Geometric Objects and Transformation

Fall 2021
10/5/2021

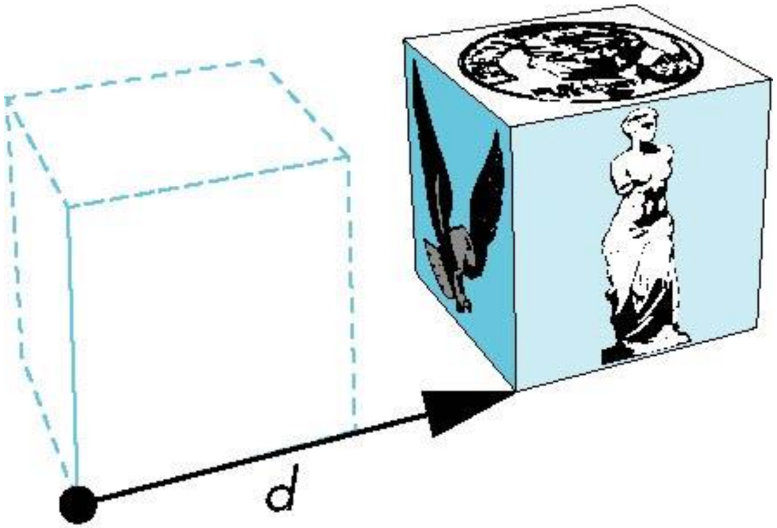
Kyoung Shin Park
Computer Engineering
Dankook University

3D Transformations

- In general, three-dimensional transformation can be thought of as an extension of two-dimensional transformation.
- The basic principles of three-dimensional translation, scaling, shearing are the same as those of two-dimensional.
- However, three-dimensional rotation is a bit more complicated.

3D Translation

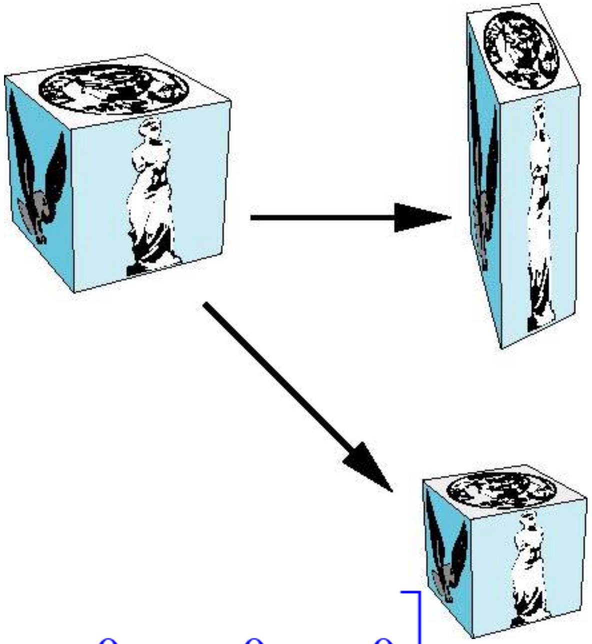
$$p' = p + d \qquad p = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad p' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \qquad d = \begin{bmatrix} dx \\ dy \\ dz \\ 0 \end{bmatrix}$$



$$p' = Tp \qquad T = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

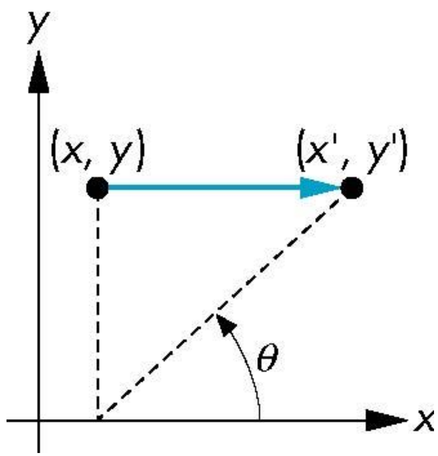
3D Scale

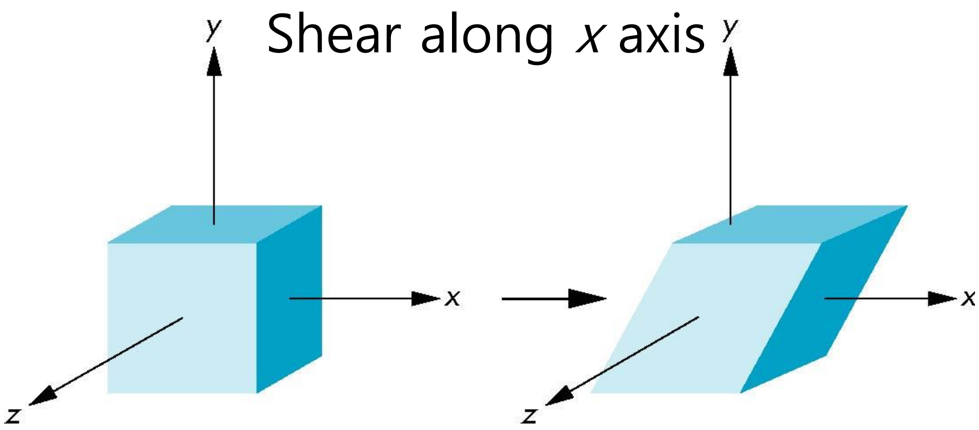
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



$$p' = Sp \qquad S = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad S^{-1} = \begin{bmatrix} \frac{1}{sx} & 0 & 0 & 0 \\ 0 & \frac{1}{sy} & 0 & 0 \\ 0 & 0 & \frac{1}{sz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Shear





x′ = x + y cot θ
y′ = y
z′ = z

$$\mathbf{H}_{xy}(\theta) = \begin{bmatrix} 1 & \mathbf{cot}\,\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\tan\theta = \frac{y}{x'-x} \Rightarrow \cot\theta = \frac{x'-x}{y}$$

