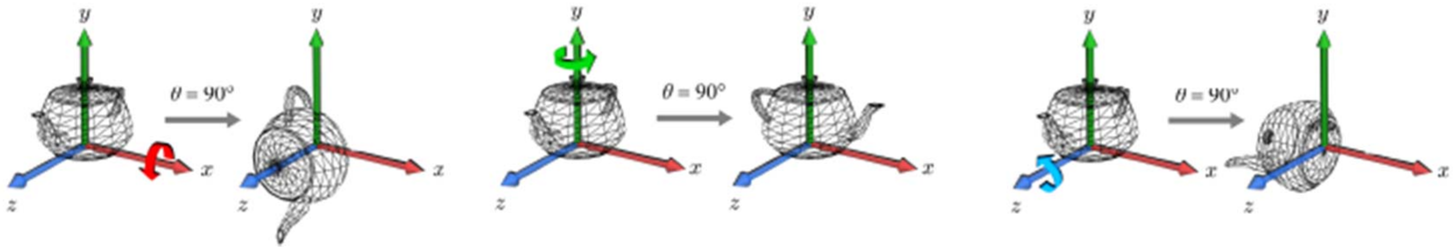
# Euler Transform and Quaternion

# *Rotations*

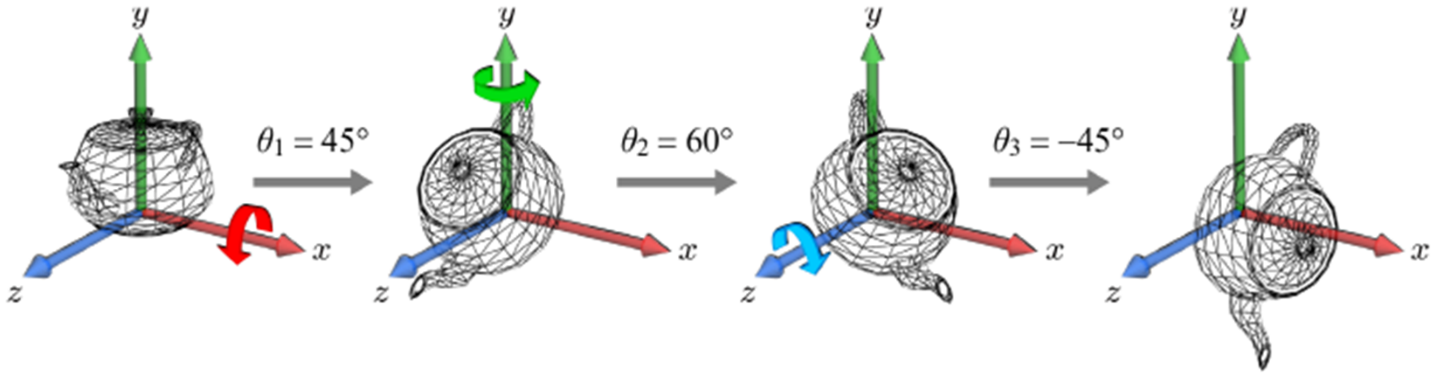
- We have learned the rotation matrices about the principal axes.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x & 0 \\ 0 & sin\theta_x & cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} cos\theta_y & 0 & sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -sin\theta_y & 0 & cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} cos\theta_z & -sin\theta_z & 0 & 0 \\ sin\theta_z & cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



# *Euler Transform*

- When we successively rotate an object about the principal axes, the object acquires an arbitrary orientation. This method of determining an object's orientation is called *Euler transform*, and the rotations angles $(\theta_1, \theta_2, \theta_3)$ or $(\theta_x, \theta_y, \theta_z)$ are called the *Euler angles*.




- Concatenating the rotation matrices produces a single matrix:

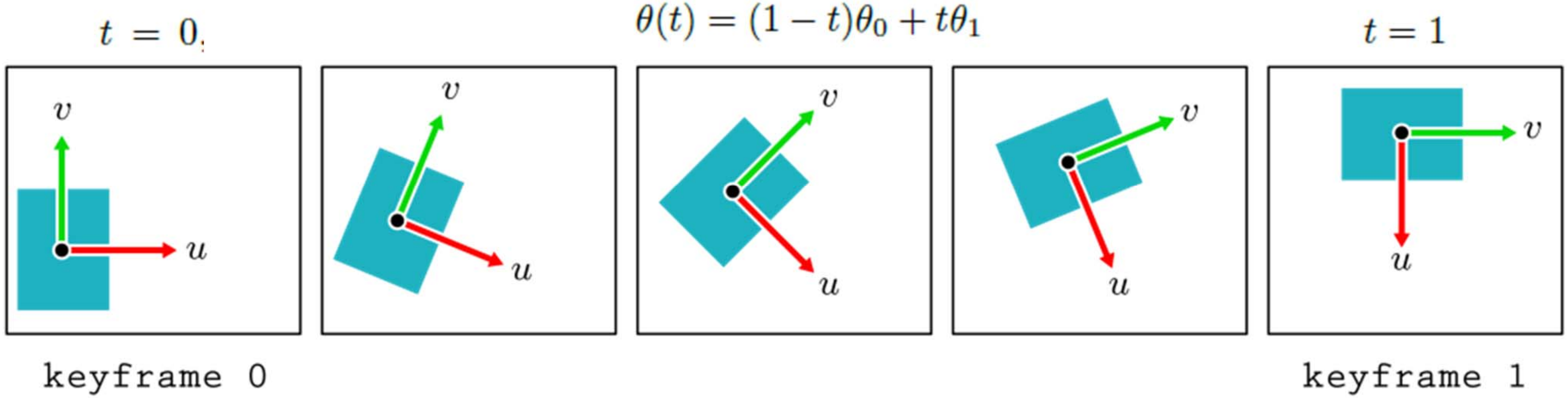
$$
R_z(-45°)R_y(60°)R_x(45°) = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 0 & \frac{\sqrt{3}}{2} \\ 0 & 1 & 0 \\ -\frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{pmatrix}
$$
$$
= \begin{pmatrix} \frac{\sqrt{2}}{4} & \frac{2+\sqrt{3}}{4} & \frac{-2+\sqrt{3}}{4} \\ -\frac{\sqrt{2}}{4} & \frac{2-\sqrt{3}}{4} & \frac{-2-\sqrt{3}}{4} \\ \frac{-\sqrt{3}}{2} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \end{pmatrix}
$$

