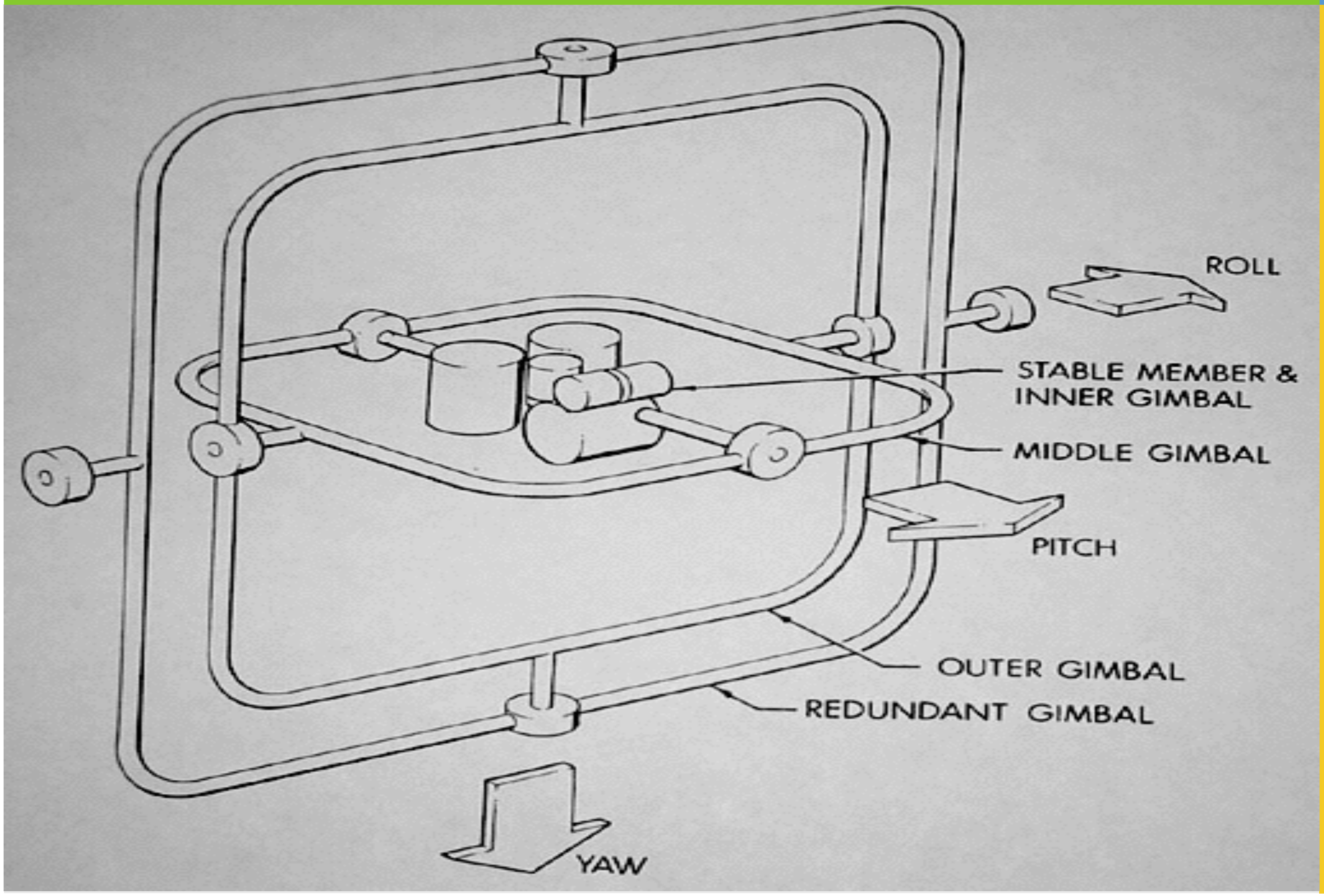
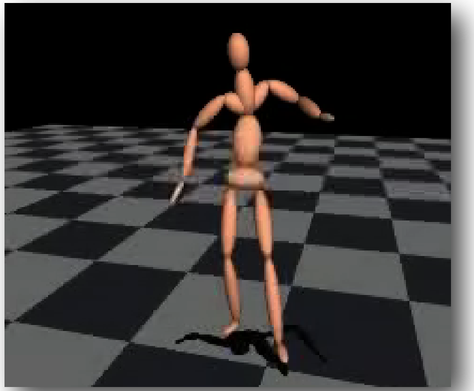
3D orientation



- Rotation matrix
- Fixed angle and Euler angle
- Axis angle
- Quaternion
- Exponential map

