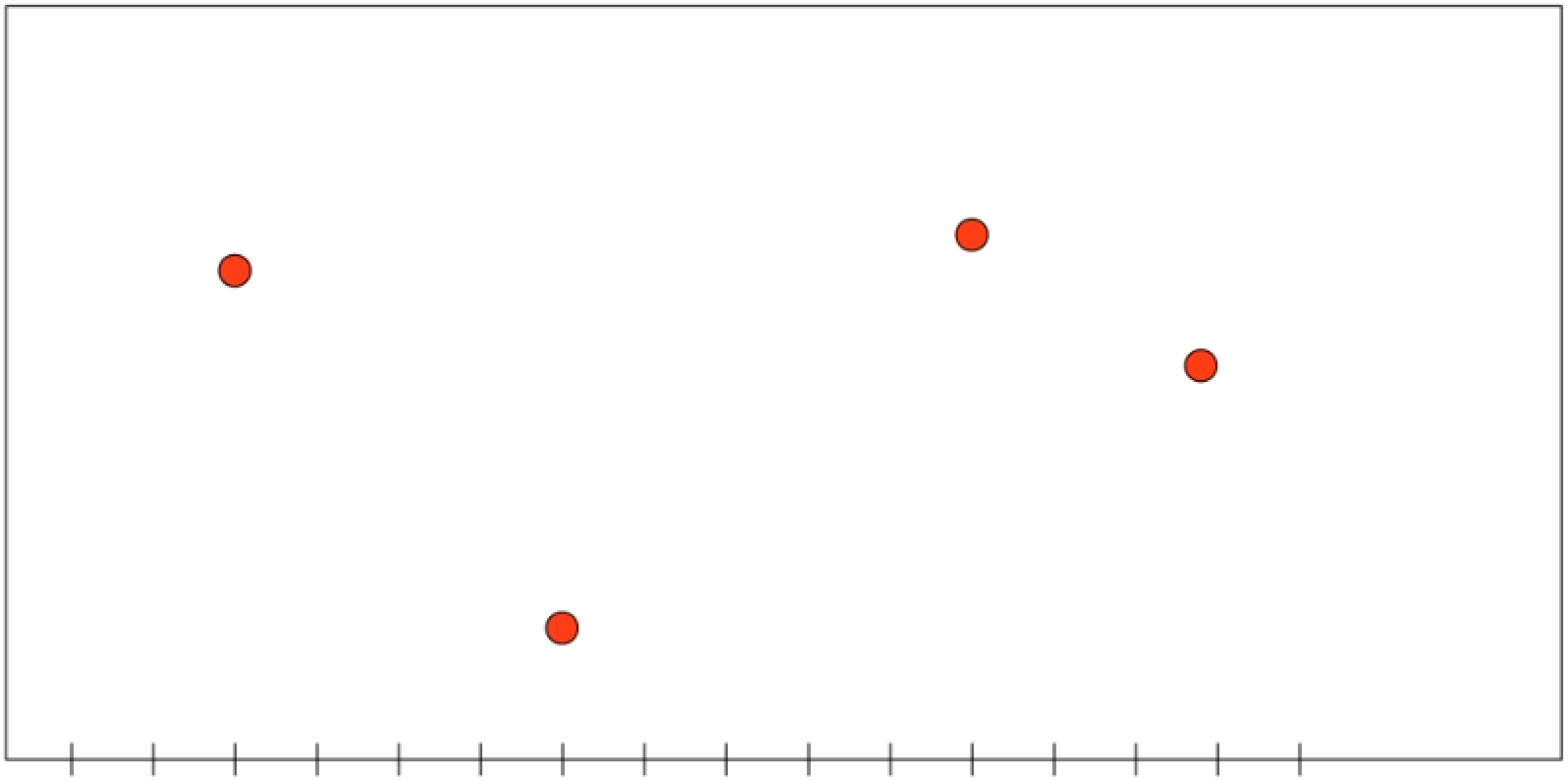
60 sec animation

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

What's the inbetween motion?