# *Keyframe Animation in 2D*

- In the traditional hand-drawn cartoon animation,
  - the senior key artist would draw the *keyframes*, and
  - the junior artist would fill the *in-between frames*.
- For a 30-fps computer animation, much fewer than 30 frames are defined per second. They are the keyframes. In real-time computer animation, the in-between frames are automatically filled at run time.
- The key data are assigned to the keyframes, and they are interpolated to generate the in-between frames. Any data that change in the time domain can be interpolated,
- In the example, the center *p* and orientation angle $\theta$ are interpolated.

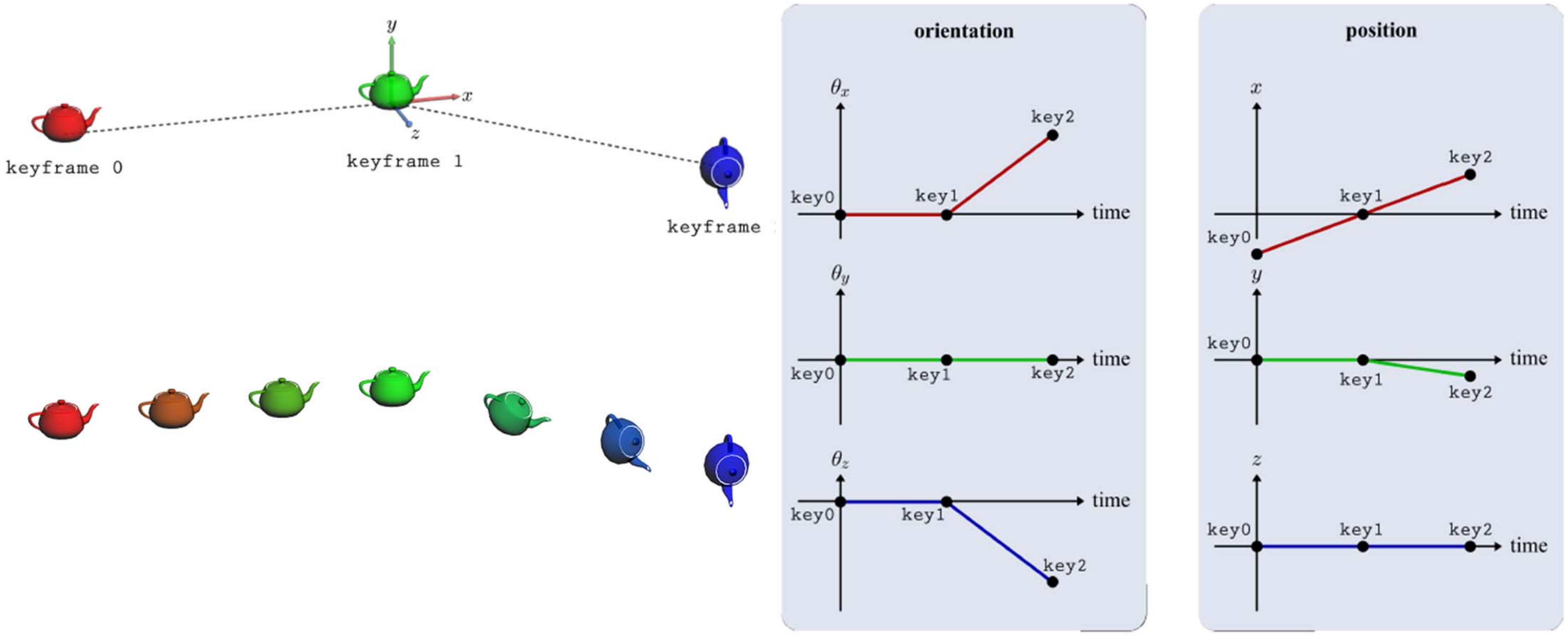
$$p(t) = (1-t)p_0 + tp_1$$

$$\theta(t) = (1-t)\theta_0 + t\theta_1$$



# *Keyframe Animation in 3D*

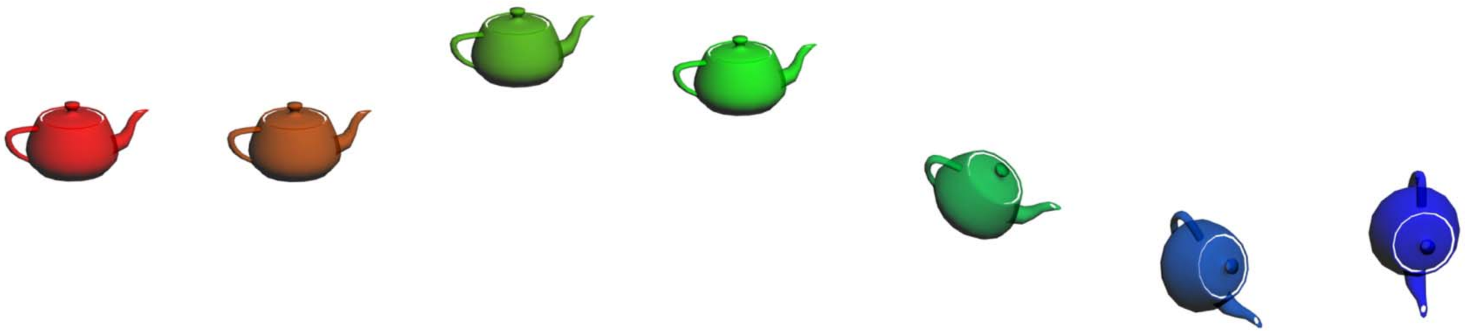
- Keyframe animation in 3D: Seven teapot instances are defined by sampling the graphs seven times.



# *Keyframe Animation in 3D (cont'd)*

- Smoother animation may often be obtained using a higher-order interpolation.

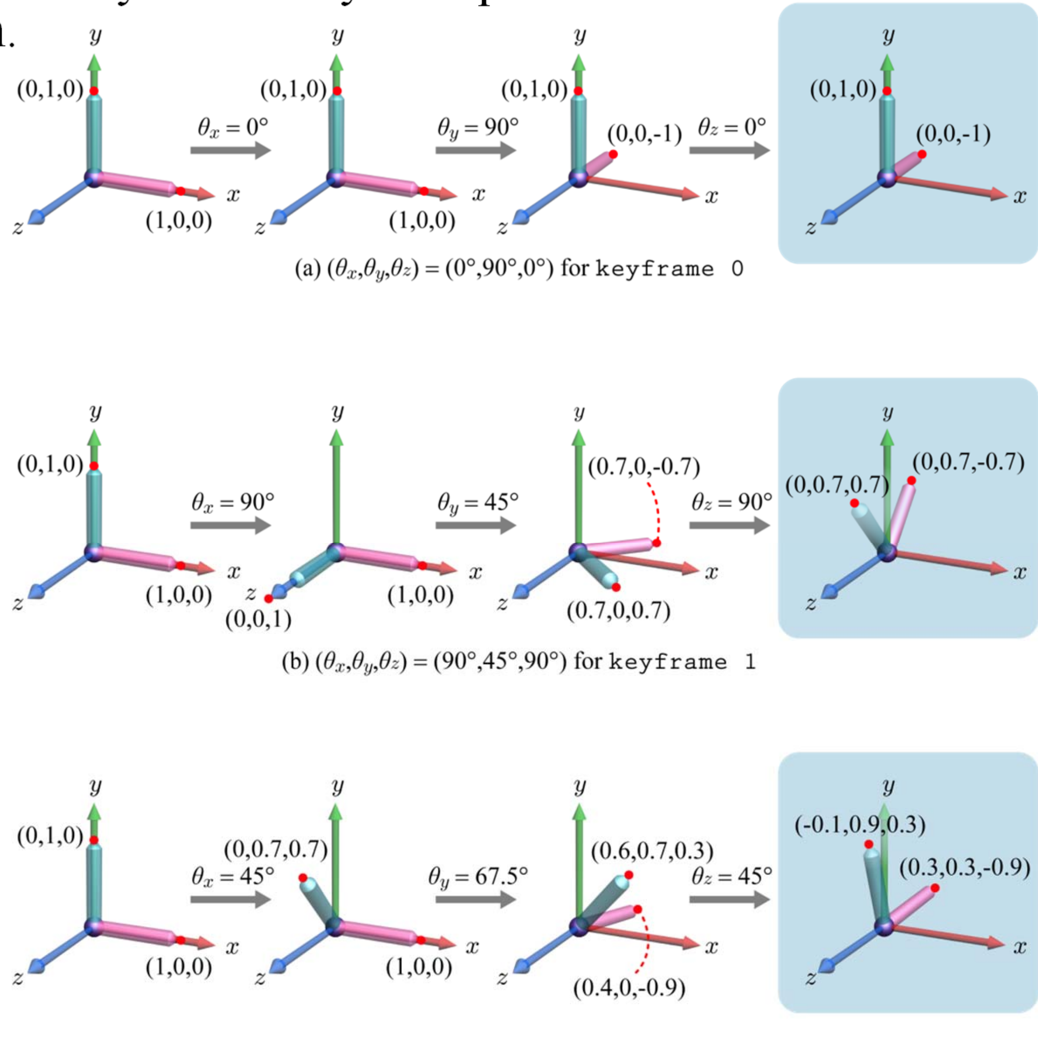

(a)



(b)

# A Problem of Euler Angles

- Euler angles are not always correctly interpolated and so are not suitable for keyframe animation.




