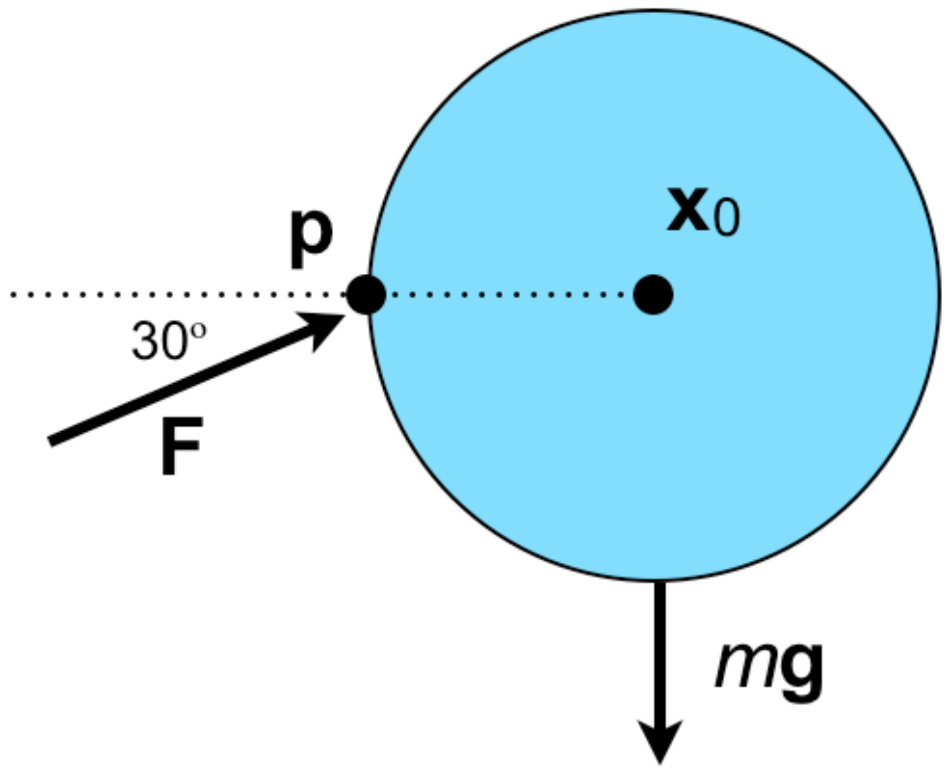
Previously in CS248B…

Quiz

- **Consider a 3D sphere with radius 1m, mass 1kg, and inertia I_{body}. The initial linear and angular velocity are both zero. The initial position and the initial orientation are x_0 and R_0. The forces applied on the sphere include gravity (g) and an initial push F applied at point p. Note that F is only applied for one time step at t_0. If we use Explicit Euler method with time step h to integrate , what are the position and the orientation of the sphere at t_2? Use the actual numbers defined as below to compute your solution (except for g and h).**



$\mathbf{x}_0 = (0, 0, 0)$

$$\mathbf{R}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$\mathbf{p} = (-1, 0, 0)$

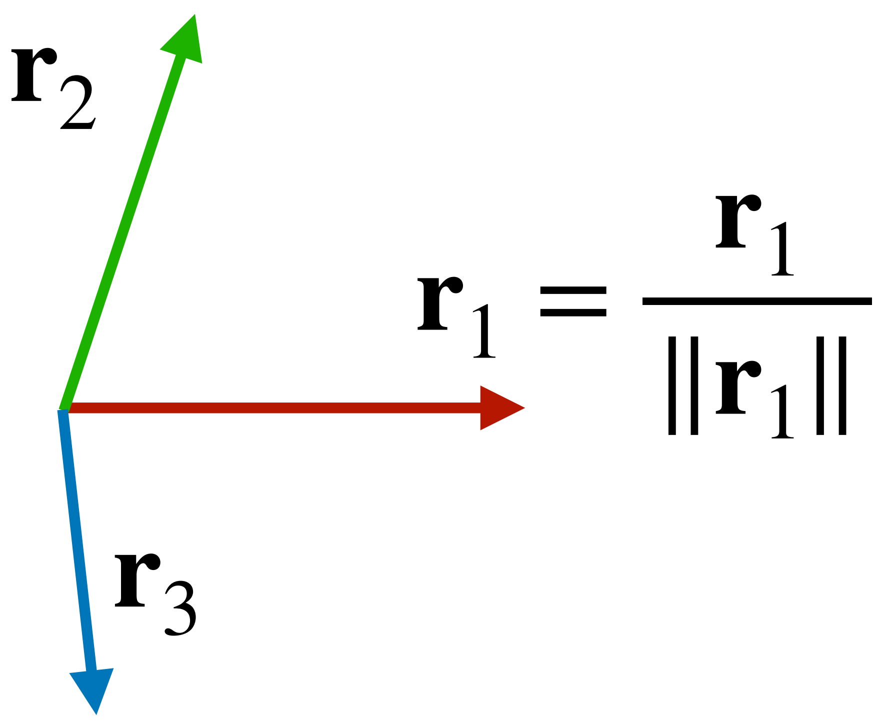
$\mathbf{F} = (4\cos(30^\circ), 4\sin(30^\circ), 0)$

$m = 1$

$$\mathbf{I}_{body} = \begin{pmatrix} 2/5 & 0 & 0 \\ 0 & 2/5 & 0 \\ 0 & 0 & 2/5 \end{pmatrix}$$

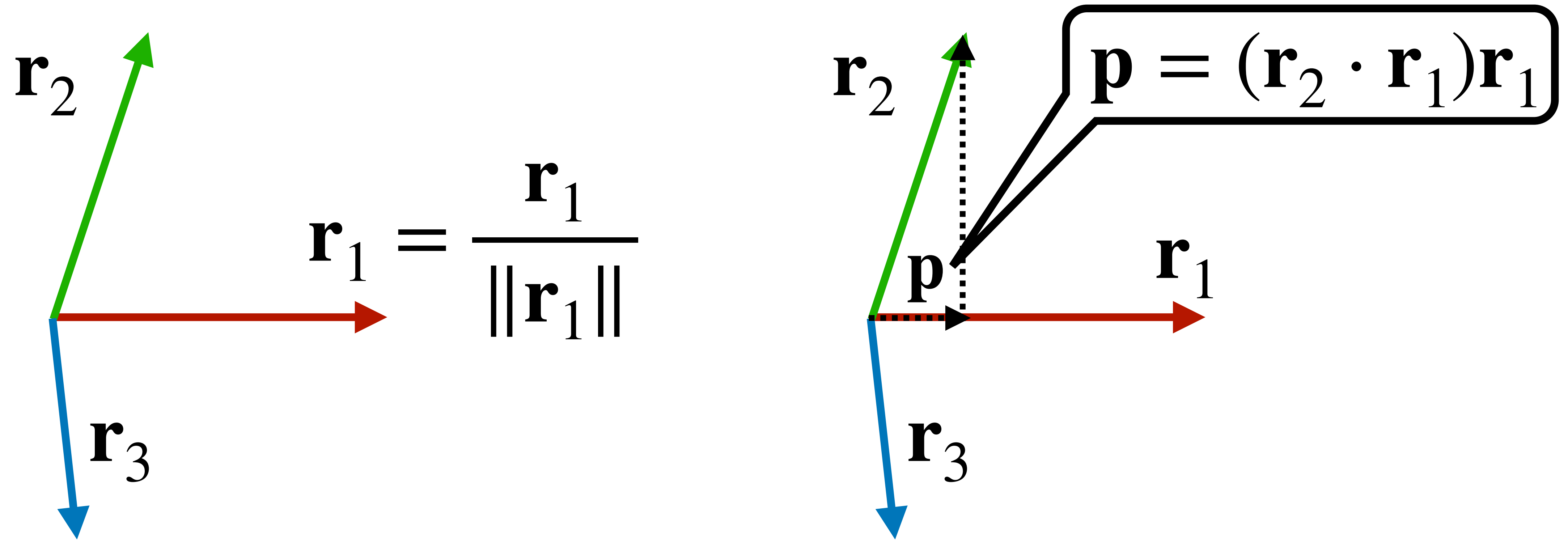
Issues with rotation matrix

- The rotational matrix might no longer be orthonormal due to accumulated numerical errors.
- Rectifying a rotational matrix is not trivial.
  - Could use Gram-Schmidt process to make $\mathbf{R}$ orthonormal.



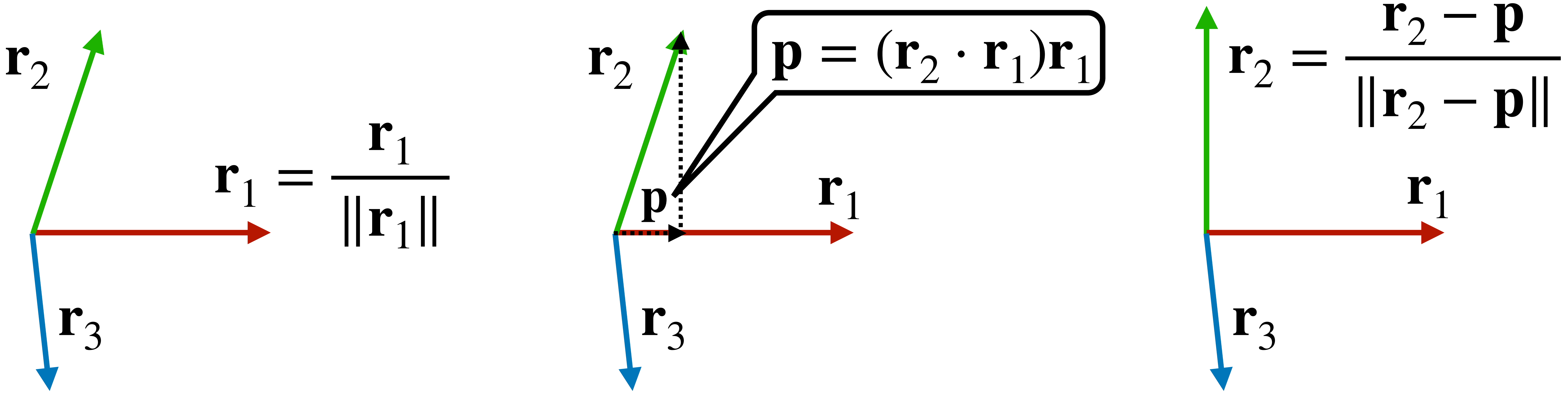
Issues with rotation matrix

- The rotational matrix might no longer be orthonormal due to accumulated numerical errors.
- Rectifying a rotational matrix is not trivial.
  - Could use Gram-Schmidt process to make $\mathbf{R}$ orthonormal.



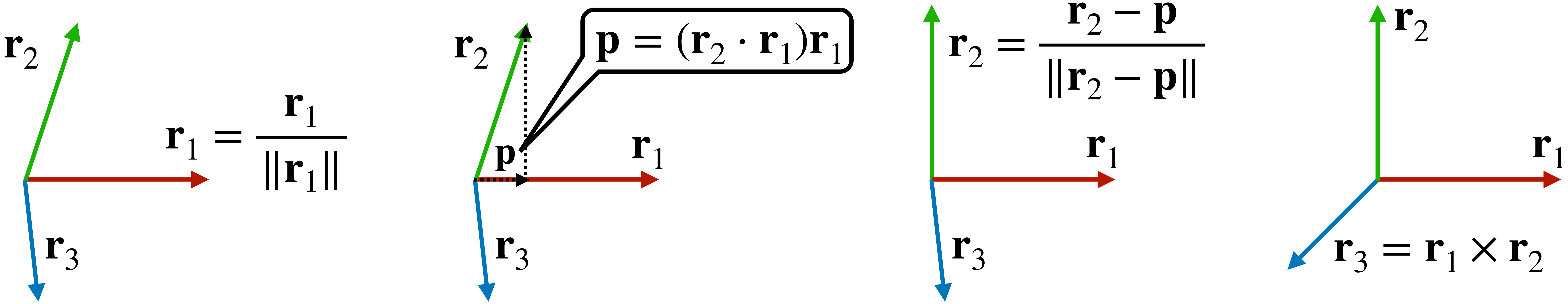
Issues with rotation matrix

- The rotational matrix might no longer be orthonormal due to accumulated numerical errors.
- Rectifying a rotational matrix is not trivial.
  - Could use Gram-Schmidt process to make $\mathbf{R}$ orthonormal.



Issues with rotation matrix

- The rotational matrix might no longer be orthonormal due to accumulated numerical errors.
- Rectifying a rotational matrix is not trivial.
  - Could use Gram-Schmidt process to make $\mathbf{R}$ orthonormal.



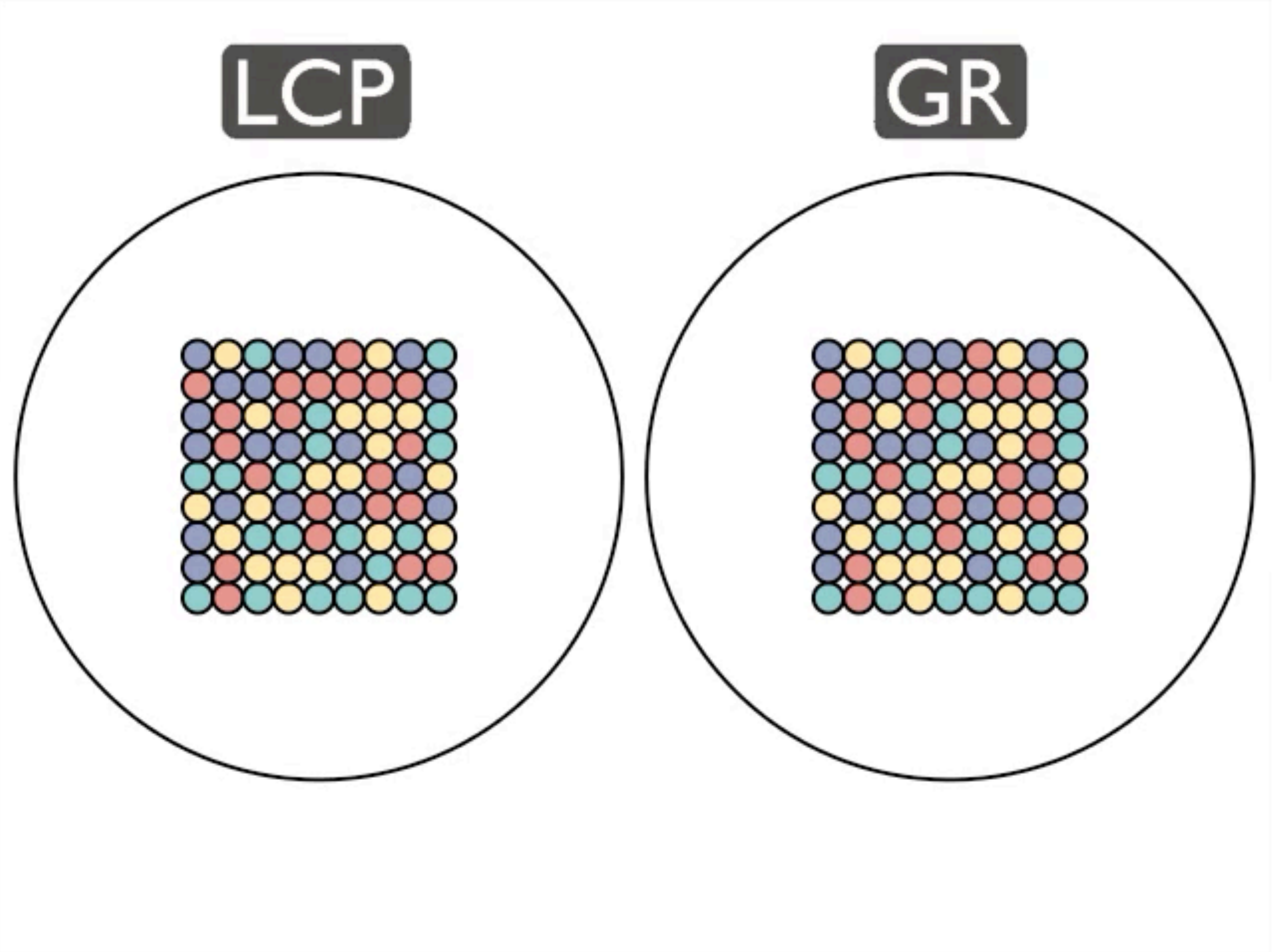
- We will use Quaternion representation for 3D orientation instead.

Momentum vs velocity

- Why do we use momentum in the state space instead of velocity?
  - Because the relation of angular momentum and torque is simpler.
  - Because the angular momentum is constant when there is no torques acting on the object.
- Use linear momentum $\mathbf{p}(t)$ to be consistent with angular velocity.

Constrained rigid body simulation

- Handling contacts and collisions is a very important topic that will be partially covered in later lectures.
- Idealized contact models can produce visually plausible results for graphics applications, but they are often a major source of error when predicting the motion of real-world objects.