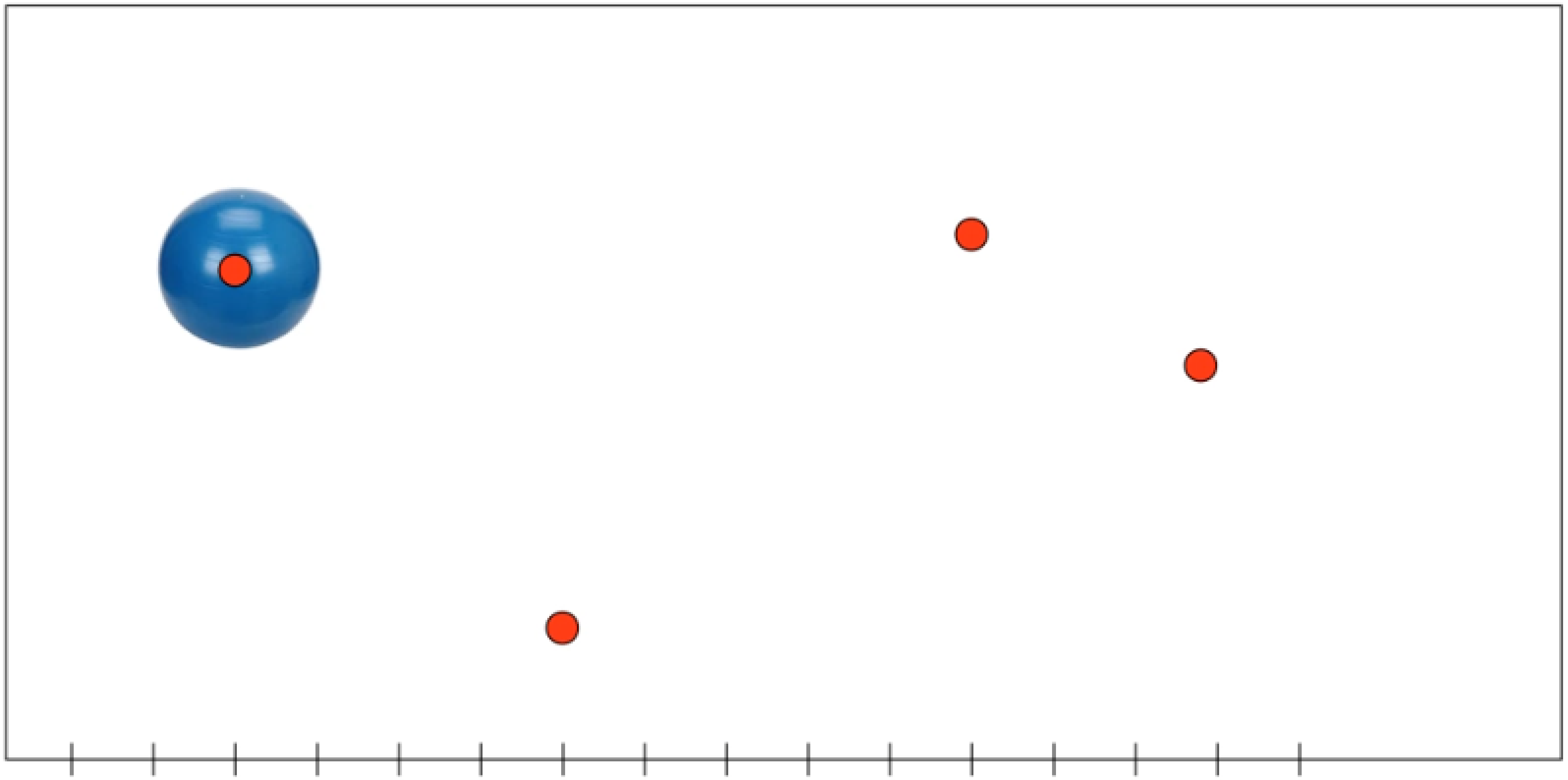


Mathematical problem: Given a set of points, what are the most reasonable points *in between*?



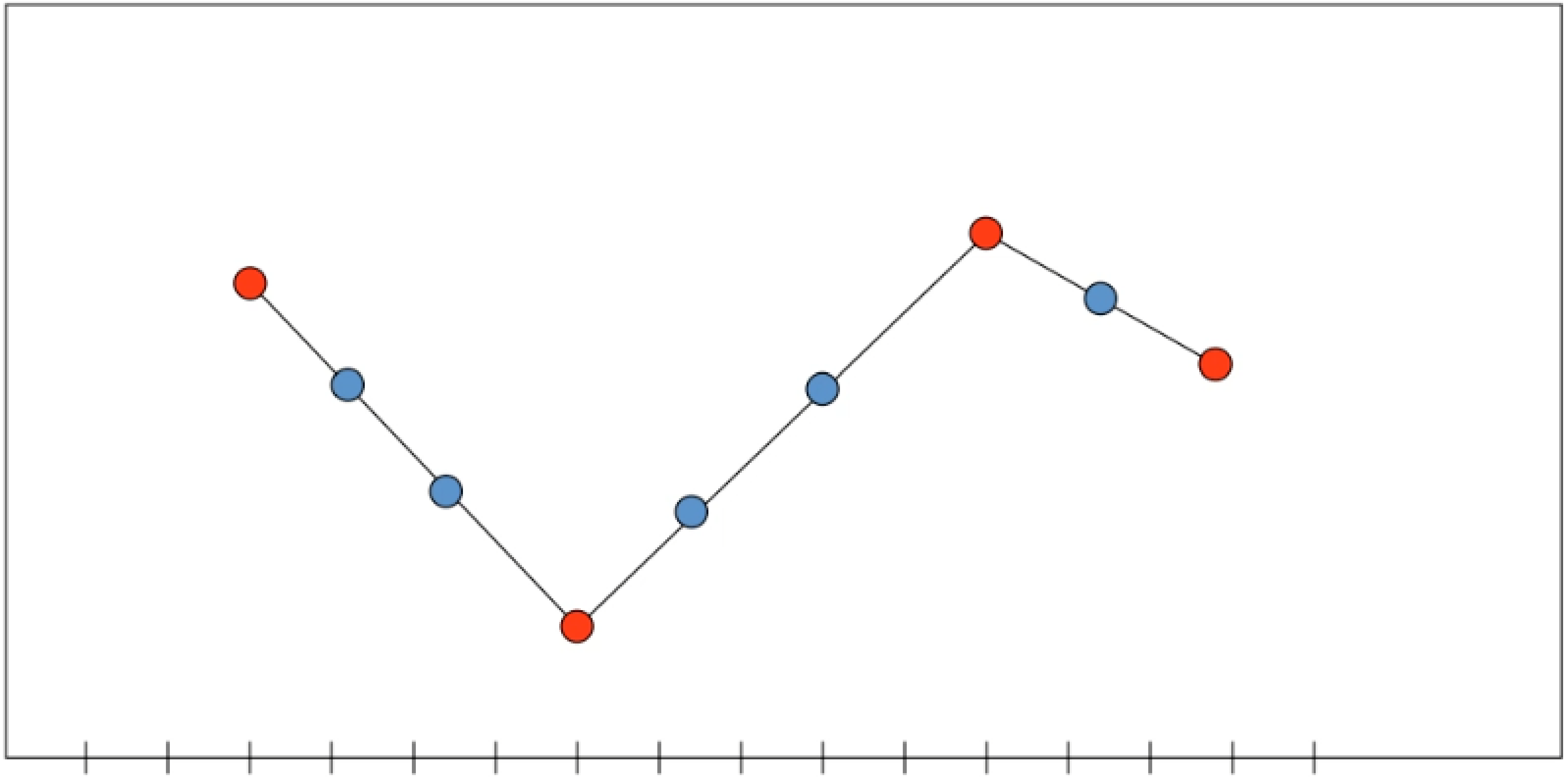
Mathematical problem: Given a set of points, what are the most reasonable points *in between*?