Joints and rotations

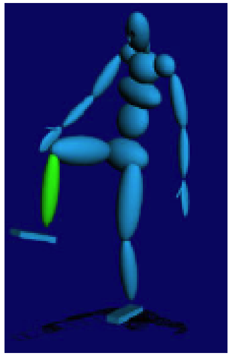
Rotational DOFs are widely used in character animation



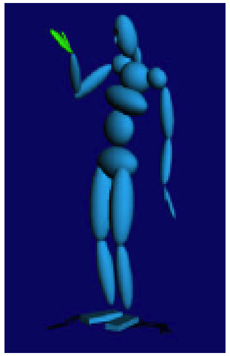
3 translational DOFs

48 rotational DOFs

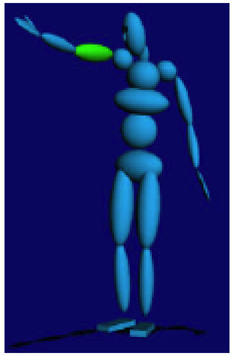
Each joint can have up to 3 DOFs



1 DOF: knee



2 DOF: wrist



3 DOF: arm

Representation of orientation

- Homogeneous coordinates (review)
  - 4X4 matrix used to represent translation, scaling, and rotation
  - a point in the space is represented as $\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$
  - Treat all transformations the same so that they can be easily combined

Translation

$$\begin{bmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

new point | translation matrix | old point

Scaling

$$\begin{bmatrix} s_x x \\ s_y y \\ s_z z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

new point scaling matrix old point

Rotation

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

X axis

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

Y axis

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

Z axis

Quiz

- True or False: Given an arbitrary rotation matrix R
  - R is always orthonormal
  - R is always symmetric
  - $RR^T = I$
  - $R_x(30)R_y(60) = R_y(60)R_x(30)$

Interpolation

- In order to “move things”, we need both translation and rotation
- Interpolation the translation is easy, but what about rotations?

Interpolation of orientation

- 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

Interpolation of orientation

Example: interpolate linearly from a positive 90 degree rotation about y axis to a negative 90 degree rotation about y

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

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

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

Properties of rotation matrix

- Easily composed? Yes
- Interpolate? No

- Rotation matrix
- Fixed angle and Euler angle
- Axis angle
- Quaternion
- Exponential map

Fixed angle

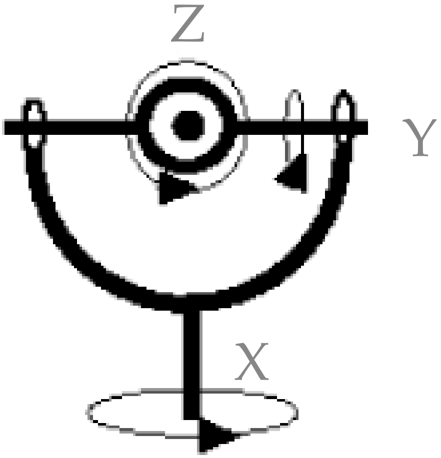
- Angles used to rotate about fixed axes
- Orientations are specified by a set of 3 ordered parameters that represent 3 ordered rotations about fixed axes
- Many possible orderings

Euler angle

- Same as fixed angles, except now the axes move with the object
- An Euler angle is a rotation about a single Cartesian axis
- Create multi-DOF rotations by concatenating Euler angles
    - evaluate each axis independently in a set order

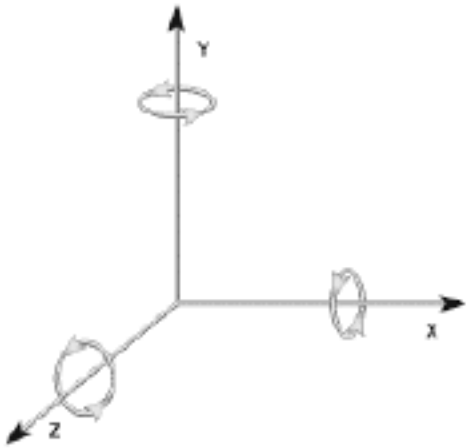
Euler angle vs. fixed angle

- $\mathbf{R}_z(90)\mathbf{R}_y(60)\mathbf{R}_x(30) = \mathbf{E}_x(30)\mathbf{E}_y(60)\mathbf{E}_z(90)$
- Euler angle rotations about moving axes written in reverse order are the same as the fixed axis rotations



Properties of Euler angle

- Easily composed? No
- Interpolate? Sometimes
- How about joint limit? Easy
- What seems to be the problem? Gimbal lock