Additional reading

- Skew symmetric matrix: https://en.wikipedia.org/wiki/Skew-symmetric_matrix
- Rigid body lecture notes from David Baraff:
  - https://www.cs.cmu.edu/~baraff/sigcourse/notesd1.pdf
  - https://www.cs.cmu.edu/~baraff/sigcourse/notesd1.pdf
- Brian Mirtich's thesis
  - https://people.eecs.berkeley.edu/~jfc/mirtich/thesis/mirtichThesis.pdf

Lecture 12:

3D Orientation

FUNDAMENTALS OF COMPUTER GRAPHICS

Animation & Simulation

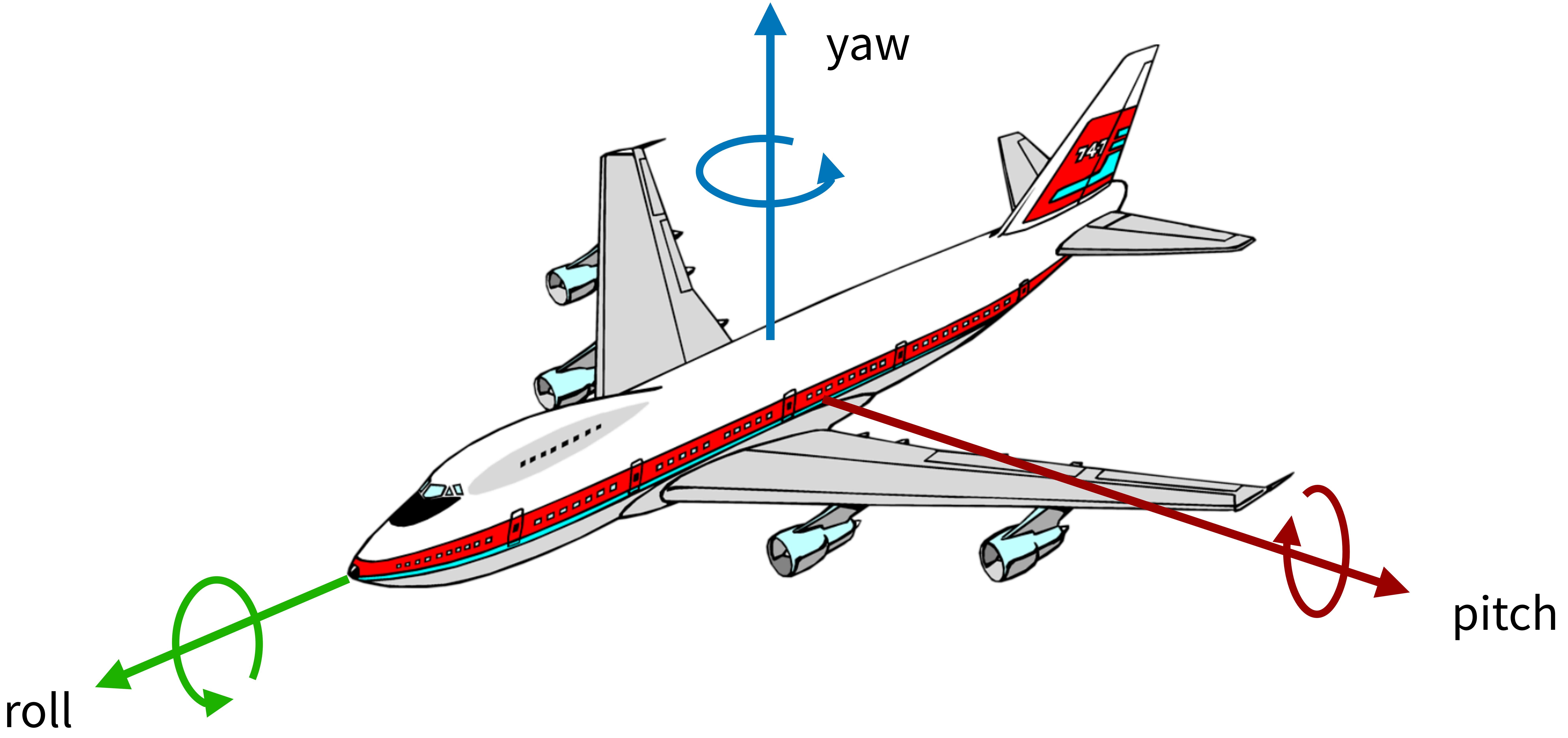
Stanford CS248B, Fall 2022

Learning Objectives

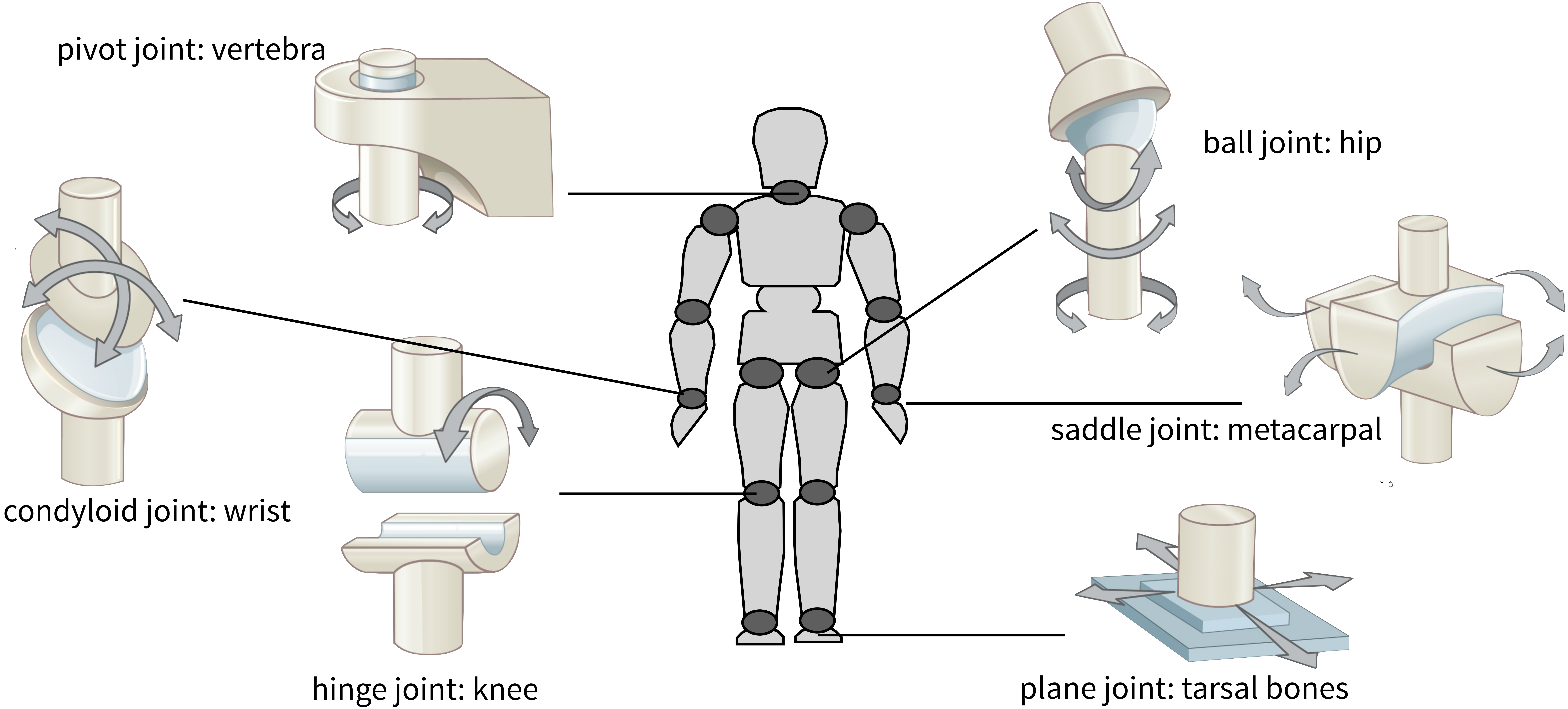
- Learn different representations for 3D orientation
- Understand the properties of each representation
- Learn the concept of quaternion and its applications

3D orientation

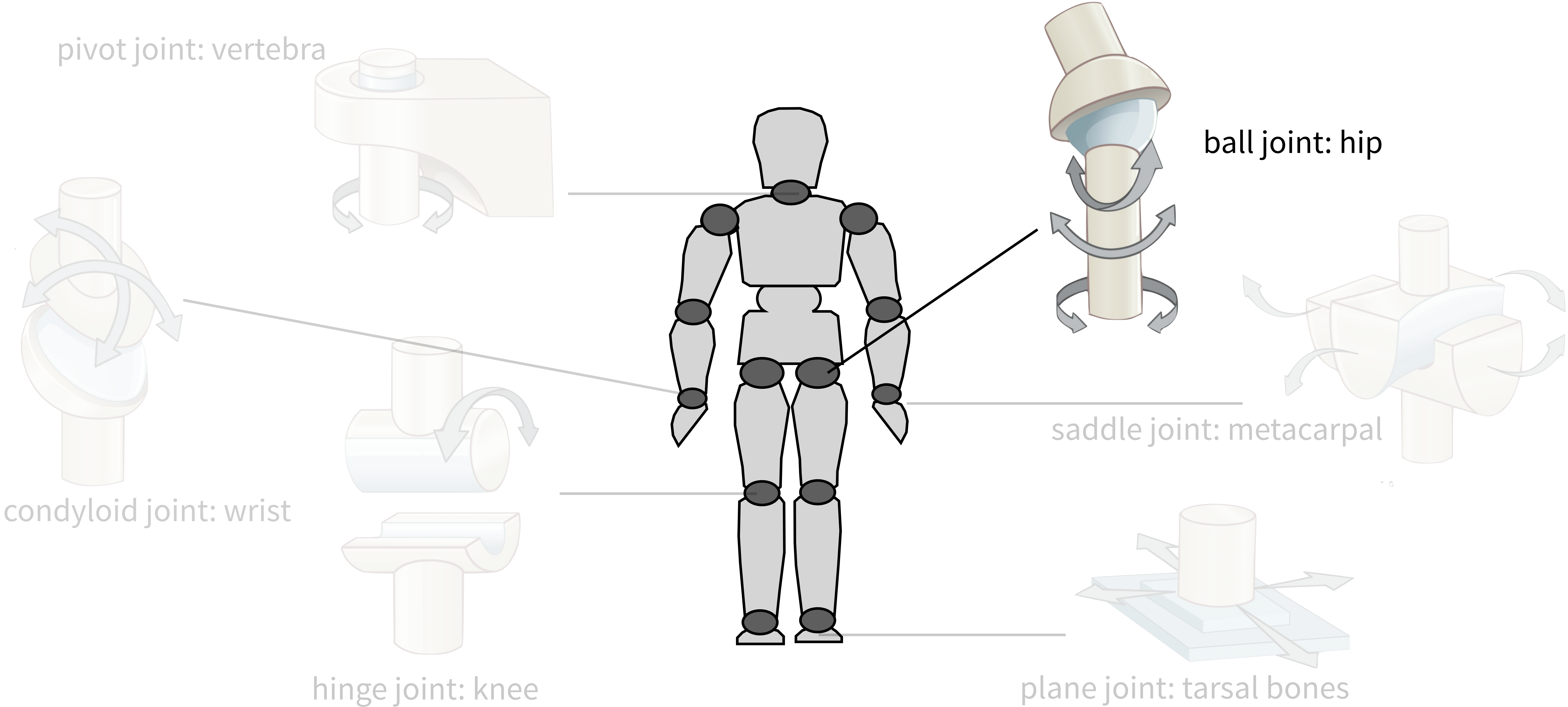
Orientation describes the angular position of a 3D object in the space it occupies.



Parameterizing human model



Parameterizing human model



Matrix representation

- 3D orientation can be represented by a matrix $\mathbf{R}$.
- $\mathbf{R}$ is an orthonormal matrix.
- $\mathbf{R}\mathbf{R}^T = \mathbf{I}$
- Multiple rotations can be compose to one matrix
  - $\mathbf{R} = \mathbf{R}_x(30)\mathbf{R}_y(60)\mathbf{R}_z(90)$

about x axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

about y axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

about z axis

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Interpolation

- Interpolating translation is easy, but what about rotations? How about interpolating each entry of the rotation matrix?
- The interpolated matrix might no longer be orthonormal, leading to nonsense for the in-between rotations.
  - Example: interpolate linearly from a $+90^\circ$ rotation about y axis to -90°rotation about y-axis:

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Interpolation

- **Interpolating translation is easy, but what about rotations? How about interpolating each entry of the rotation matrix?**
- **The interpolated matrix might no longer be orthonormal, leading to nonsense for the in-between rotations.**
  - **Example: interpolate linearly from a $+90^\circ$ rotation about y axis to -90°rotation about y-axis:**

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Linearly interpolate each component and halfway between, you get this...

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

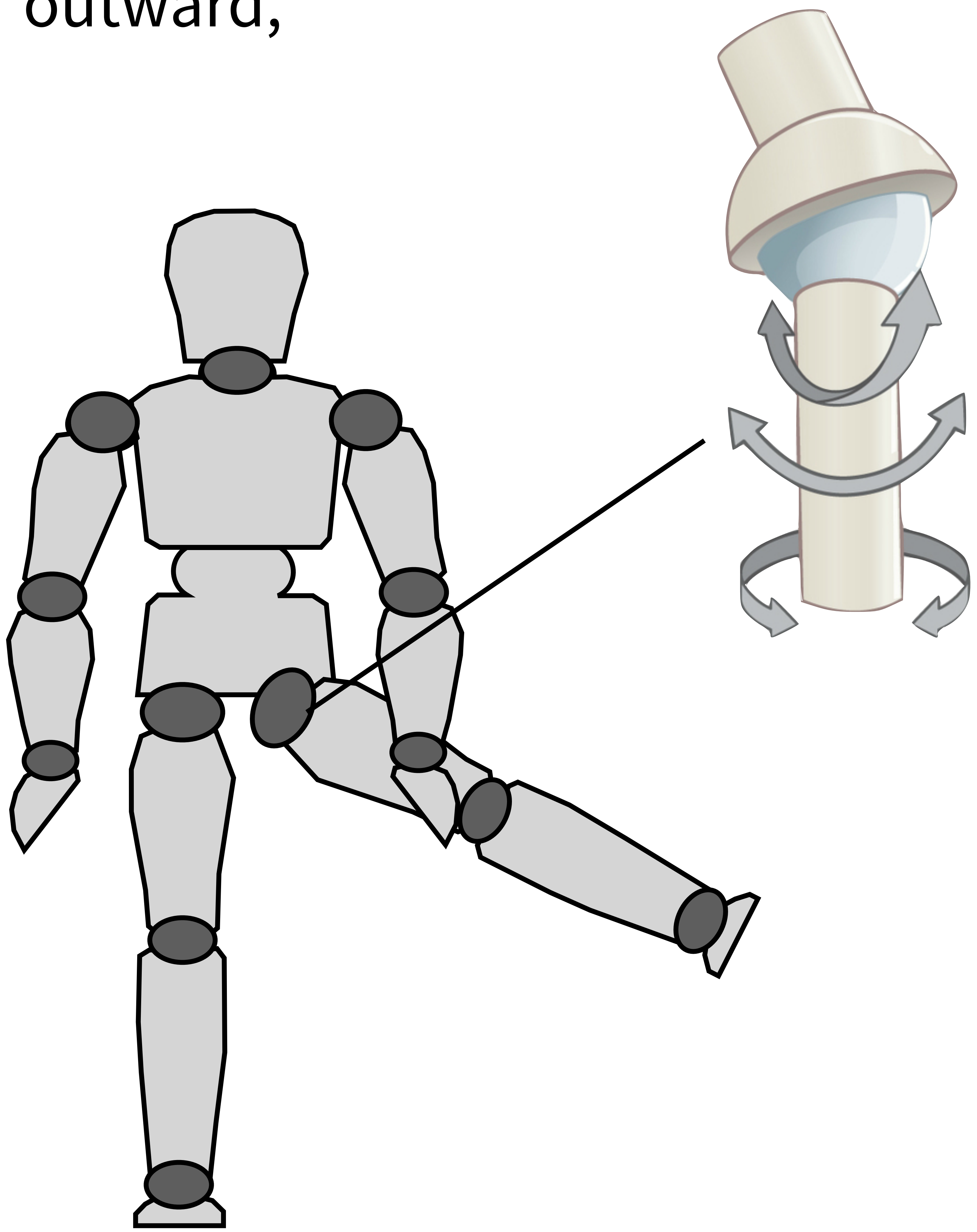
Enforce joint limits

Suppose the hip joint can not abduct more than $80°$ outward,

Does $\mathbf{R}_{l-hip} = \begin{bmatrix} -0.069 & 0.012 & 0.998 \\ 0.601 & 0.799 & 0.032 \\ -0.796 & 0.601 & -0.062 \end{bmatrix}$

violate the joint limits?

Hard to tell!



Properties of rotation matrix

- Easy to compose? ✓
- Easy to interpolate? ✗
- Easy to enforce joint limits? ✗

Fixed angle and Euler angle

- Both fixed angle and Euler angle representations specify 3D orientation by 3 parameters that represent angles used to rotate about 3 ordered axes.
  - Each angle is a rotation about a single Cartesian axis.
  - Concatenating 3 rotations to create a multi-DOF joint.
  - For example, $\mathbf{R}_x(30)\mathbf{R}_y(60)\mathbf{R}_z(90)$ indicates rotating about x-axis by $30°$, about y-axis by $60°$, and finally about z-axis by $90°$.
- Different rotating orders result in different orientations
  - For example, $\mathbf{R}_x(30)\mathbf{R}_y(60)\mathbf{R}_z(90)$ is different from $\mathbf{R}_z(90)\mathbf{R}_y(60)\mathbf{R}_x(30)$

Fixed angle and Euler angle

Two different conventions to rotate $\mathbf{R}_x(30)\mathbf{R}_y(60)\mathbf{R}_z(90)$

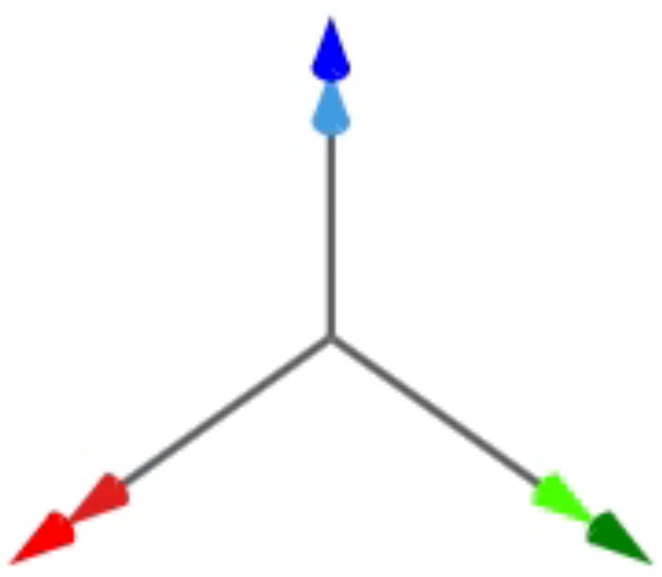
Fixed angle: axes do not move with object

Fixed angle and Euler angle

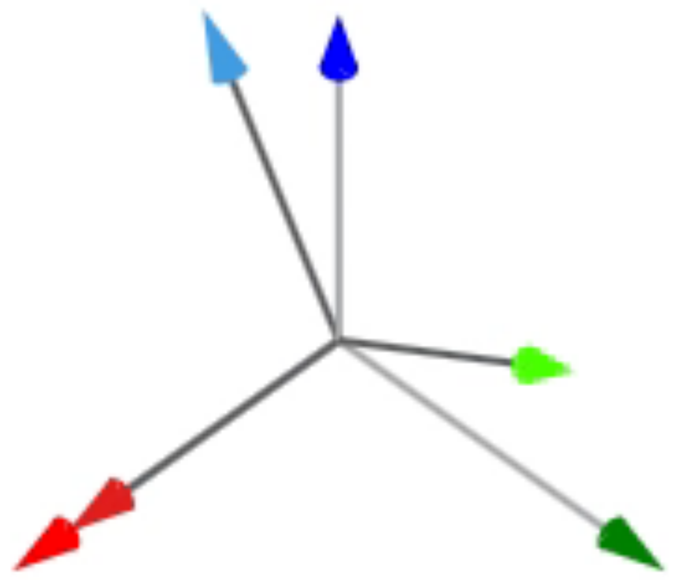
Two different conventions to rotate $\mathbf{R}_x(30)\mathbf{R}_y(60)\mathbf{R}_z(90)$