Interpolation

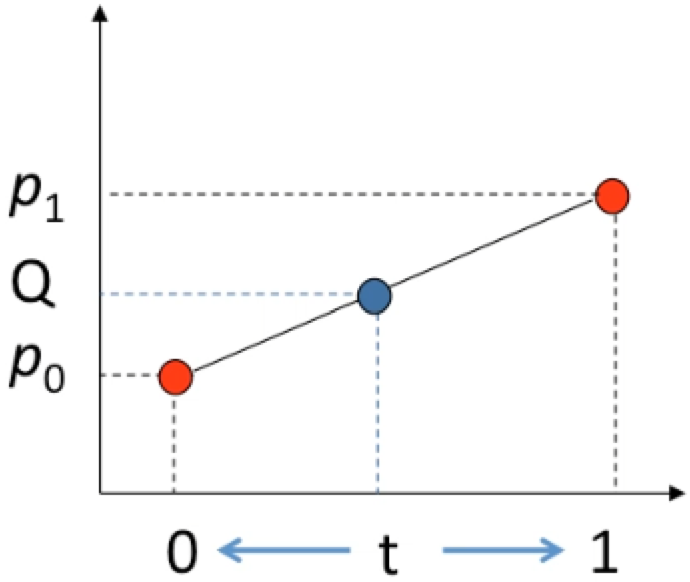


Mathematical problem: Given a set of points,
what are the most reasonable points *in between*?