(c) Interpolated Euler angles $(\theta_x,\theta_y,\theta_z) = (45°,67.5°,45°)$

# *Quaternion*

- A quaternion is an extended complex number.

$$q_x i + q_y j + q_z k + q_w = (q_x, q_y, q_z, q_w) = (\mathbf{q}_v, q_w)$$

$$\begin{aligned} i^2 &= j^2 = k^2 = -1 \\ ij &= k, ji = -k \\ jk &= i, kj = -i \\ ki &= j, ik = -j \end{aligned}$$

$$\begin{aligned} \mathbf{p} &= (p_x, p_y, p_z, p_w) \\ \mathbf{q} &= (q_x, q_u, q_z, q_w) \\ \mathbf{pq} &= (p_x i + p_y j + p_z k + p_w)(q_x i + q_y j + q_z k + q_w) \\ &= (p_x q_w + p_y q_z - p_z q_y + p_w q_x)i + \\ &\quad (-p_x q_z + p_y q_w + p_z q_x + p_w q_y)j + \\ &\quad (p_x q_y - p_y q_x + p_z q_w + p_w q_z)k + \\ &\quad (-p_x q_x - p_y q_y - p_z q_z + p_w q_w) \end{aligned}$$

- Conjugate

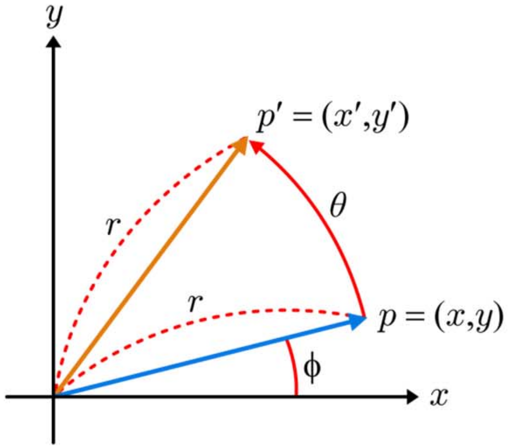
$$\begin{aligned} \mathbf{q}^* &= (-\mathbf{q}_v, q_w) \\ &= (-q_x, -q_y, -q_z, q_w) \\ &= -q_x i - q_y j - q_z k + q_w \end{aligned}$$

- Magnitude (If the magnitude of a quaternion is 1, it's called a unit quaternion.)

$$\|\mathbf{q}\| = \sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2}$$

# 2D Rotation through Quaternion

- Recall 2D rotation




$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} xcos\theta - ysin\theta \\ xsin\theta + ycos\theta \end{pmatrix}$$

- Let us represent ($x$,$y$) by a complex number $x$+$yi$, and denote it by **p**.
- Given the rotation angle $\theta$, let us consider a unit-length complex number, $cos\theta+sin\theta i$. We denote it by **q**. Then, we have the following:

$$\begin{aligned} \mathbf{pq} &= (x+yi)(cos\theta + sin\theta i) \\ &= (xcos\theta - ysin\theta) + (xsin\theta + ycos\theta)i \end{aligned}$$

- Surprisingly, the real and imaginary parts of **pq** represent the rotated coordinates.

# *3D Rotation through Quaternion*

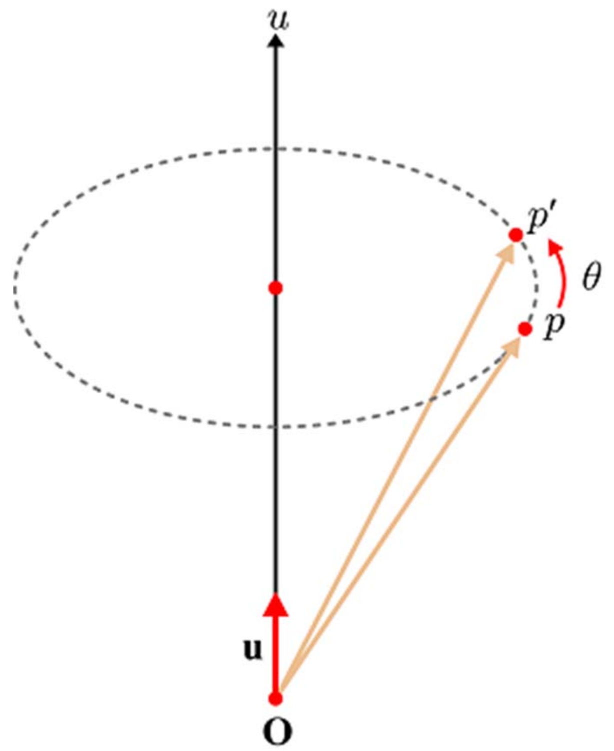
- As extended complex numbers, quaternions can be used to describe 3D rotation.
- Consider rotating a 3D vector $p$ about an axis $u$ by an angle $\theta$. Both "the vector to be rotated" and "the rotation" are represented in quaternions.
  - Vector $p$ to a quaternion **p**

$$\begin{aligned}\mathbf{p} &= (\mathbf{p}_v, p_w) \\ &= (p, 0)\end{aligned}$$

  - The rotation axis $u$ and rotation angle $\theta$ define another quaternion **q**. (The axis $u$ is divided by its length to make a unit vector **u**.)

$$\begin{aligned}\mathbf{q} &= (\mathbf{q}_v, q_w) \\ &= (sin\tfrac{\theta}{2}\mathbf{u}, cos\tfrac{\theta}{2})\end{aligned}$$

  - Then, the rotated vector is equivalent to the imaginary part of $\mathbf{q}\mathbf{p}\mathbf{q}^*$ .