Fixed angle: axes do not move with object

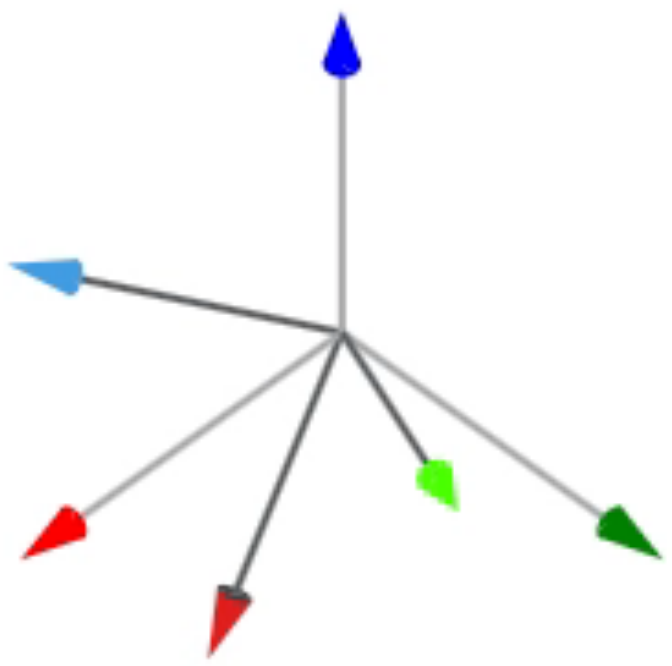
$\mathbf{R}_x^F(30)$



$\mathbf{R}_y^F(60)$



$\mathbf{R}_z^F(90)$

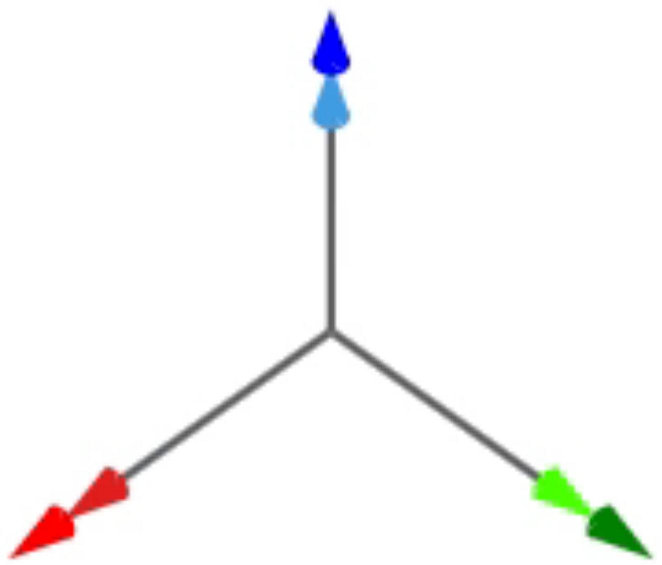


Fixed angle and Euler angle

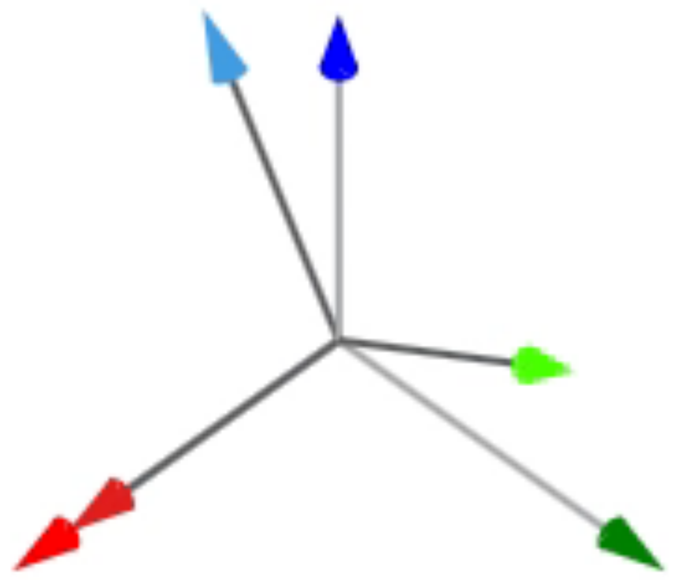
Two different conventions to rotate $\mathbf{R}_x(30)\mathbf{R}_y(60)\mathbf{R}_z(90)$

Fixed angle: axes do not move with object

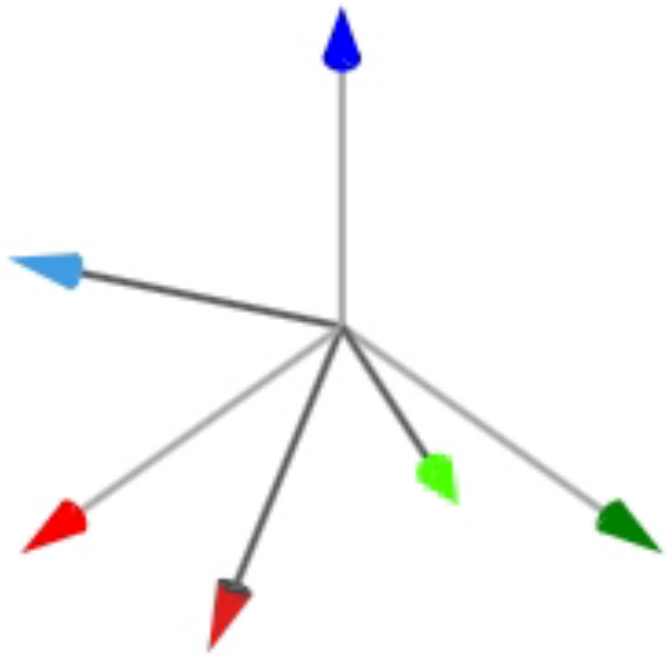
$\mathbf{R}_x^F(30)$



$\mathbf{R}_y^F(60)$

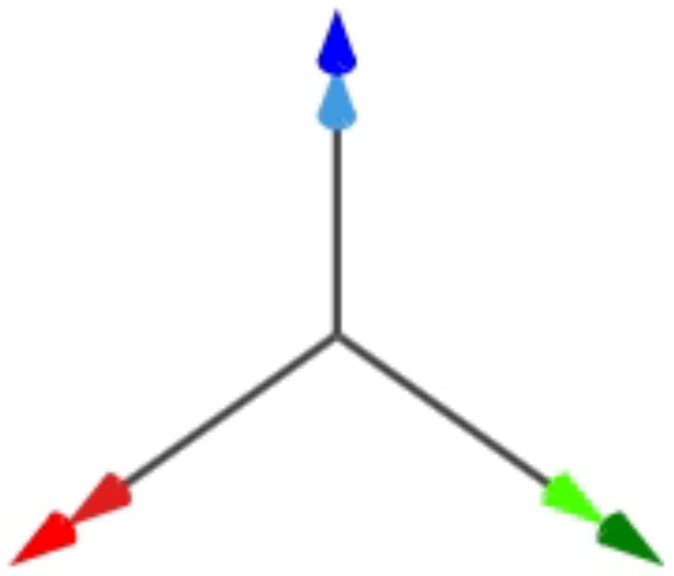


$\mathbf{R}_z^F(90)$

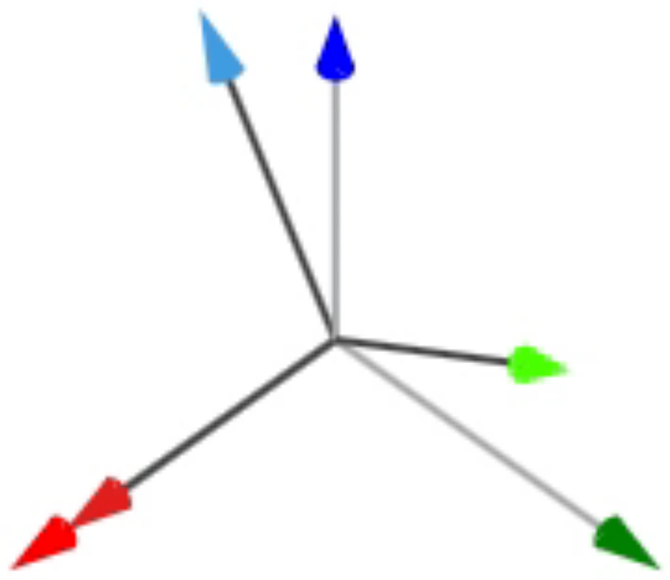


Euler angle: axes move with object

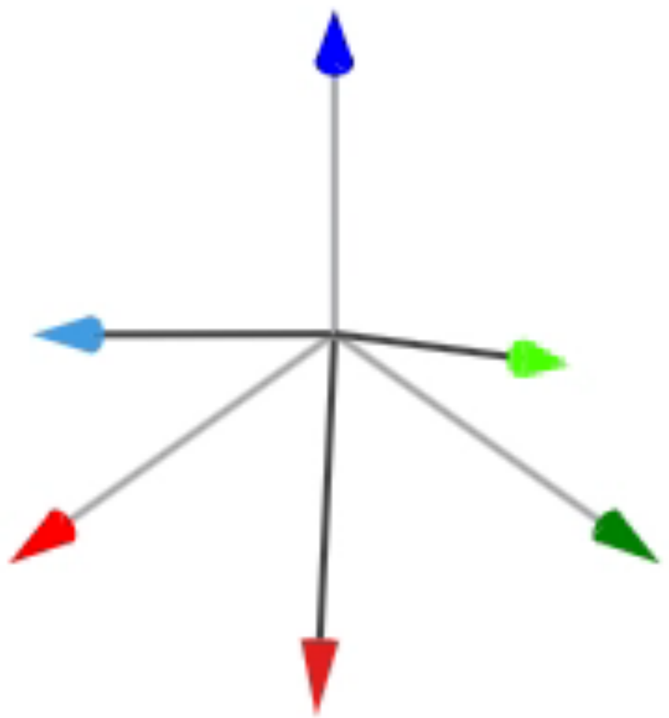
$\mathbf{R}_x^E(30)$



$\mathbf{R}_y^E(60)$



$\mathbf{R}_z^E(90)$



Fixed angle and Euler angle

- Euler angle is the same as fixed angle, except now the axes move with the object.
- Euler angle rotations about moving axes written in reverse order are the same as the fixed axis rotations.

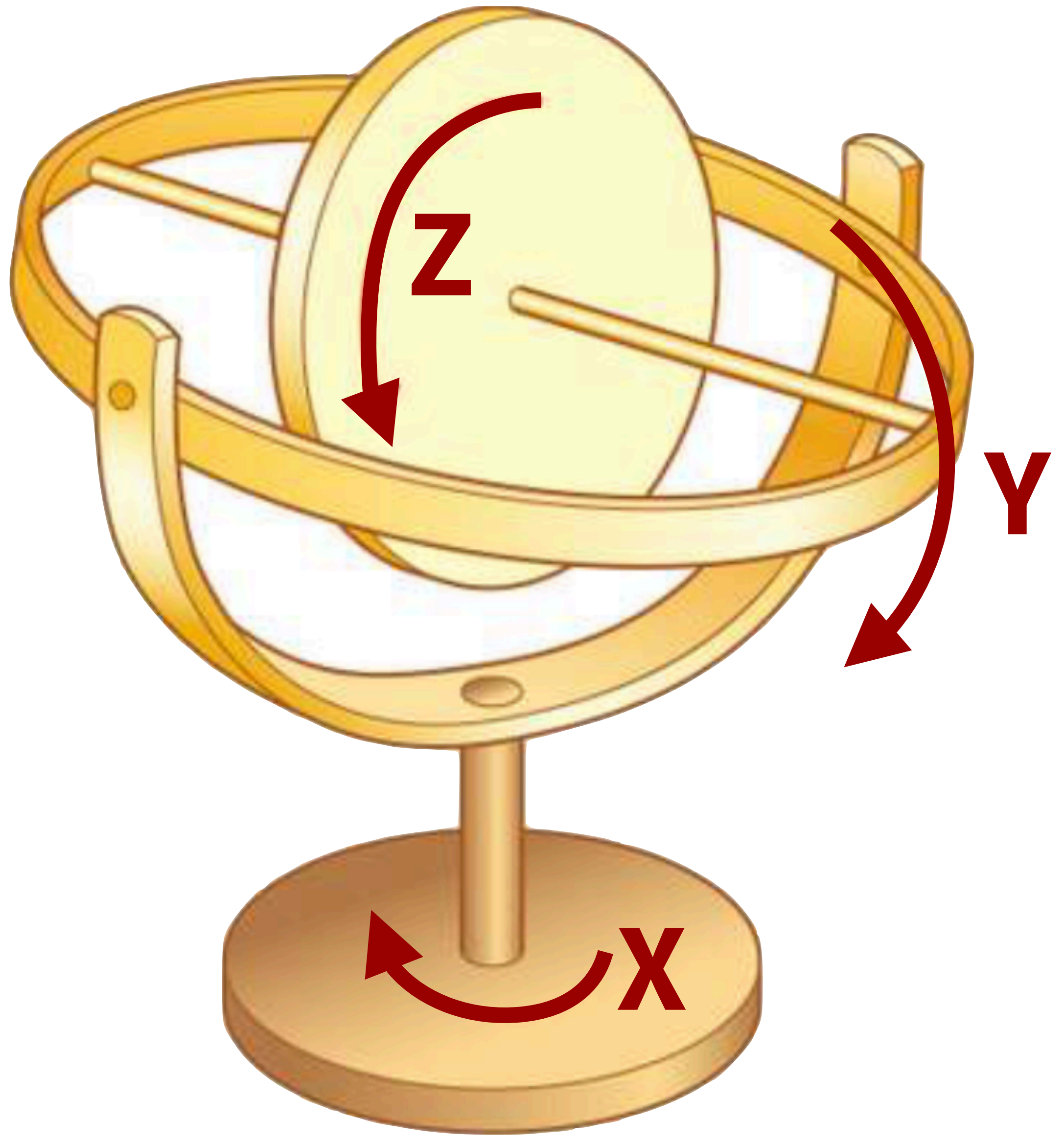
$$\mathbf{R}_x^F(30)\mathbf{R}_y^F(60)\mathbf{R}_z^F(90) = \mathbf{R}_z^E(90)\mathbf{R}_y^E(60)\mathbf{R}_x^E(30)$$

Fixed angle and Euler angle

- Euler angle is the same as fixed angle, except now the axes move with the object.
- Euler angle rotations about moving axes written in reverse order are the same as the fixed axis rotations.

$$\mathbf{R}_x^F(30)\mathbf{R}_y^F(60)\mathbf{R}_z^F(90) = \mathbf{R}_z^E(90)\mathbf{R}_y^E(60)\mathbf{R}_x^E(30)$$

If we rotate the gyroscope in XYZ order, are we using Fixed angle or Euler angle convention?



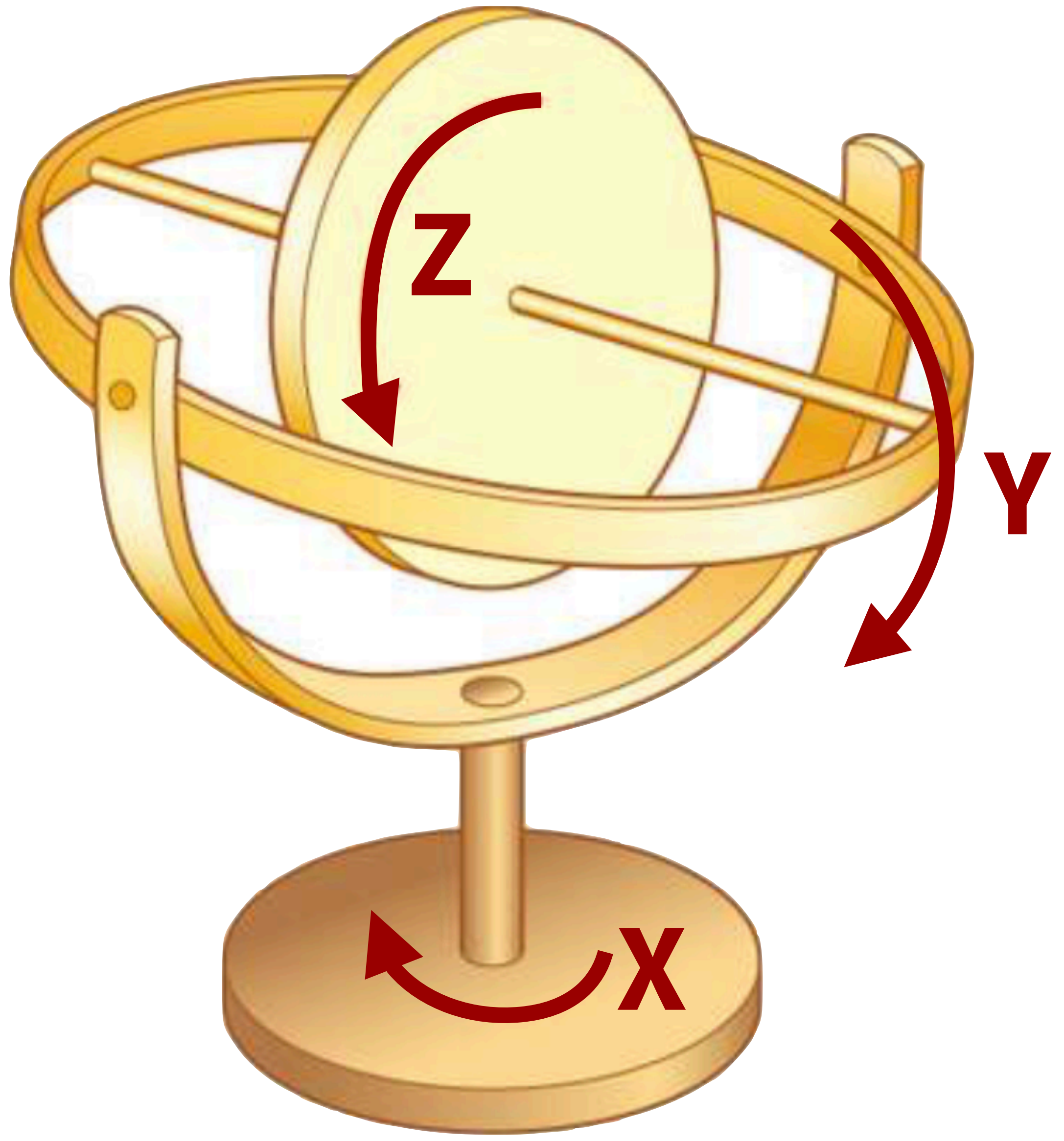
Fixed angle and Euler angle

- Euler angle is the same as fixed angle, except now the axes move with the object.
- Euler angle rotations about moving axes written in reverse order are the same as the fixed axis rotations.

$$\mathbf{R}_x^F(30)\mathbf{R}_y^F(60)\mathbf{R}_z^F(90) = \mathbf{R}_z^E(90)\mathbf{R}_y^E(60)\mathbf{R}_x^E(30)$$

If we rotate the gyroscope in XYZ order, are we using Fixed angle or Euler angle convention?

Euler angle.