Linear Interpolation



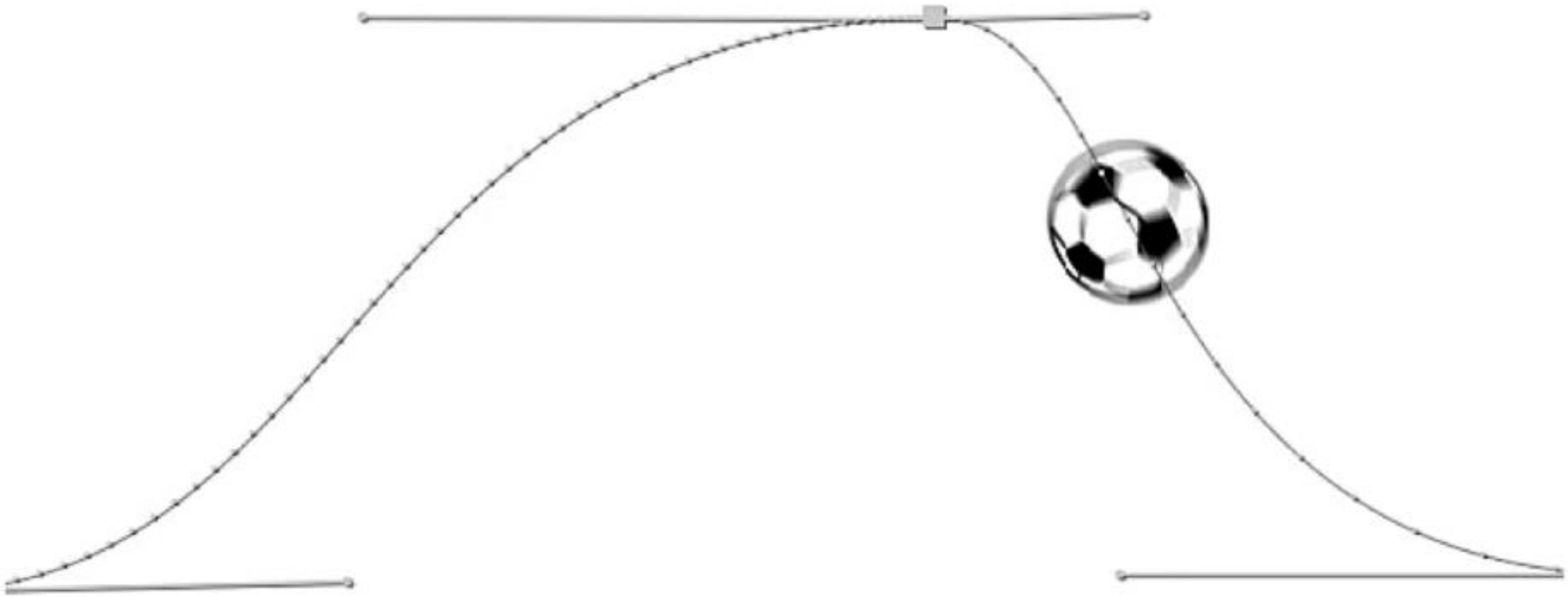
Linear Interpolation



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

Linear Interpolation: Limitations

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



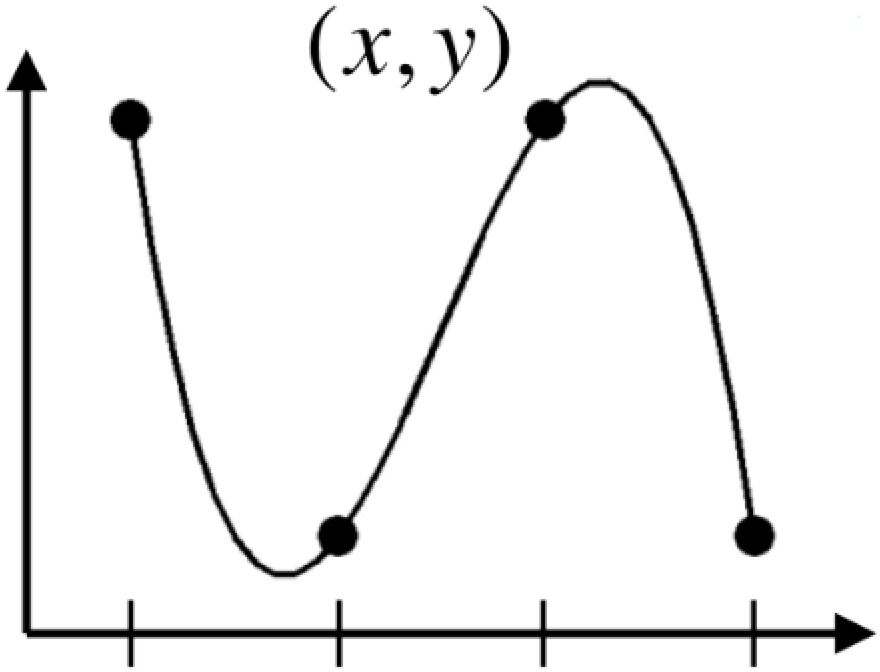
Parametric Curves

- Define a continuous smooth curve f passing through the data points

- Explicit form y = f(x)

- Implicit form f(x, y) = 0

- <u>Parametric form x = f(t), y = g(t)</u>



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

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

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

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

$$\boxed{0 \le t \le 1}$$

Cubic Curves

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

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

$$\boxed{0 \le t \le 1}$$

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

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

- Vector Form:

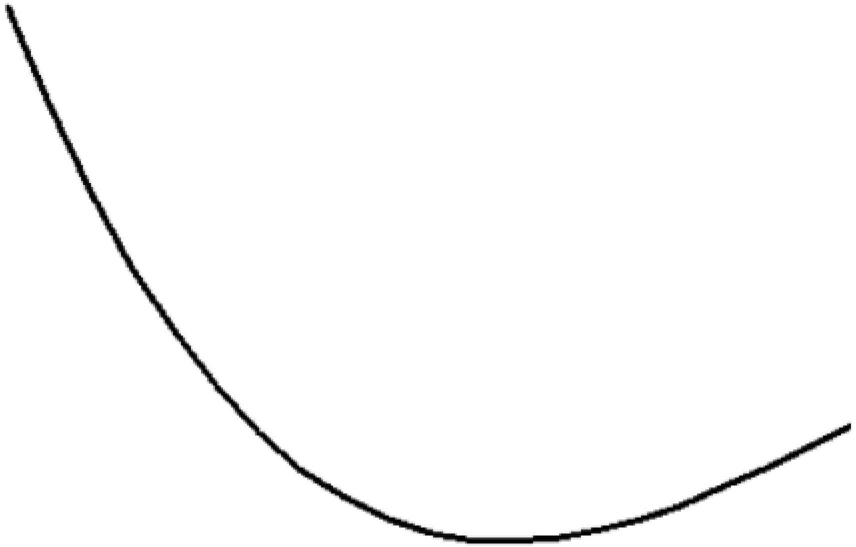
$$\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}$$

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

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

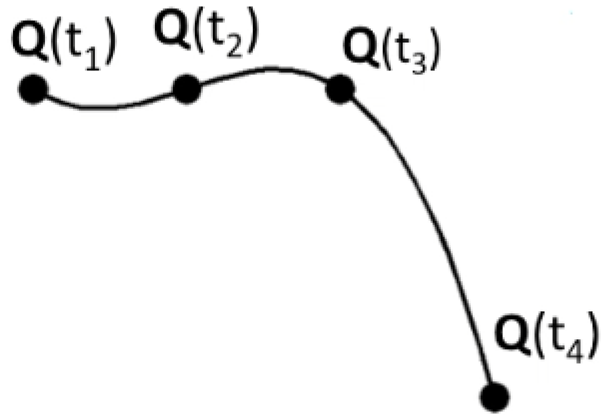
- How many constraints do we need to determine a cubic curve?

