Geometric Objects and Transformation

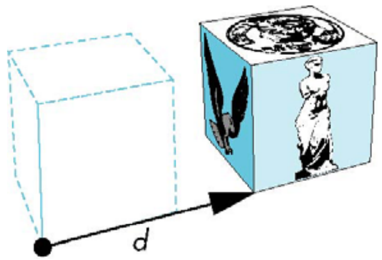
321190
2014년 봄학기
4/11/2014
박경신

3D Transformations

- 일반적으로 3차원 변환은 2차원 변환의 확장으로 생각하면 된다.
- 변환 행렬은 각각 한 행과 열이 추가된다.
- 3차원의 이동 (Translation), 크기변환 (Scaling), 밀림변환 (Shearing)의 기본 원리는 2차원과 같다.
- 그러나, 3차원의 회전은 좀 더 복잡하다.

3D Translation

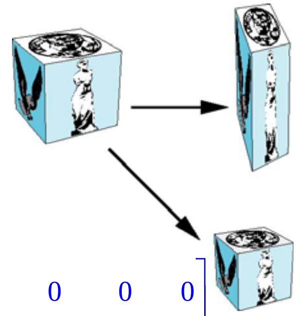
$$p' = p + d \quad p = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad p' = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \quad d = \begin{bmatrix} dx \\ dy \\ dz \\ 0 \end{bmatrix}$$



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

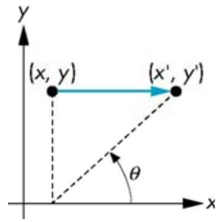
3D Scale

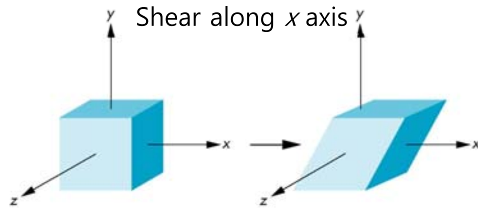
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



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

3D Shear





x′ = x + y cot θ
y′ = y
z′ = z

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

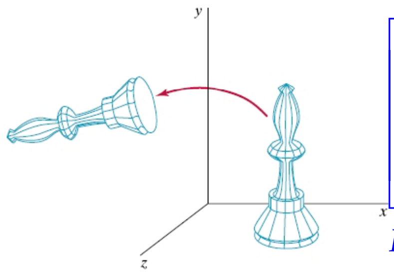
$$\tan\theta = \frac{y}{x'-x} \Rightarrow \cot\theta = \frac{x'-x}{y}$$

3D Rotation

- Z-축으로 회전

x′ = x cosθ − y sinθ
y′ = x sinθ + y cosθ
z′ = z

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



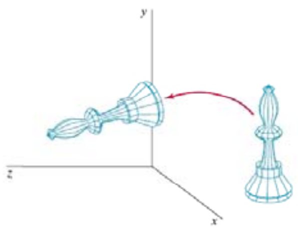
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

3D Rotation

- X-축으로 회전

y′ = y cosθ − z sinθ
z′ = y sinθ + z cosθ
x′ = x



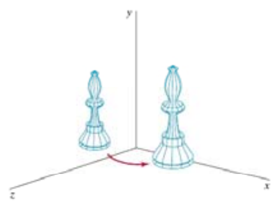
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

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

3D Rotation

- Y-축으로 회전

x′ = x cosθ + z sinθ
z′ = -x sinθ + z cosθ
y′ = y



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

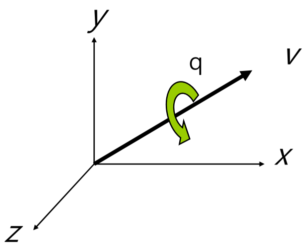
$$P' = R_Y(\theta) \cdot P$$

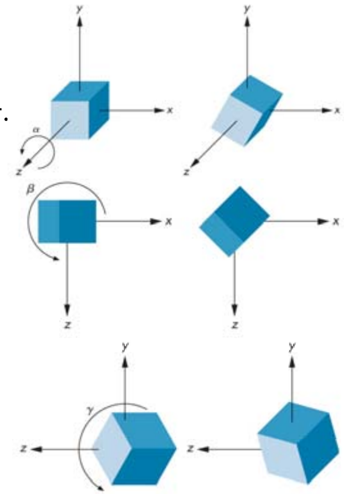
3D Rotation about the Origin

- 원점(0, 0, 0)에서 임의의 회전은 세 축에 대한 세 번의 연속적인 회전으로 구성할 수 있다.

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

  θ_X, θ_Y, θ_Z를 오일러 앵글이라 부른다.