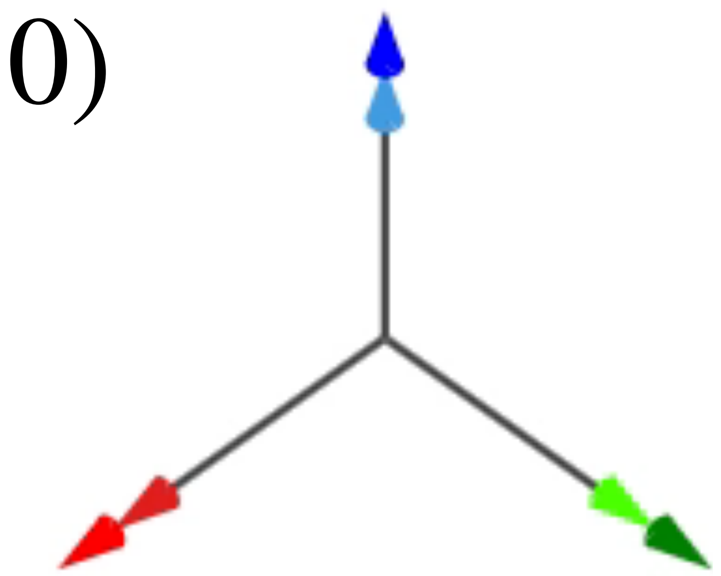
Properties of Fixed/Euler angle

- **Easy to compose?** ✓, but can only consolidate consecutive rotations about the same axis.
- **Easy to interpolate?** ✓, but interpolating each axis separately might not lead to a smooth interpolated path.
- **Easy to enforce joint limits?** ✓
- **What seems to be the problem?** Gimbal lock!

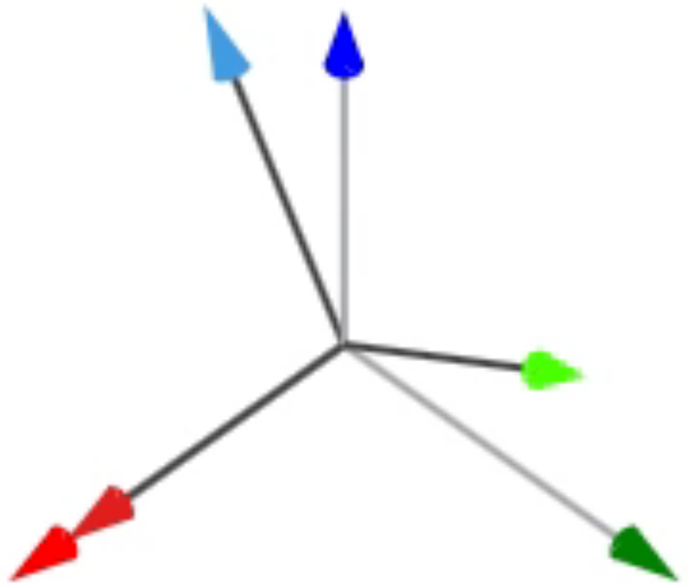
Gimbal lock

When two rotational axis of an object pointing in the same direction, the rotation ends up losing one degree of freedom.

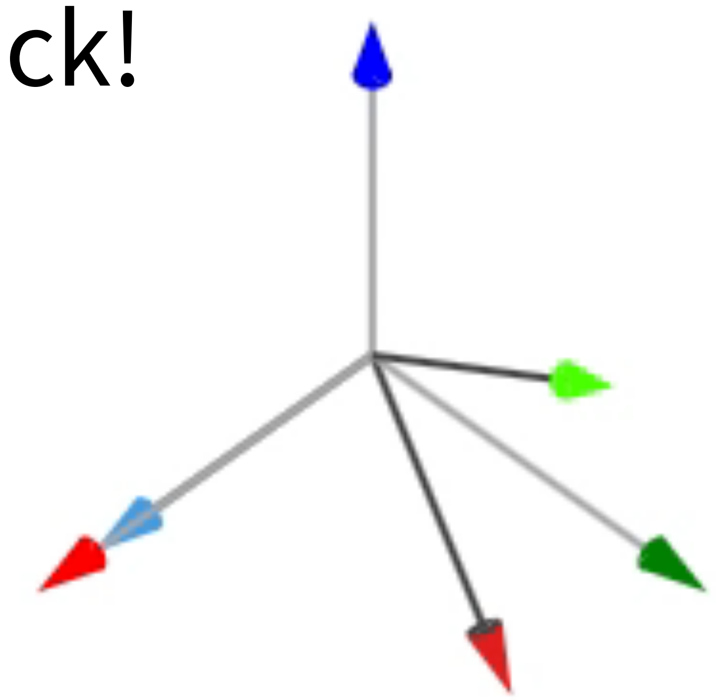
$\mathbf{R}_x^E(30)$



$\mathbf{R}_y^E(90)$



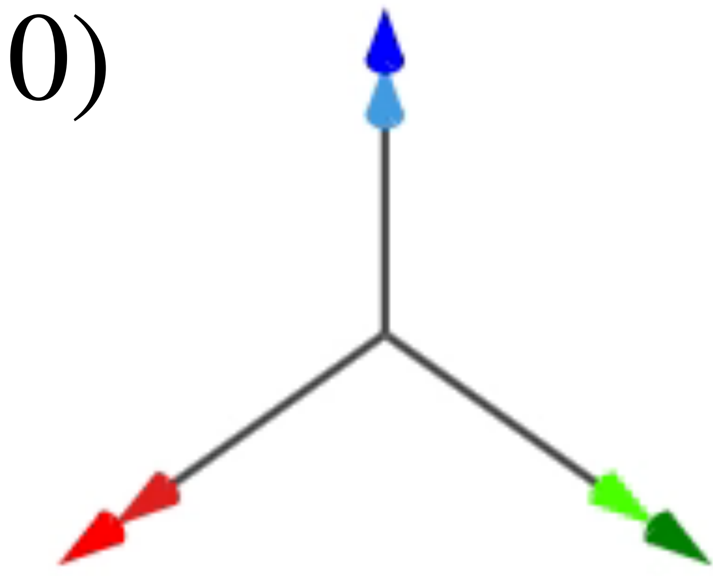
3rd axis aligned with 1st axis
Gimbal lock!



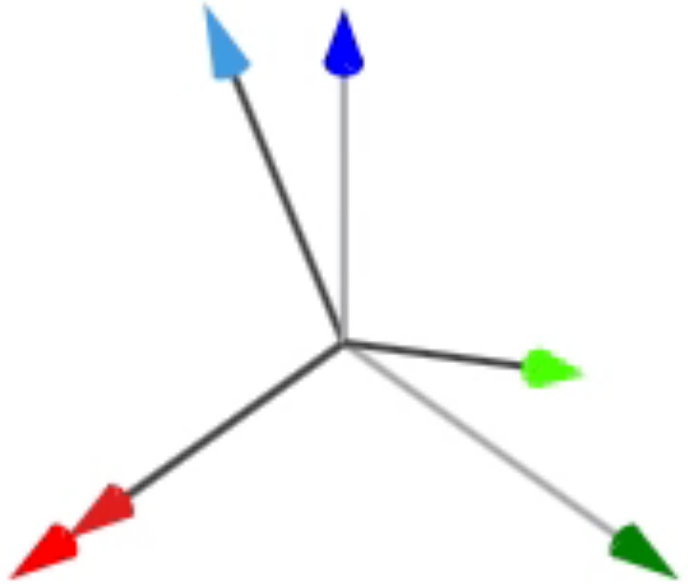
Gimbal lock

When two rotational axis of an object pointing in the same direction, the rotation ends up losing one degree of freedom.

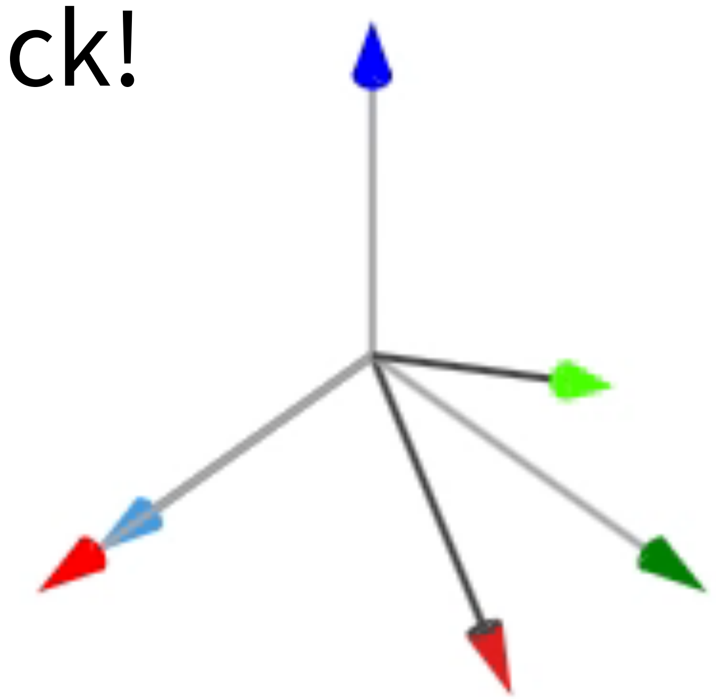
$\mathbf{R}_x^E(30)$

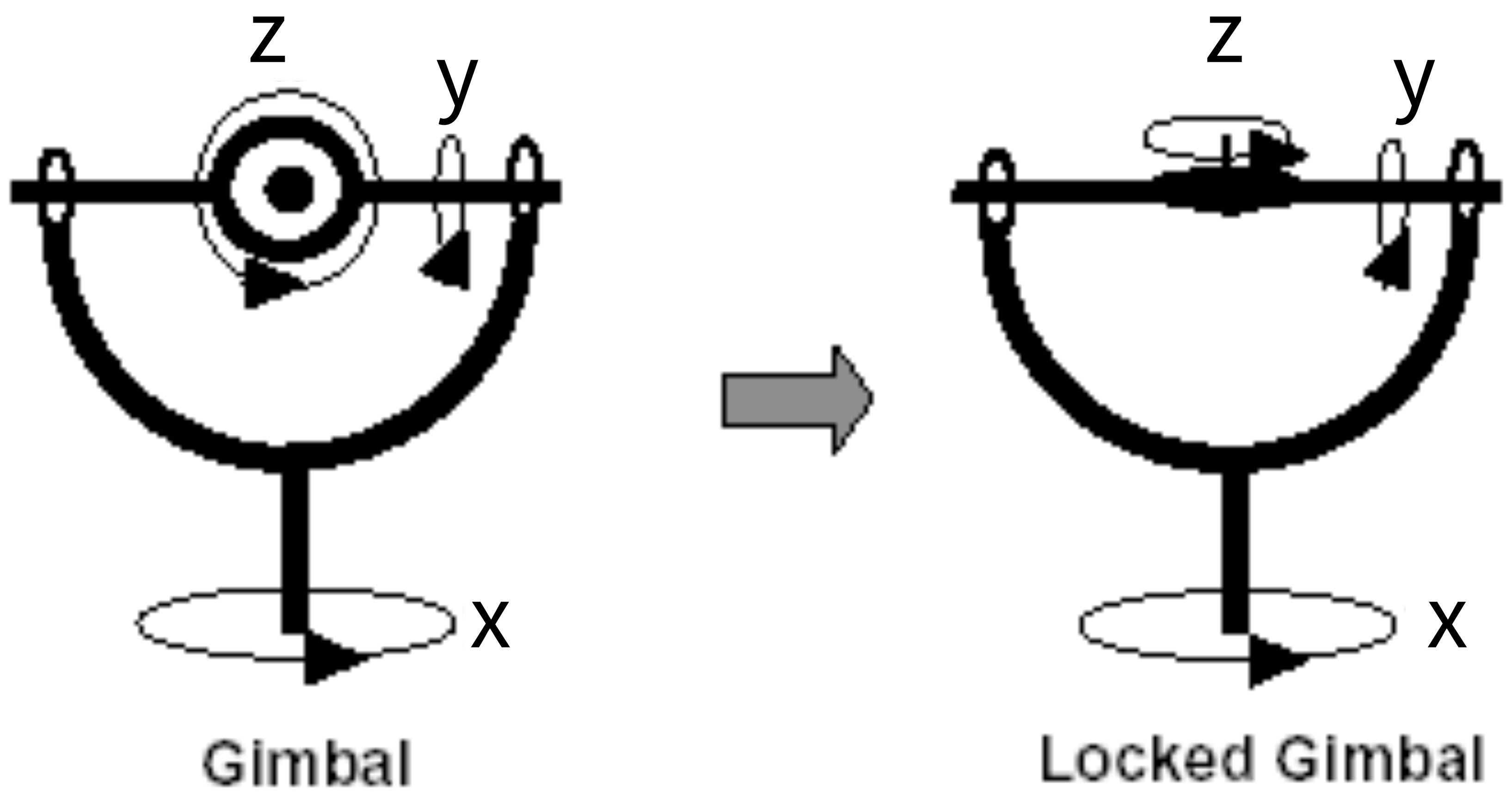


$\mathbf{R}_y^E(90)$



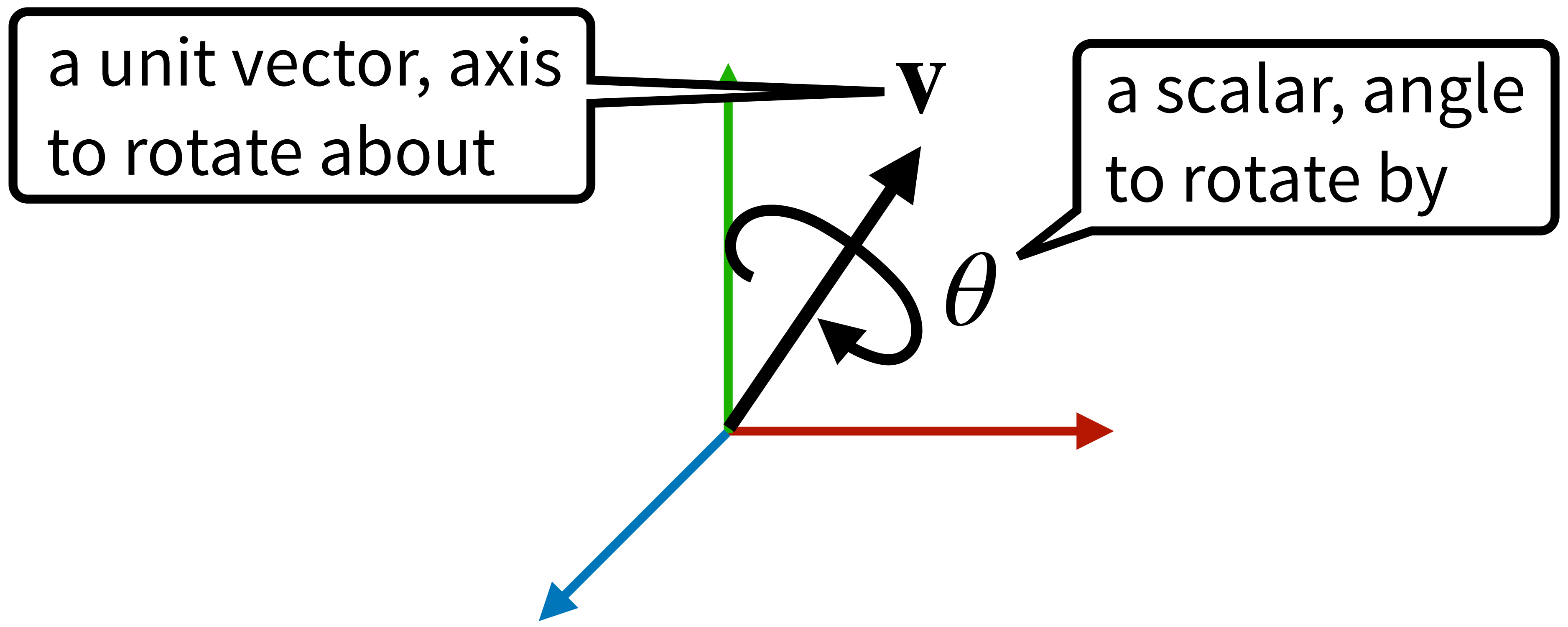
3rd axis aligned with 1st axis
Gimbal lock!





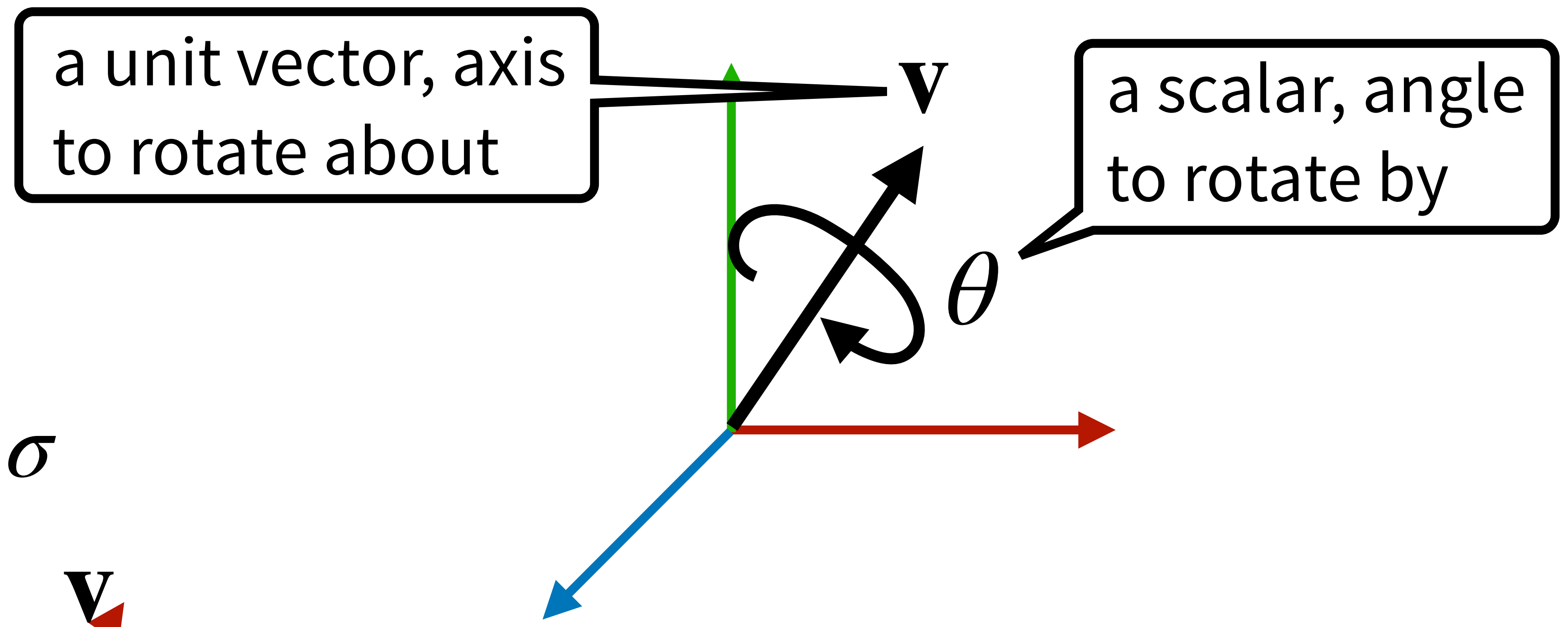
Axis angle

Represent orientation as a scalar and a vector $(\theta, \mathbf{v})$



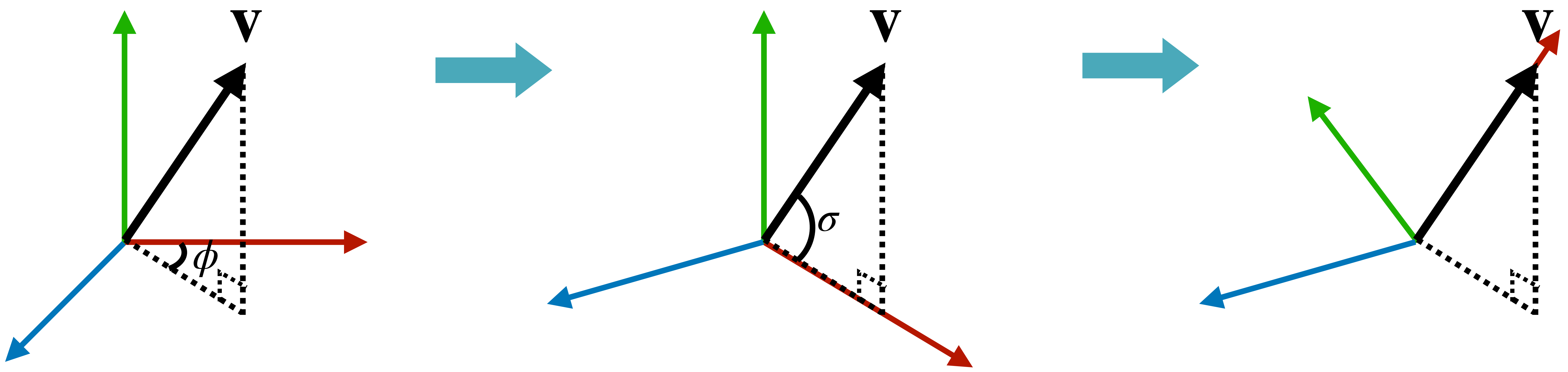
Axis angle

Represent orientation as a scalar and a vector $(\theta, \mathbf{v})$



$\mathbf{R}_y(\phi)$: rotate about y by ϕ

$\mathbf{R}_z(\sigma)$: rotate about z by σ



Now that x-axis is aligned with $\mathbf{v}$, we can rotate θ about x axis, $\mathbf{R}_x(\theta)$.

