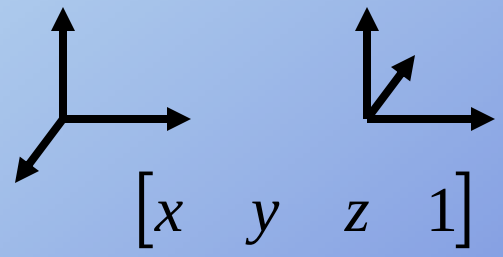
Computer Animation Algorithms and Techniques

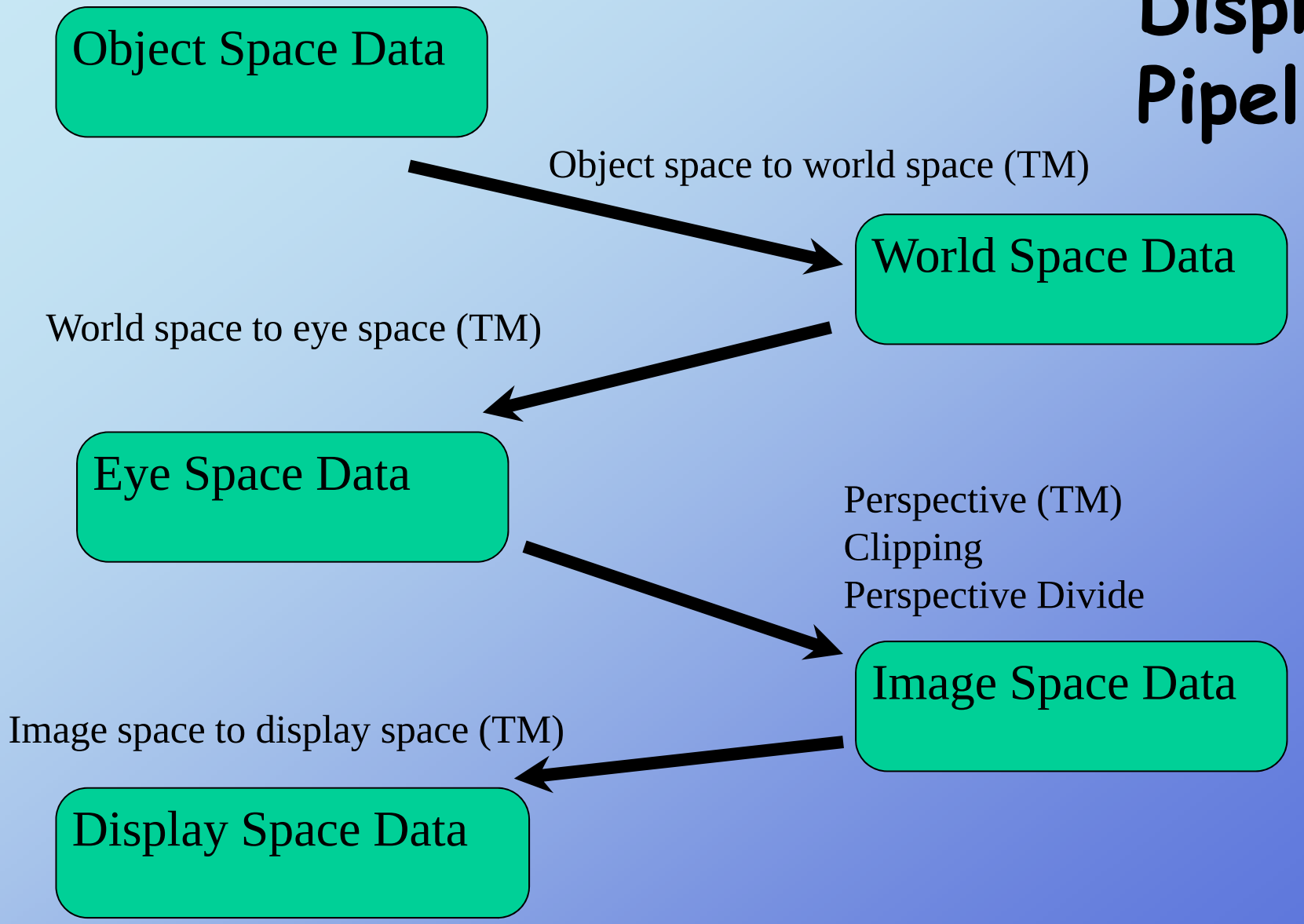
Technical Background

Spaces and Transformations

Left-handed v. right handed
Homogeneous coordinates:
4x4 transformation matrix (TM)
Concatenating TMs
Basic transformations (TMs)
Display pipeline

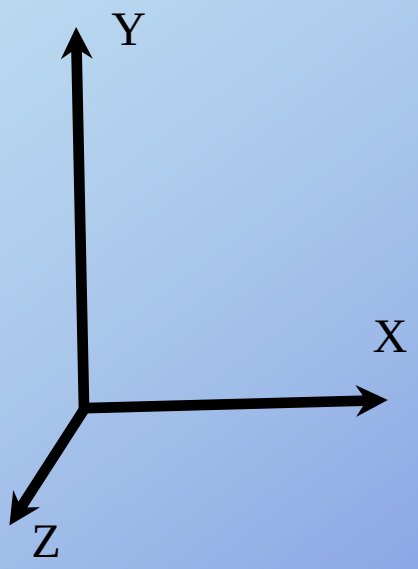


Display Pipeline



Representing an orientation

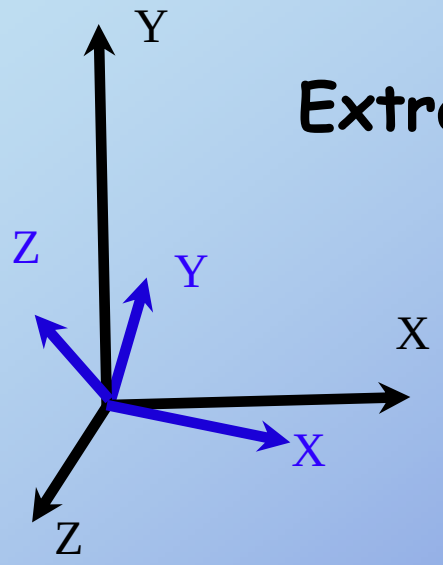
Example: fixed angles - rotate around global axes



Orientation: $(\alpha \quad \beta \quad \gamma)$

$$P' = R_z(\gamma)R_y(\beta)R_x(\alpha)P$$

Working with fixed angles and Rotation Matrices (RMs)



Orthonormalizing a RM

Extracing fixed angles from an orientation

Extracing fixed angles from a RM

Making a RM from fixed angles

Making a RM from transformed unit coordinate system (TUCS)

Transformations in pipeline

object -> world: often rigid transforms
world -> eye: rigid transforms
perspective matrix: uses 4^{th} component of homo. coords
perspective divide
image -> screen: 2D map to screen coordinates
Clipping: procedure that considers view frustum

Error considerations

Accumulated round-off error - transform data:

transform world data by delta RM

update RM by delta RM; apply to object data

update angle; form RM; apply to object data

orthonormalization

rotation matrix: orthogonal, unit-length columns

iterate update by taking cross product of 2 vectors

scale to unit length

considerations of scale

miles-to-inches can exeed single precision arithmetic

Orientation Representation

Rotation matrix
Fixed angles: rotate about global coordinate system
Euler angles: rotate about local coordinate system
Axis-angle: arbitrary axis and angle
Quaternions: mathematically handy axis-angle 4-tuple
Exponential map: 3-tuple version of quaternions

Transformation Matrix

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

Transformation Matrix

$$\begin{bmatrix} a & b & c & t_x \\ e & f & g & t_y \\ i & j & k & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