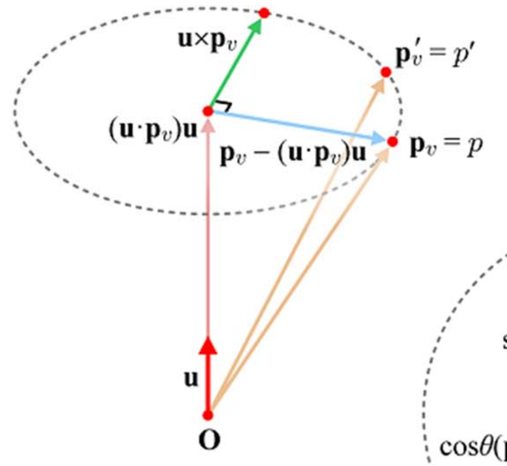
# 3D Rotation through Quaternion (cont'd)

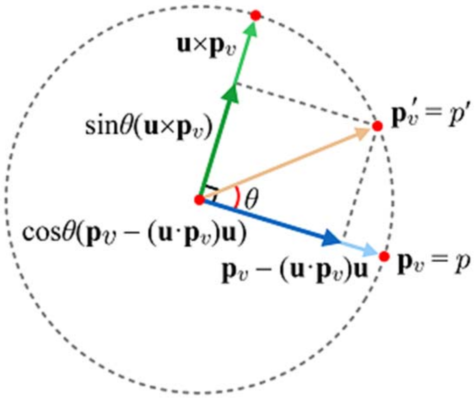
- Proof

$$\mathbf{p} = (\mathbf{p}_v, p_w) \\ = (p, 0)$$

$$\mathbf{q} = (\mathbf{q}_v, q_w) \\ = (sin\tfrac{\theta}{2}\mathbf{u}, cos\tfrac{\theta}{2})$$

$$\begin{aligned}\mathbf{pq} = &(p_x q_w + p_y q_z - p_z q_y + p_w q_x)i + \\ &(-p_x q_z + p_y q_w + p_z q_x + p_w q_y)j + \\ &(p_x q_y - p_y q_x + p_z q_w + p_w q_z)k + \\ &(-p_x q_x - p_y q_y - p_z q_z + p_w q_w) \\ = &(\mathbf{p}_v \times \mathbf{q}_v + q_w \mathbf{p}_v + p_w \mathbf{q}_v, p_w q_w - \mathbf{p}_v \cdot \mathbf{q}_v)\end{aligned}$$