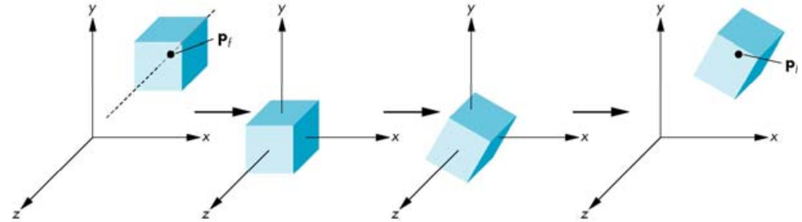




Rotation About a Pivot other than the Origin

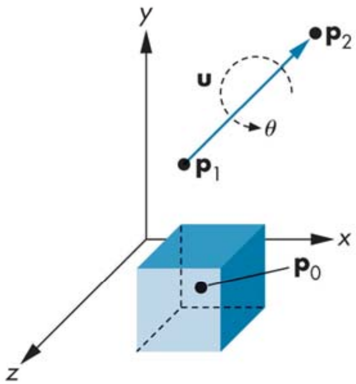
- 고정점 Pivot (P_f)을 원점(0, 0, 0)으로 이동 후, 회전 후, 다시 Pivot으로 이동한다.
- **M** = **T**(p_f) **R$_Z$** (θ) **T**($-p_f$)

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



3D Rotation about an Arbitrary Axis

- P_0 를 원점으로 이동한다.
- 임의의 축 u를 z-축에 정렬 (align) 시키기 위해 두 번 회전을 수행한다.
- Z-축으로 θ만큼 회전한다.
- 두 번의 회전을 되돌린다 (undo alignment).
- 다시 P_0 로 이동한다.



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

3D Rotation about an Arbitrary Axis

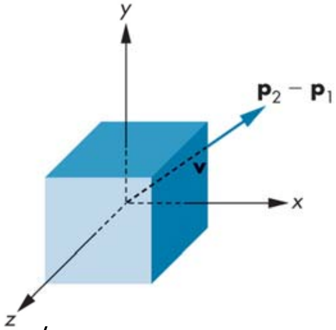
- The translation matrix, $T(-P_0)$

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

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:

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

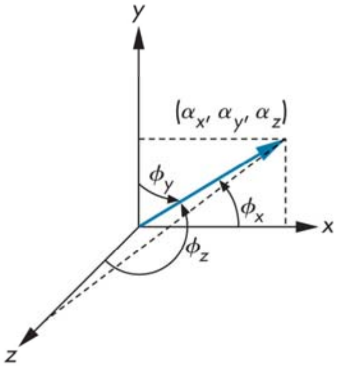


- Rotate along x-axis until *v* hits *xz-plane*
- Rotate along y-axis until *v* hits *z-axis*

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:

$$\cos\phi_x = \alpha_x$$
$$\cos\phi_y = \alpha_y$$
$$\cos\phi_z = \alpha_z$$
$$\cos^2\phi_x + \cos^2\phi_y + \cos^2\phi_z = 1$$

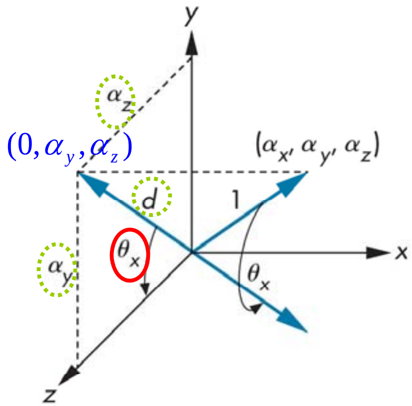


3D Rotation about an Arbitrary Axis

- Compute x-rotation θ_x

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

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



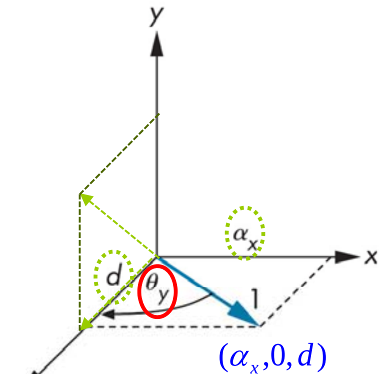
$$\cos\theta_x = \frac{\alpha_z}{d}$$
$$\sin\theta_x = \frac{\alpha_y}{d}$$

3D Rotation about an Arbitrary Axis

- Compute y-rotation θ_y

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

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



주의: y-축에 대하여 시계방향

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

3D Rotation about an Arbitrary Axis

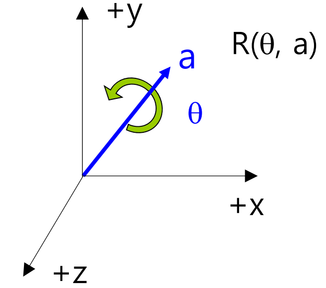
- Rotation about the z axis

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

- 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

- 임의의 회전축 (axis)에 대한 하나의 회전각도 (angle) 4개의 숫자로 표현한다.
- 임의의 회전축을 나타내는 단위벡터 a (x, y, z)와 단위 벡터 주위로 회전각도를 나타내는 θ (0~360)값으로 구성된다.
- 궁극적으로 axis/angle을 하나의 3D rotation vector로 바꿀 수 있다.