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

Constraints on the Cubics

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

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

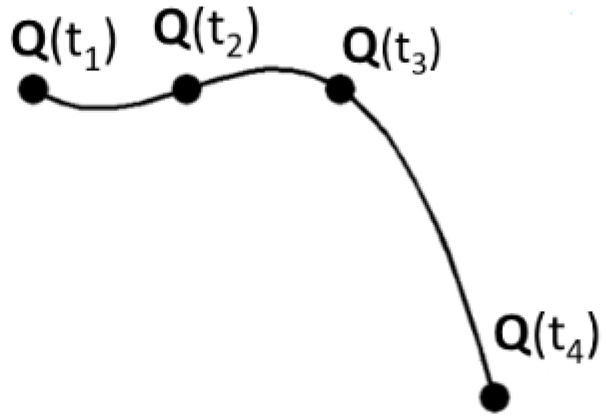


Constraints on the Cubics

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

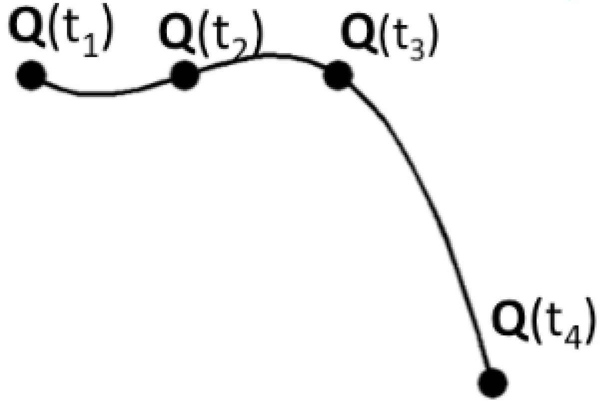
$$\mathbf{Q}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

