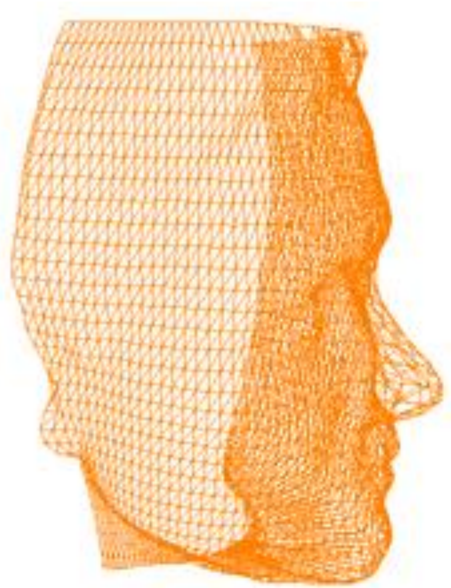
COSC422 Advanced Computer Graphics



8 Quaternions

Semester 2
2021

R. Mukundan (mukundan@canterbury.ac.nz)
Department of Computer Science and Software Engineering
University of Canterbury, New Zealand.

Quaternions in CG

- Mathematics
- physics

- Quaternions are being increasingly used by graphics and games programmers for implementing advanced **rotation interpolations**.
- GLM: `<glm/gtc/quaternion.hpp>`
- Assimp: `aiQuaternion q = (chnl->rotnKey).mValue`
- Almost every book on games programming contains a separate chapter/section on Quaternion algebra.
- Quaternions are also used in several other areas such as robotics, mechanics, aerospace technology and computer vision.

Outline

- General rigid body transformations
- **3D rotation interpolation**
- Extraction of rotation parameters from transformation matrices
- **Quaternion algebra**
- Representation of general 3D rotations using quaternions
- Quaternion interpolation
  - Quaternion Linear Interpolation (LERP)
  - Quaternion Spherical Linear Interpolation (**SLERP**)

Quaternions: Games and Animation

Extensively used for rotation interpolation, but least understood by developers.

Time: T_1
Matrix: `m_init`

How should an intermediate transformation be computed?

Time: T_2
Matrix: `m_final`

A sample implementation:

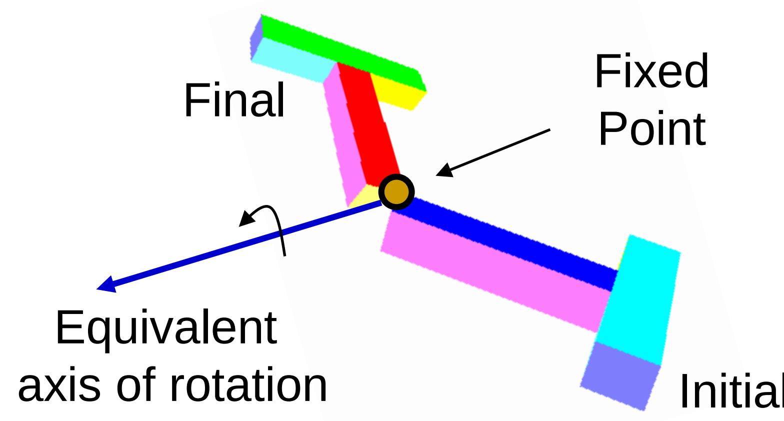
```
glm::mat4x4 m_init, m_final, m_inter; //Rotation matrices
glm::quat   q_init, q_final, q_inter;
q_init  = glm::quat_cast(m_init);     //Convert initial matrix to quat
q_final = glm::quat_cast(m_final);    //Convert final matrix to quat

for (float t = 0; t < 1; t += 0.1)    //Keyframe interpolation
{
   q_inter = glm::mix(q_init, q_final, t);  //Quat SLERP interpolation
   m_inter = glm::mat4_cast(q_inter);       //quat to matrix
   ...          //Use the matrix in transformations
}
```

Fundamental Properties of Transformations

- A **rigid-body transformation** comprises of only translations and rotations.
- Any rigid-body transformation with a fixed point is a rotation. The fixed point need not be part of the object.
- **Euler's Rotation Theorem**: Any general rigid-body transformation of an object with a fixed point can be represented by a single rotation about an axis through the fixed point.