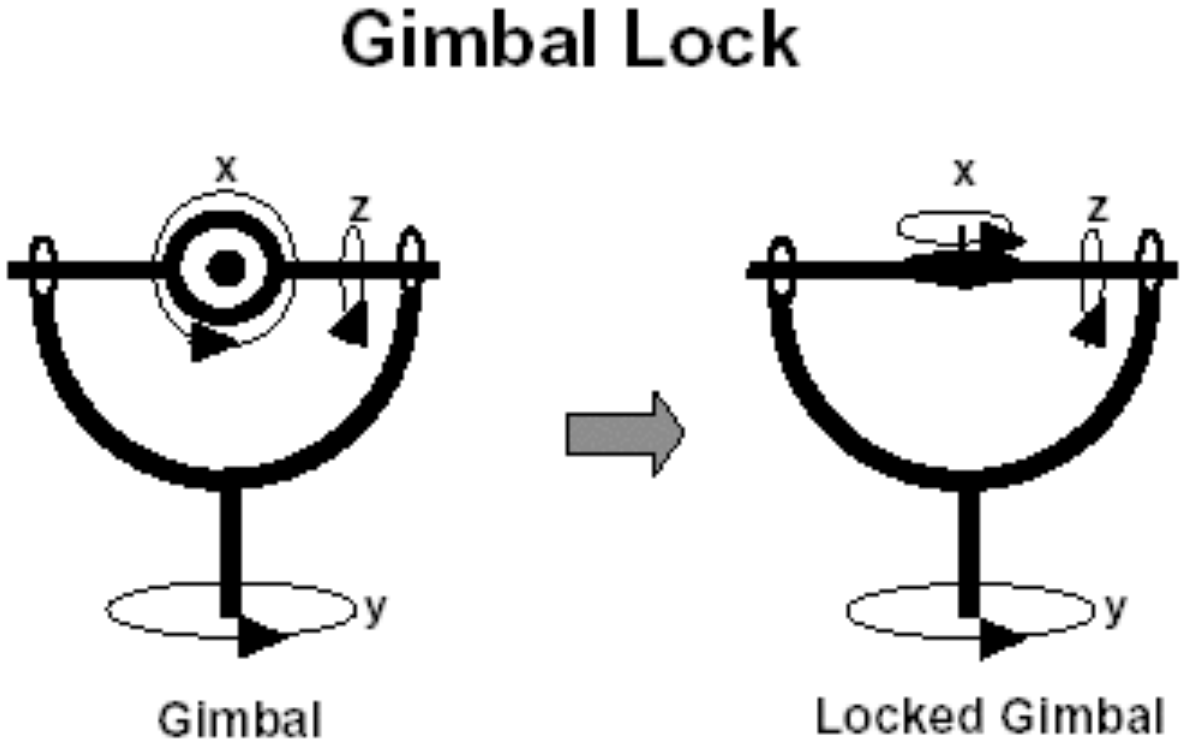
Gimbal Lock

A Gimbal is a hardware implementation of Euler angles used for mounting gyroscopes or expensive globes

Gimbal lock is a basic problem with representing 3D rotation using Euler angles or fixed angles

Gimbal lock

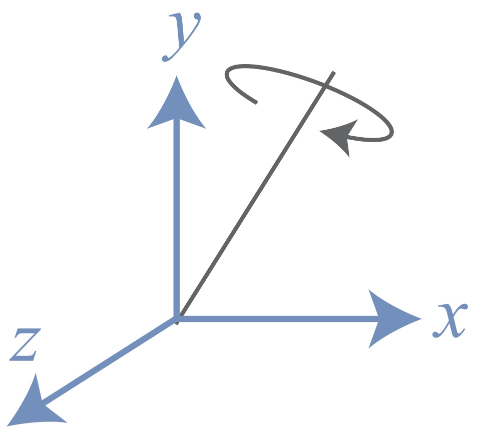
When two rotational axis of an object pointing in the same direction, the rotation ends up losing one degree of freedom



- Rotation matrix
- Fixed angle and Euler angle
- Axis angle
- Quaternion
- Exponential map

Axis angle

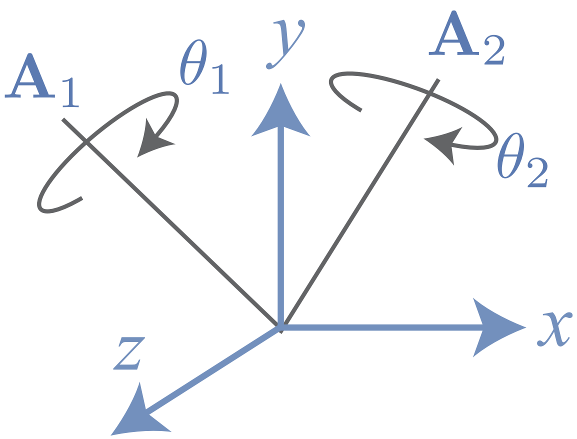
- Represent orientation as a vector and a scalar
  - vector is the axis to rotate about
  - scalar is the angle to rotate by



Properties of axis angle

- Can avoid Gimbal lock. Why?
    - It does 3D orientation in one step
- Can interpolate the vector and the scalar separately. How?