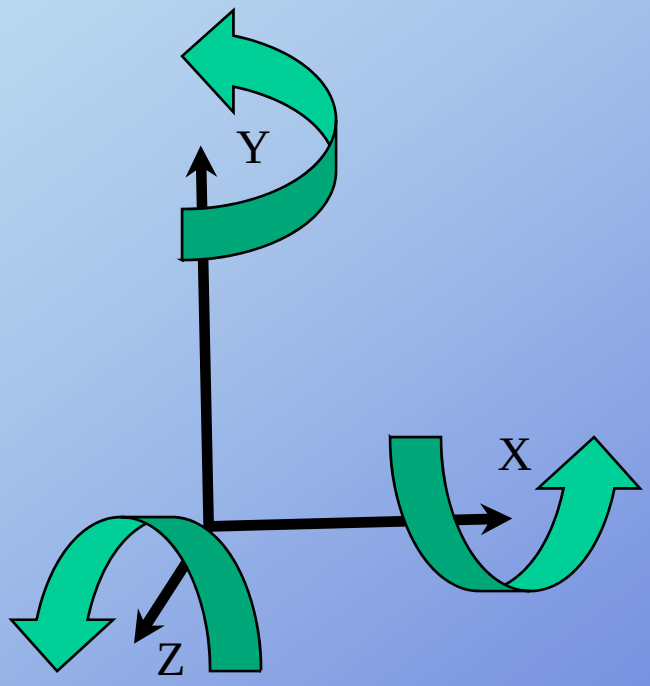
Rotation Matrices

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fixed Angles



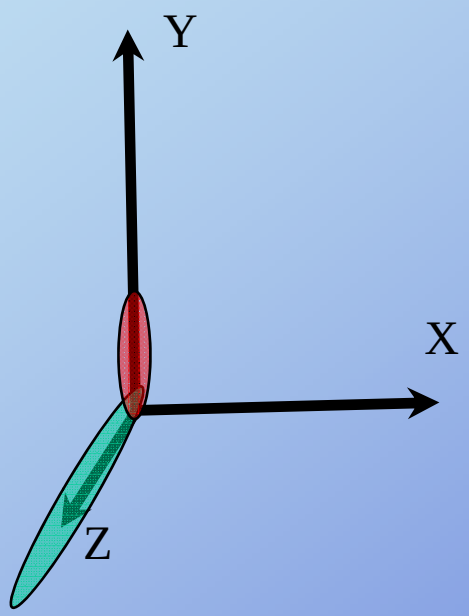
$$(\alpha \quad \beta \quad \gamma) \longrightarrow P' = R_z(\gamma)R_y(\beta)R_x(\alpha)P$$

Fixed order: e.g., x, y, z; also could be x, y, x
Global axes

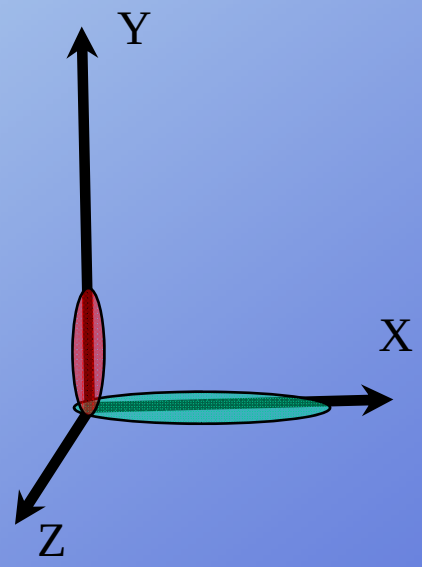
Gimbal Lock

Fixed angle: e.g., x, y, z

$(0 \quad 0 \quad 0)$

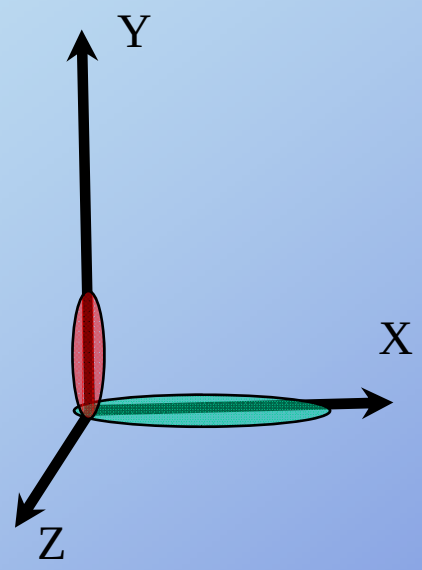


$(0 \quad 90 \quad 0)$



Gimbal Lock

$(0 \quad 90 \quad 0)$



Fixed order of rotations: x, y, z

What do these epsilon rotations do?