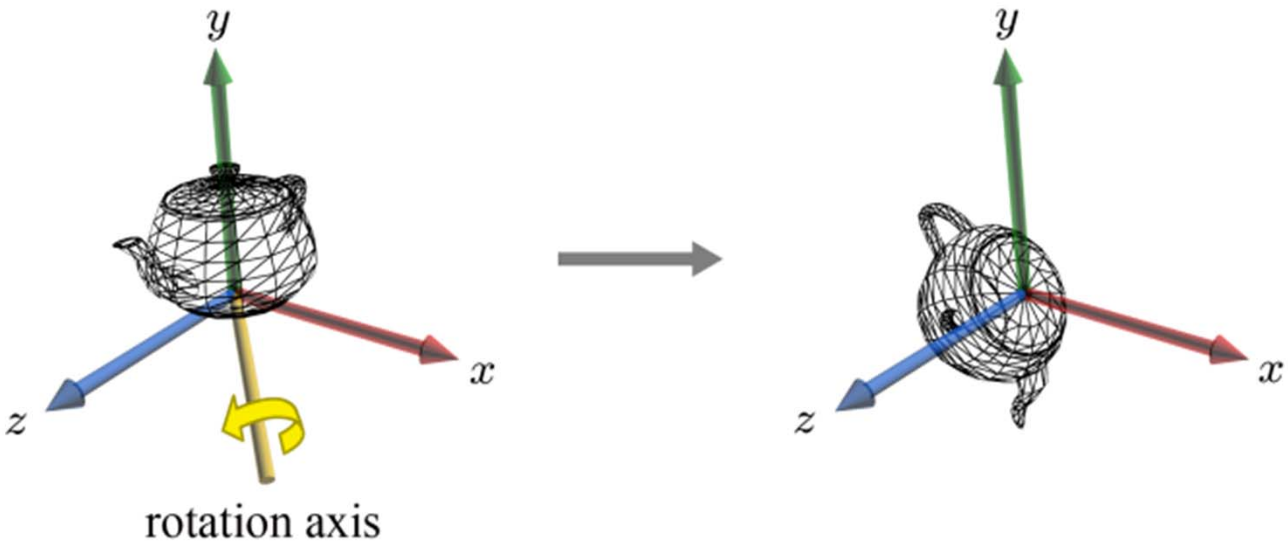



$$\begin{aligned}\mathbf{qpq}^* &= (\mathbf{q}_v \times \mathbf{p}_v + q_w \mathbf{p}_v, -\mathbf{q}_v \cdot \mathbf{p}_v)\mathbf{q}^* \\ &= (\mathbf{q}_v \times \mathbf{p}_v + q_w \mathbf{p}_v, -\mathbf{q}_v \cdot \mathbf{p}_v)(-\mathbf{q}_v, q_w) \\ &= ((\mathbf{q}_v \times \mathbf{p}_v + q_w \mathbf{p}_v) \times (-\mathbf{q}_v) + q_w(\mathbf{q}_v \times \mathbf{p}_v + q_w \mathbf{p}_v) + (-\mathbf{q}_v \cdot \mathbf{p}_v)(-\mathbf{q}_v), \\ &\quad (-\mathbf{q}_v \cdot \mathbf{p}_v)q_w - (\mathbf{q}_v \times \mathbf{p}_v + q_w \mathbf{p}_v) \cdot (-\mathbf{q}_v)) \\ &= ((\mathbf{q}_v \cdot \mathbf{p}_v)\mathbf{q}_v - (\mathbf{q}_v \cdot \mathbf{q}_v)\mathbf{p}_v + 2q_w(\mathbf{q}_v \times \mathbf{p}_v) + q_w^2 \mathbf{p}_v + (\mathbf{q}_v \cdot \mathbf{p}_v)\mathbf{q}_v, 0) \\ &= (2(\mathbf{q}_v \cdot \mathbf{p}_v)\mathbf{q}_v + (q_w^2 - \|\mathbf{q}_v\|^2)\mathbf{p}_v + 2q_w(\mathbf{q}_v \times \mathbf{p}_v), 0) \\ &= (2sin^2\tfrac{\theta}{2}(\mathbf{u} \cdot \mathbf{p}_v)\mathbf{u} + (cos^2\tfrac{\theta}{2} - sin^2\tfrac{\theta}{2})\mathbf{p}_v + 2cos\tfrac{\theta}{2}sin\tfrac{\theta}{2}(\mathbf{u} \times \mathbf{p}_v), 0) \\ &= ((1 - cos\theta)(\mathbf{u} \cdot \mathbf{p}_v)\mathbf{u} + cos\theta\mathbf{p}_v + sin\theta(\mathbf{u} \times \mathbf{p}_v), 0) \\ &= ((\mathbf{u} \cdot \mathbf{p}_v)\mathbf{u} + cos\theta(\mathbf{p}_v - (\mathbf{u} \cdot \mathbf{p}_v)\mathbf{u}) + sin\theta(\mathbf{u} \times \mathbf{p}_v), 0)\end{aligned}$$

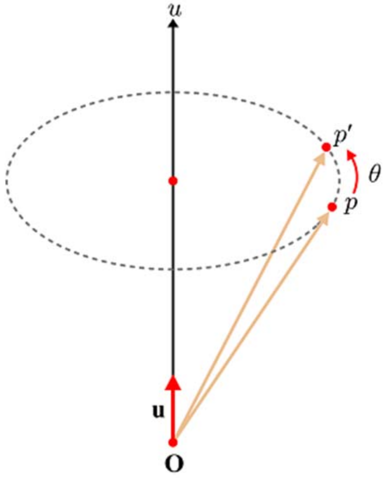
# *3D Rotation through Quaternion (cont'd)*

- Rotation about an arbitrary axis that is not limited to a principal axis.




# *3D Rotation through Quaternion (cont'd)*

- Consider rotating *p´* by another quaternion **r**.