Axis angle interpolation



$$\theta_k = (1-k)\theta_1 + k\theta_2$$

$$\mathbf{B} = \mathbf{A}_1 \times \mathbf{A}_2$$

$$\phi = \cos^{-1}\left(\frac{\mathbf{A}_1 \cdot \mathbf{A}_2}{|\mathbf{A}_1||\mathbf{A}_2|}\right)$$

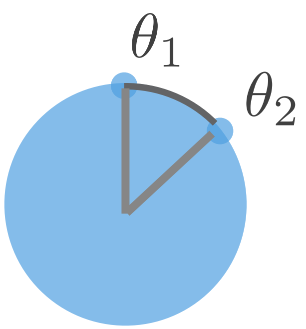
$$\mathbf{A}_k = \mathbf{R}_B(k\phi)\mathbf{A}_1$$

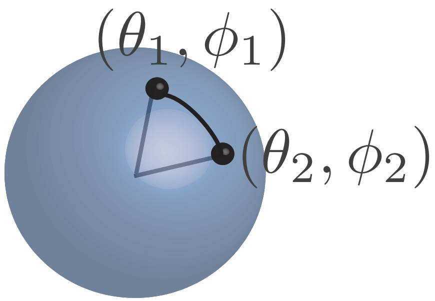
Properties of axis angle

- Easily composed? No, must convert back to matrix form
- Interpolate? Yes
- Joint limit? Yes
- Avoid Gimbal lock? Yes

- Rotation matrix
- Fixed angle and Euler angle
- Axis angle
- Quaternion
- Exponential map

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?

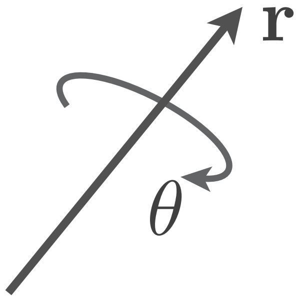
A unit quaternion is a point on the 4D sphere

Quaternion: algebraic view

4 tuple of real numbers: w, x, y, z

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

Same information as axis angles but in a different form



$$\mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mathbf{r} \end{bmatrix}$$

Basic quaternion definitions

- Unit quaternion $|\mathbf{q}| = 1$

$$x^2 + y^2 + z^2 + w^2 = 1$$

- Inverse quaternion $\mathbf{q}^{-1} = \dfrac{\mathbf{q}^*}{|\mathbf{q}|}$

Conjugate $\mathbf{q}^* = \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix}^* = \begin{bmatrix} w \\ -\mathbf{v} \end{bmatrix}$

- Identity

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

Quaternion multiplication

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

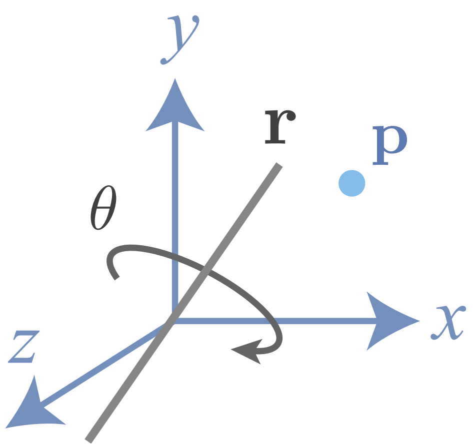
- Commutativity

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

- Associativity

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

Quaternion Rotation



$$\mathbf{q}_p = \begin{bmatrix} 0 \\ \mathbf{p} \end{bmatrix} \qquad \mathbf{q} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2)\mathbf{r} \end{bmatrix}$$

If $\mathbf{q}$ is a unit quaternion and

then $\mathbf{q}\mathbf{q}_p\mathbf{q}^{-1}$ results in $\mathbf{p}$ rotating about $\mathbf{r}$ by θ

proof: see *Quaternions* by Shoemaker

Quaternion Rotation

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

$$= \begin{bmatrix} w \\ \mathbf{v} \end{bmatrix} \begin{bmatrix} \mathbf{p} \cdot \mathbf{v} \\ w\mathbf{p} - \mathbf{p} \times \mathbf{v} \end{bmatrix}$$