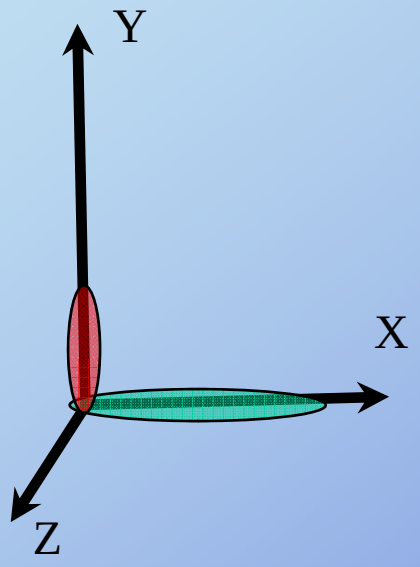
$$(0 \pm \varepsilon \quad 90 \quad 0)$$

$$(0 \quad 90 \pm \varepsilon \quad 0)$$

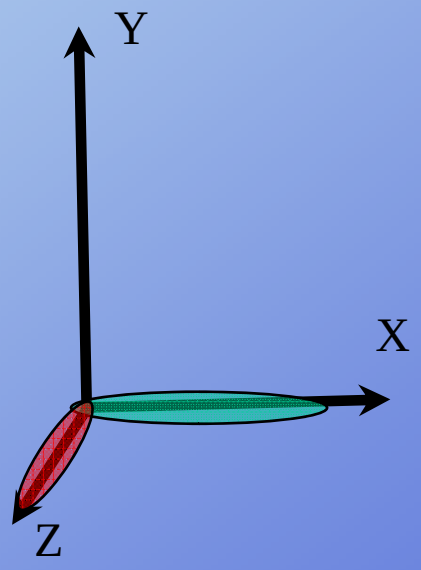
$$(0 \quad 90 \quad 0 \pm \varepsilon)$$

Gimbal Lock

$(0 \quad 90 \quad 0)$



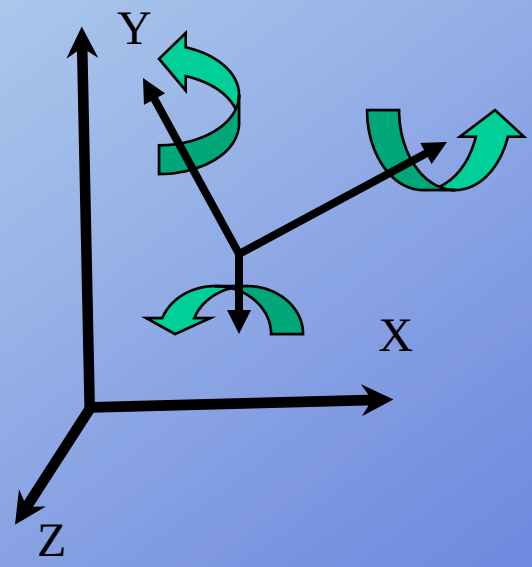
$(90 \quad 0 \quad 90)$



Interpolating FA representations does not produce intuitive rotation because of gimbal lock

