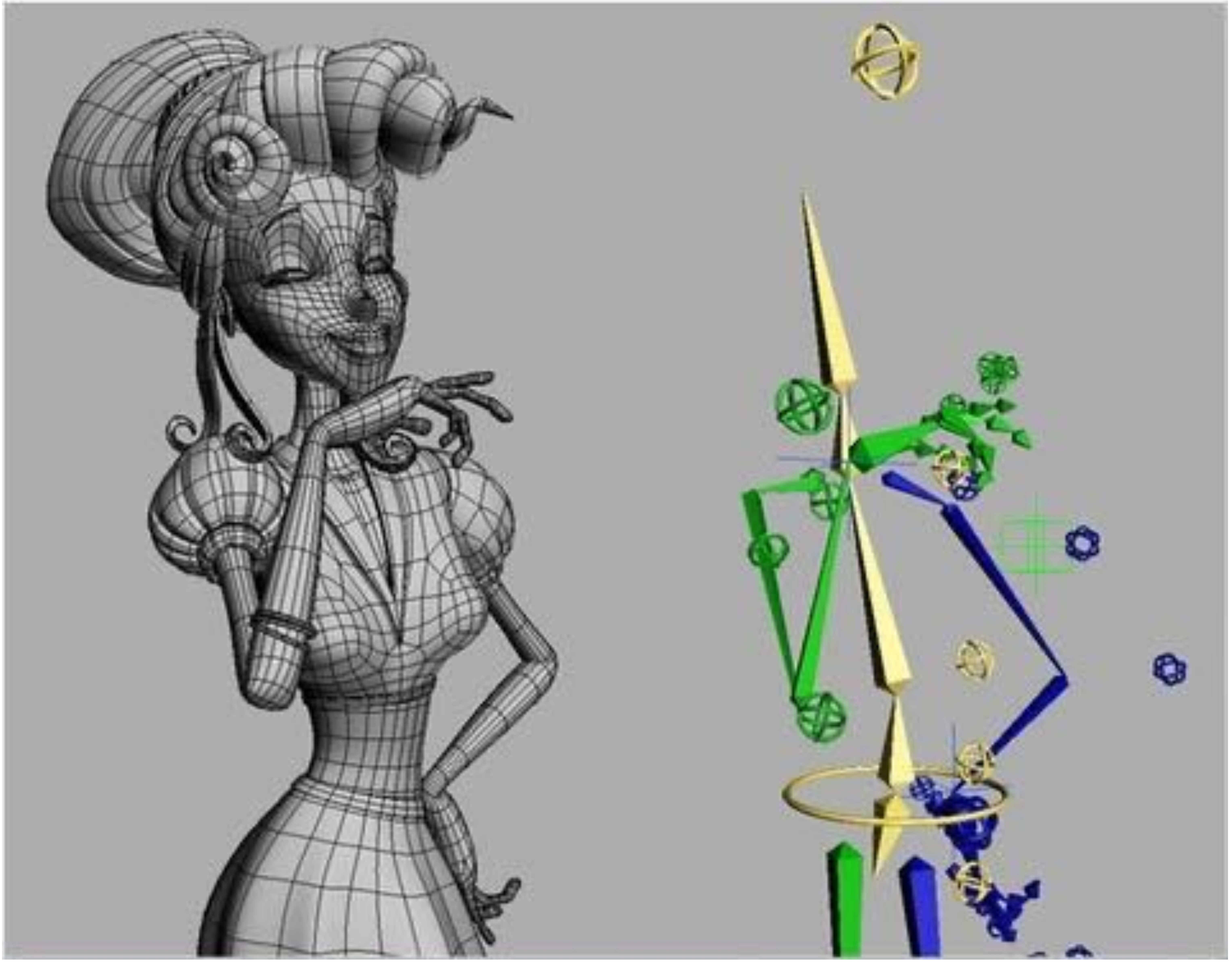
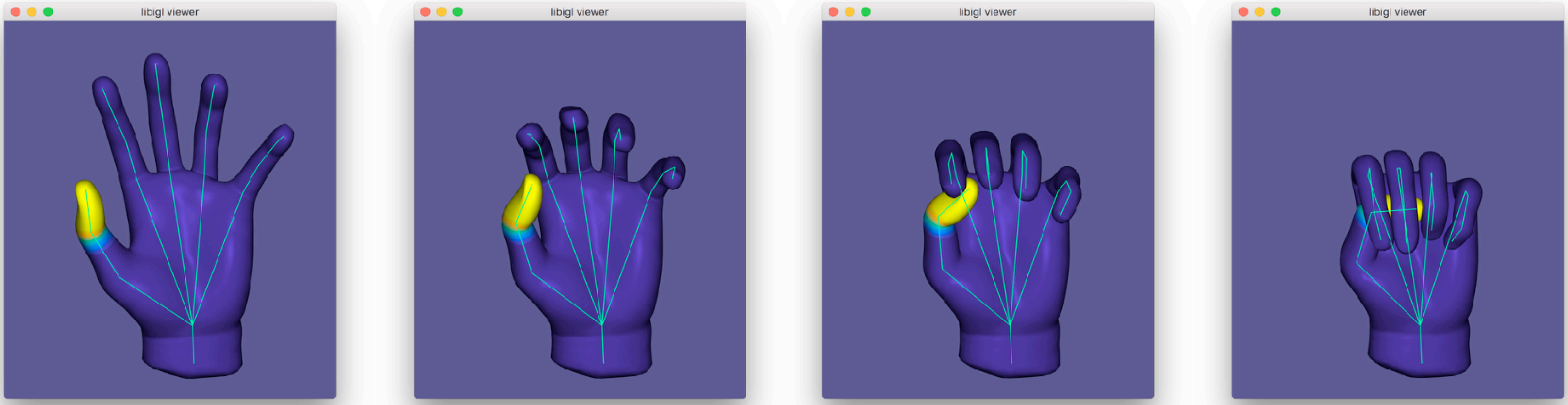
COL781: Computer Graphics

28. Skeletal Animation

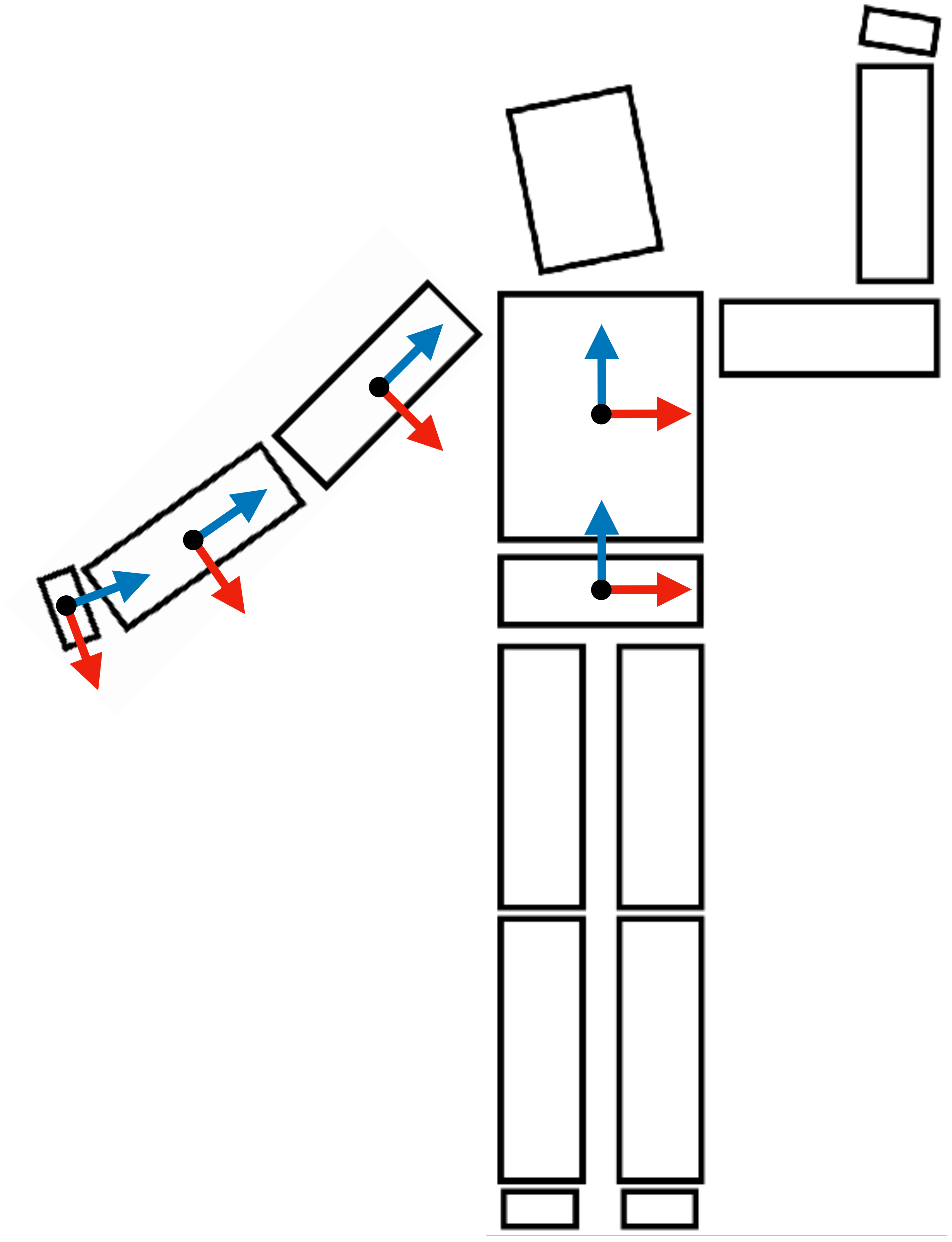
Pixar



Skeletal motion: transformation of bones based on joint angles



Skinning: displacement of surface vertices based on bone transformations



Recall hierarchical transformations from Lec. 6!