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



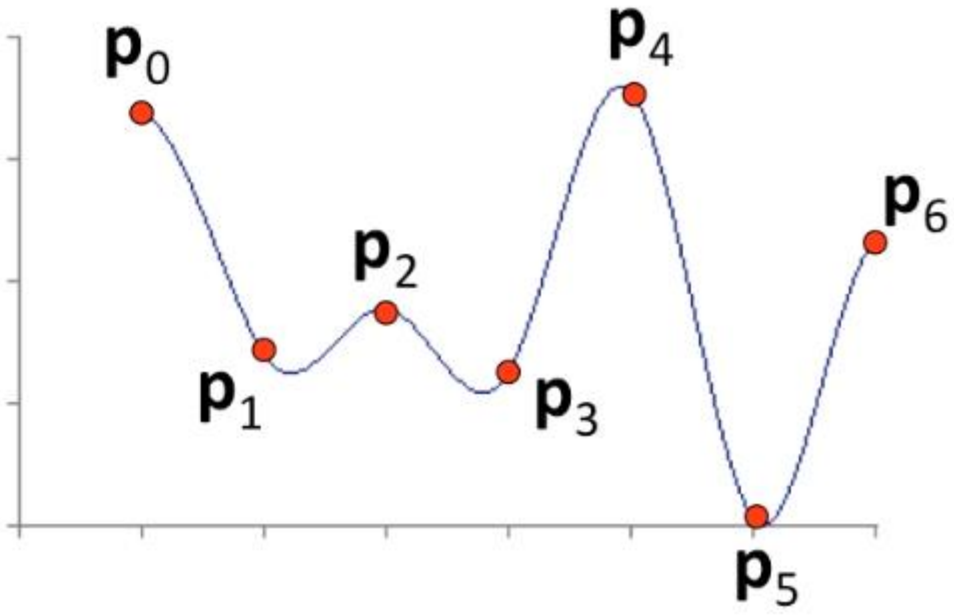
Natural Cubic Curves

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

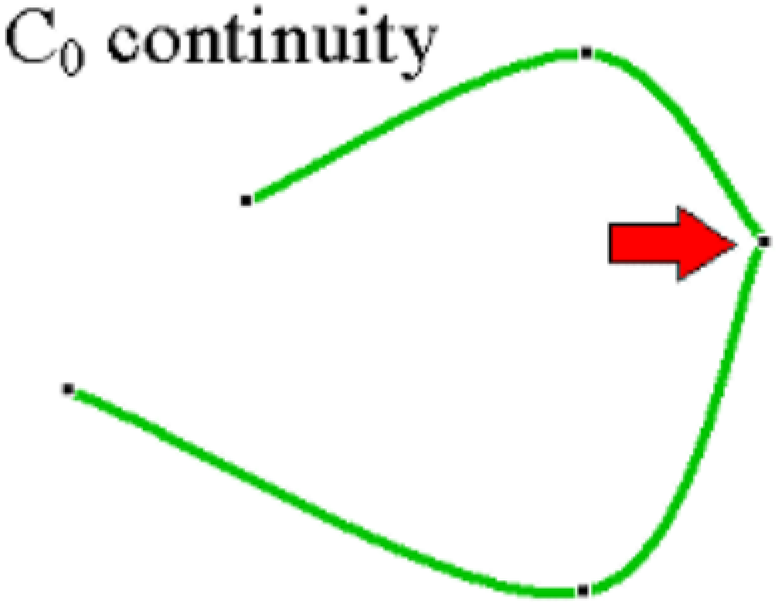


Natural Cubic Spline

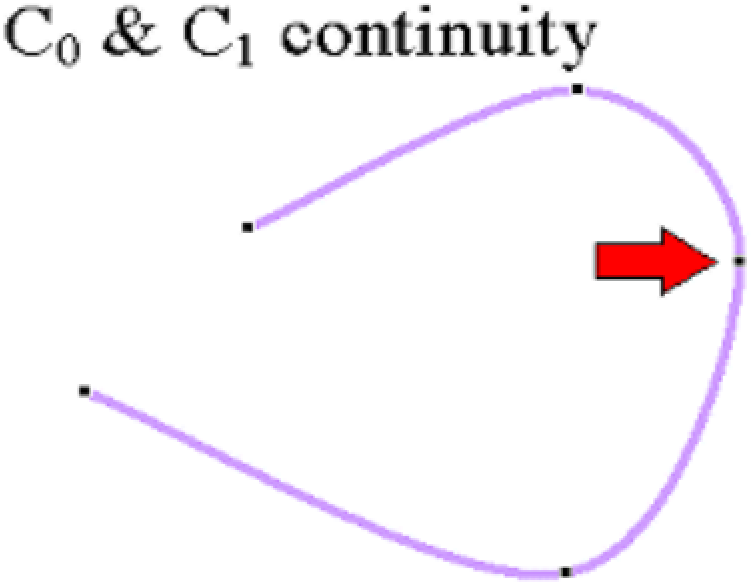
- A **spline** is a curve that is piecewise-defined and is smooth at the places where the pieces connect