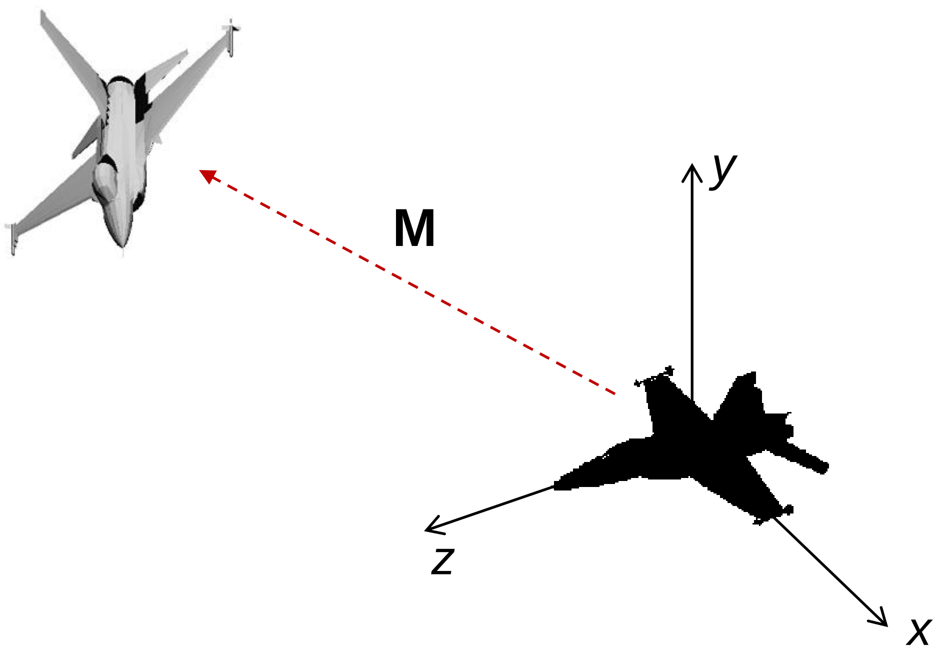
Axis: (l, m, n)
Angle: θ



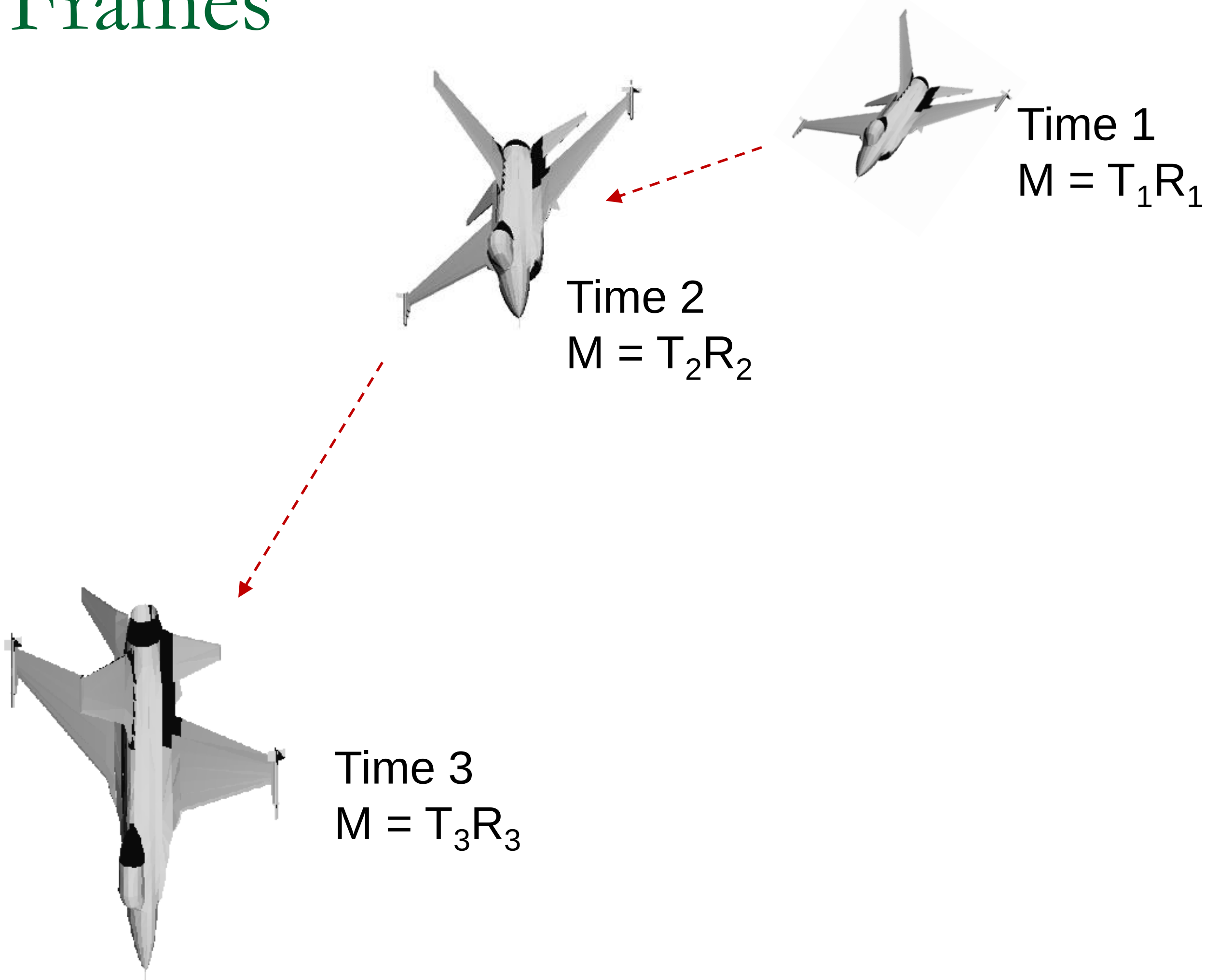
General 3D Rigid Body Transformations

- Every rigid-body transformation of an object can be expressed as a single rotation about an axis through **the origin**, followed by a single translation.
- Transformation matrix:

$$\mathbf{M} = \mathbf{TR} = \begin{bmatrix} \boxed{\begin{matrix} m_0 & m_4 & m_8 \\ m_1 & m_5 & m_9 \\ m_2 & m_6 & m_{10} \end{matrix}} & \boxed{\begin{matrix} m_{12} \\ m_{13} \\ m_{14} \end{matrix}} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix}$$