3D Rotation

- 3D rotation in Z-axis

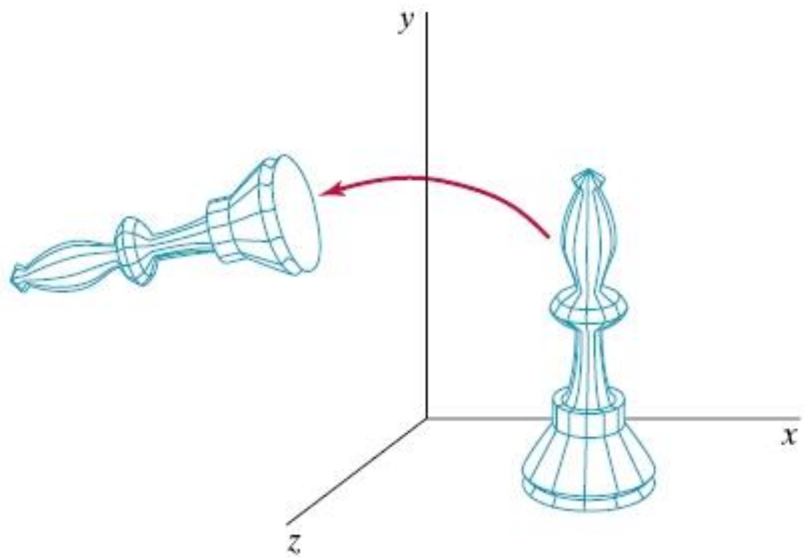
  $x' = x\cos\theta - y\sin\theta$

  $y' = x\sin\theta + y\cos\theta$

  $z' = z$

$$R^{-1}(\theta) = R(-\theta)$$

$$R^{-1}(\theta) = R^{T}(\theta)$$



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

$$P' = R_Z(\theta) \cdot P$$

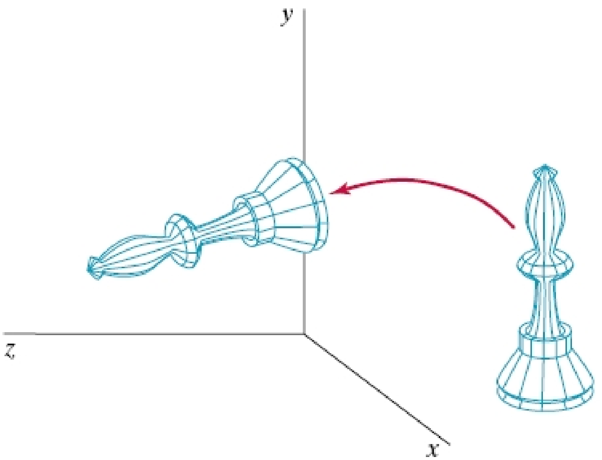
3D Rotation

- 3D rotation in X-axis

  $y' = y\cos\theta - z\sin\theta$

  $z' = y\sin\theta + z\cos\theta$

  $x' = x$



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

$$P' = R_X(\theta) \cdot P$$

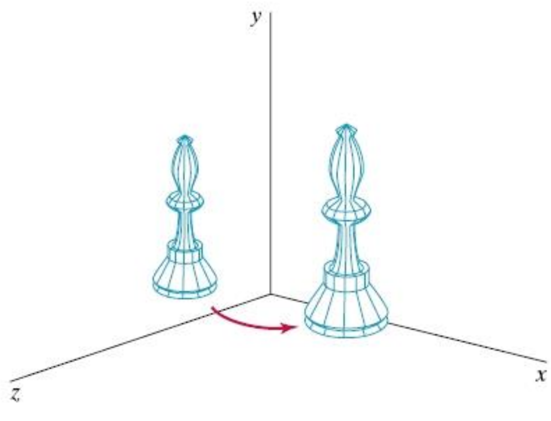
3D Rotation

- 3D rotation in Y-axis

  x′ = x cosθ + z sinθ

  z′ = -x sinθ + z cosθ

  y′ = y



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

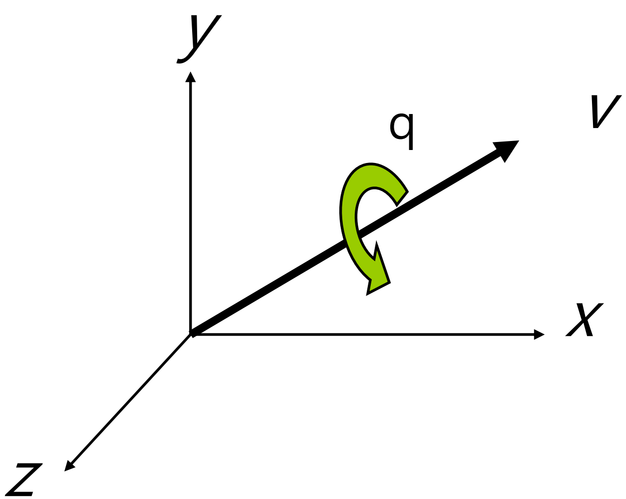
$$P' = R_Y(\theta) \cdot P$$

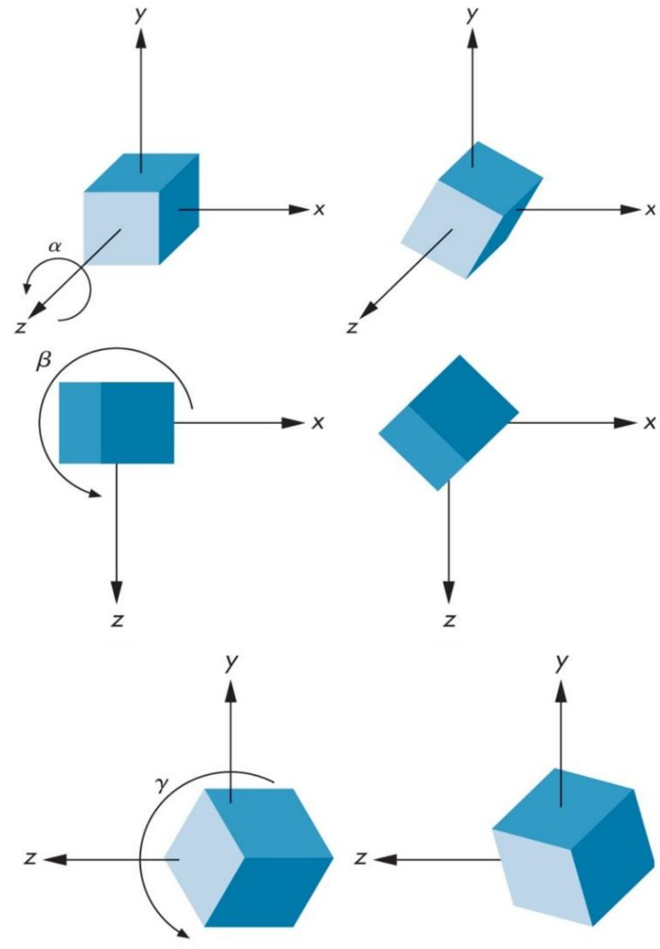
3D Rotation about the Origin

- A rotation by q about an arbitrary axis can be decomposed into the concatenation of rotations about the x, y, and z axes.

$$R(\theta) = R_Z(\theta_Z)R_Y(\theta_Y)R_X(\theta_X)$$

θ_X, θ_Y, θ_Z are called the Euler angles.