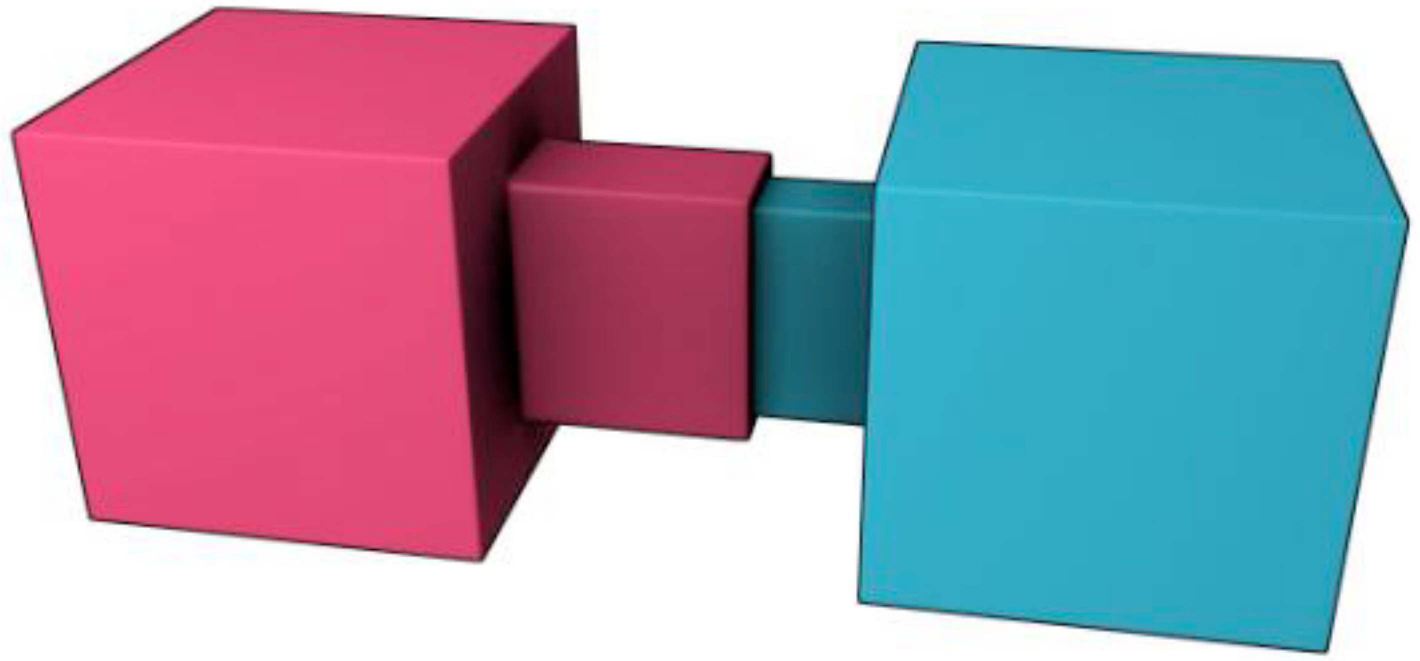
Each bone is transformed relative to its parent

$$\mathbf{M}^{\text{world}}_{\text{lHand}} = \mathbf{M}^{\text{world}}_{\text{hip}} \mathbf{M}^{\text{hip}}_{\text{chest}} \mathbf{M}^{\text{chest}}_{\text{ulArm}} \mathbf{M}^{\text{ulArm}}_{\text{llArm}} \mathbf{M}^{\text{llArm}}_{\text{lHand}}$$

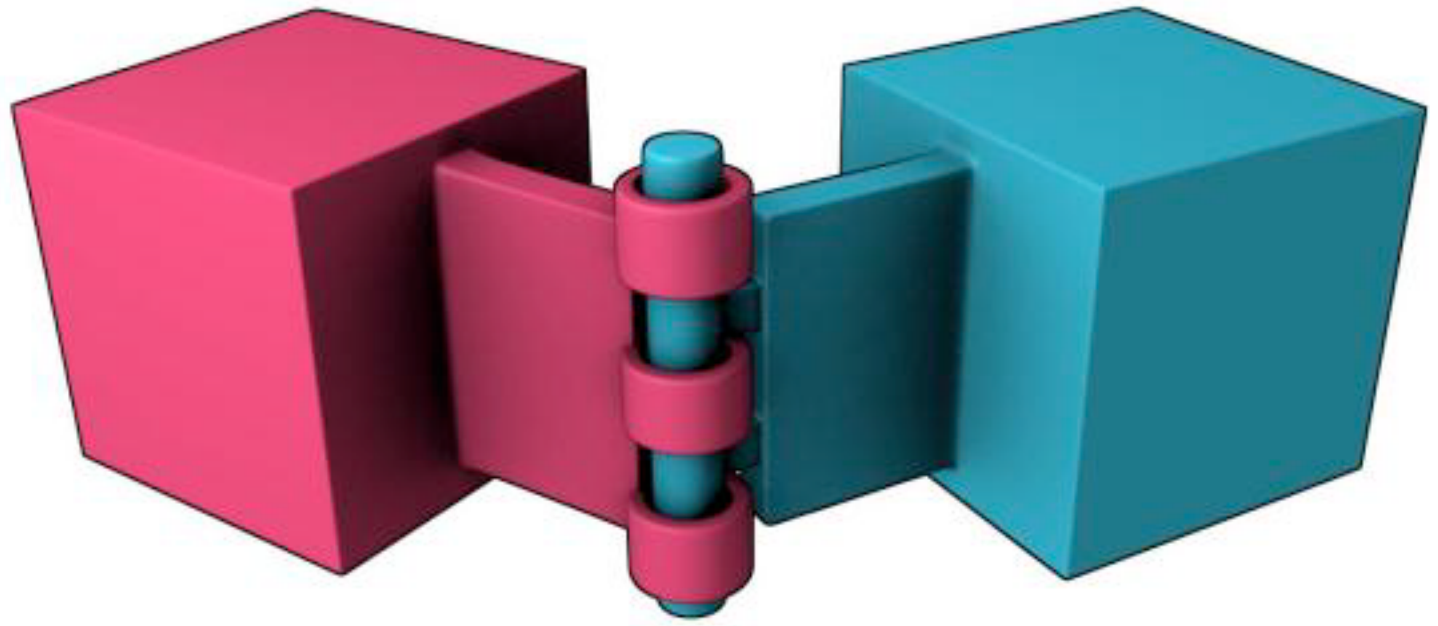
$$\mathbf{M}^{\text{world}}_{\text{lHand}} = \mathbf{M}^{\text{world}}_{\text{llArm}} \mathbf{M}^{\text{llArm}}_{\text{lHand}}$$

OK, how to control transformation of child bone?

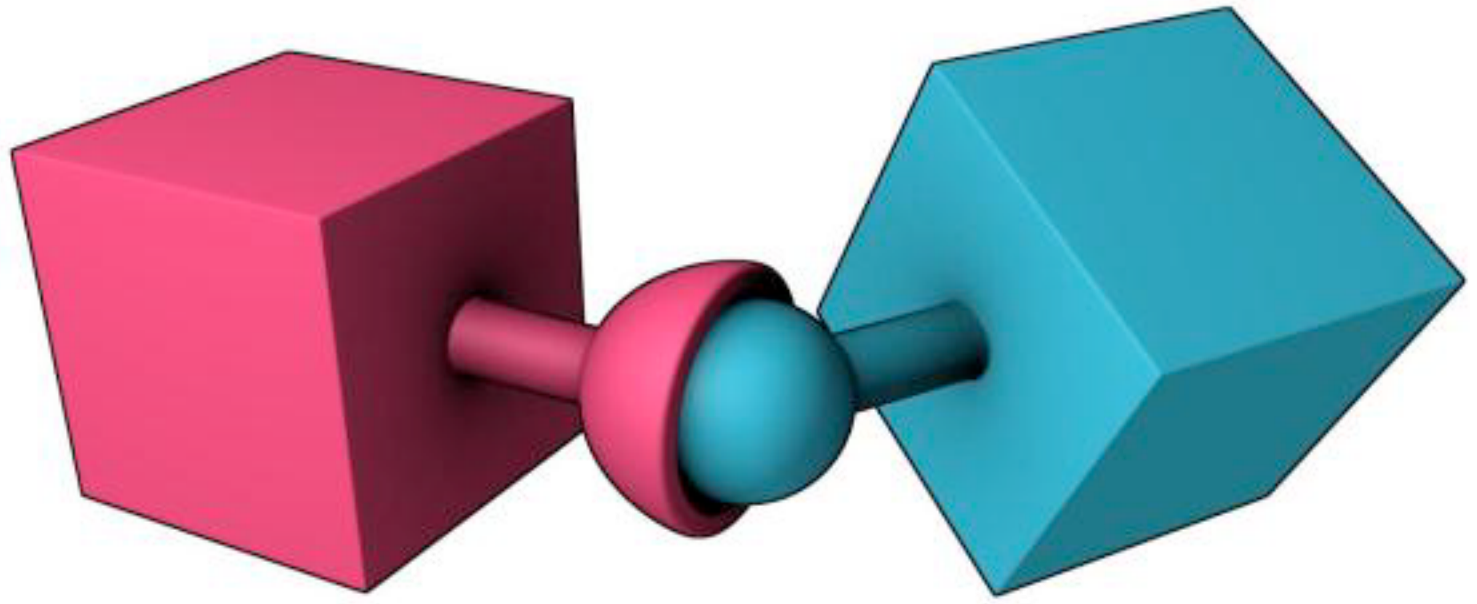
$\mathbf{M}_{\text{child}}^{\text{parent}}$ cannot be arbitrary! Child's relative motion is constrained by some kind of **joint**, e.g.:



Slider
(1 DOF)



Hinge
(1 DOF)



Ball-and-socket
(3 DOFs)

Bullet manual

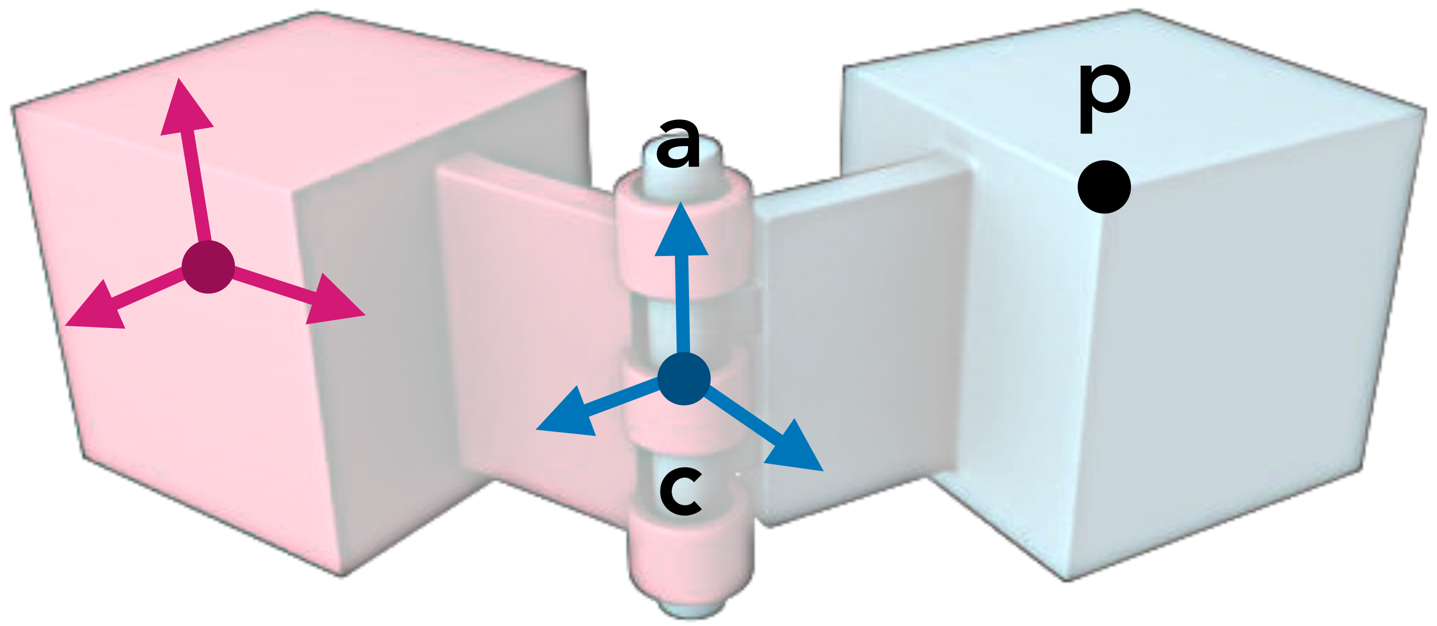
Example: Hinge joint

Suppose hinge center is at **c** in parent's coordinates, origin in child's coordinates

Hinge axis vector is **a** in both coordinate systems (why?)

Consider a point at coordinates $\mathbf{p}^c$ on the child bone.

- After rotation about hinge axis: $\mathbf{R}(\theta, \mathbf{a})\ \mathbf{p}^c$
- In parent's coordinate system: $\mathbf{T}(\mathbf{c})\ \mathbf{R}(\theta, \mathbf{a})\ \mathbf{p}^c$



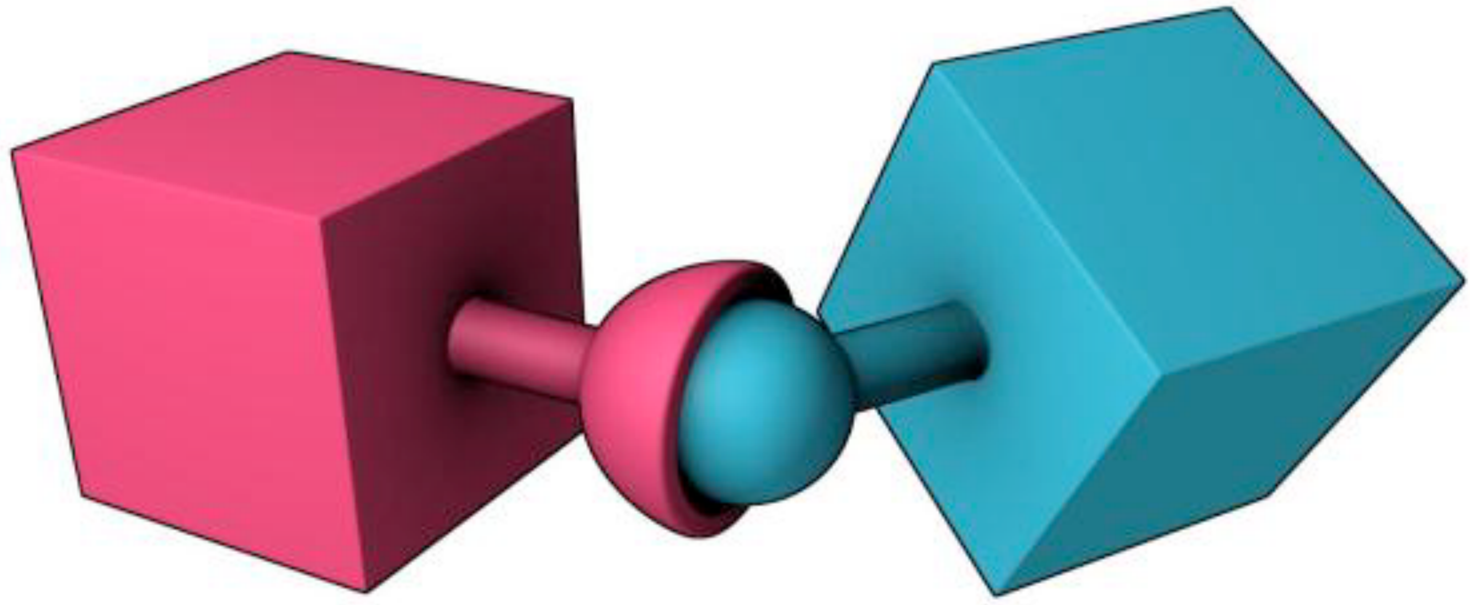
Full transformation:

$$\mathbf{p}^p = \mathbf{T}(\mathbf{c})\ \mathbf{R}(\theta, \mathbf{a})\ \mathbf{p}^c$$

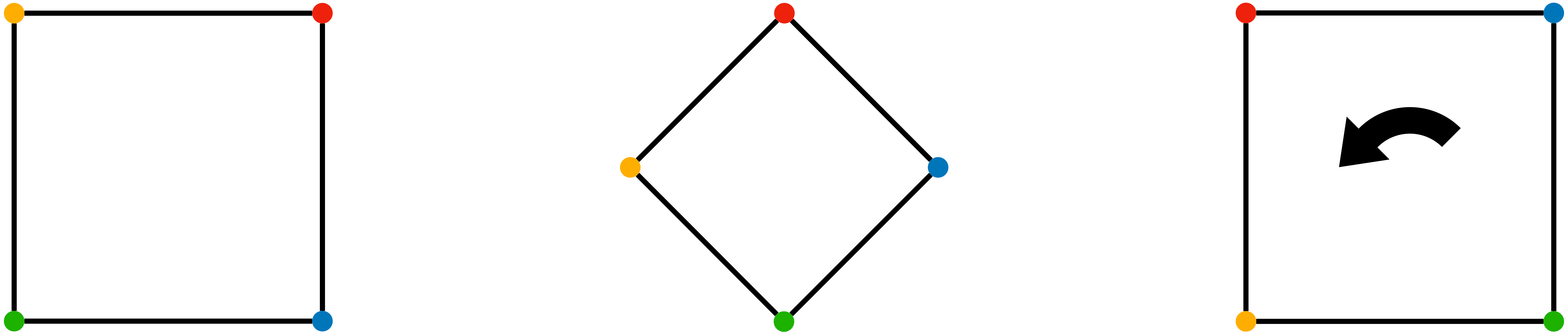
$$\mathbf{M}_c^p = \mathbf{T}(\mathbf{c})\ \mathbf{R}(\theta, \mathbf{a})$$

How to represent the rotation of a ball joint?