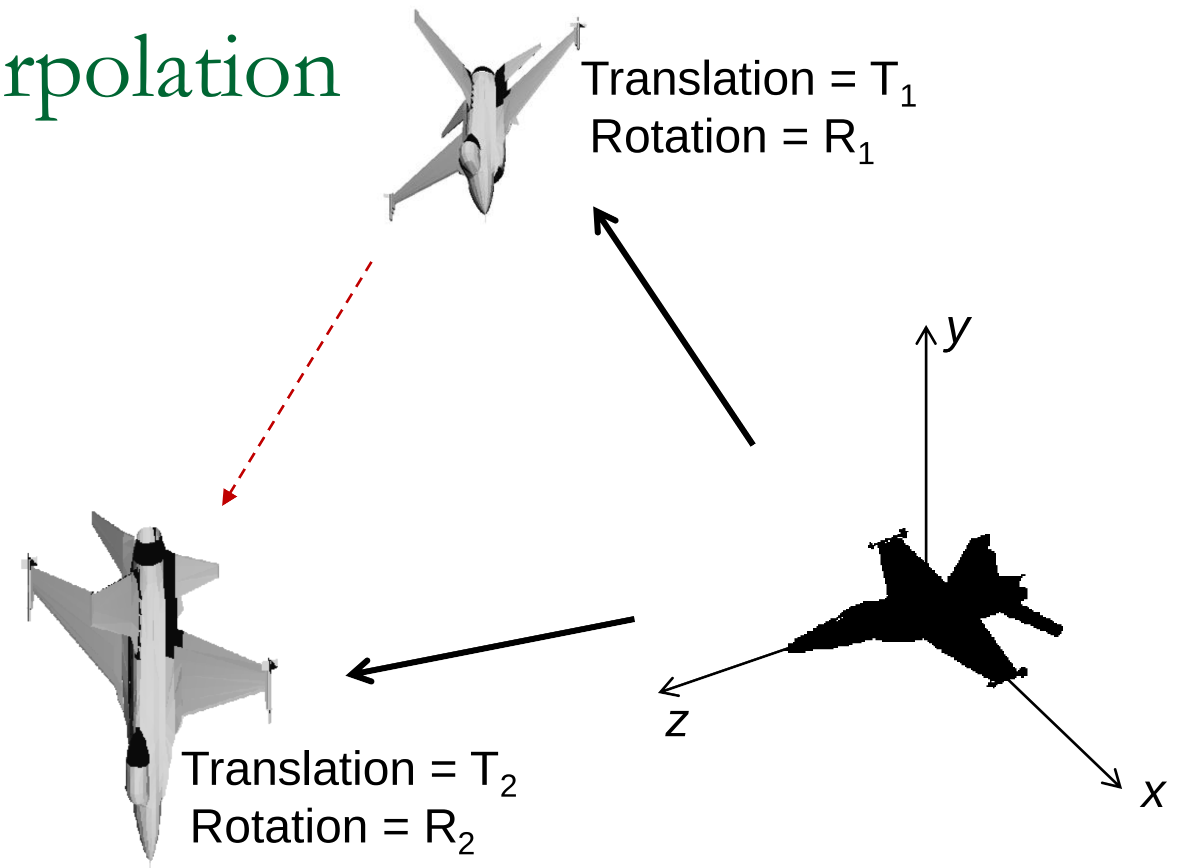


Key Frames



Keyframes specify the transformation parameters at discrete points in time in an object’s motion sequence

Frame Interpolation



We can perform a linear interpolation between the translation parameters T_1, T_2 to get an intermediate position of the object.

Can we perform a linear interpolation between two rotation matrices R_1, R_2 to get an intermediate orientation of the object?

Rotation Interpolation

- Rotational transforms are always represented by orthogonal matrices:

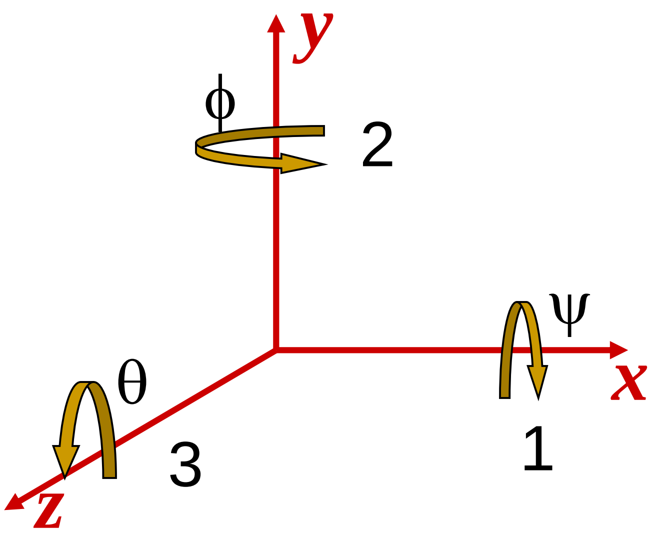
  $R^{-1} = R^T$. Equivalently, $R^T R = I$.

- Problem: Even if R_1 and R_2 are orthogonal matrices, $R = (1-t)R_1 + tR_2$ need not be orthogonal. Thus a linear interpolation will not give a rotational transformation matrix.

- We could however try to perform a linear interpolation between rotational parameters

Euler Angles

- A convenient representation of the most general form of rotation using three angles.
- Any rotation about the origin can be decomposed into a sequence of rotations about principal axes directions
  1. A rotation ψ about x
  2. followed by a rotation ϕ about y
  3. followed by a rotation θ about z.