$$\mathbf{R}_y(\phi)\mathbf{R}_z(\sigma)\mathbf{R}_x(\theta) = \mathbf{R}_\mathbf{v}(\theta) = \begin{bmatrix} \cos\theta + v_x^2(1-\cos\theta) & v_xv_y(1-\cos\theta) - v_z\sin\theta & v_xv_z(1-\cos\theta) + v_y\sin\theta \\ v_yv_x(1-\cos\theta) + v_z\sin\theta & \cos\theta + v_y^2(1-\cos\theta) & v_yv_z(1-\cos\theta) - v_x\sin\theta \\ v_zv_x(1-\cos\theta) - v_y\sin\theta & v_zv_y(1-\cos\theta) + v_x\sin\theta & \cos\theta + v_z^2(1-\cos\theta) \end{bmatrix}$$

Axis angle

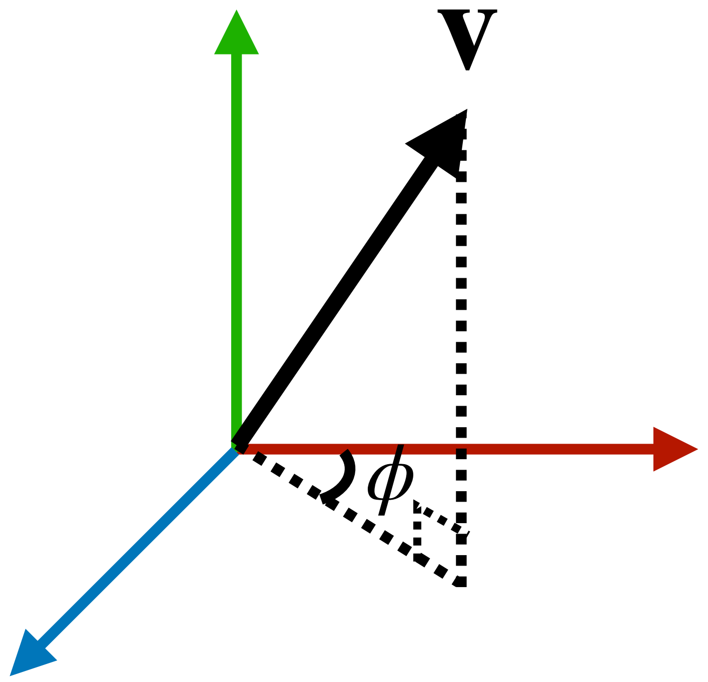
Represent orientation as a scalar and a vector $(\theta, \mathbf{v})$

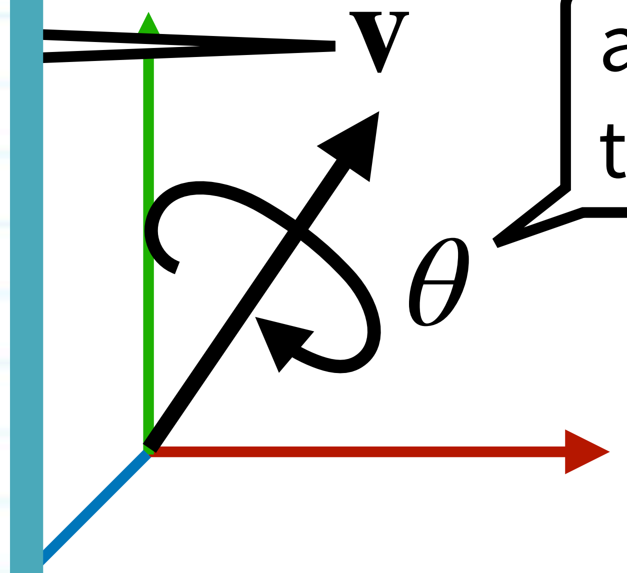
What is the 3D rotation matrix $\mathbf{R}$ for a rotation by θ about axis $\mathbf{k}$?

$$
\begin{aligned}
\mathbf{v}_{\text{rot}} &= \mathbf{v}_{\|} + \mathbf{v}_{\perp\text{rot}} \\
&= \left(\mathbf{v} - \mathbf{v}_{\perp}\right) + \mathbf{v}_{\perp\text{rot}} \\
&= \left(\mathbf{v} + \mathbf{k} \times (\mathbf{k} \times \mathbf{v})\right) + \left(-\cos\theta\, \mathbf{k} \times (\mathbf{k} \times \mathbf{v}) + \sin\theta\, \mathbf{k} \times \mathbf{v}\right) \\
&= \mathbf{v} + (1 - \cos\theta)\, \mathbf{k} \times (\mathbf{k} \times \mathbf{v}) + \sin\theta\, \mathbf{k} \times \mathbf{v} \\
&= \left[\mathbf{I} + (1 - \cos\theta)\, \mathbf{K}^2 + \sin\theta\, \mathbf{K}\right] \mathbf{v} \quad \text{where} \quad \mathbf{K} \equiv \text{"}\mathbf{k}\times\text{"} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \\
&= \mathbf{R}\,\mathbf{v}
\end{aligned}
$$

We just derived the famous Rodrigues' Formula!!!

$\mathbf{R}_y(\phi)$: rot





a scalar, angle to rotate by

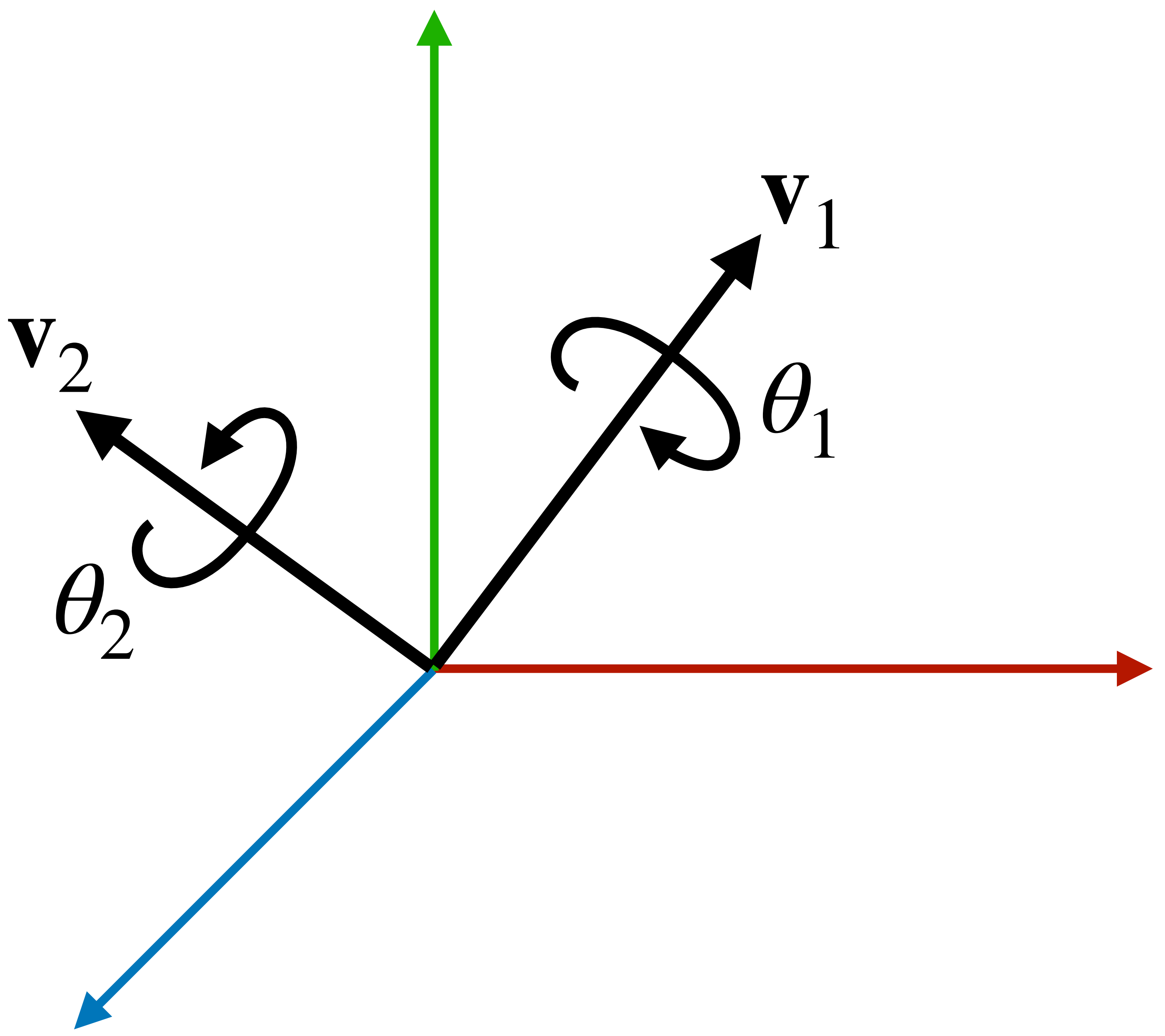
axis is aligned

an rotate θ

, $\mathbf{R}_x(\theta)$.

$$
\mathbf{R}_y(\phi)\mathbf{R}_z(\sigma)\mathbf{R}_x(\theta) = \mathbf{R}_{\mathbf{v}}(\theta) = \begin{bmatrix} \cos\theta + v_x^2(1-\cos\theta) & v_x v_y(1-\cos\theta) - v_z\sin\theta & v_x v_z(1-\cos\theta) + v_y\sin\theta \\ v_y v_x(1-\cos\theta) + v_z\sin\theta & \cos\theta + v_y^2(1-\cos\theta) & v_y v_z(1-\cos\theta) - v_x\sin\theta \\ v_z v_x(1-\cos\theta) - v_y\sin\theta & v_z v_y(1-\cos\theta) + v_x\sin\theta & \cos\theta + v_z^2(1-\cos\theta) \end{bmatrix}
$$

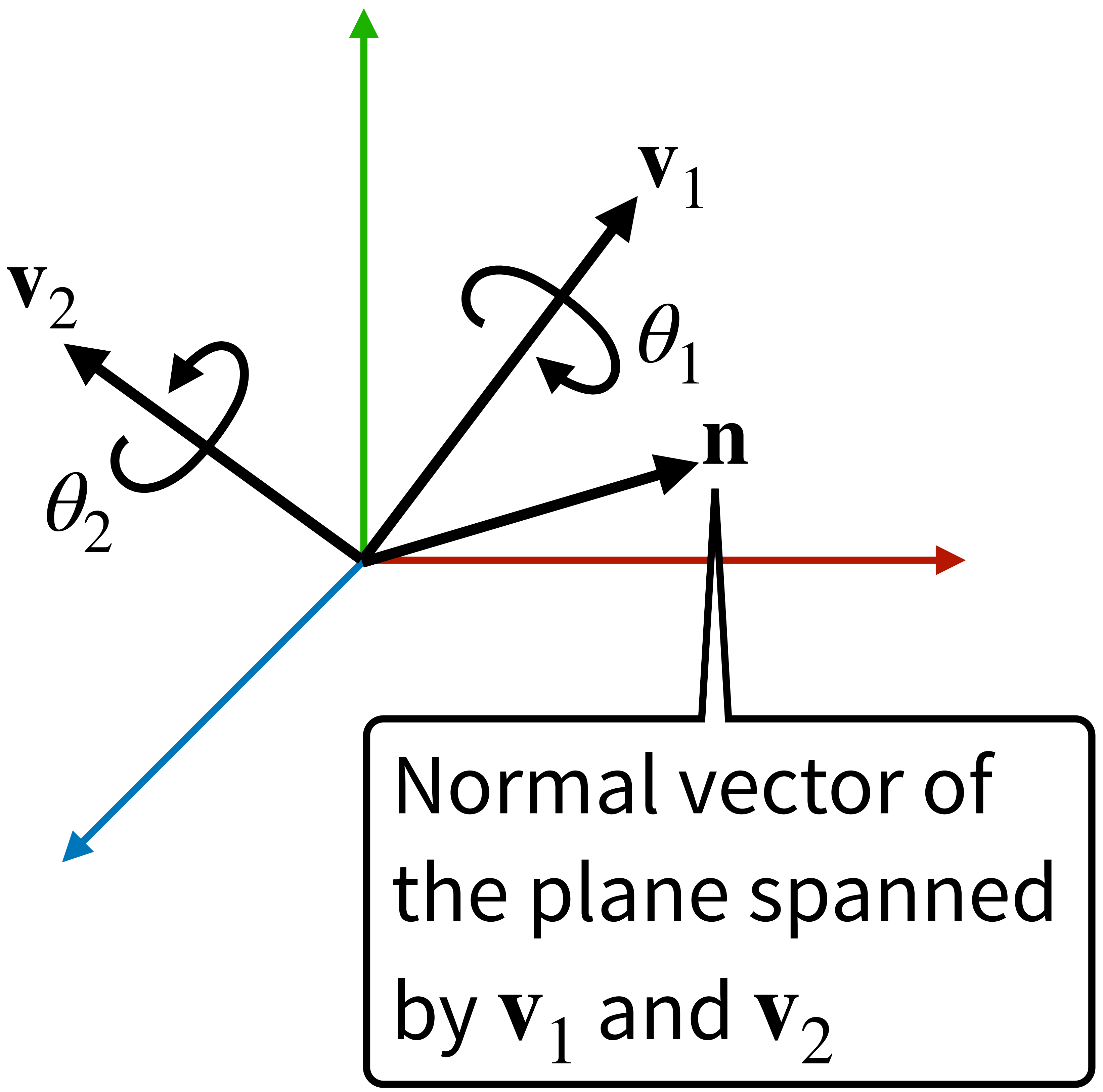
Axis angle interpolation



- **Scalar interpolation:** $\theta_k = (1-k)\theta_1 + k\theta_2$**,** **where** $0 \leq k \leq 1$
- **Vector interpolation is more involved**

Axis angle interpolation

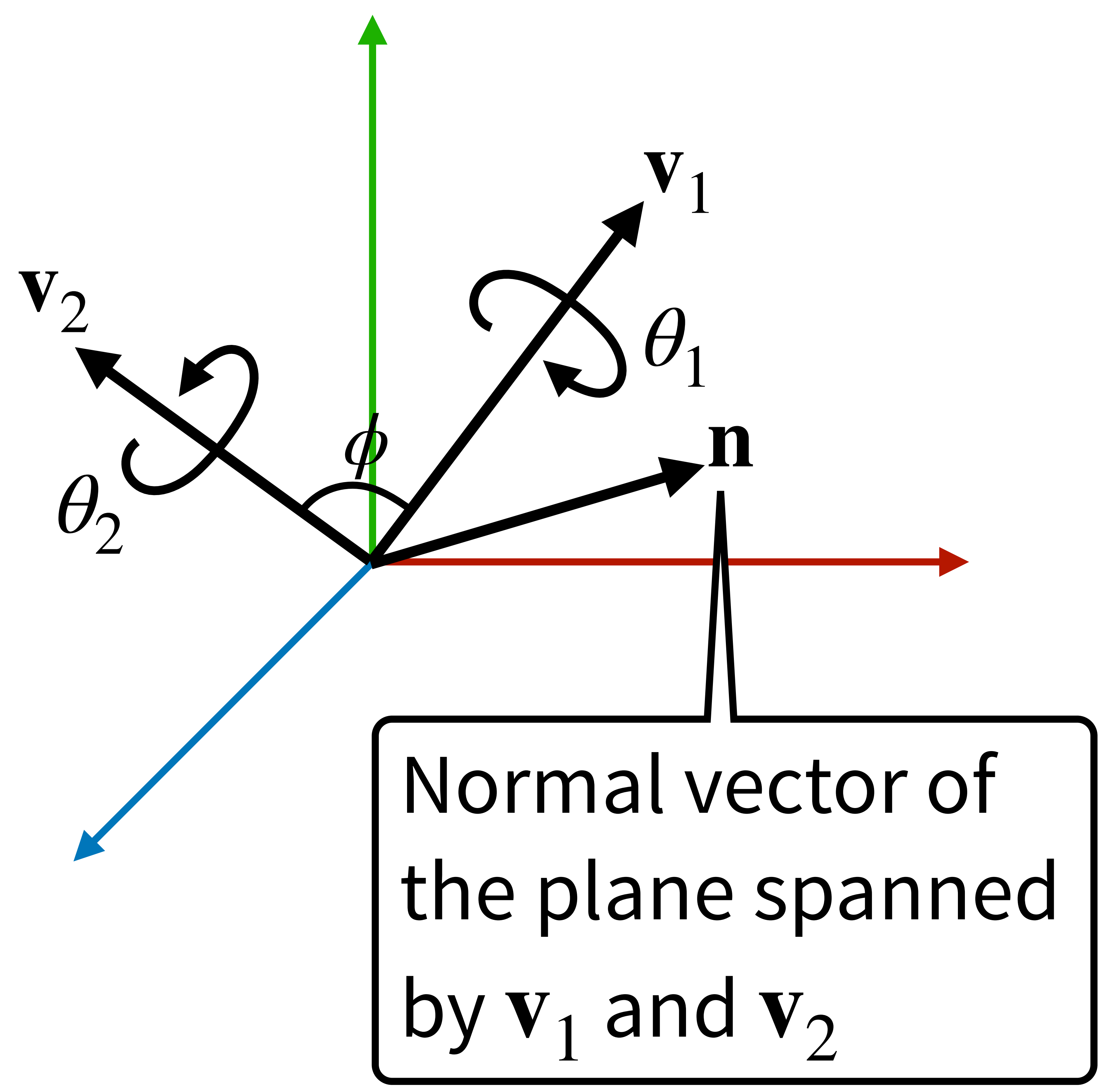
- **Scalar interpolation:** $\theta_k = (1-k)\theta_1 + k\theta_2$, **where** $0 \leq k \leq 1$
- **Vector interpolation is more involved**