- 3×3 rotation matrix $\mathbf{R}$?
- Euler angles $(\theta_x, \theta_y, \theta_z)$?
- Quaternions $\boldsymbol{q} = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$?



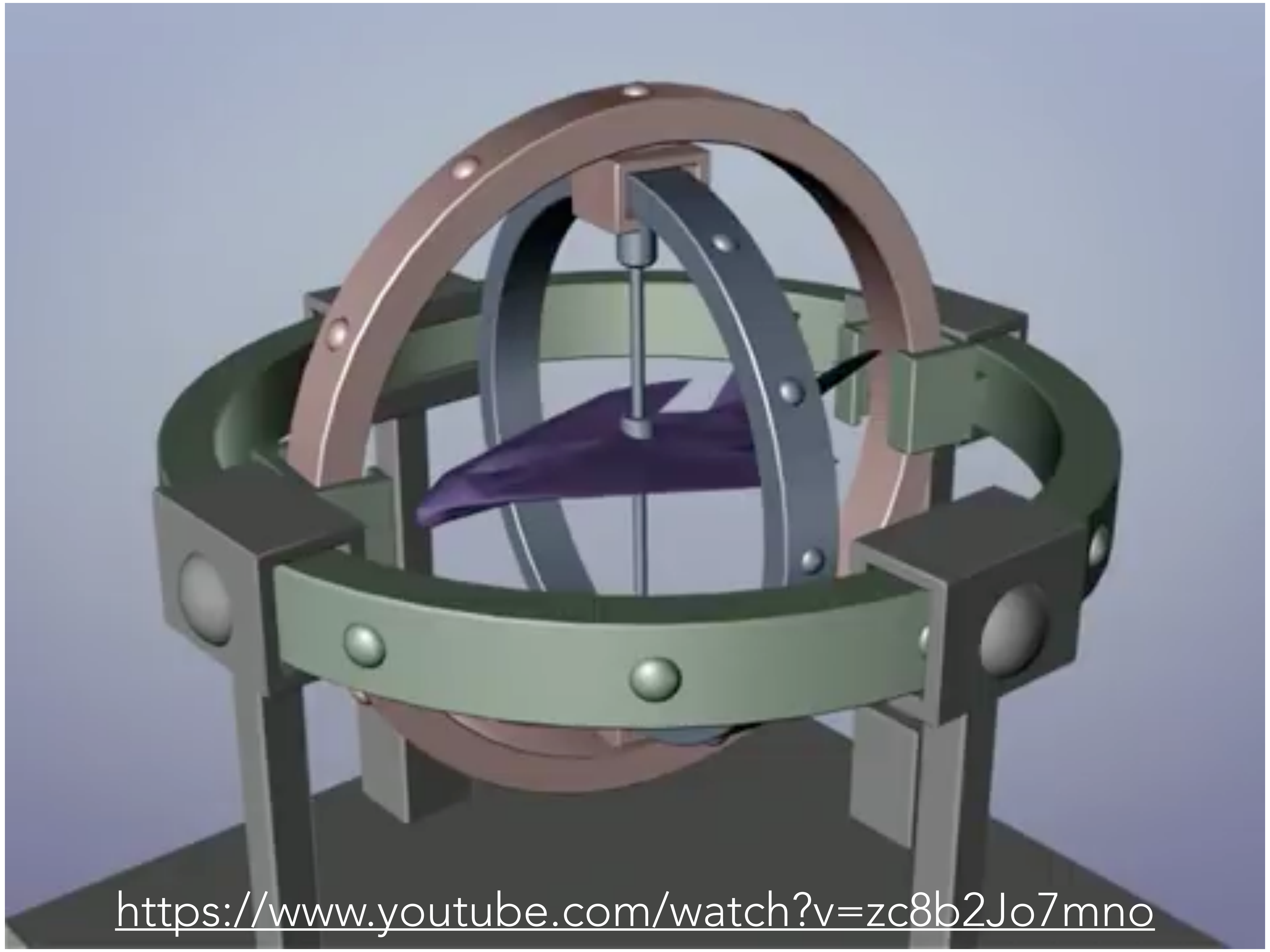
Problems with rotation matrices



$\mathbf{R}_1$ $\qquad$ $\frac{1}{2}(\mathbf{R}_1 + \mathbf{R}_2)$ $\qquad$ $\mathbf{R}_2$