3D Rotation about an Arbitrary Axis

- 회전축(axis)/각(angle)로부터 다음과 같은 회전행렬 (rotation matrix)을 만든다.

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

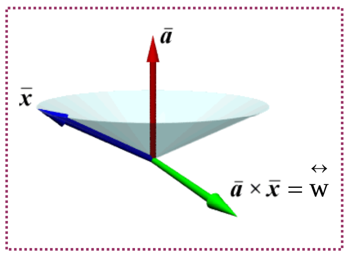
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\cos\theta + \begin{bmatrix} a_x^2 & a_x a_y & a_x a_z \\ a_x a_y & a_y^2 & a_y a_z \\ a_x a_z & a_y a_z & a_z^2 \end{bmatrix}(1-\cos\theta) + \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}\sin\theta$$

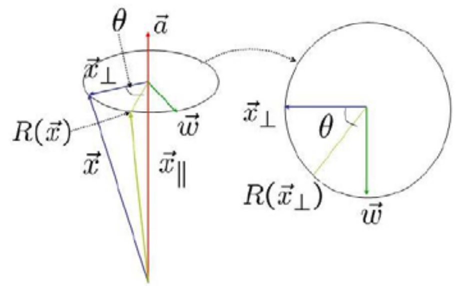
$$= \begin{bmatrix} a_x^2 + \cos\theta(1-a_x^2) & a_x a_y(1-\cos\theta) - a_z\sin\theta & a_x a_z(1-\cos\theta) + a_y\sin\theta \\ a_x a_y(1-\cos\theta) + a_z\sin\theta & a_y^2 + \cos\theta(1-a_y^2) & a_y a_z(1-\cos\theta) - a_x\sin\theta \\ a_x a_z(1-\cos\theta) - a_y\sin\theta & a_y a_z(1-\cos\theta)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

3D Rotation as Vector Components

- 임의의 회전축 a =[a_x, a_y, a_z] 를 중심으로 θ만큼 회전변환

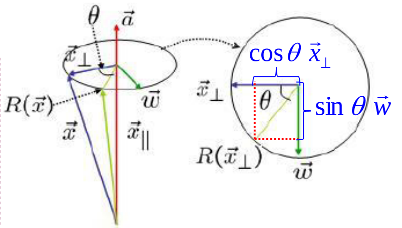
$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \left(\textbf{Symmetric}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right)(1-\cos\theta) + \textbf{Skew}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right)\sin\theta + \mathbf{I}\cos\theta \right)\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$





3D Rotation as Vector Components

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



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

$$\vec{x}_{||} = (\vec{a}\cdot\vec{x})\vec{a}$$
$$\vec{x}_\perp = \vec{x} - \vec{x}_{||} = \vec{x} - (\vec{a}\cdot\vec{x})\vec{a}$$

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

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 q.
- 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.

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

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

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.

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

$$\text{Skew}(\bar{a})\bar{x} = \bar{a} \times \bar{x}$$

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

- 사원수(Quaternion)란 3차원 그래픽스에서 회전을 표현할 때, 행렬 대신에 사용하는 수학적 개념으로 사원수는 4차원 복소수 공간 (complex space) 벡터이다.
- 실제로 회전의 표현에 있어서 가장 효과적인 방법이다.
- 사원수 (quaternion)는 4개의 구성요소로 표현한다.

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

Quaternion (Imaginary Space)

- 사원수는 실제로 복소수(complex numbers)의 확장이다.
- 4개 중에 하나는 실수 (scalar number)이고 다른 세 개는 허수의 공간 i, j, k에 있는 복소수이다.

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

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

$$i=jk=-kj$$

$$j=ki=-ik$$

$$k=ij=-ji$$

Quaternion (Scalar/Vector)

- 사원수는 또한 스칼라 값 s와 벡터 값 v 로 표현된다.

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

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

$$s = w$$

Identity Quaternion

- 벡터와는 달리 2개의 항등 사원수 (Identity quaternion)가 있다.
- 곱셈 항등 사원수 (multiplication identity quaternion) – 그래서 이 곱셈 항등 사원수와 곱해진 어떤 사원수도 변하지 않는다:

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

- 덧셈 단위 사원수 (addition identity quaternion) – 여기서는 사용하지 않는다:

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

Unit Quaternion

- 사원수 연산의 편리함을 위하여 단위 사원수 (unit length quaternion)을 사용한다.
- 단위 사원수 (unit length quaternion)는 사원수의 크기가 1이다. 이것은 4차원 공간에서 단위 길이를 가지는 구 (hypersphere)의 surface (즉, 4차원 공간에서의 3차원 부피)를 형성하는 벡터를 이룬다.