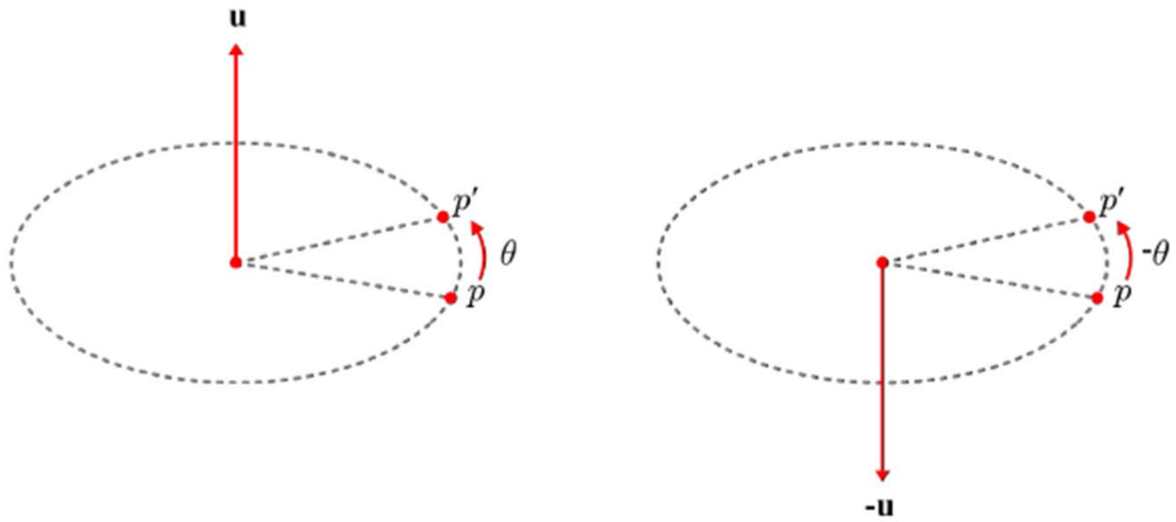



$$
\begin{aligned}
\mathbf{r}\mathbf{p}'\mathbf{r}^* &= \mathbf{r}(\mathbf{q}\mathbf{p}\mathbf{q}^*)\mathbf{r}^* \\
&= (\mathbf{r}\mathbf{q})\mathbf{p}(\mathbf{q}^*\mathbf{r}^*) \\
&= (\mathbf{r}\mathbf{q})\mathbf{p}(\mathbf{r}\mathbf{q})^*
\end{aligned}
$$

- The composite quaternion **rq** represents the combined rotation.

# *3D Rotation through Quaternion (cont'd)*

- "Rotation about **u** by $\theta$" equals "rotation about -**u** by -$\theta$."




- It can be proven: $$\begin{aligned}\mathbf{q}' &= (sin\tfrac{-\theta}{2}(-\mathbf{u}), cos\tfrac{-\theta}{2}) \\ &= (sin\tfrac{\theta}{2}\mathbf{u}, cos\tfrac{\theta}{2})\end{aligned}$$

- Consider the quaternion for "rotation about **u** by $2\pi+\theta$."

$$\begin{aligned}(sin\tfrac{2\pi+\theta}{2}\mathbf{u}, cos\tfrac{2\pi+\theta}{2}) &= (sin(\pi+\tfrac{\theta}{2})\mathbf{u}, cos(\pi+\tfrac{\theta}{2})) \\ &= (-sin\tfrac{\theta}{2}\mathbf{u}, -cos\tfrac{\theta}{2}) \\ &= -\mathbf{q}\end{aligned}$$

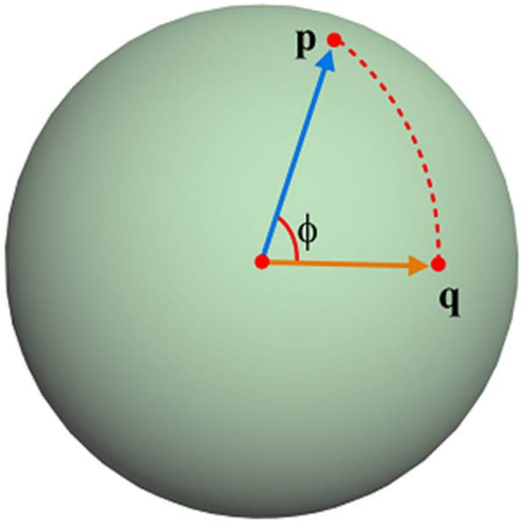
- This shows that -**q** and **q** represent the same rotation.

# *Interpolation of Quaternions*

- Consider two unit quaternions, **p** and **q**, which represent rotations. They can be interpolated using parameter *t* in the range of [0,1]:

$$\frac{sin(\phi(1-t))}{sin\phi}\mathbf{p} + \frac{sin(\phi t)}{sin\phi}\mathbf{q}$$

$$cos\phi = \mathbf{p}\cdot\mathbf{q} = (p_x, p_y, p_z, p_w)\cdot(q_x, q_y, q_z, q_w) = p_x q_x + p_y q_y + p_z q_z + p_w q_w.$$




- This is called spherical linear interpolation (slerp).

# *Interpolation of Quaternions (cont'd)*

- Proof

$$\mathbf{r} = l_1\mathbf{p} + l_2\mathbf{q}$$

$$sin\phi = \frac{h_1}{l_1}$$

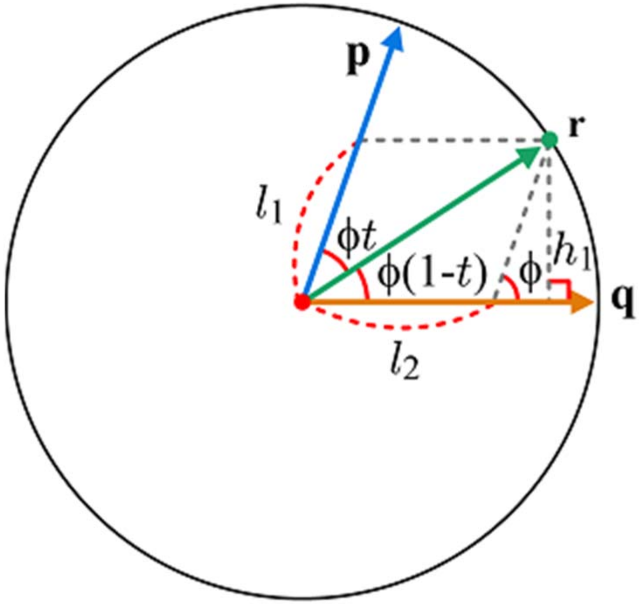
$$l_1 = \frac{h_1}{sin\phi}$$

$$h_1 = sin(\phi(1-t))$$

$$l_1 = \frac{sin(\phi(1-t))}{sin\phi}$$

$$l_2 = \frac{sin(\phi t)}{sin\phi}$$

$$\frac{sin(\phi(1-t))}{sin\phi}\mathbf{p} + \frac{sin(\phi t)}{sin\phi}\mathbf{q}$$