Continuity



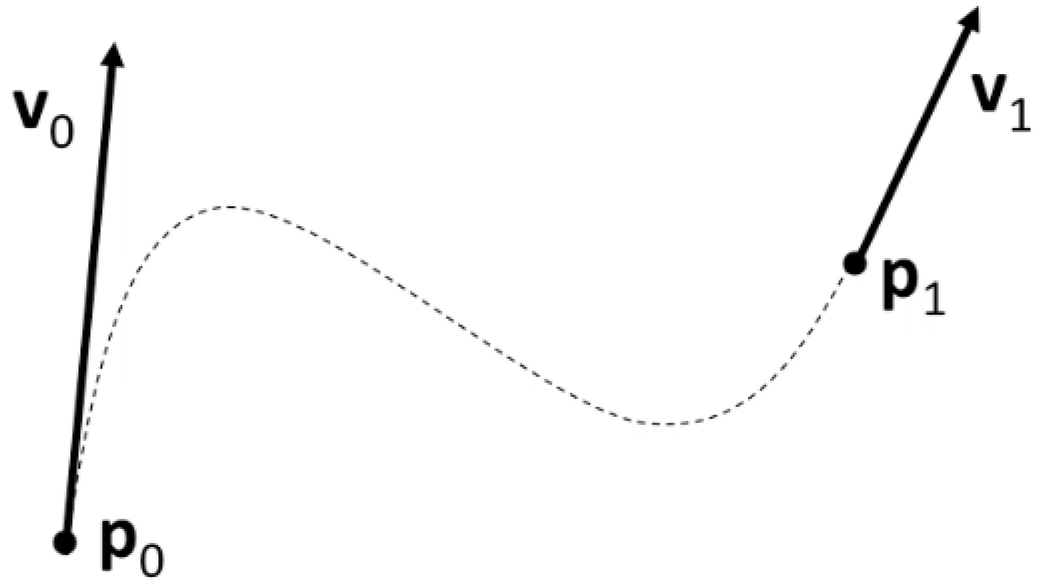
Positions of splines align



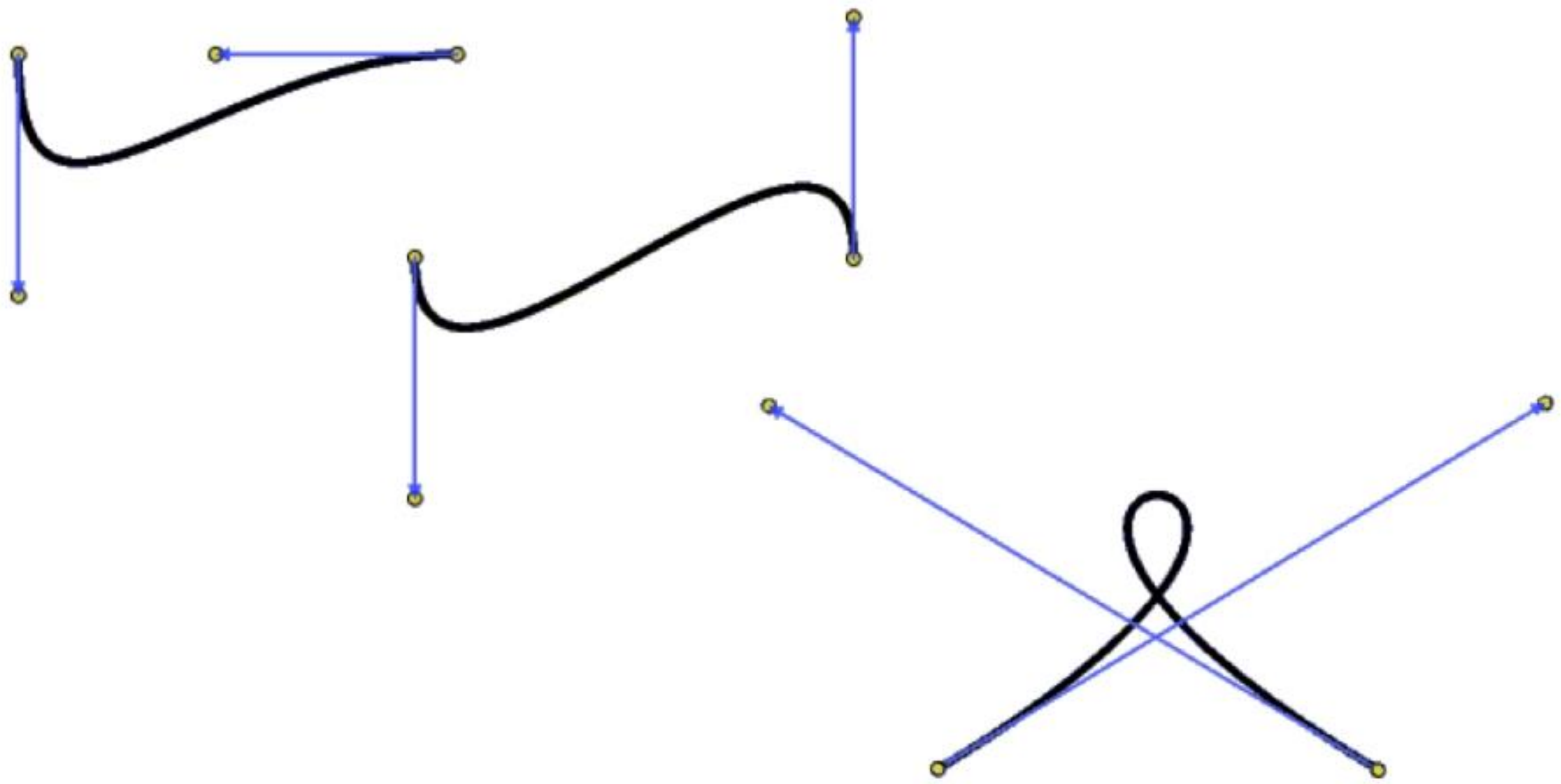
Positions and tangents of splines align

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)

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

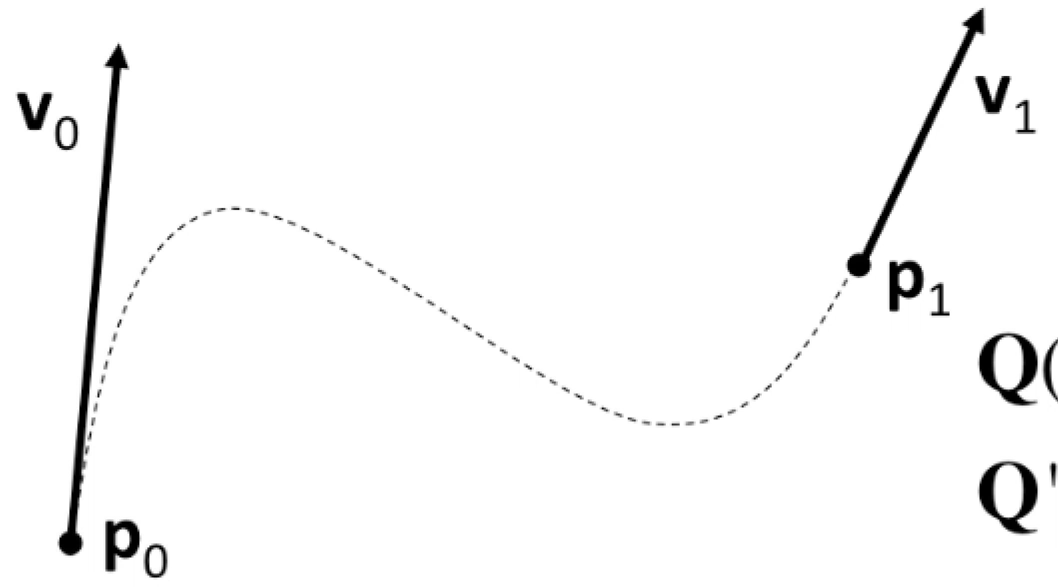
Tangents (Derivatives)

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

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

Hermite Curves

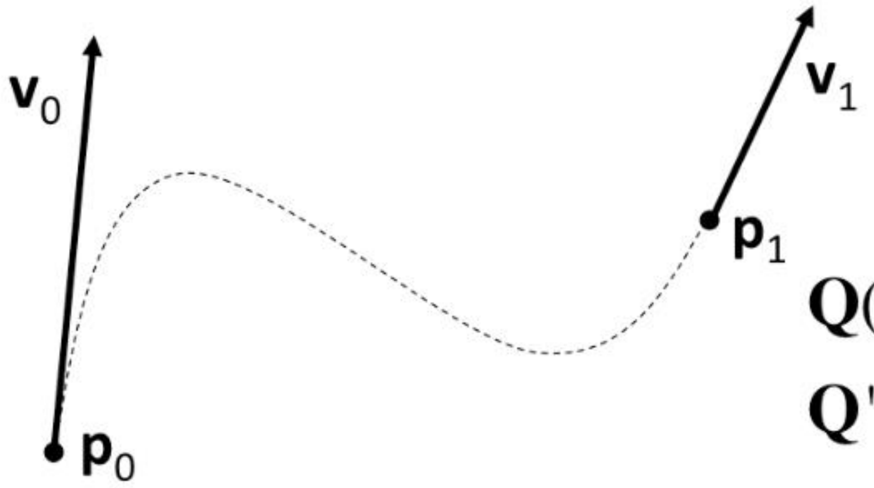
- The value of the curve is $\mathbf{Q}(0)=p_0$ at $t=0$ and $\mathbf{Q}(1)=p_1$ at $t=1$
- The derivative of the curve to be v_0 at $t=0$ and v_1 at $t=1$