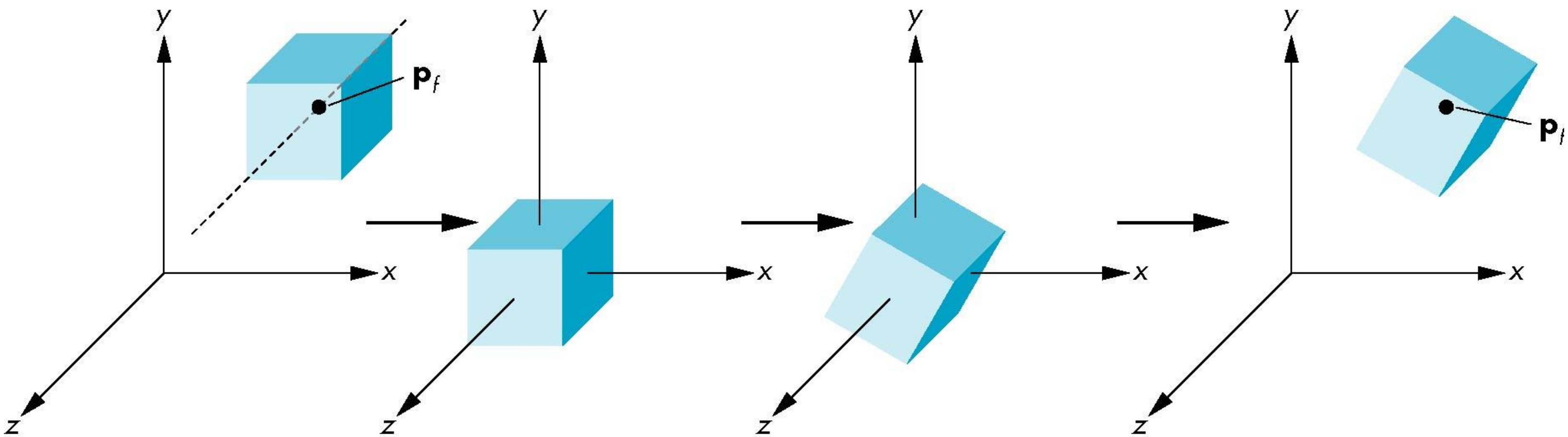




Rotation About a Pivot other than the Origin

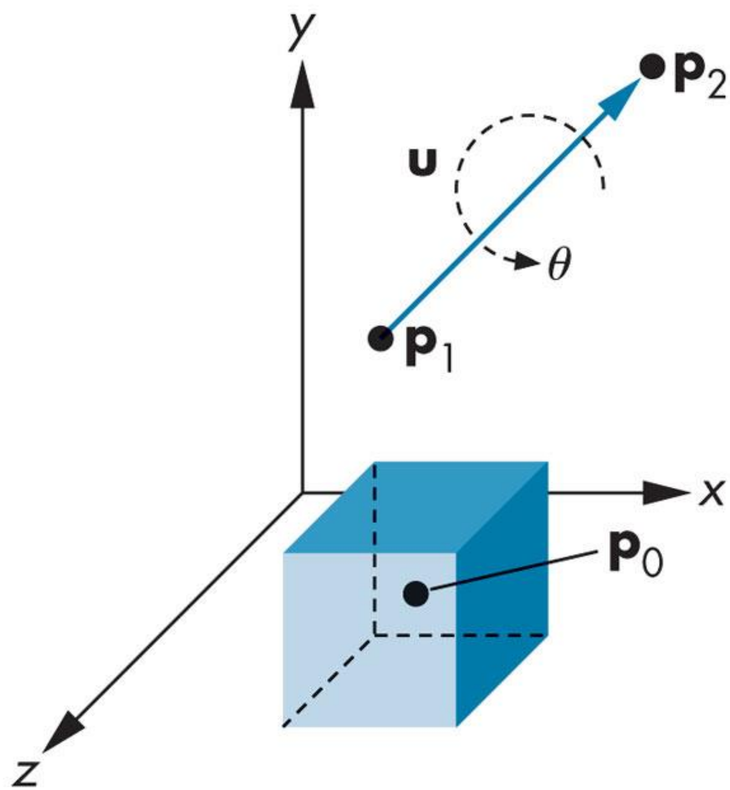
- Move fixed point to origin, rotate, and then move fixed point back.
- $\mathbf{M} = \mathbf{T}(p_f)\ \mathbf{R_Z}(\theta)\ \mathbf{T}(-p_f)$

$$M = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & x_f - x_f\cos\theta + y_f\sin\theta \\ \sin\theta & \cos\theta & 0 & y_f - x_f\sin\theta - y_f\cos\theta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3D Rotation about an Arbitrary Axis

- Move P_0 to the origin.
- Rotate twice to align the arbitrary axis u with the Z-axis.
- Rotate by θ in Z-axis.
- Undo two rotations (undo alignment).
- Move back to P_0.



$$M = T(P_0)R_x(-\theta_x)R_y(-\theta_y)R_z(\theta)R_y(\theta_y)R_x(\theta_x)T(-P_0)$$

3D Rotation about an Arbitrary Axis

- The translation matrix, $T(-P_0)$

$$T = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

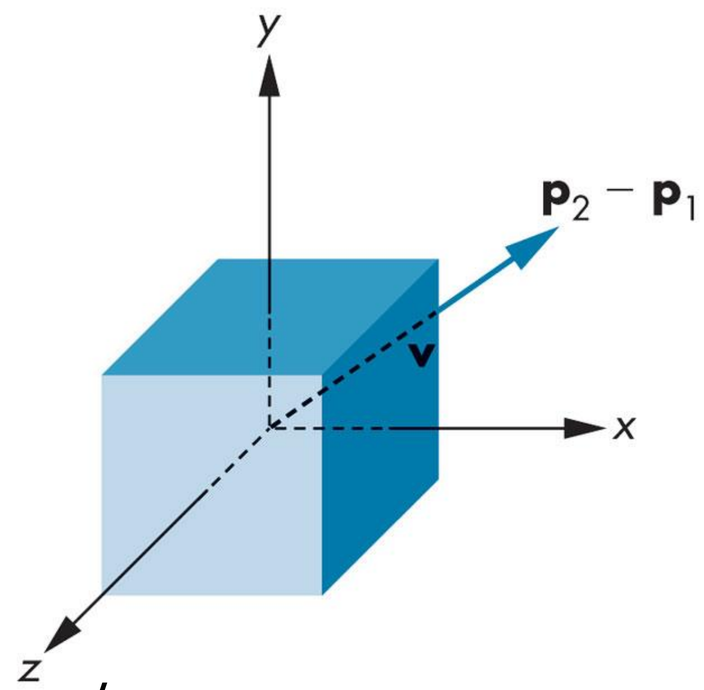
3D Rotation about an Arbitrary Axis

- The rotation-axis vector

$$u = P_2 - P_1$$
$$= (x_2 - x_1,\ y_2 - y_1,\ z_2 - z_1)$$

- Normalize u:

$$v = \frac{u}{\|u\|} = \begin{bmatrix} \alpha_x \\ \alpha_y \\ \alpha_z \end{bmatrix}$$