No easy way to “normalize” an arbitrary matrix **M** so it becomes a rotation

Euler angles

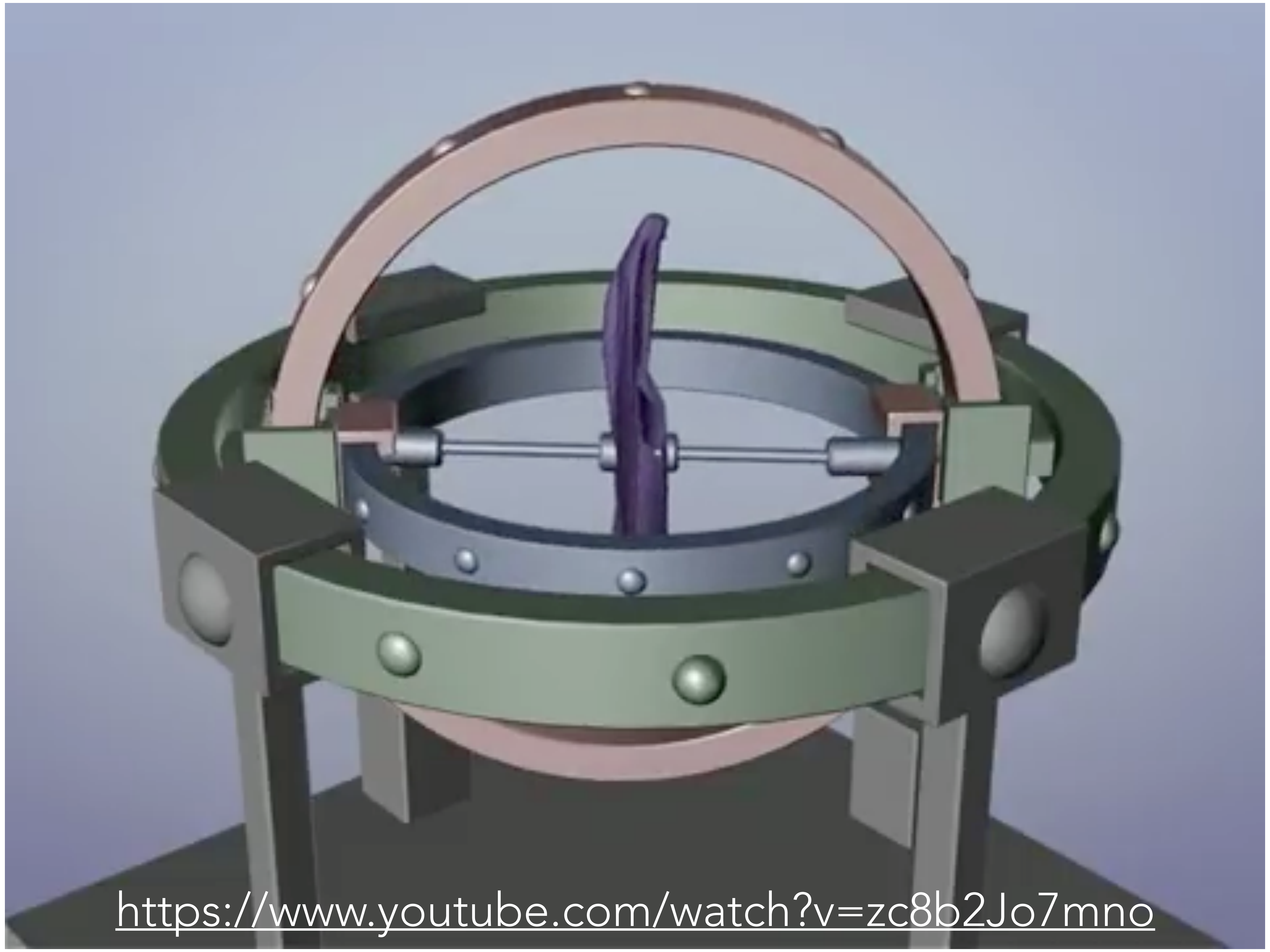


https://www.youtube.com/watch?v=zc8b2Jo7mno

GuerillaCG

Problems with Euler angles

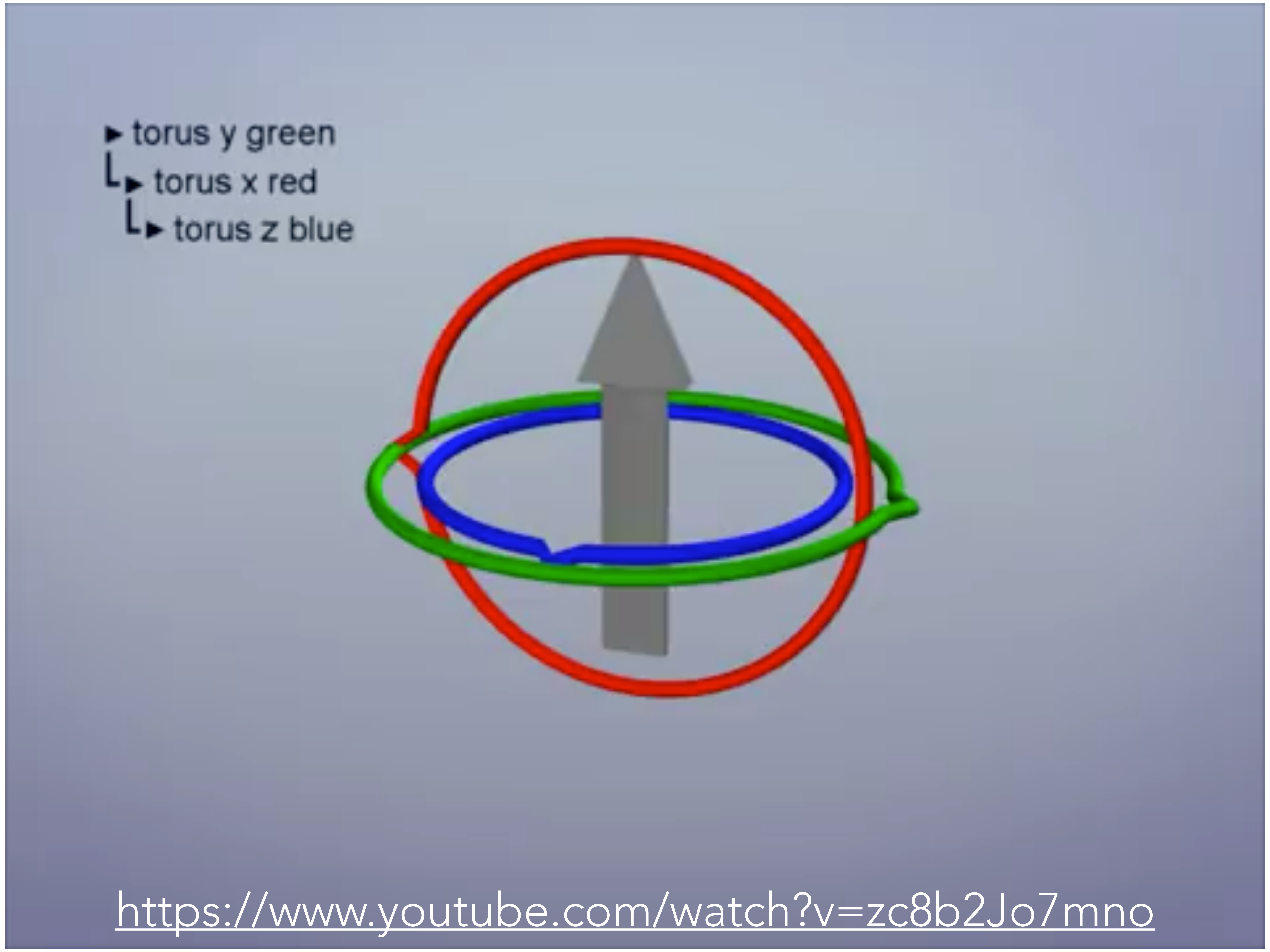
- Gimbal lock
- Unnatural interpolation



GuerillaCG

Problems with Euler angles

- Gimbal lock
- Unnatural interpolation



GuerillaCG

Warm-up: 2D rotations via complex numbers

Complex numbers are quantities of the form $a + ib$ where $i^2 = -1$.

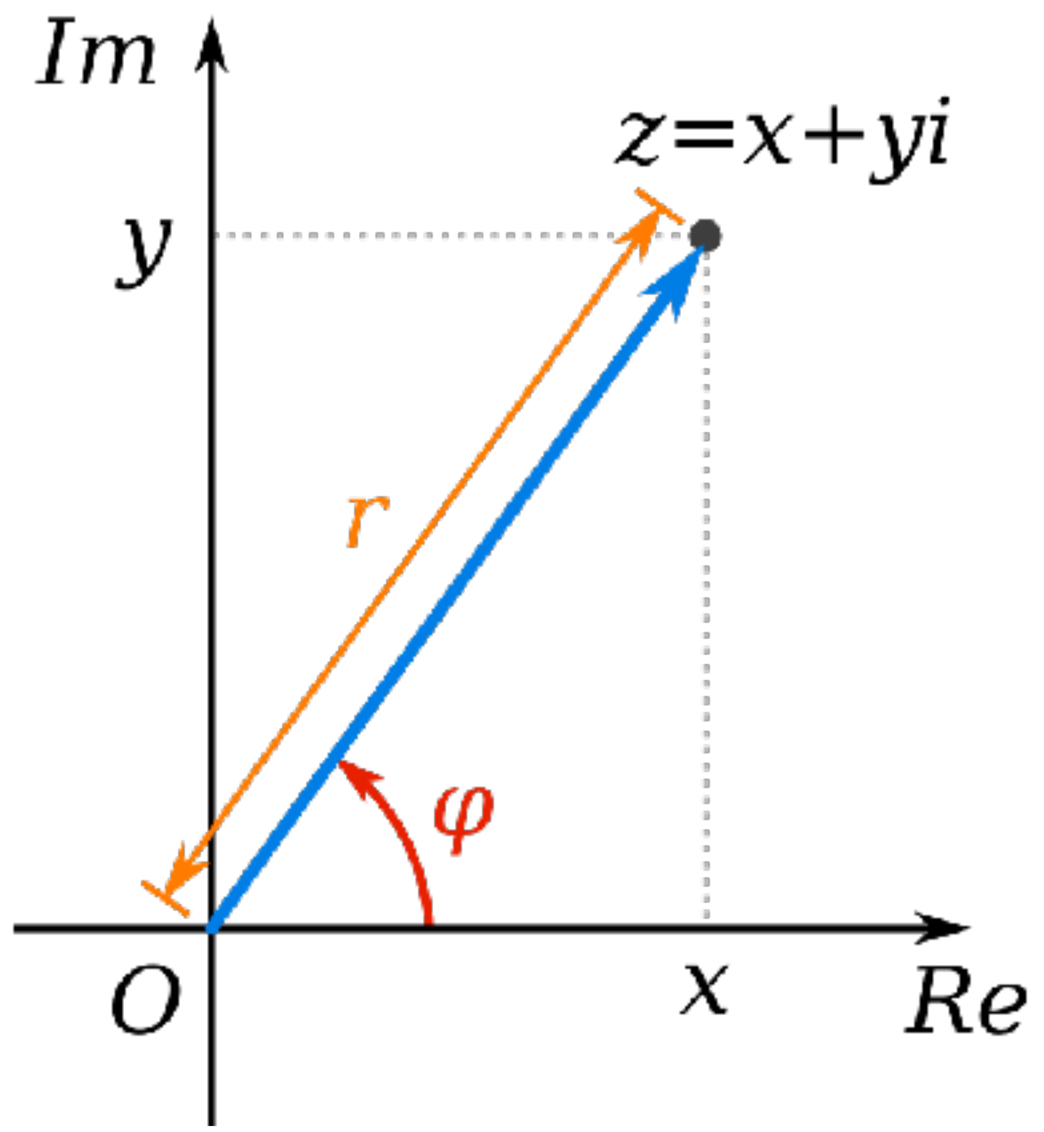
Any unit complex number ($a^2 + b^2 = 1$) represents a 2D rotation.

Rotation by angle θ:

$$z = \cos(\theta) + i \sin(\theta)$$

How to rotate a vector $\mathbf{v} = (x, y) \in \mathbb{R}^2$?

- Interpret it as a complex number $v = x + iy$
- Rotated number is $v' = zv$. Convert back to vector $\mathbf{v}' = (\operatorname{Re} v', \operatorname{Im} v')$



Quaternions

Quaternions are quantities of the form $\boldsymbol{q} = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$ where

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1$$

Useful to separate into scalar part and vector part: $\boldsymbol{q} = (a, \mathbf{u})$

- Multiplication: Not commutative! $\boldsymbol{q}_1 \boldsymbol{q}_2 \neq \boldsymbol{q}_2 \boldsymbol{q}_1$ in general

$$(a, \mathbf{u})(b, \mathbf{v}) = (ab - \mathbf{u} \cdot \mathbf{v}, a\mathbf{v} + b\mathbf{u} + \mathbf{u} \times \mathbf{v})$$

- Conjugate: $\boldsymbol{q}^* = a - b\boldsymbol{i} - c\boldsymbol{j} - d\boldsymbol{k} = (a, -\mathbf{u})$.
- Norm: $|\boldsymbol{q}| = \sqrt{\boldsymbol{q}^*\boldsymbol{q}} = \sqrt{a^2 + b^2 + c^2 + d^2}$