$$\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$$

Axis angle interpolation

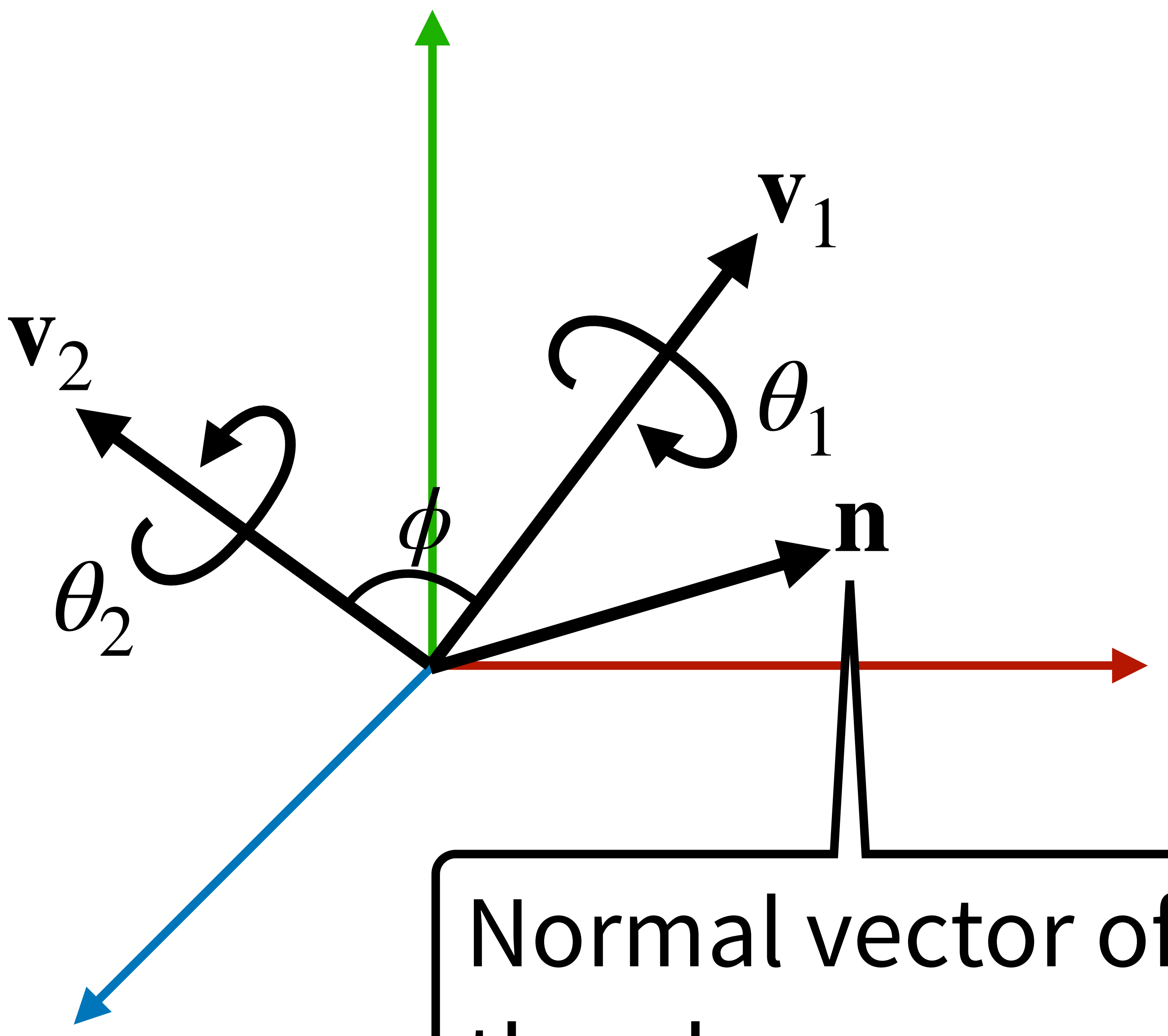
- **Scalar interpolation:** $\theta_k = (1-k)\theta_1 + k\theta_2$, **where** $0 \leq k \leq 1$
- **Vector interpolation is more involved**



$$\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$$

$$\phi = \cos^{-1}(\mathbf{v}_1 \cdot \mathbf{v}_2)$$

Axis angle interpolation



Normal vector of the plane spanned by $\mathbf{v}_1$ and $\mathbf{v}_2$

- **Scalar interpolation:** $\theta_k = (1-k)\theta_1 + k\theta_2$, where $0 \le k \le 1$
- **Vector interpolation is more involved**

$$\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$$

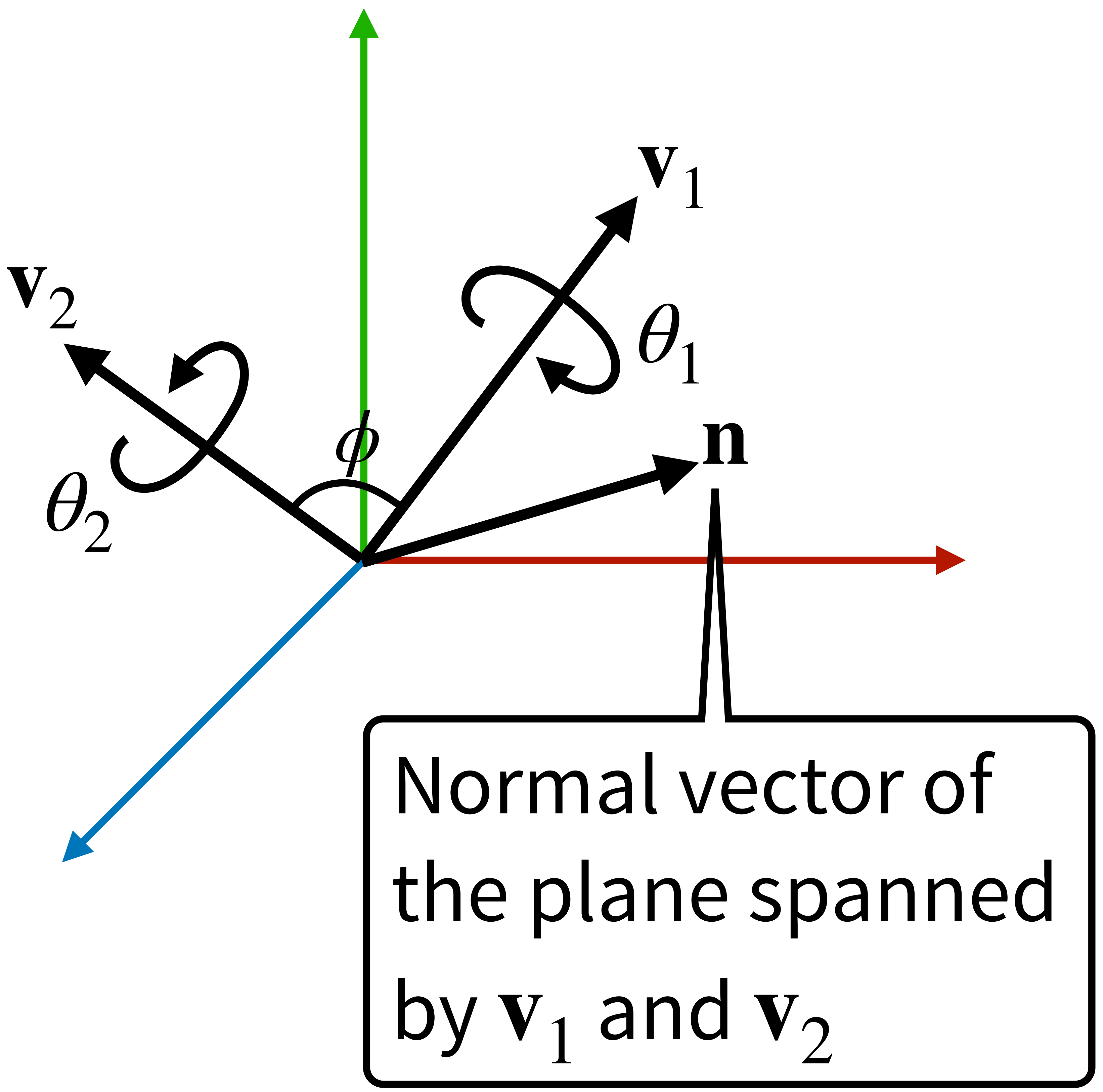
$$\phi = \cos^{-1}(\mathbf{v}_1 \cdot \mathbf{v}_2)$$

$$\mathbf{v}_k = \mathbf{R}_\mathbf{n}(k\phi)\mathbf{v}_1$$

Convert axis angle $(k\phi, \mathbf{n})$ to matrix $\mathbf{R}_\mathbf{n}(k\phi)$

Axis angle interpolation

- **Scalar interpolation:** $\theta_k = (1-k)\theta_1 + k\theta_2$, **where** $0 \le k \le 1$
- **Vector interpolation is more involved**



$$\mathbf{n} = \mathbf{v}_1 \times \mathbf{v}_2$$

$$\phi = \cos^{-1}(\mathbf{v}_1 \cdot \mathbf{v}_2)$$

$$\mathbf{v}_k = \mathbf{R}_\mathbf{n}(k\phi)\mathbf{v}_1 = \frac{\sin\big((1-k)\phi\big)}{\sin\phi}\mathbf{v}_1 + \frac{\sin(k\phi)}{\sin\phi}\mathbf{v}_2$$

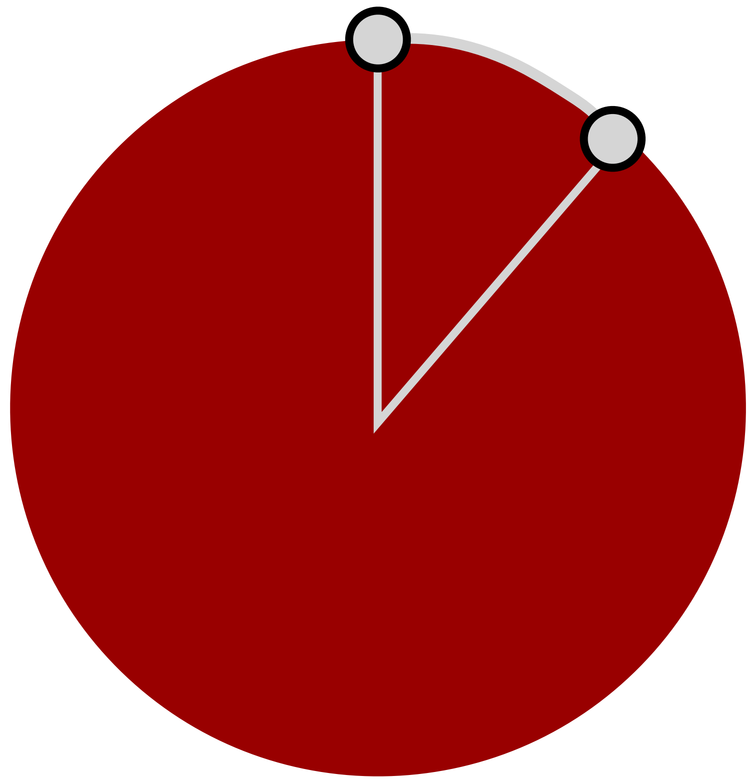
Or, we can use Spherical Linear Interpolation (SLERP) formula.

Convert axis angle $(k\phi, \mathbf{n})$ to matrix $\mathbf{R}_\mathbf{n}(k\phi)$

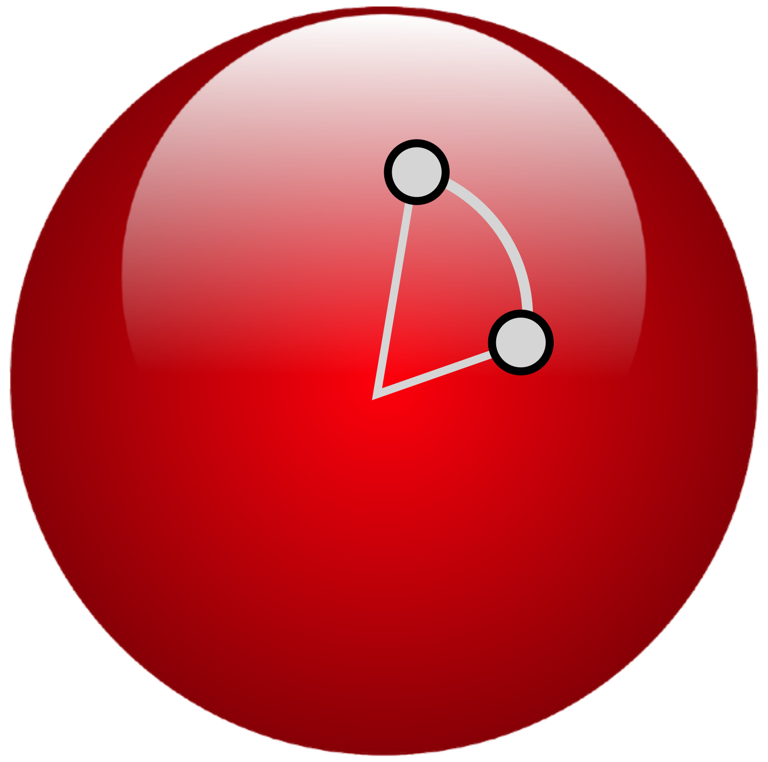
Properties of axis angle

- Easy to compose? ✗ Must convert back to matrix form.
- Easy to interpolate? ✓
- Easy to enforce joint limits? ✓
- Avoid gimbal lock? ✓ It rotates all three axes at once.

Quaternion: geometric view



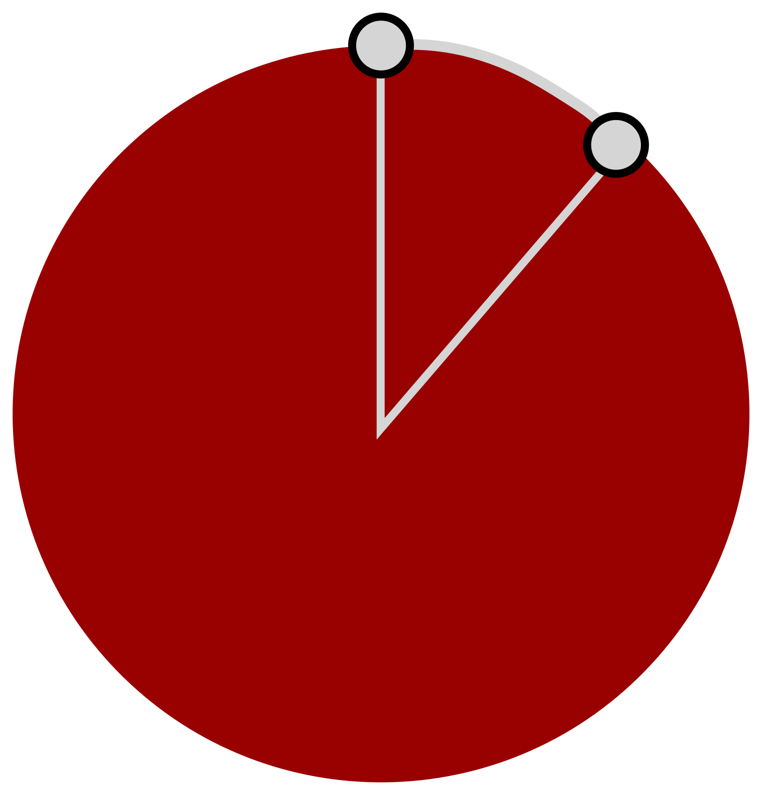
1-angle rotation can be represented by a unit circle



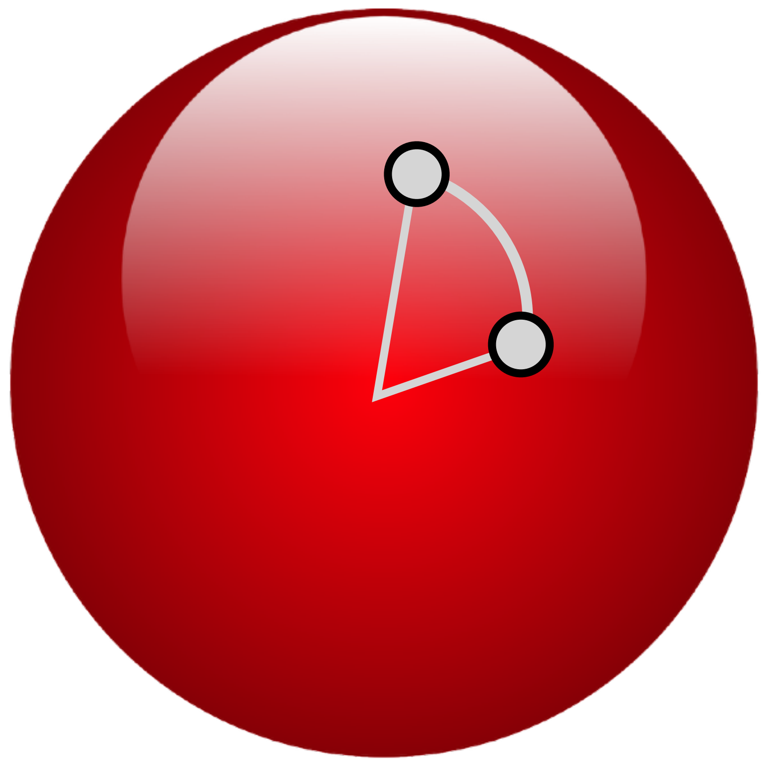
2-angle rotation can be represented by a unit sphere

What about 3-angle rotation?

Quaternion: geometric view



1-angle rotation can be
represented by a unit circle



2-angle rotation can be
represented by a unit sphere

What about 3-angle rotation?

A unit quaternion is a point on the 4D sphere!