- Rotate along x-axis until *v* hits *xz-plane*

- Rotate along y-axis until *v* hits *z-axis*

3D Rotation about an Arbitrary Axis

- Find θ_x and θ_y

  $v = (\alpha_x, \alpha_y, \alpha_z)$

  $\alpha_x^2 + \alpha_y^2 + \alpha_z^2 = 1$

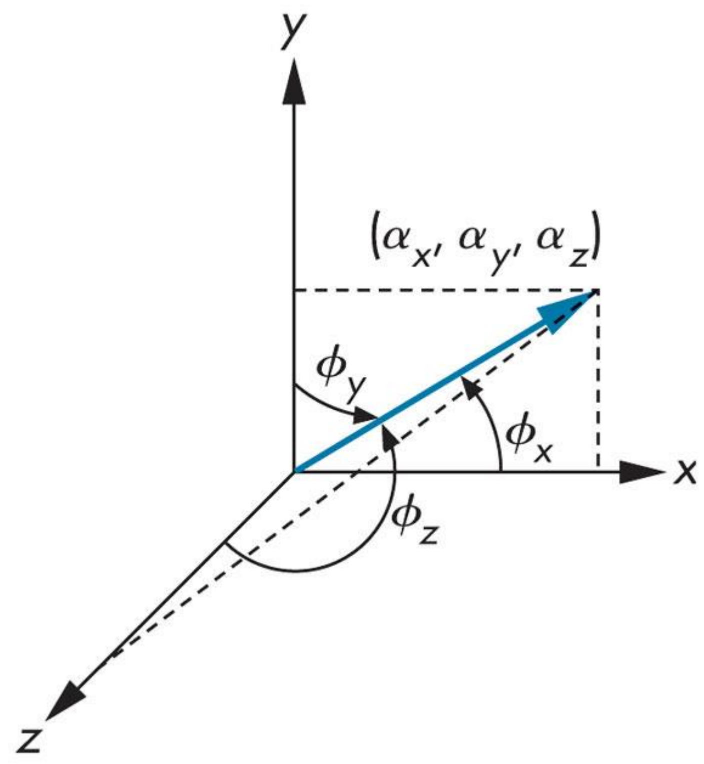
- Direction cosines:

$$\cos\phi_x = \alpha_x$$

$$\cos\phi_y = \alpha_y$$

$$\cos\phi_z = \alpha_z$$

$$\cos^2\phi_x + \cos^2\phi_y + \cos^2\phi_z = 1$$

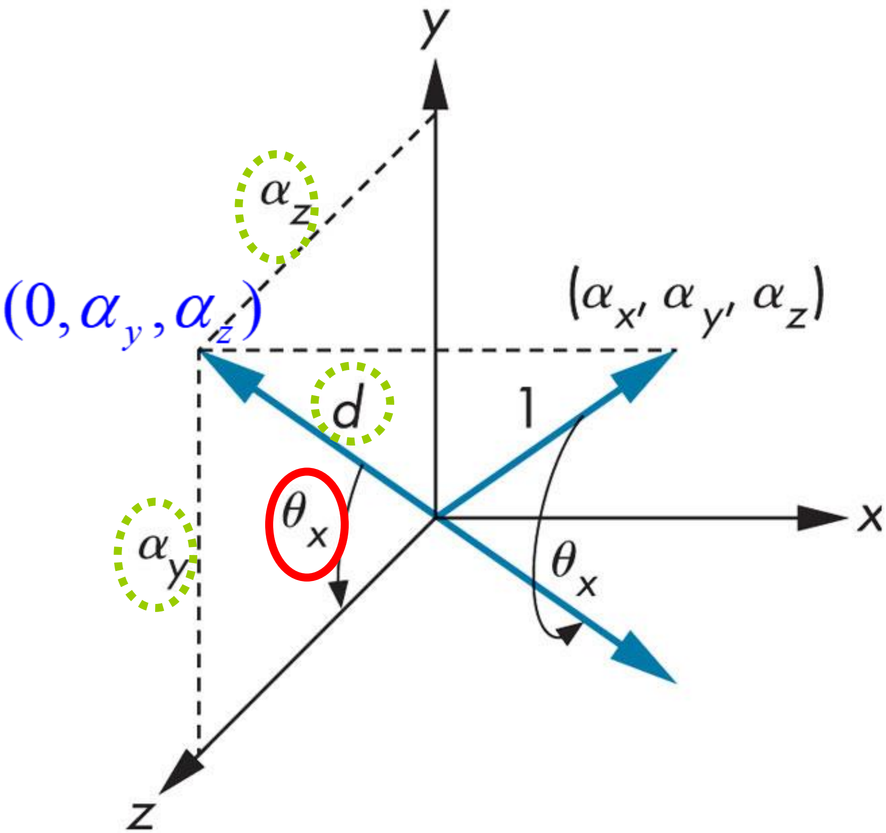


3D Rotation about an Arbitrary Axis

- Compute x-rotation θ_x

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\alpha_z}{d} & -\frac{\alpha_y}{d} & 0 \\ 0 & \frac{\alpha_y}{d} & \frac{\alpha_z}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$d = \sqrt{\alpha_y^2 + \alpha_z^2}$$