# *Quaternion and Matrix*

- A quaternion **q** representing a rotation can be converted into a matrix form. If **q** = $(q_x,q_y,q_z,q_w)$, the rotation matrix is defined as follows:

$$\begin{pmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y-q_wq_z) & 2(q_xq_z+q_wq_y) & 0 \\ 2(q_xq_y+q_wq_z) & 1-2(q_x^2+q_z^2) & 2(q_yq_z-q_wq_x) & 0 \\ 2(q_xq_z-q_wq_y) & 2(q_yq_z+q_wq_x) & 1-2(q_x^2+q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# *Quaternion and Matrix (cont'd)*

- Proof

$$
\begin{aligned}
\mathbf{pq} &= (p_x i + p_y j + p_z k + p_w)(q_x i + q_y j + q_z k + q_w) \\
&= (p_x q_w + p_y q_z - p_z q_y + p_w q_x) i + \\
&\quad (-p_x q_z + p_y q_w + p_z q_x + p_w q_y) j + \\
&\quad (p_x q_y - p_y q_x + p_z q_w + p_w q_z) k + \\
&\quad (-p_x q_x - p_y q_y - p_z q_z + p_w q_w)
\end{aligned}
$$

$$
\mathbf{pq} = \begin{pmatrix} q_w & q_z & -q_y & q_x \\ -q_z & q_w & q_x & q_y \\ q_y & -q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ p_w \end{pmatrix} = M_{\mathbf{q}}\mathbf{p}
\qquad
\mathbf{pq} = \begin{pmatrix} p_w & -p_z & p_y & p_x \\ p_z & p_w & -p_x & p_y \\ -p_y & p_x & p_w & p_z \\ -p_x & -p_y & -p_z & p_w \end{pmatrix} \begin{pmatrix} q_x \\ q_y \\ q_z \\ q_w \end{pmatrix} = N_{\mathbf{p}}\mathbf{q}
$$

$$
\begin{aligned}
\mathbf{qpq}^* &= (\mathbf{qp})\mathbf{q}^* \\
&= M_{\mathbf{q}^*}(\mathbf{qp}) \\
&= M_{\mathbf{q}^*}(N_{\mathbf{q}}\mathbf{p}) \\
&= (M_{\mathbf{q}^*}N_{\mathbf{q}})\mathbf{p} \\
&= \begin{pmatrix} q_w & -q_z & q_y & -q_x \\ q_z & q_w & -q_x & -q_y \\ -q_y & q_x & q_w & -q_z \\ q_x & q_y & q_z & q_w \end{pmatrix} \begin{pmatrix} q_w & -q_z & q_y & q_x \\ q_z & q_w & -q_x & q_y \\ -q_y & q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ p_w \end{pmatrix}
\end{aligned}
$$

$$(q_w^2 - q_z^2 - q_y^2 + q_x^2) = (1 - 2(q_y^2 + q_z^2))$$

# *Quaternion and Matrix*

- Conversely, given a rotation matrix, we can compute the corresponding quaternion.
- Its proof requires to extract $\{q_x, q_y, q_z, q_w\}$ from the following matrix:

$$\begin{pmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y-q_wq_z) & 2(q_xq_z+q_wq_y) & 0 \\ 2(q_xq_y+q_wq_z) & 1-2(q_x^2+q_z^2) & 2(q_yq_z-q_wq_x) & 0 \\ 2(q_xq_z-q_wq_y) & 2(q_yq_z+q_wq_x) & 1-2(q_x^2+q_y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - Compute the sum of all diagonal elements.

$$4-4(q_x^2+q_y^2+q_z^2) = 4-4(1-q_w^2) = 4q_w^2$$

  - So, we obtain $q_w$.
  - Subtract $m_{12}$ from $m_{21}$ of the above matrix.

$$m_{21}-m_{12} = 2(q_xq_y+q_wq_z)-2(q_xq_y-q_wq_z) = 4q_wq_z$$

  - As we know $q_w$, we can compute $q_z$. Similarly, we can compute $q_x$ and $q_y$.
- Note that $q_w$ has two values. Will it make a problem? No.

# *Quaternion - Summary*

- Summary
  - An arbitrary 3D rotation is represented in a quaternion, as well as in Euler transform.
  - Quaternions are well interpolated through spherical linear interpolation.
  - A quaternion can be converted into a rotation matrix.

- If quaternions are defined for the keyframes,
  - they are spherically interpolated for the in-between frames and
  - they rotate the vectors througl $qpq^*$ .

- If Euler angles are defined for the keyframes,
  - the Euler angles for each keyframe determine a matrix,
  - the matrix is converted into a quaternion,
  - The quaternions are spherically interpolated for the in-between frames, and
  - they rotate the vectors through $qpq^*$ .

.