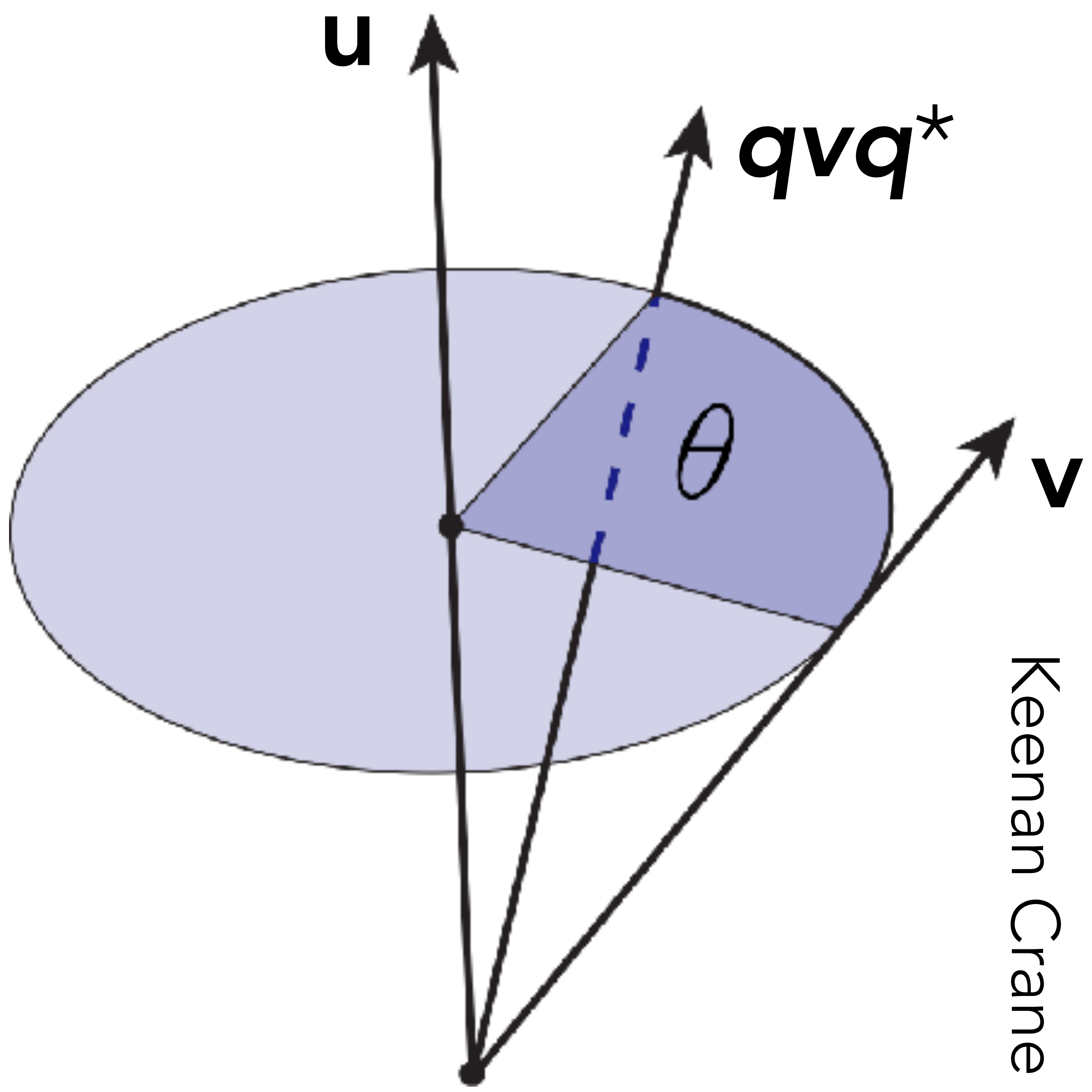
	1	**i**	**j**	**k**
1	1	**i**	**j**	**k**
i	**i**	−1	**k**	−**j**
j	**j**	−**k**	−1	**i**
k	**k**	**j**	−**i**	−1

Any unit quaternion represents a 3D rotation. Rotation by angle θ about axis $\mathbf{u}$:

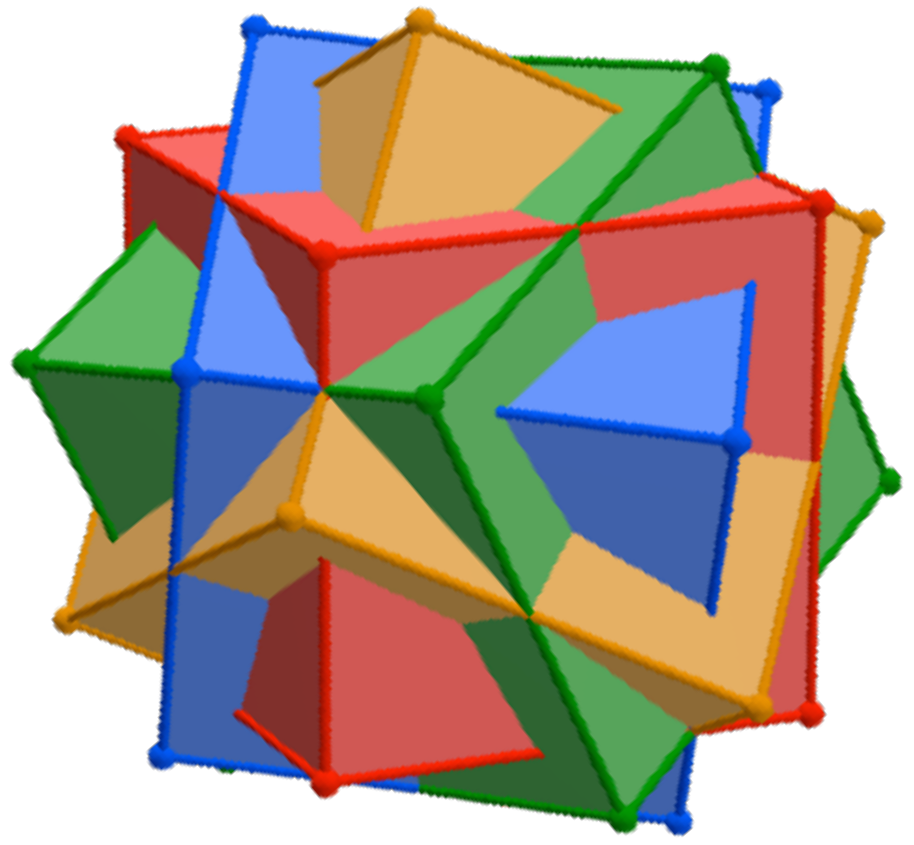
$$\boldsymbol{q} = (\cos(\theta/2),\ \mathbf{u}\sin(\theta/2))$$

How to rotate a vector $\mathbf{v} = (x, y, z) \in \mathbb{R}^3$?

- Interpret as a purely imaginary quaternion $\boldsymbol{v} = 0 + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$.
- After rotation, $\boldsymbol{v}' = \boldsymbol{q}\boldsymbol{v}\boldsymbol{q}^{-1}$ ($= \boldsymbol{q}\boldsymbol{v}\boldsymbol{q}^*$ because $|\boldsymbol{q}| = 1$).
  Surprisingly, still purely imaginary! $\boldsymbol{v}' = 0 + x'\boldsymbol{i} + y'\boldsymbol{j} + z'\boldsymbol{k}$.