$$\cos\theta_x = \frac{\alpha_z}{d}$$

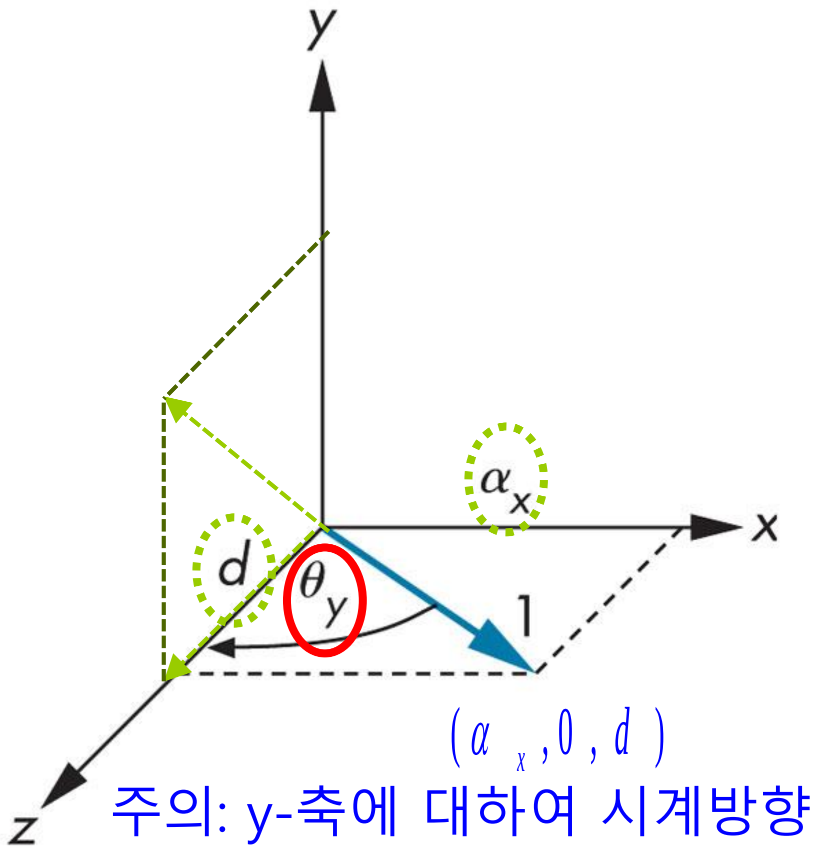
$$\sin\theta_x = \frac{\alpha_y}{d}$$

3D Rotation about an Arbitrary Axis

- Compute y-rotation θ_y

$$R_y(\theta_y) = \begin{bmatrix} d & 0 & -\alpha_x & 0 \\ 0 & 1 & 0 & 0 \\ \alpha_x & 0 & d & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$d = \sqrt{\alpha_y^2 + \alpha_z^2}$$



$$\cos\theta_y = d$$
$$\sin\theta_y = \alpha_x$$

3D Rotation about an Arbitrary Axis

- Rotation about the z axis

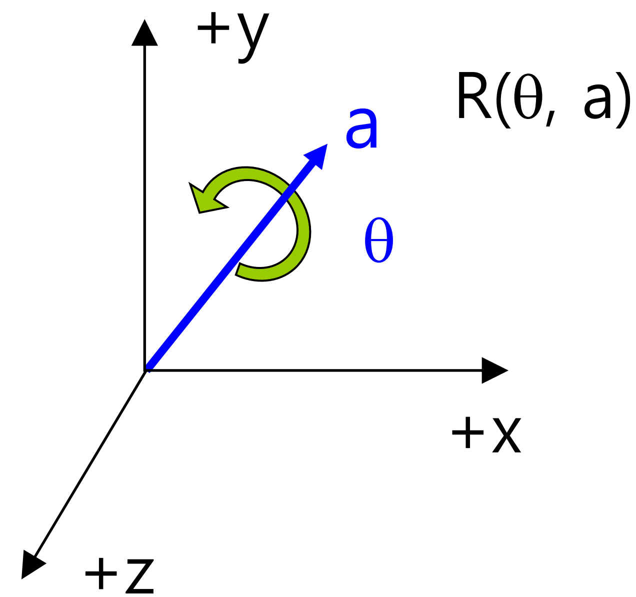
$$R_z(\theta_z) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Undo alignment, $R_x(-\theta_x)R_y(-\theta_y)$
- Undo translation, $T(P_0)$

- $M = T(P_0)R_x(-\theta_x)R_y(-\theta_y)R_z(\theta)R_y(\theta_y)R_x(\theta_x)T(-P_0)$

3D Rotation about an Arbitrary Axis Using Rotation Vectors

- 3D rotation can be expressed as 4 numbers of one angle of rotation about an arbitrary axis (ax, ay, az).
- It consists of a unit vector a (x, y, z) representing an arbitrary axis of rotation and a value of θ (0~360 degrees) representing the rotation angle around the unit vector.
- 3D rotation vector



3D Rotation about an Arbitrary Axis

- From axis/angle, we make the following rotation matrix.

$$R = \mathrm{I}\cos\theta + \textbf{Symmetric}\ (1-\cos\theta) + \textbf{Skew}\ \sin\theta$$

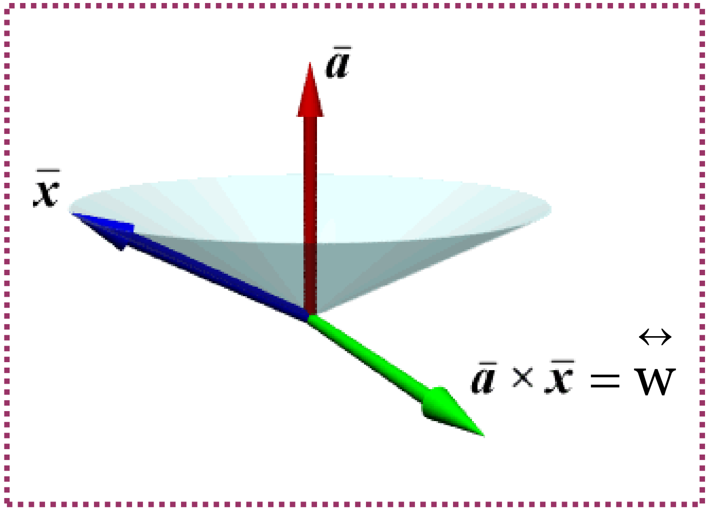
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cos\theta + \begin{bmatrix} a_x^2 & a_x a_y & a_x a_z \\ a_x a_y & a_y^2 & a_y a_z \\ a_x a_z & a_y a_z & a_z^2 \end{bmatrix} (1-\cos\theta) + \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \sin\theta$$