Quaternion: algebraic view

- **A quaternion can be represented by a 4-tuple of real numbers:** w, x, y, z

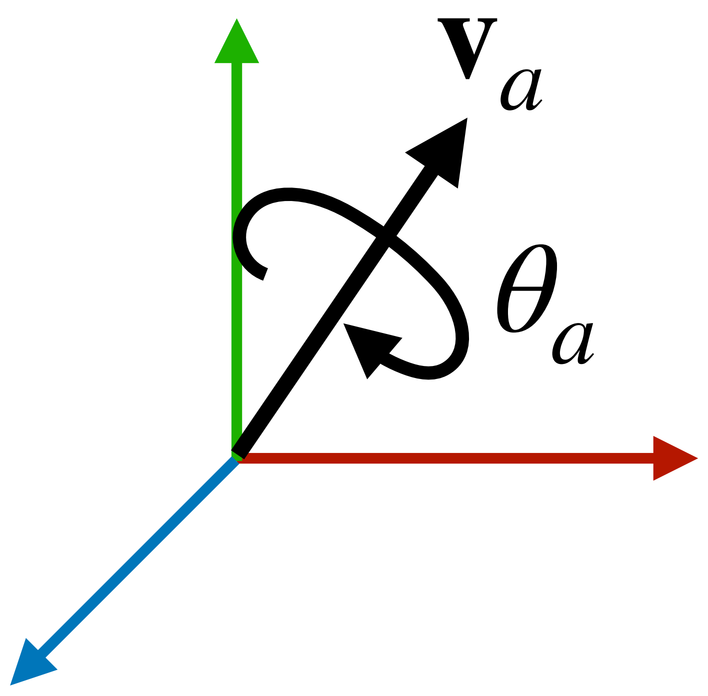
$$\mathbf{q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} \begin{array}{l} \text{scalar} \\ \text{3D vector} \end{array}$$

Quaternion: algebraic view

- **A quaternion can be represented by a 4-tuple of real numbers:** w, x, y, z

$$\mathbf{q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} \begin{matrix} \text{scalar} \\ \text{3D vector} \end{matrix}$$

- **Same information as axis angles but in a different form:**



axis angle $(\theta_a, \mathbf{v}_a)$ ➡

$$\mathbf{q} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta_a}{2} \\ \sin\frac{\theta_a}{2}\mathbf{v}_a \end{bmatrix}$$

Quaternion basic definitions

- **Unit quaternion**

$$\|\mathbf{q}\| = \sqrt{x^2 + y^2 + z^2 + w^2} = 1$$

- **Inverse quaternion**

$$\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|} \qquad \mathbf{q}^* = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix}^* = \begin{bmatrix} w \\ -\mathbf{v} \end{bmatrix}$$

- **Identity**

$$\mathbf{q}\mathbf{q}^{-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Quaternion multiplication

$$\mathbf{q}_1 \otimes \mathbf{q}_2 = \begin{bmatrix} w_1 \\ \mathbf{v}_1 \end{bmatrix} \begin{bmatrix} w_2 \\ \mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} w_1 w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2 \\ w_1 \mathbf{v}_2 + w_2 \mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2 \end{bmatrix}$$

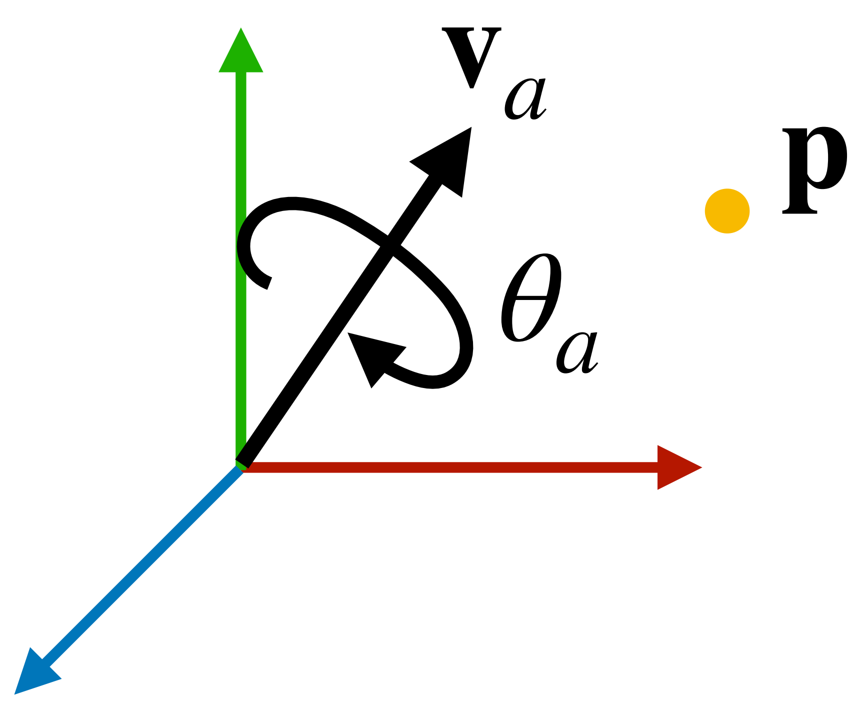
- **Commutativity** ✗

$$\mathbf{q}_1\mathbf{q}_2 \neq \mathbf{q}_2\mathbf{q}_1$$

- **Associativity** ✓

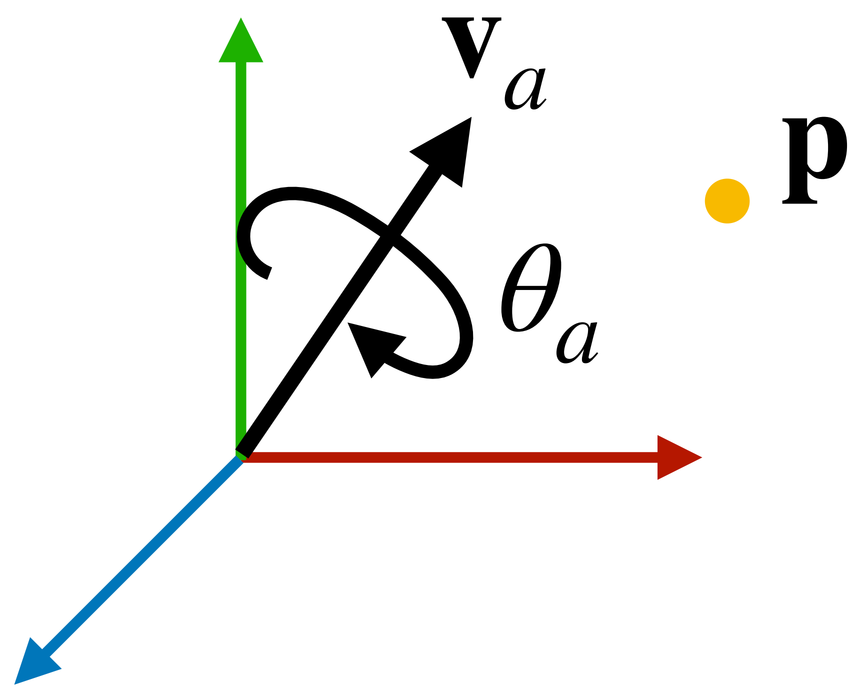
$$\mathbf{q}_1(\mathbf{q}_2\mathbf{q}_3) = (\mathbf{q}_1\mathbf{q}_2)\mathbf{q}_3$$

Quaternion rotation



Let a unit quaternion $\mathbf{q} \equiv \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix}$ be $\begin{bmatrix} \cos\frac{\theta_a}{2} \\ \sin\frac{\theta_a}{2}\mathbf{v}_a \end{bmatrix}$. Let a vector $\mathbf{q}_p \in \mathbb{R}^4$ be $\begin{bmatrix} 0 \\ \mathbf{p} \end{bmatrix}$, where $\mathbf{p}$ is a 3D point.

Quaternion rotation



Let a unit quaternion $\mathbf{q} \equiv \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix}$ be $\begin{bmatrix} \cos\frac{\theta_a}{2} \\ \sin\frac{\theta_a}{2}\mathbf{v}_a \end{bmatrix}$. Let a vector $\mathbf{q}_p \in \mathbb{R}^4$ be $\begin{bmatrix} 0 \\ \mathbf{p} \end{bmatrix}$, where $\mathbf{p}$ is a 3D point.

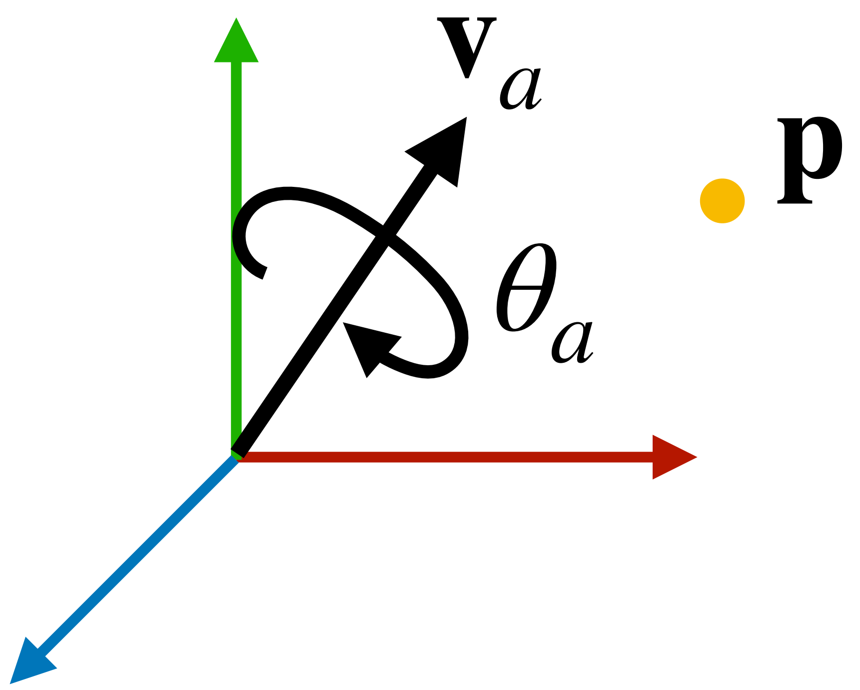
Then $\mathbf{p}'$ is the result of $\mathbf{p}$ rotating about $\mathbf{v}_a$ by θ_a, where

$$\mathbf{q}\mathbf{q}_p\mathbf{q}^{-1} = \begin{bmatrix} 0 \\ \mathbf{p}' \end{bmatrix}.$$

These are "quaternion multiplications", which product is in the form of $[0, \mathbf{p}']$

Proof: see Quaternions by Shoemaker

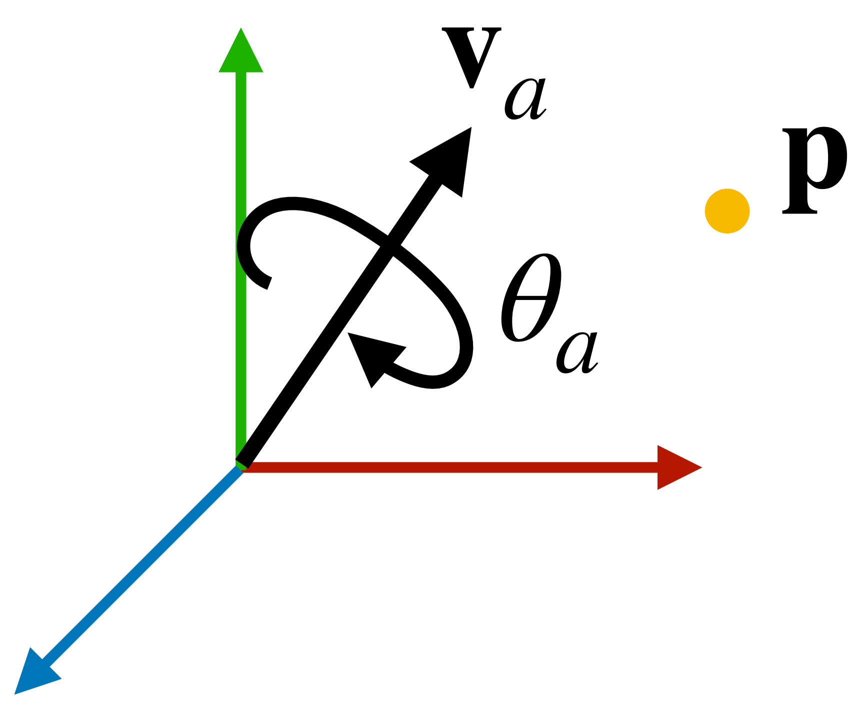
Quaternion rotation



$$\mathbf{q}\mathbf{q}_p\mathbf{q}^{-1} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{p} \end{bmatrix} \begin{bmatrix} w \\ -\mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ w(w\mathbf{p} - \mathbf{p} \times \mathbf{v}) + (\mathbf{p} \cdot \mathbf{v})\mathbf{v} + \mathbf{v} \times (w\mathbf{p} - \mathbf{p} \times \mathbf{v}) \end{bmatrix}$$

where $w = \cos\frac{\theta_a}{2}$ and $\mathbf{v} = \sin\frac{\theta_a}{2}\mathbf{v}_a$

Quaternion rotation



$$\mathbf{q}\mathbf{q}_p\mathbf{q}^{-1} = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{p} \end{bmatrix} \begin{bmatrix} w \\ -\mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ w(w\mathbf{p} - \mathbf{p} \times \mathbf{v}) + (\mathbf{p} \cdot \mathbf{v})\mathbf{v} + \mathbf{v} \times (w\mathbf{p} - \mathbf{p} \times \mathbf{v}) \end{bmatrix}$$

where $w = \cos\dfrac{\theta_a}{2}$ and $\mathbf{v} = \sin\dfrac{\theta_a}{2}\mathbf{v}_a$

Express $w(w\mathbf{p} - \mathbf{p} \times \mathbf{v}) + (\mathbf{p} \cdot \mathbf{v})\mathbf{v} + \mathbf{v} \times (w\mathbf{p} - \mathbf{p} \times \mathbf{v})$ in terms of $\mathbf{v}_a$ and θ_a and compare it with

Convert the axis angle to a rotation matrix

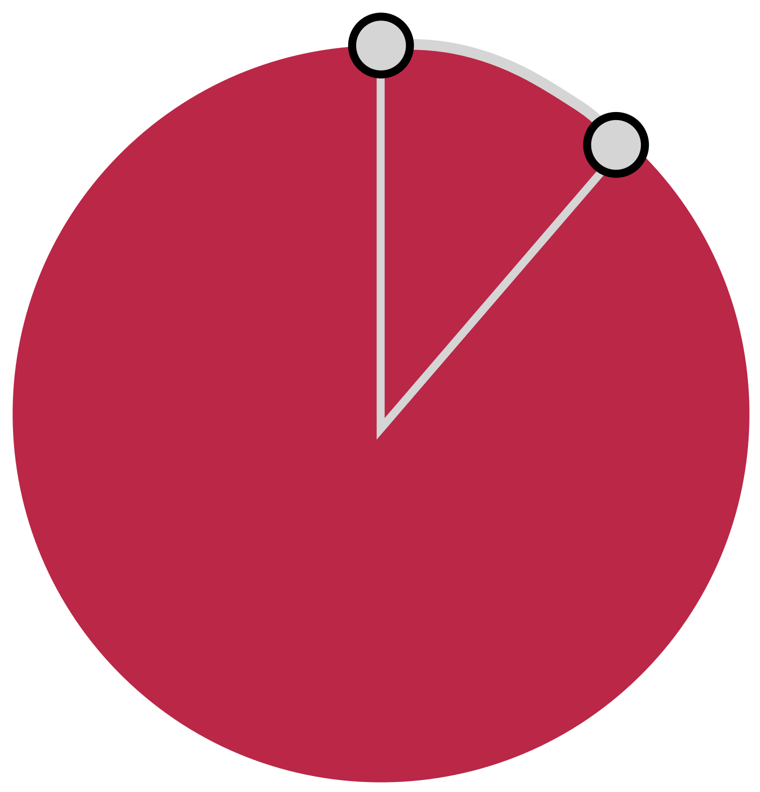
$$\mathbf{R}_{\mathbf{v}_a}(\theta_a)\mathbf{p} = \begin{bmatrix} \cos\theta_a + v_x^2(1-\cos\theta_a) & v_xv_y(1-\cos\theta_a) - v_z\sin\theta_a & v_xv_z(1-\cos\theta_a) + v_y\sin\theta_a \\ v_yv_x(1-\cos\theta_a) + v_z\sin\theta_a & \cos\theta_a + v_y^2(1-\cos\theta_a) & v_yv_z(1-\cos\theta_a) - v_x\sin\theta_a \\ v_zv_x(1-\cos\theta_a) - v_y\sin\theta_a & v_zv_y(1-\cos\theta_a) + v_x\sin\theta_a & \cos\theta_a + v_z^2(1-\cos\theta_a) \end{bmatrix}\mathbf{p}$$

Quaternion composition

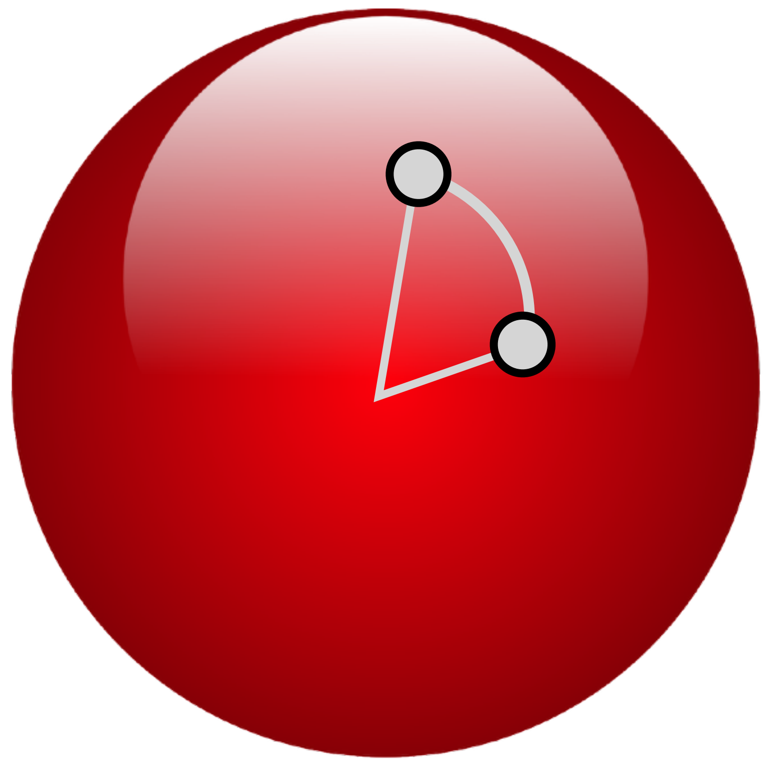
- If $\mathbf{q}_1$ and $\mathbf{q}_2$ are unit quaternions, the combined rotation of first rotating by $\mathbf{q}_1$ and then by $\mathbf{q}_2$ is equivalent to $\mathbf{q}_3 = \mathbf{q}_2\mathbf{q}_1$.
- Quaternion $\mathbf{q} = [w, x, y, z]$ can also be converted to a rotation matrix in homogeneous coordinate.