Euler Angles

$$(\alpha \quad \beta \quad \gamma)$$



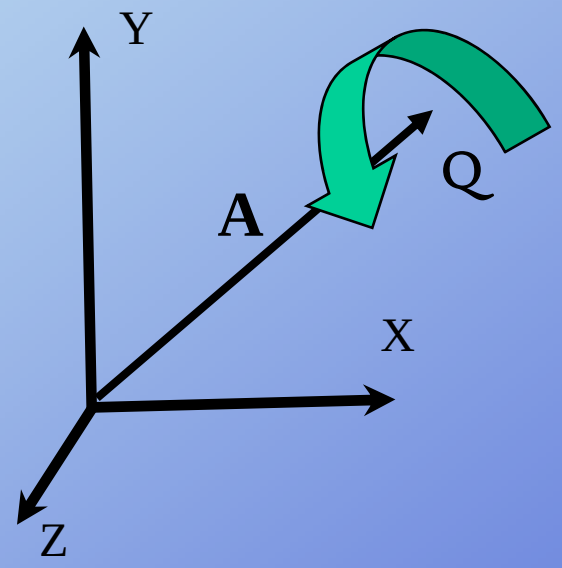
Prescribed order: e.g., x, y, z or x, y, x
Rotate around (rotated) local axes

Note: fixed angles are same as Euler angles in reverse order and vice versa

$$(\alpha \quad \beta \quad \gamma) \rightarrow P' = R_x(\alpha)R_y(\beta)R_z(\gamma)P$$

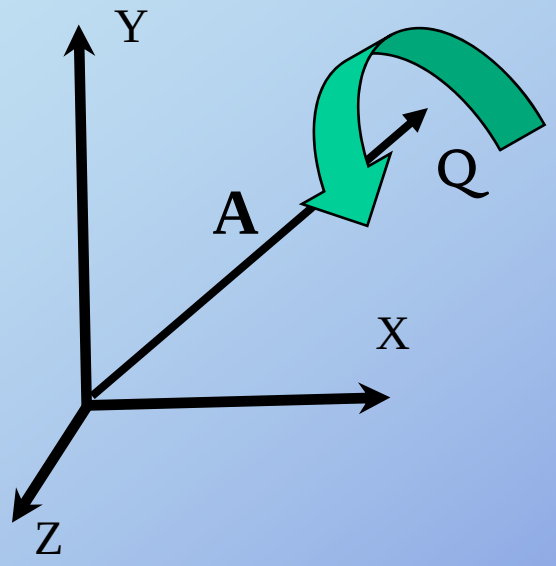
Axis-Angle

$$[\theta \quad A]$$
$$[\theta \quad (x \quad y \quad z)]$$



Rotate about given axis
Euler's Rotation Theorem
OpenGL
Fairly easy to interpolate between orientations
Difficult to concatenate rotations

Axis-angle to rotation matrix

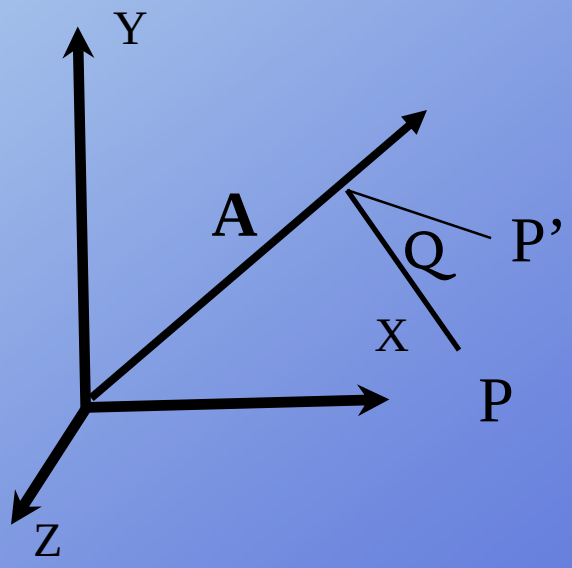


Concantenate the following:
Rotate A around z to x-z plane
Rotate A' around y to x-axis
Rotate theta around x
Undo rotation around y-axis
Undo rotation around z-axis

Axis-angle to rotation matrix

$$\hat{A} = \begin{bmatrix} a_x a_x & a_x a_y & a_x a_z \\ a_y a_x & a_y a_y & a_y a_z \\ a_z a_x & a_z a_y & a_z az \end{bmatrix}$$