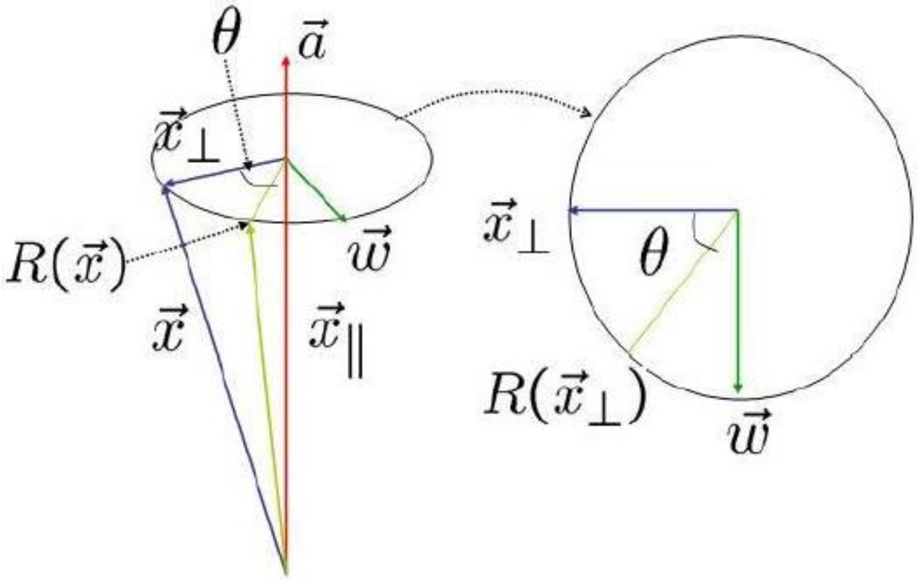
$$= \begin{bmatrix} a_x^2 + \cos\theta(1-a_x^2) & a_x a_y(1-\cos\theta) - a_z \sin\theta & a_x a_z(1-\cos\theta) + a_y \sin\theta \\ a_x a_y(1-\cos\theta) + a_z \sin\theta & a_y^2 + \cos\theta(1-a_y^2) & a_y a_z(1-\cos\theta) - a_x \sin\theta \\ a_x a_z(1-\cos\theta) - a_y \sin\theta & a_y a_z(1-\cos\theta) + a_x \sin\theta & a_z^2 + \cos\theta(1-a_z^2) \end{bmatrix}$$

3D Rotation as Vector Components

- 3D rotation by θ around the arbitrary axis a =[a_x, a_y, a_z]

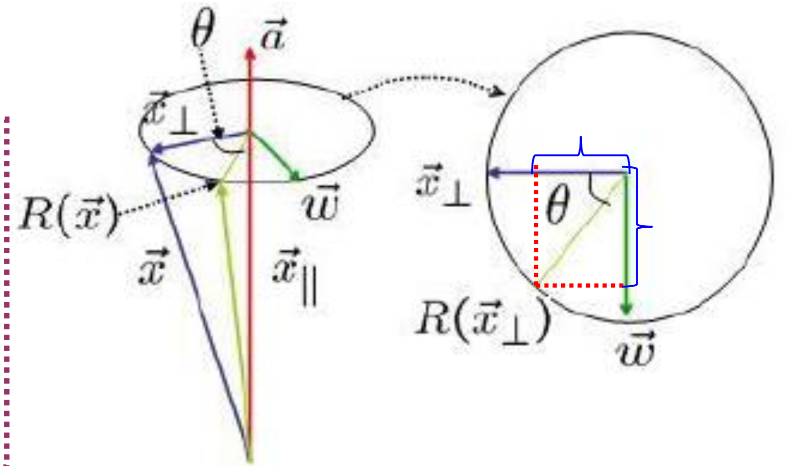
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \left(\mathbf{Symmetric}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right)(1-\cos\theta) + \mathbf{Skew}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right)\sin\theta + \mathbf{I}\cos\theta \right) \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$





3D Rotation as Vector Components

$$
\begin{aligned}
\vec{w} &= \vec{a} \times \vec{x}_\perp \\
&= \vec{a} \times (\vec{x} - \vec{x}_{||}) \\
&= (\vec{a} \times \vec{x}) - (\vec{a} \times \vec{x}_{||}) \\
&= \vec{a} \times \vec{x}
\end{aligned}
$$



$$R(\vec{x}_\perp) = \cos\theta \vec{x}_\perp + \sin\theta \vec{w}$$

$$
\begin{aligned}
\vec{x}_{||} &= (\vec{a} \cdot \vec{x})\vec{a} \\
\vec{x}_\perp &= \vec{x} - \vec{x}_{||} = \vec{x} - (\vec{a} \cdot \vec{x})\vec{a}
\end{aligned}
$$

$$
\begin{aligned}
R(\vec{x}) &= R(\vec{x}_{||}) + R(\vec{x}_\perp) \\
&= R(\vec{x}_{||}) + \cos\theta \vec{x}_\perp + \sin\theta \vec{w} \\
&= (\vec{a} \cdot \vec{x})\vec{a} + \cos\theta(\vec{x} - (\vec{a} \cdot \vec{x})\vec{a}) + \sin\theta \vec{w} \\
&= \cos\theta \vec{x} + (1 - \cos\theta)(\vec{a} \cdot \vec{x})\vec{a} + \sin\theta(\vec{a} \times \vec{x})
\end{aligned}
$$

Symmetric

Skew

3D Rotation as Vector Components

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \left(\mathbf{Symmetric}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \right)(1-\cos\theta) + \mathbf{Skew}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \right)\sin\theta + \mathbf{I}\cos\theta \right) \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

- The vector *a* specifies the axis of rotation. This axis vector must be normalized.
- The rotation angle is given by θ.
- The basic idea is that *any rotation can be decomposed into weighted contributions from three different vectors.*

3D Rotation as Vector Components

- *The symmetric matrix of a vector* generates a vector in the direction of the axis.
- The symmetric matrix is composed of the outer product of a row vector and an column vector of the same value.

$$\text{Symmetric}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right) = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \end{bmatrix} = \begin{bmatrix} a_x^2 & a_x a_y & a_x a_z \\ a_x a_y & a_y^2 & a_y a_z \\ a_x a_z & a_y a_z & a_z^2 \end{bmatrix}$$

$$\text{Symmetric}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \overline{a}(\overline{a} \cdot \overline{x})$$

3D Rotation as Vector Components

- *Skew symmetric matrix of a vector* generates a vector that is perpendicular to both the axis and it's input vector.

$$\mathrm{Skew}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right) = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

$$\mathrm{Skew}(\overline{a})\overline{x} = \overline{a} \times \overline{x}$$

3D Rotation as Vector Components

- First, consider a rotation by 0. :

$$Rotate\left(\begin{bmatrix}a_x\\a_y\\a_z\end{bmatrix},0\right)=\begin{bmatrix}a_x^2 & a_xa_y & a_xa_z\\a_xa_y & a_y^2 & a_ya_z\\a_xa_z & a_ya_z & a_z^2\end{bmatrix}(1-1)+\begin{bmatrix}0 & -a_z & a_y\\a_z & 0 & -a_x\\-a_y & a_x & 0\end{bmatrix}0+\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}1=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$$

- For instance, a rotation about the x-axis:

$$Rotate\left(\begin{bmatrix}1\\0\\0\end{bmatrix},\theta\right)=\begin{bmatrix}1 & 0 & 0\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}(1-\cos\theta)+\begin{bmatrix}0 & 0 & 0\\0 & 0 & -1\\0 & 1 & 0\end{bmatrix}\sin\theta+\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}\cos\theta$$

$$Rotate\left(\begin{bmatrix}1\\0\\0\end{bmatrix},\theta\right)=\begin{bmatrix}1 & 0 & 0\\0 & \cos\theta & -\sin\theta\\0 & \sin\theta & \cos\theta\end{bmatrix}$$