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1-2y^2-2z^2 & 2xy+2wz & 2xz-2wy & 0 \\ 2xy-2wz & 1-2x^2-2z^2 & 2yz+2wx & 0 \\ 2xz+2wy & 2yz-2wx & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Quaternion interpolation



1-angle rotation can be represented by a unit circle



2-angle rotation can be represented by a unit sphere

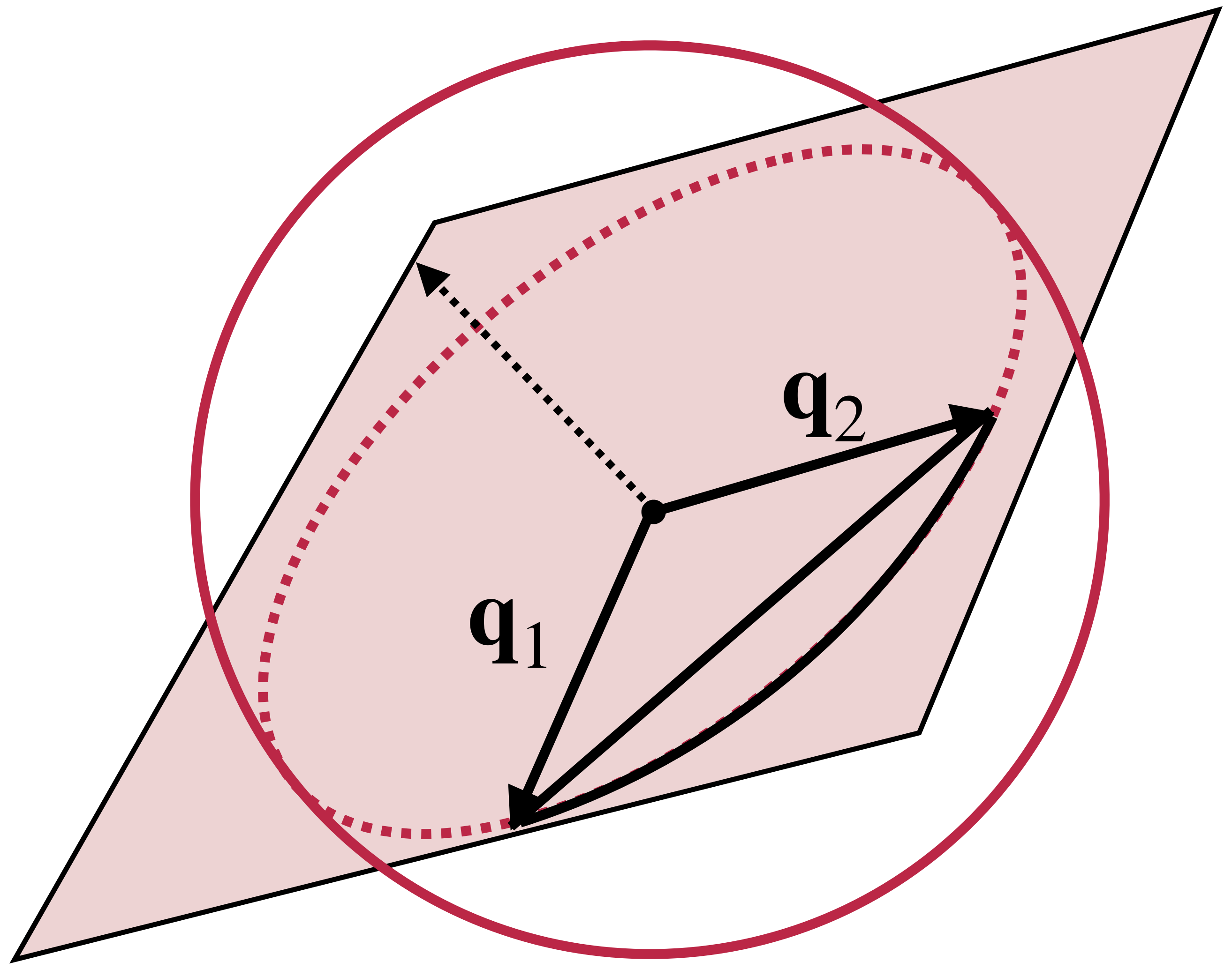
Interpolating quaternion means moving on the surface of 4-D sphere — move with constant angular velocity along the great circle between two points on the 4D unit sphere.

Quaternion interpolation

- Direct linear interpolation does not give motion with constant angular velocity.
- Convert $\mathbf{q}_1$ and $\mathbf{q}_2$ to two axis angles and use Spherical Linear Interpolation (SLERP) to interpolate.

$$\theta_a = 2\cos^{-1}(w), \quad \mathbf{v}_a = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

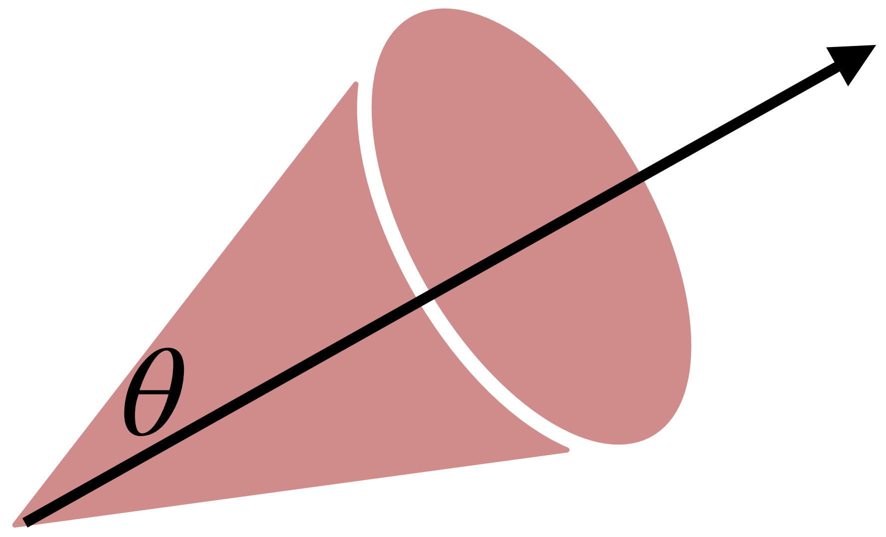
- Use Quaternion SLEPR: $\mathbf{q}_k = \mathbf{q}_1(\mathbf{q}_1^{-1}\mathbf{q}_2)^k$



Interpolating quaternion means moving on the surface of 4-D sphere — move with constant angular velocity along the great circle between two points on the 4D unit sphere.

Quaternion constraints

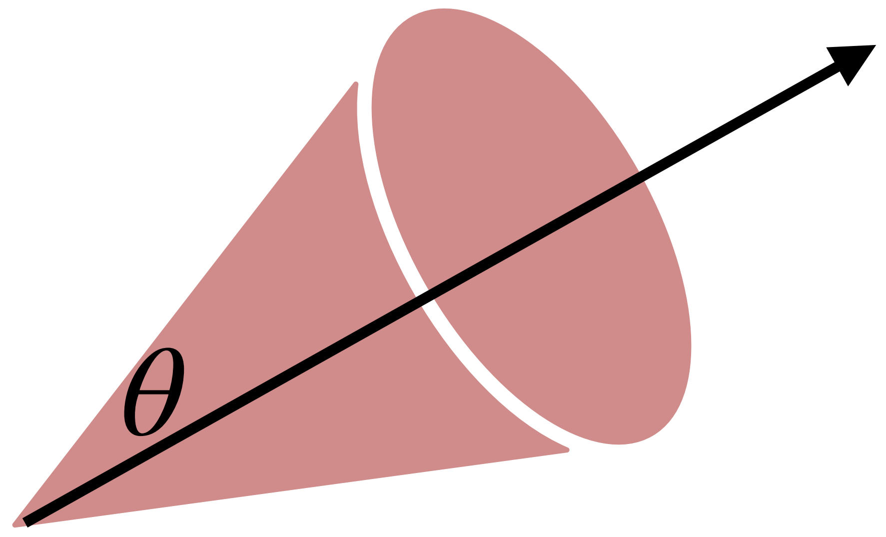
- **Given a quaternion $\mathbf{q} = [w, x, y, z]$, we can enforce**
  - **Cone constraint, e.g. constrain orientations within a cone aligned with x-axis.**



$$\frac{1 - \cos\theta}{2} = y^2 + z^2$$

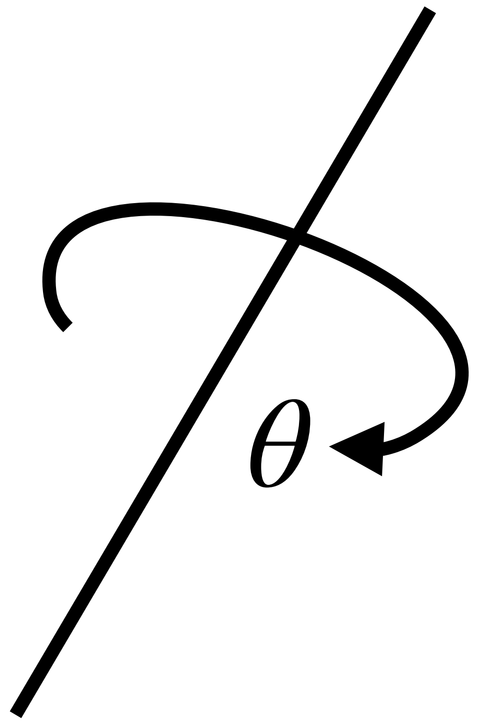
Quaternion constraints

- Given a quaternion $\mathbf{q} = [w, x, y, z]$, we can enforce
  - Cone constraint, e.g. constrain orientations within a cone aligned with x-axis.



$$\frac{1 - \cos\theta}{2} = y^2 + z^2$$

  - Twist constraint, e.g. constrain the twist about x-axis.



$$\tan\frac{\theta}{2} = \frac{x}{w}$$

Properties of quaternion

- Easy to compose? ✓
- Easy to interpolate? ✓
- Easy to enforce joint limits? ✓
- Avoid Gimbal lock? ✓
- Anything bad about quaternion? Need to be normalized.

Quiz

- **What representation is the best for each of the following requirements?**
  - **Intuitive animation input:** Euler/fixed angles
  - **Enforce joint limits:** Euler/fixed angles, axis angle, quaternion
  - **Interpolation:** axis angle, quaternion
  - **Composition:** rotation matrix, quaternion
  - **Avoid gimbal lock:** rotation matrix, axis angle, quaternion
  - **Rendering:** rotation matrix

Back to rigid bodies…

$$\mathbf{Y}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{R}(t) \\ \mathbf{p}(t) \\ \mathbf{L}(t) \end{bmatrix}$$

$$\dot{\mathbf{Y}}(t) = \begin{bmatrix} \mathbf{v}(t) \\ [\boldsymbol{\omega}(t)]\mathbf{R}(t) \\ \mathbf{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix}$$

Given the current state $\mathbf{Y}_n$, how to evaluate $\dot{\mathbf{Y}}_n$, assuming the mass, M, and inertia in the body space, $\mathbf{I}_b$ are known?

$$\mathbf{v}(t) = \frac{\mathbf{p}}{M}$$

$$[\boldsymbol{\omega}(t)]\mathbf{R}(t) = [\mathbf{I}(t)^{-1}\mathbf{L}(t)]\mathbf{R}(t) = [\mathbf{R}^T(t)\mathbf{I}_b^{-1}\mathbf{R}(t)\mathbf{L}(t)]\mathbf{R}(t)$$

How to compute $\mathbf{f}(t)$?

Evaluate all the forces, $\mathbf{f}_1, \cdots, \mathbf{f}_n$ currently applied on the rigid body.

$$\boldsymbol{\tau}(t) = \sum_{i=1}^{n} \big(\mathbf{r}_i(t) - \mathbf{x}(t)\big) \times \mathbf{f}_i(t)\big)$$

Back to rigid bodies…

$$\mathbf{Y}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \cancel{\mathbf{R}(t)} \\ \mathbf{p}(t) \\ \mathbf{L}(t) \end{bmatrix} \quad \mathbf{q}(t)$$

Given the current state $\mathbf{Y}_n$, how to evaluate $\dot{\mathbf{Y}}_n$, assuming the mass, M, and inertia in the body space, $\mathbf{I}_b$ are known?

$$\mathbf{v}(t) = \frac{\mathbf{p}}{M}$$

$$\cancel{[\boldsymbol{\omega}(t)]\mathbf{R}(t) = [\mathbf{I}(t)^{-1}\mathbf{L}(t)]\mathbf{R}(t) = [\mathbf{R}^T(t)\mathbf{I}_b^{-1}\mathbf{R}(t)\mathbf{L}(t)]\mathbf{R}(t)}$$

$$\dot{\mathbf{Y}}(t) = \begin{bmatrix} \mathbf{v}(t) \\ \cancel{[\boldsymbol{\omega}(t)]\mathbf{R}(t)} \\ \mathbf{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} \quad \dot{\mathbf{q}}(t)$$

How to compute $\mathbf{f}(t)$?

Evaluate all the forces, $\mathbf{f}_1, \cdots, \mathbf{f}_n$ currently applied on the rigid body.

$$\boldsymbol{\tau}(t) = \sum_{i=1}^{n} \left(\mathbf{r}_i(t) - \mathbf{x}(t)\right) \times \mathbf{f}_i(t)\Big)$$

Back to rigid bodies…

$$\mathbf{Y}(t) = \begin{bmatrix} \mathbf{x}(t) \\ \cancel{\mathbf{R}(t)} \\ \mathbf{p}(t) \\ \mathbf{L}(t) \end{bmatrix} \quad \mathbf{q}(t)$$

$$\dot{\mathbf{Y}}(t) = \begin{bmatrix} \mathbf{v}(t) \\ \cancel{[\boldsymbol{\omega}(t)]\mathbf{R}(t)} \\ \mathbf{f}(t) \\ \boldsymbol{\tau}(t) \end{bmatrix} \quad \dot{\mathbf{q}}(t)$$

Given the current state $\mathbf{Y}_n$, how to evaluate $\dot{\mathbf{Y}}_n$, assuming the mass, M, and inertia in the body space, $\mathbf{I}_b$ are known?

$$\mathbf{v}(t) = \frac{\mathbf{p}}{M}$$

$$\cancel{[\boldsymbol{\omega}(t)]\mathbf{R}(t) = [\mathbf{I}(t)^{-1}\mathbf{L}(t)]\mathbf{R}(t) = [\mathbf{R}^{T}}$$

$$\boxed{\begin{aligned} \dot{\mathbf{q}}(t) &= \frac{1}{2}\begin{bmatrix} 0 \\ \boldsymbol{\omega}(t) \end{bmatrix} \otimes \mathbf{q}(t) \\ \boldsymbol{\omega}(t) &= \mathbf{R}^T(t)\mathbf{I}_b^{-1}\mathbf{R}(t)\mathbf{L}(t) \\ \mathbf{R}(t) &= \mathrm{quatToMatrix}(\mathbf{q}(t)) \end{aligned}}$$

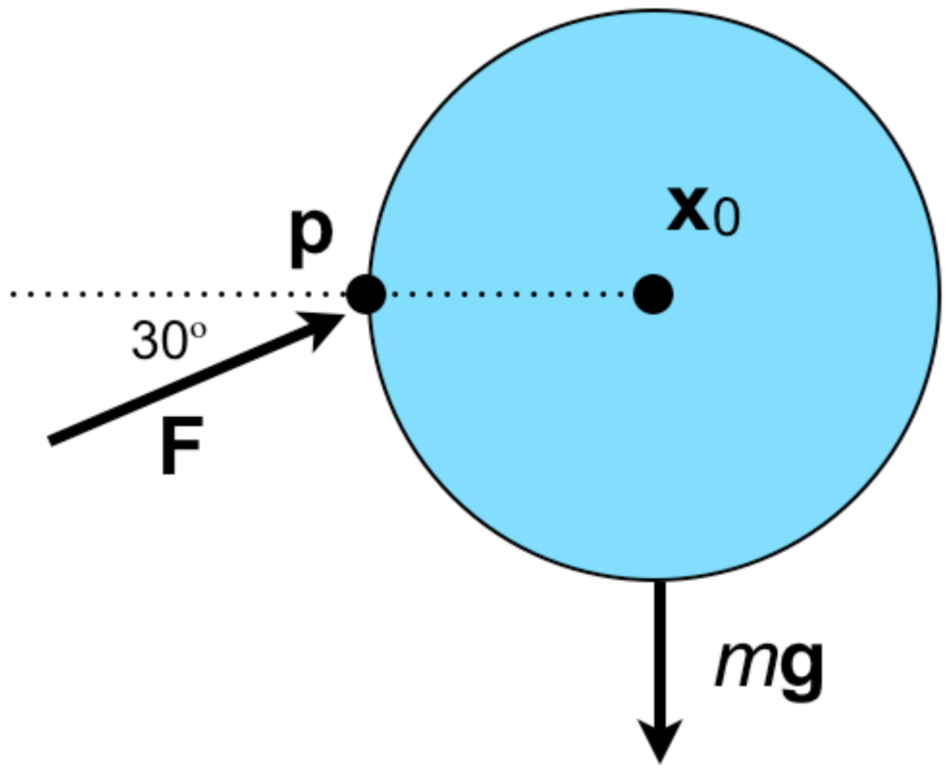
How to compute $\mathbf{f}(t)$?

Evaluate all the forces, $\mathbf{f}_1, \cdots, \mathbf{f}_n$ currently applied on the rigid body.

$$\boldsymbol{\tau}(t) = \sum_{i=1}^{n} \big(\mathbf{r}_i(t) - \mathbf{x}(t)\big) \times \mathbf{f}_i(t)\big)$$

Quiz

- Consider a 3D sphere with radius 1m, mass 1kg, and inertia I_{body}. The initial linear and angular velocity are both zero. The initial position and the initial orientation are x_0 and R_0. The forces applied on the sphere include gravity (g) and an initial push F applied at point p. Note that F is only applied for one time step at t_0. If we use Explicit Euler method with time step h to integrate, what are the position and the orientation of the sphere at t_2? Use Quaternion to represent orientation.



$\mathbf{x}_0 = (0, 0, 0)$

$$\mathbf{R}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$\mathbf{p} = (-1, 0, 0)$

$\mathbf{F} = (4\cos(30^\circ), 4\sin(30^\circ), 0)$

$m = 1$

$$\mathbf{I}_{body} = \begin{pmatrix} 2/5 & 0 & 0 \\ 0 & 2/5 & 0 \\ 0 & 0 & 2/5 \end{pmatrix}$$

Additional reading

- Euler/fixed angles Java script:
  - https://www.mecademic.com/resources/Euler-angles/Euler-angles
- Quaternions by Shoemake:
  - http://www.cs.ucr.edu/~vbz/resources/quatut.pdf
- Quaternions, Interpolation and Animation
  - https://web.mit.edu/2.998/www/QuaternionReport1.pdf
- Spherical Linear Interpolation (SLERP)
  - https://en.wikipedia.org/wiki/Slerp