Actually $\boldsymbol{q}$ and $-\boldsymbol{q}$ represent the same rotation…

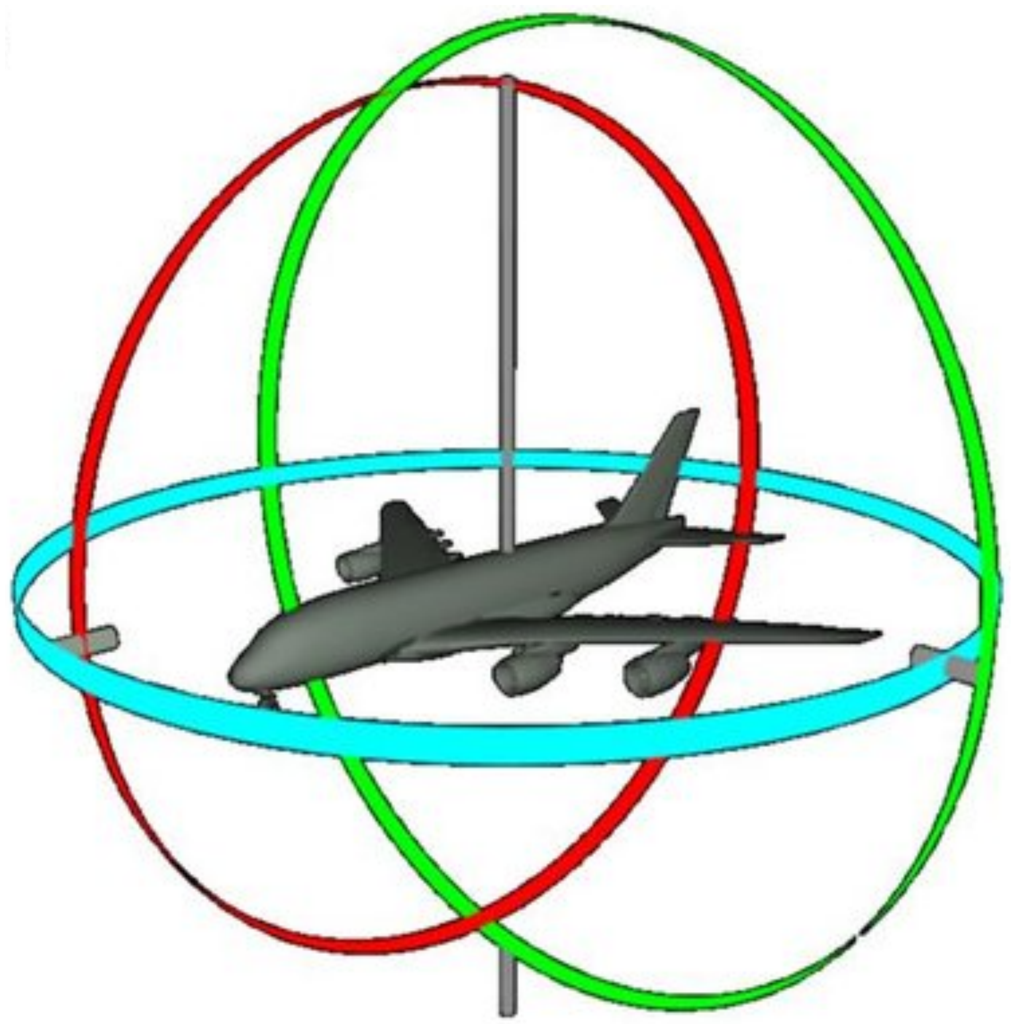


Puzzle:

In complex numbers we had $z_1 z_2 = z_2 z_1$, but in quaternions we generally have $\mathbf{q}_1 \mathbf{q}_2 \neq \mathbf{q}_2 \mathbf{q}_1$.

Why is this a very good thing, if we intend to use quaternions to model 3D rotations?

…And why did that reason not apply in 2D?



Why quaternions?

No gimbal lock (unlike Euler angles)

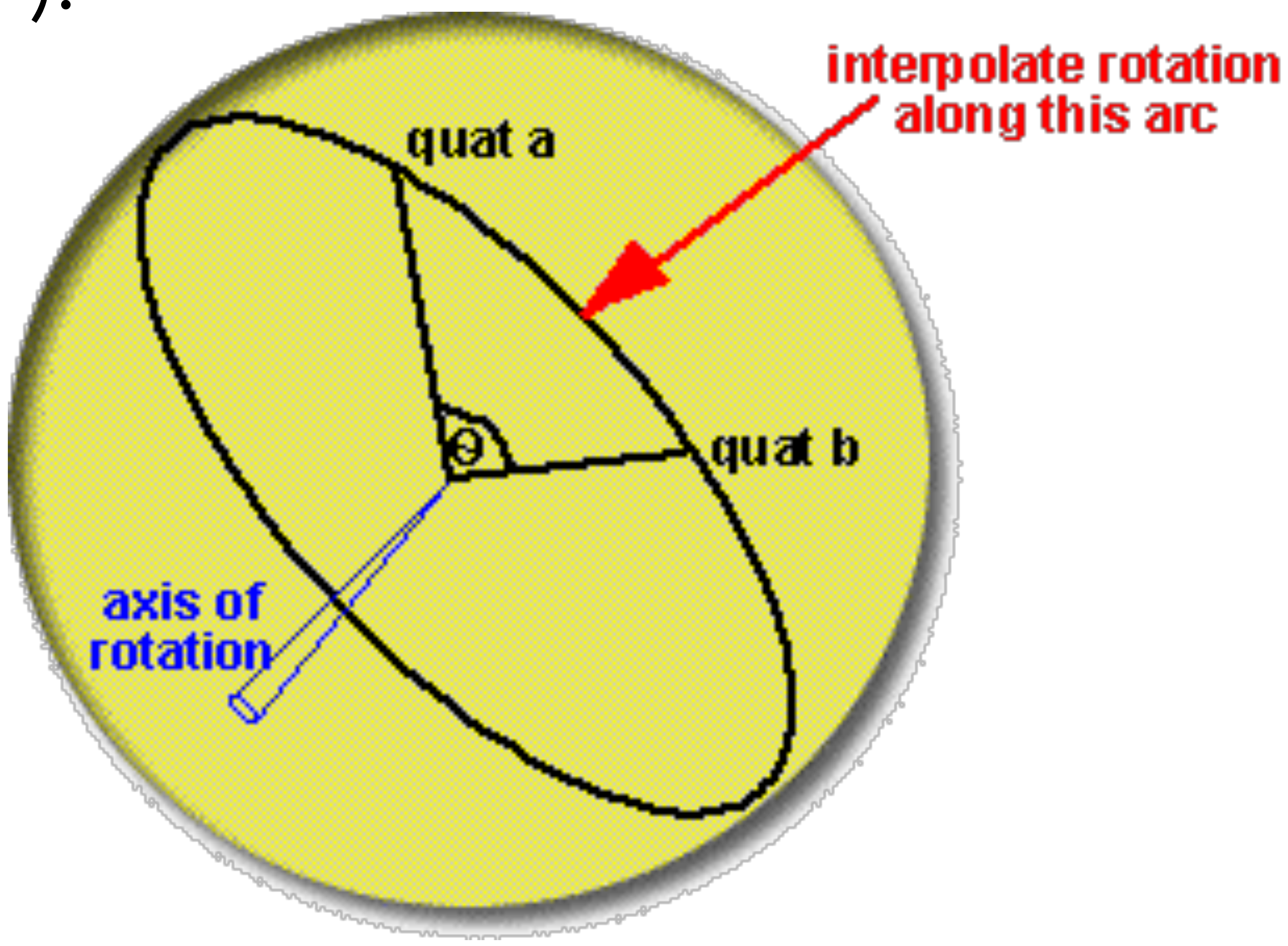
Easy to normalize: just divide by $|\boldsymbol{q}|$ (unlike rotation matrices)

Easy to interpolate via spherical linear interpolation ("**Slerp**"):

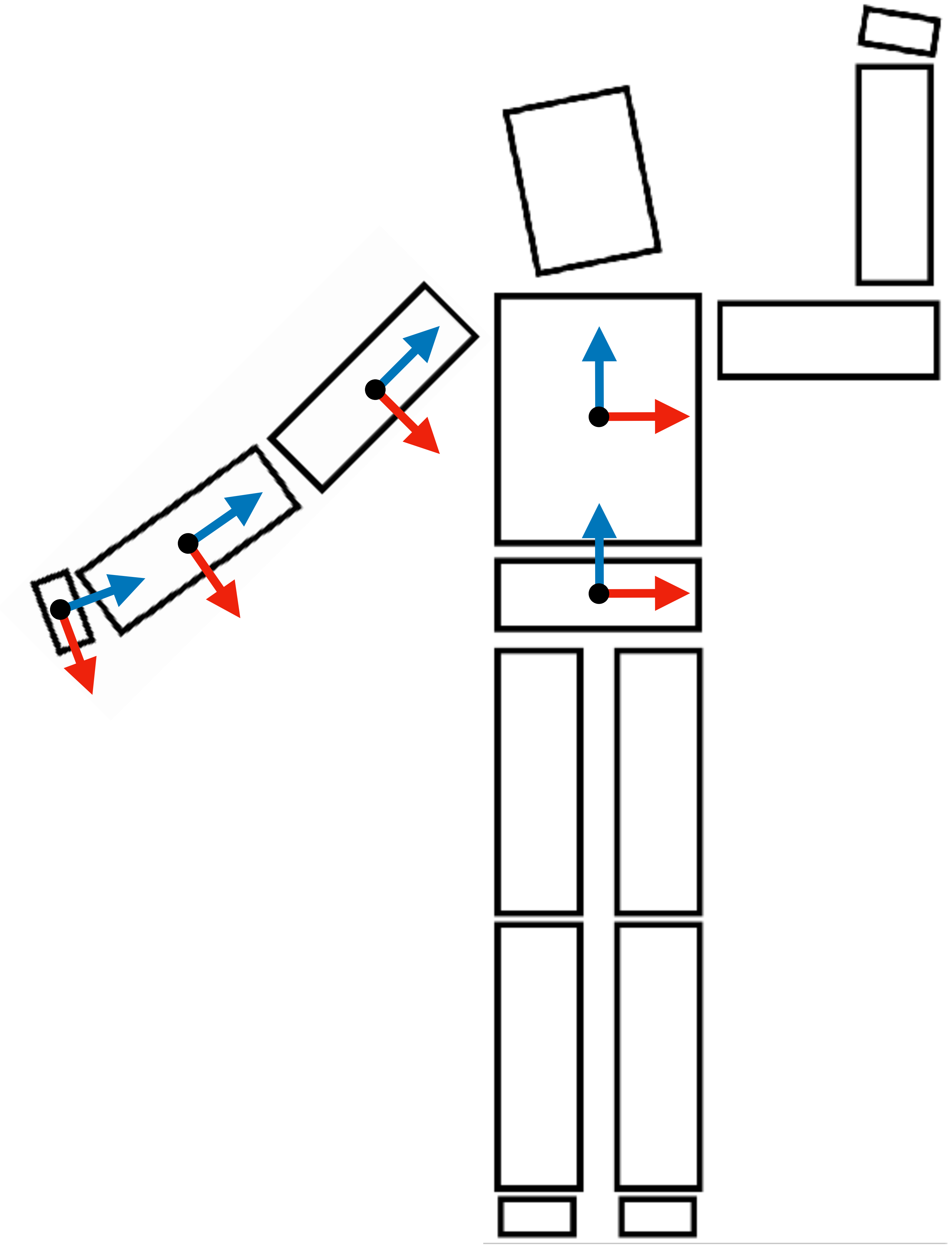
$$\mathrm{Slerp}(\boldsymbol{q}_0, \boldsymbol{q}_1, t) = (\boldsymbol{q}_1\, \boldsymbol{q}_0^{-1})^t\, \boldsymbol{q}_0$$

$$= \frac{\sin((1-t)\Omega)}{\sin\Omega}\boldsymbol{q}_0 + \frac{\sin(t\Omega)}{\sin\Omega}\boldsymbol{q}_1$$

where $\cos\Omega = \mathrm{Re}(\boldsymbol{q}_0{}^*\boldsymbol{q}_1) = \boldsymbol{q}_0 \cdot \boldsymbol{q}_1$ treated as vectors