3D Rotation as Vector Components

- For instance, a rotation about the y-axis:

$$Rotate\left(\begin{bmatrix}0\\1\\0\end{bmatrix},\theta\right)=\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix}(1-\cos\theta)+\begin{bmatrix}0&0&1\\0&0&0\\-1&0&0\end{bmatrix}\sin\theta+\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\cos\theta$$

$$Rotate\left(\begin{bmatrix}0\\1\\0\end{bmatrix},\theta\right)=\begin{bmatrix}\cos\theta&0&\sin\theta\\0&1&0\\-\sin\theta&0&\cos\theta\end{bmatrix}$$

- For instance, a rotation about the z-axis:

$$Rotate\left(\begin{bmatrix}0\\0\\1\end{bmatrix},\theta\right)=\begin{bmatrix}0&0&0\\0&0&0\\0&0&1\end{bmatrix}(1-\cos\theta)+\begin{bmatrix}0&-1&0\\1&0&0\\0&0&0\end{bmatrix}\sin\theta+\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\cos\theta$$

$$Rotate\left(\begin{bmatrix}0\\0\\1\end{bmatrix},\theta\right)=\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}$$

Quaternion

- Quaternion is a 4D complex space vector. It is a mathematical concept used in place of a matrix when expressing 3D rotation in computer graphics.
- It is actually the most effective way to express rotation.
- Quaternion has four components.

$$\mathbf{q} = \langle x \quad y \quad z \quad w \rangle$$

Quaternions (Imaginary Space)

- Quaternions are actually an extension to complex numbers.
- Of the 4 components, one is a 'real' scalar number, and the other 3 form a vector in imaginary *ijk* space!

$$\mathbf{q} = xi + yj + zk + w$$

$$i^2 = j^2 = k^2 = ijk = -1$$

$$i = jk = -kj$$

$$j = ki = -ik$$

$$k = ij = -ji$$

Quaternion (Scalar/Vector)

- The quaternion is also expressed as a scalar value s and a vector value v.

$$\mathbf{q} = \langle \mathbf{v}, s \rangle$$

$$v = (x, y, z)$$

$$s = w$$

Identity Quaternion

- Unlike vectors, there are two identity quaternions.
- The multiplication identity quaternion is

$$\mathbf{q} = \langle 0,0,0,1 \rangle = 0i + 0j + 0k + 1$$

- The addition identity quaternion (which we do not use) is

$$\mathbf{q} = \langle 0,0,0,0 \rangle$$

Unit Quaternion

- For convenience, we will use only unit length quaternions, as they will make things a little easier

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

- These correspond to the set of vectors that form the 'surface' of a 4D hyper-sphere of radius 1
- The 'surface' is actually a 3D volume in 4D space, but it can sometimes be visualized as an extension to the concept of a 2D surface on a 3D sphere
- Quaternion normalization:

$$q = {}^{q}\!\big/_{|\mathbf{q}|} = {}^{q}\!\big/_{\sqrt{x^2 + y^2 + z^2 + w^2}}$$

Quaternion as Rotations

- A quaternion can represent a rotation by an angle **q** around a unit axis **a** **(a_x, a_y, a_z)** :

$$\mathbf{q} = \left[a_x \sin\frac{\theta}{2}, \quad a_y \sin\frac{\theta}{2}, \quad a_z \sin\frac{\theta}{2}, \quad \cos\frac{\theta}{2} \right]$$

or

$$\mathbf{q} = \left[\mathbf{a} \sin\frac{\theta}{2}, \quad \cos\frac{\theta}{2} \right]$$

- If **a** has unit length, then **q** will also has unit length

Quaternion as Rotations

$$
\begin{aligned}
|\mathbf{q}| &= \sqrt{x^2 + y^2 + z^2 + w^2} \\
&= \sqrt{a_x^2 \sin^2\frac{\theta}{2} + a_y^2 \sin^2\frac{\theta}{2} + a_z^2 \sin^2\frac{\theta}{2} + \cos^2\frac{\theta}{2}} \\
&= \sqrt{\sin^2\frac{\theta}{2}\left(a_x^2 + a_y^2 + a_z^2\right) + \cos^2\frac{\theta}{2}} \\
&= \sqrt{\sin^2\frac{\theta}{2}|\mathbf{a}|^2 + \cos^2\frac{\theta}{2}} = \sqrt{\sin^2\frac{\theta}{2} + \cos^2\frac{\theta}{2}} \\
&= \sqrt{1} = 1
\end{aligned}
$$

Quaternion to Rotation Matrix

- Equivalent rotation matrix representing a quaternion is:

$$\begin{bmatrix} x^2 - y^2 - z^2 + w^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & -x^2 + y^2 - z^2 + w^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & -x^2 - y^2 + z^2 + w^2 \end{bmatrix}$$

- Using unit quaternion that $x^2+y^2+z^2+w^2=1$, we can reduce the matrix to:

$$\begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2wz & 2xz + 2wy \\ 2xy + 2wz & 1 - 2x^2 - 2z^2 & 2yz - 2wx \\ 2xz - 2wy & 2yz + 2wx & 1 - 2x^2 - 2y^2 \end{bmatrix}$$

Quaternion to Axis/Angle

- To convert a quaternions to a rotation axis, a (ax, ay, az) and an angle θ :

$$scale = \sqrt{x^2 + y^2 + z^2} \quad or \quad \sin(\mathbf{acos}(w))$$

$$ax = x / scale$$

$$ay = y / scale$$

$$az = z / scale$$

$$\theta = 2\mathbf{acos}(w)$$

Quaternion Dot Product

- The dot product of two quaternions works in the same way as the dot product of two vectors:

$$\mathbf{p}\cdot\mathbf{q} = x_p x_q + y_p y_q + z_p z_q + w_p w_q = |\mathbf{p}||\mathbf{q}|\cos\varphi$$

- The angle between two quaternions in 4D space is half the angle one would need to rotate from one orientation to the other in 3D space.