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

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

Quaternion composition

If $\mathbf{q}_1$ and $\mathbf{q}_2$ are unit quaternion

the combined rotation of first rotating by $\mathbf{q}_1$ and then by $\mathbf{q}_2$ is equivalent to

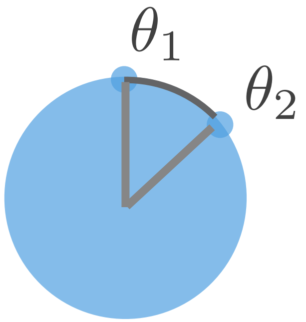
$$\mathbf{q}_3 = \mathbf{q}_2 \cdot \mathbf{q}_1$$

Matrix form

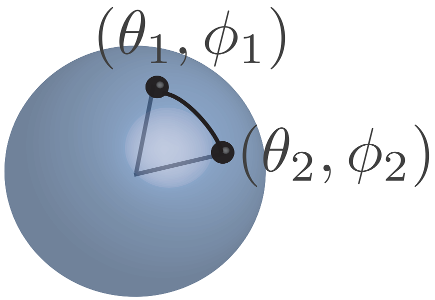
$$\mathbf{q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}$$

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

Quaternion interpolation



1-angle rotation can be represented by a unit circle



2-angle rotation can be represented by a unit sphere

- Interpolation means moving on n-D sphere

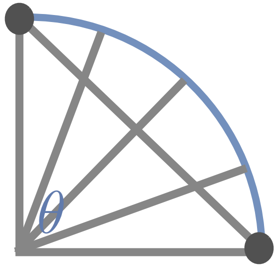
Quaternion interpolation

- Moving between two points on the 4D unit sphere
  - a unit quaternion at each step - another point on the 4D unit sphere
  - move with constant angular velocity along the great circle between the two points on the 4D unit sphere

Quaternion interpolation

Direct linear interpolation does not work

Linearly interpolated intermediate points are not uniformly spaced when projected onto the circle



Spherical linear interpolation (SLERP)

$$slerp(\mathbf{q}_1, \mathbf{q}_2, u) = \mathbf{q}_1 \frac{\sin((1-u)\theta)}{\sin\theta} + \mathbf{q}_2 \frac{\sin(u\theta)}{\sin\theta}$$

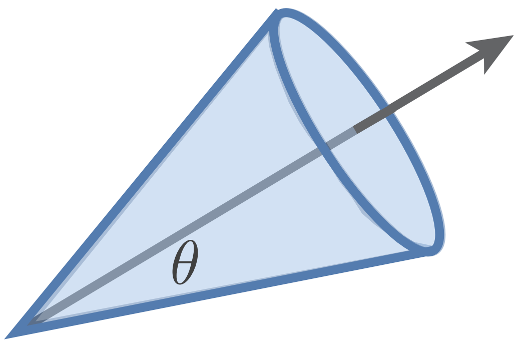
Normalize to regain unit quaternion

Quaternion constraints

Cone constraint

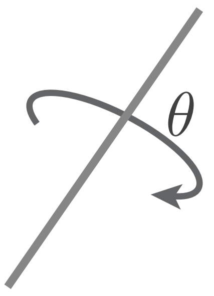
e.g. a cone along-x axis

$$\mathbf{q} = \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix}$$



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

Twist constraint



$$\tan(\theta/2) = \frac{q_{axis}}{w}$$

q_{axis} is the element of twist axis, e.g. z-axis

Properties of quaternion

- Easily composed?
- Interpolate?
- Joint limit?
- Avoid Gimbal lock?
- So what's bad about Quaternion?

- Rotation matrix
- Fixed angle and Euler angle
- Axis angle
- Quaternion
- Exponential map

Exponential map

- Represent orientation as a vector
  - direction of the vector is the axis to rotate about
  - magnitude of the vector is the angle to rotate by
- Zero vector represents the identity rotation

Properties of exponential map

- No need to re-normalize the parameters
- Fewer DOFs
- Good interpolation behavior
- Singularities exist but can be avoided

Choose a representation

- Choose the best representation for the task
  - common animation input: Euler angles
  - joint limits: Euler angles, axis angle, quaternion (harder)
  - interpolation: axis angle, quaternion
  - composition: quaternion or orientation matrix
  - avoid gimbal lock: axis and angle, quaternion
  - rendering: orientation matrix