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

- 사원수의 정규화 (normalization)은 아래와 같이 구한다.

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

Quaternion as Rotations

- 사원수는 벡터의 회전과 밀접한 관계가 있는데 회전축 (axis **a**) 와 각도 (angle **θ**)로 나타낼 수 있다.

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

or

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

- 회전축 **a** 가 단위길이를 갖는다면, 사원수 **q**도 마찬가지로 단위길이를 갖는다.

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

- 최종적으로 얻어진 사원수를 실제 프로그램에서 회전에 사용하기 위해서는 다음과 같은 행렬로 변환:

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

- 단위 사원수가 $x^2+y^2+z^2+w^2=1$를 갖는 점을 이용하여, 회전행렬을 좀더 줄이면:

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

Quaternion to Axis/Angle

- 사원수를 3차원 공간에서의 임의 회전축 a (a_x, a_y, a_z)과 각도(θ)에 의한 표현으로 변환:

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

$$ax = {}^{x}\!/_{scale}$$

$$ay = {}^{y}\!/_{scale}$$

$$az = {}^{z}\!/_{scale}$$

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

Quaternion Dot Product

- 두 개의 사원수 간의 내적 (dot product)은 두 개의 벡터 간의 내적과 같은 방식으로 계산하면 된다.

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

Quaternion Multiplication

- 단위 사원수는 3차원 공간에서의 한 방향을 표현하기 때문에, 두 개의 단위 사원수 간의 곱은 두 개의 단위 회전을 결합한 회전을 나타내는 단위 사원수가 된다.
- 사원수의 곱은 순서가 중요하다. 사원수의 곱은 교환법칙이 성립되지 않는다. qq′ ≠ q′q

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

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

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 q = [v sin q , cos q]
  - log(q) = log(v sin q + cos q) = log(exp(v q)) = v q

Quaternion Interpolation

- 사원수는 키 프레임 (key frames)간에 회전 보간 (interpolation)을 가장 효과적으로 표현할 수 있다.

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

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

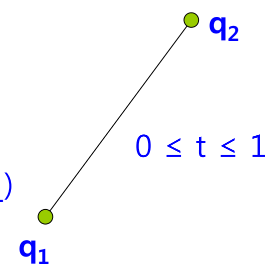
Linear Interpolation (LERP)

- 가장 쉬운 방식으로 두 개의 사원수간의 선형보간 (linear interpolation) 방식이 있다.

$$\text{Lerp}(\mathbf{q_1}, \mathbf{q_2}, t) = (1-t)\ \mathbf{q_1} + (t)\ \mathbf{q_2}$$
where $0 \le t \le 1$

- 선형보간 공식의 또 다른 표현:

$$\text{Lerp}(\mathbf{q_1}, \mathbf{q_2}, t) = \mathbf{q_1} + t(\mathbf{q_2} - \mathbf{q_1})$$

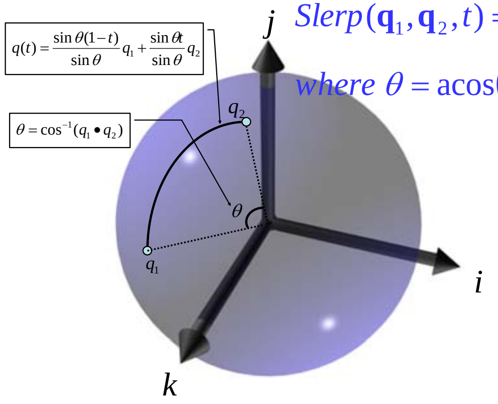


Spherical Linear Interpolation (SLERP)

- 구면 선형 보간 (spherical linear interpolation)은 벡터 q_1가 길이를 유지한 채로 회전해서 q_2가 되었다고 했을 때 회전한 그 사이 값을 보간하는 방법이다.

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

$$where\ \theta = \text{acos}\left(\mathbf{q}_1 \cdot \mathbf{q}_2\right)$$



Spherical Cubic Interpolation (SQUAD)

- 두 단위 사원수 q_i, q_{i+1} 사이에 a_i, a_{i+1}이라는 사원수를 도입한다. 구면 입방 보간 (spherical cubic interpolation)은 아래와 같이 정의한다.

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

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

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

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

- a_i 들은 초기 방향들에서의 접선 방향 (tangent orientation)을 표시하는데 사용된다.