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

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

Hermite Curves



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

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

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

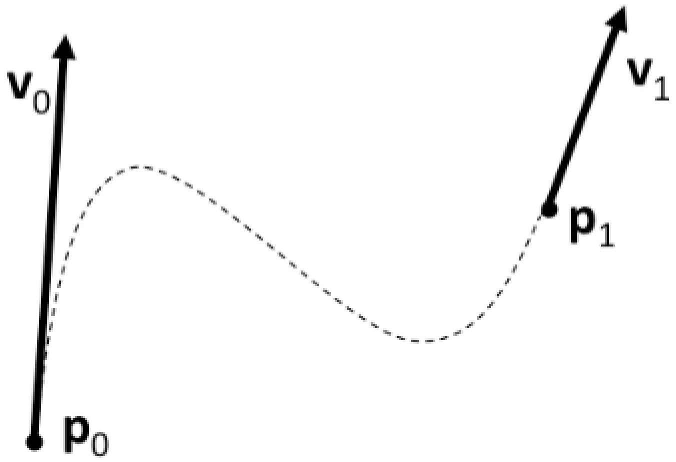
Hermite Curves

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

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

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

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



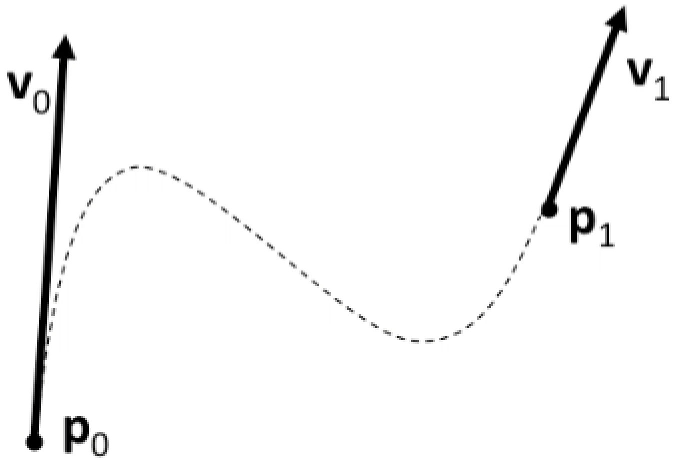
Hermite Curves

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

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

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

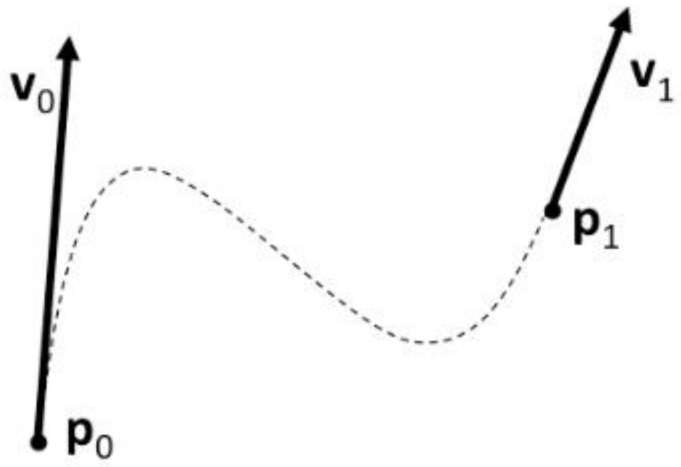
$$\mathbf{v}_1 = 3\mathbf{a} + 2\mathbf{b} + \mathbf{c}$$



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

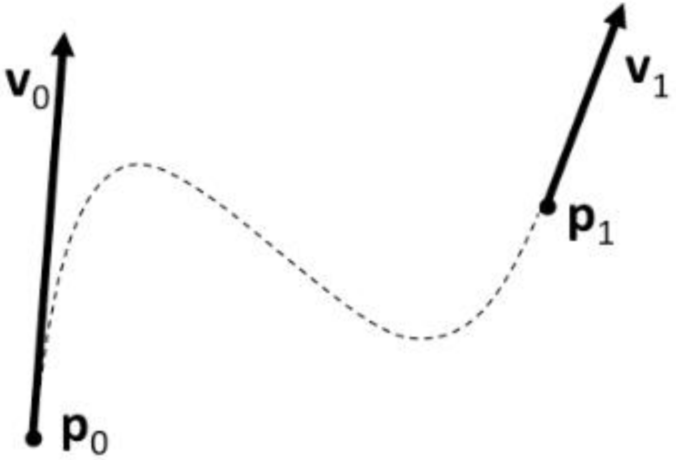
Hermite Interpolation

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



Hermite Interpolation

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



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