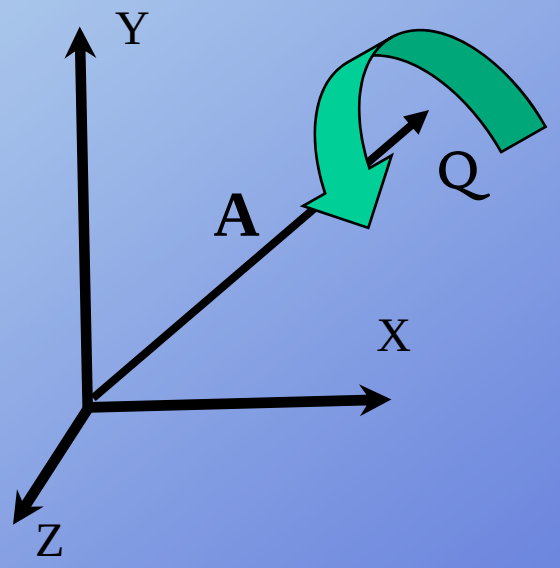
$$A^* = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$



$$Rot_{(\theta \quad (x \quad y \quad z))} = \hat{A} + \cos(\theta)(I - \hat{A}) + \sin(\theta)A^*$$

Quaternion

$$Rot_{(\theta \quad A)} = \left[\cos\left(\frac{\theta}{2}\right) \quad \sin(\frac{\theta}{2}) * A \right]$$



Same as axis-angle, but different form
Still rotate about given axis
Mathematically convenient form

$$q \quad \left[s \quad v \right]$$

Note: in this form v is a scaled version of the given axis of rotation, A

Quaternion Arithmetic

Addition $$\left[s_1 + s_2 \quad v_1 + v_2\right] = \left[s_1 \quad v_1\right] + \left[s_2 \quad v_2\right]$$

Multiplication

$$q_1 q_2 = \left[s_1 s_2 - v_1 \cdot v_2 \quad s_2 v_1 + s_1 v_2 + v_1 \times v_2\right]$$

Inner Product $$q_1 \cdot q_2 = s_1 s_2 + v_1 \cdot v_2$$

Length $$\|q\| = \sqrt{q \cdot q}$$

Quaternion Arithmetic

Inverse $$q^{-1} = \frac{1}{\|q\|^2} \begin{bmatrix} s & -v \end{bmatrix}$$

$$qq^{-1} = q^{-1}q = \begin{bmatrix} 1 & (0 & 0 & 0) \end{bmatrix}$$

$$(pq)^{-1} = q^{-1}p^{-1}$$

Unit quaternion $$\hat{q} = \frac{q}{\|q\|}$$

Quaternion Represention

Vector $\begin{bmatrix} 0 & v \end{bmatrix}$

Transform $v' = Rot_q(v) = qvq^{-1}$

Quaternion Geometric Operations

$$Rot_q(v) = Rot_{-q}(v)$$

$$Rot_q(v) = Rot_{kq}(v)$$

$$v'' = Rot_q(Rot_p(v)) = Rot_{qp}(v)$$

$$v'' = Rot_{q^{-1}}(Rot_q(v)) = q^{-1}(qvq^{-1})q = v$$

Unit Quaternion Conversions

$$Rot_{[s\ \ x\ \ y\ \ z]} = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2sz & 2xz-2sy \\ 2xy-2sz & 1-2x^2-2z^2 & 2yz-2sx \\ 2xz-2sy & 2yz-2sx & 1-2x^2-2y^2 \end{bmatrix}$$

Axis-Angle

$$\begin{cases} \theta = 2\cos^{-1}(s) \\ (x, y, z) = v / \|v\| \end{cases}$$

Quaternions

Avoid gimbal lock

Easy to rotate a point

Easy to convert to a rotation matrix

Easy to concatenate – quaternion multiply

Easy to interpolate – interpolate 4-tuples

How about smoothly (in both space and time) interpolate?