Quaternion Multiplication

- If **q** represents a rotation and **q'** represents a rotation, then **qq'** represents **q** rotated by **q'**
- This follows very similar rules as matrix multiplication (I.e., non-commutative) qq' ≠ q'q

$$\mathbf{q}\mathbf{q}' = (xi + yj + zk + w)(x'i + y'j + z'k + w')$$

$$= \langle s\mathbf{v}' + s'\mathbf{v} + \mathbf{v}' \times \mathbf{v}, ss' - \mathbf{v} \cdot \mathbf{v}' \rangle$$

Quaternion Operations

- Negation of quaternion, -q
  - -[v s] = [-v –s] = [-x, –y, –z, –w]
- Addition of two quaternion, p + q
  - p + q = [pv, ps] + [qv, qs] = [pv + qv, ps + qs]
- Magnitude of quaternion, |q|
  - $|\mathbf{q}| = \sqrt{x^2 + y^2 + z^2 + w^2}$
- Conjugate of quaternion, q* (켤레 사원수)
  - q* = [v s]* = [–v s] = [–x, –y, –z , w]
- Multiplicative inverse of quaternion, q^{-1} (역수) $q\, q^{-1} = q^{-1}\, q = 1$
  - $q^{-1} = q^*/|q|$
- Exponential of quaternion
  - exp(v q) = v sin q + cos q
- Logarithm of quaternion
  - log(q) = log(v sin q + cos q) = log(exp(v q)) = v q

Quaternion Interpolation

- One of the key benefits of using a quaternion representation is the ability to interpolate between key frames.

```
alpha = fraction value in between frame0 and frame1
q1 = Euler2Quaternion(frame0)
q2 = Euler2Quaternion(frame1)
qr = QuaternionInterpolation(q1, q2, alpha)
qr.Quaternion2Euler()
```

- Quaternion Interpolation
  - Linear Interpolation (LERP)
  - Spherical Linear Interpolation (SLERP)
  - Spherical Cubic Interpolation (SQUAD)

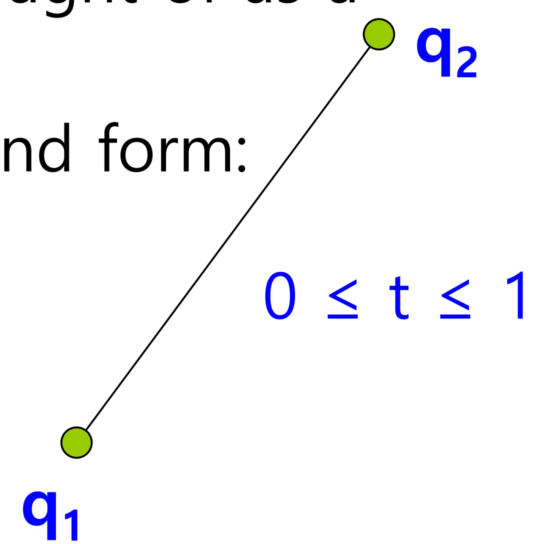
Linear Interpolation (LERP)

- If we want to do a direct interpolation between two quaternions **p** and **q** by alpha:

$$\mathrm{Lerp}(\mathbf{p}, \mathbf{q}, t) = (1-t)\mathbf{p} + (t)\mathbf{q}$$
$$\text{where } 0 \le t \le 1$$

- Note that the Lerp operation can be thought of as a weighted average (convex)
- We could also write it in it's additive blend form:

$$\mathrm{Lerp}(\mathbf{q}_1, \mathbf{q}_2, t) = \mathbf{q}_1 + t(\mathbf{q}_2 - \mathbf{q}_1)$$

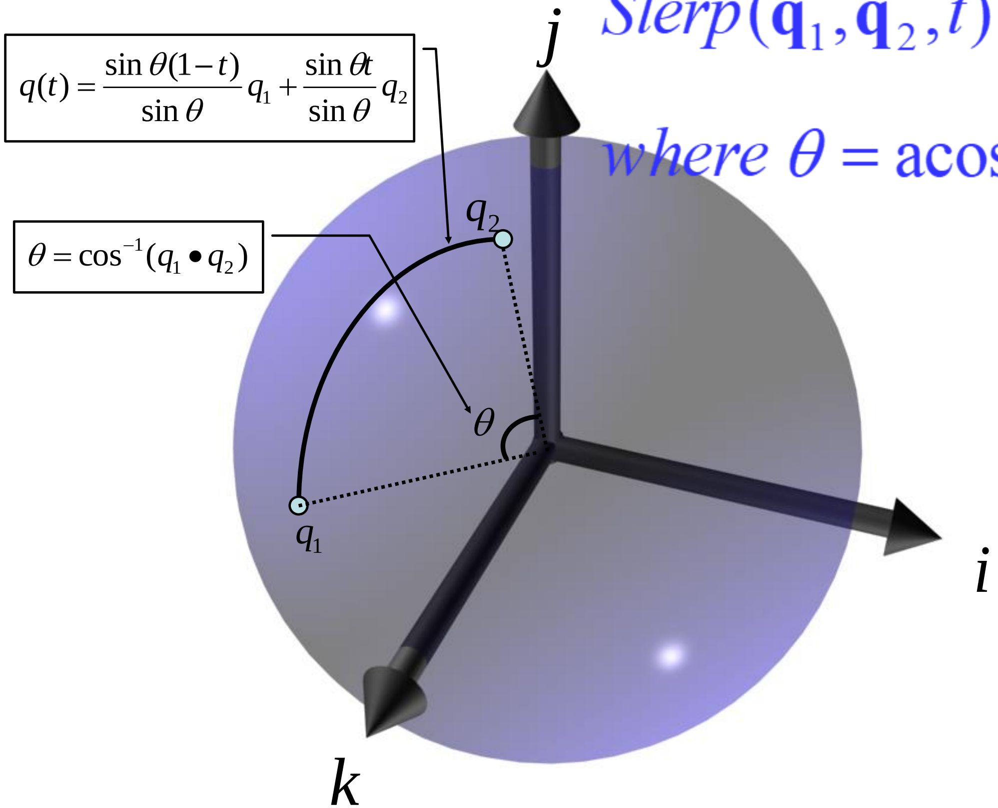


Spherical Linear Interpolation (SLERP)

- If we want to interpolate between two points on a sphere (or hypersphere), we will travel across the surface of the sphere by following a 'great arc.'

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

$$where\ \theta = \mathrm{acos}(\mathbf{q}_1 \cdot \mathbf{q}_2)$$



Spherical Cubic Interpolation (SQUAD)

- To achieve C^2 continuity between curve segments, a cubic interpolation must be done.
- Squad does a cubic interpolation between four quaternions by t

$$Squad(q_i, q_{i+1}, a_i, a_{i+1}, t)$$

$$= slerp(slerp(q_i, q_{i+1}, t), slerp(a_i, a_{i+1}, t), 2t(1-t))$$

$$a_i = q_i * \exp\left(\frac{-\log(q_i^{-1} * q_{i-1}) + \log(q_i^{-1} * q_{i+1})}{4} \right)$$

$$a_{i+1} = q_{i+1} * \exp\left(\frac{-\log(q_{i+1}^{-1} * q_i) + \log(q_{i+1}^{-1} * q_{i+2})}{4} \right)$$

  - a_i, a_{i+1} are inner quadrangle quaternions between q1 and q2. And you have to choose carefully so that continuity is guaranteed across segments.