Generalized coordinates

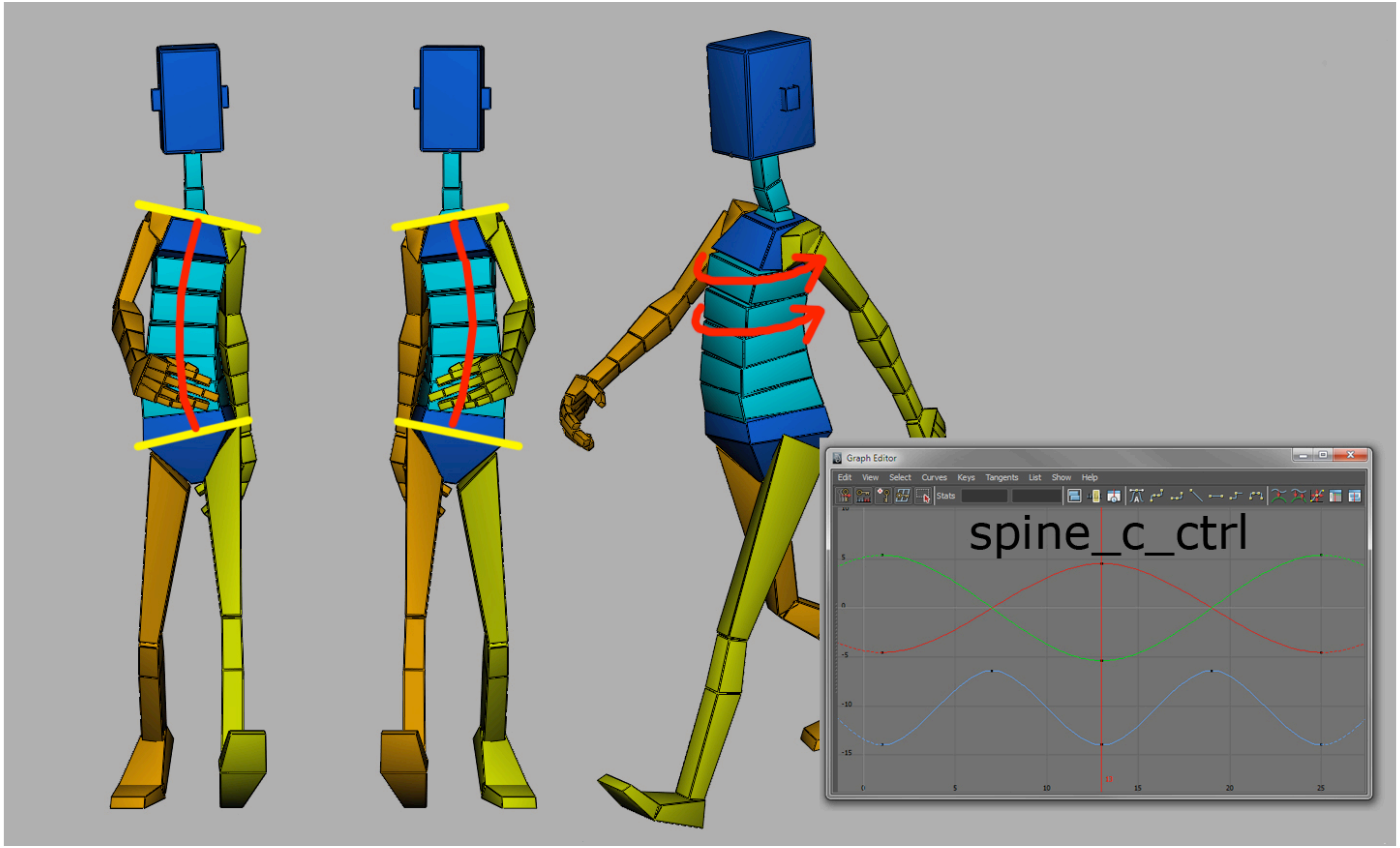


A single vector to specify the pose of the entire body

$$\mathbf{q} = (q_1, q_2, q_3, q_4, \ldots, q_n)$$

- Location of root
- Orientation of root
- Joint angles of all joints

To animate a character, specify function $t \mapsto \mathbf{q}$

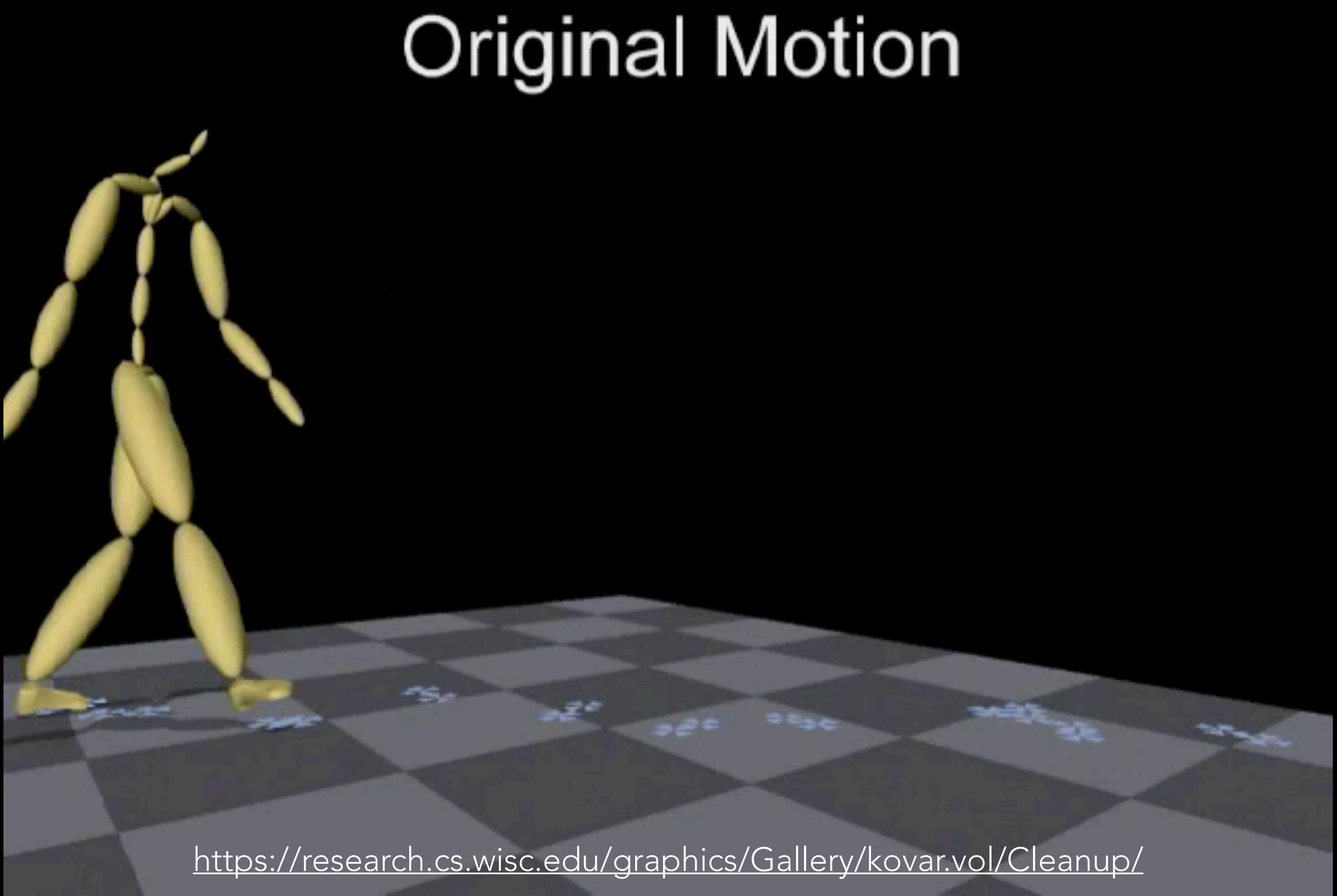


CAVE Academy

Given joint angles, compute transformation of points: **forward kinematics**

Liu 2009

Given desired transformation of end points, how to find the joint angles that achieve it?
Inverse kinematics



https://research.cs.wisc.edu/graphics/Gallery/kovar.vol/Cleanup/

Kovar et al. 2002