Euler Angles to Transformation Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boxed{3}} \underbrace{\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boxed{2}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi & 0 \\ 0 & \sin\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boxed{1}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} c_2c_3 & s_1s_2c_3 - c_1s_3 & c_1s_2c_3 + s_1s_3 & 0 \\ c_2s_3 & s_1s_2s_3 + c_1c_3 & c_1s_2s_3 - s_1c_3 & 0 \\ -s_2 & c_2s_1 & c_2c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$c_1 = \cos\psi$ $\quad c_2 = \cos\phi$ $\quad c_3 = \cos\theta$

$s_1 = \sin\psi$ $\quad s_2 = \sin\phi$ $\quad s_3 = \sin\theta$

Transformation Matrix to Euler Angles

General Rotation Matrix

$$\begin{bmatrix} \boxed{\begin{matrix} m_0 & m_4 & m_8 \\ m_1 & m_5 & m_9 \\ m_2 & m_6 & m_{10} \end{matrix}} & \begin{matrix} m_{12} \\ m_{13} \\ m_{14} \end{matrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{bmatrix} \Longleftrightarrow \begin{bmatrix} c_2c_3 & s_1s_2c_3 - c_1s_3 & c_1s_2c_3 + s_1s_3 & 0 \\ c_2s_3 & s_1s_2s_3 + c_1c_3 & c_1s_2s_3 - s_1c_3 & 0 \\ -s_2 & c_2s_1 & c_2c_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Euler Rotation Matrix

$c_1 = \cos\psi$ $\quad c_2 = \cos\phi$ $\quad c_3 = \cos\theta$

$s_1 = \sin\psi$ $\quad s_2 = \sin\phi$ $\quad s_3 = \sin\theta$

$$\boxed{\psi = \tan^{-1}\left(\frac{m_6}{m_{10}}\right)} \quad \boxed{\phi = \tan^{-1}\left(\frac{-m_2}{\sqrt{m_0^2 + m_1^2}}\right)} \quad \boxed{\theta = \tan^{-1}\left(\frac{m_1}{m_0}\right)}$$

<u>*Note*</u>: When $\phi = 90$ degs, $c_2 = 0$. $\therefore\ m_0 = m_1 = m_6 = m_{10} = 0$

$\Rightarrow$ (Singularity!) ψ, θ are undefined.

Euler Angle Interpolation

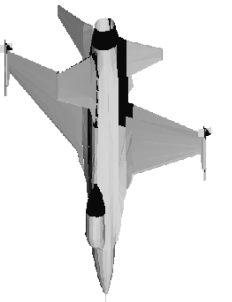
Can we find an intermediate orientation by interpolating between corresponding Euler angles?

R_1

$\{\psi_1, \phi_1, \theta_1\}$

R_t

$\{\psi_t, \phi_t, \theta_t\}$



R_2

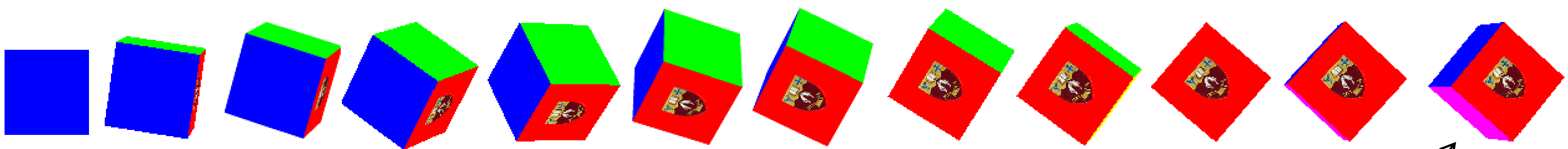
$\{\psi_2, \phi_2, \theta_2\}$

t

$$\boxed{\begin{aligned} \psi_t &= (1-t)\,\psi_1 + t\,\psi_2 \\ \phi_t &= (1-t)\,\phi_1 + t\,\phi_2 \\ \theta_t &= (1-t)\,\theta_1 + t\,\theta_2 \end{aligned}}$$

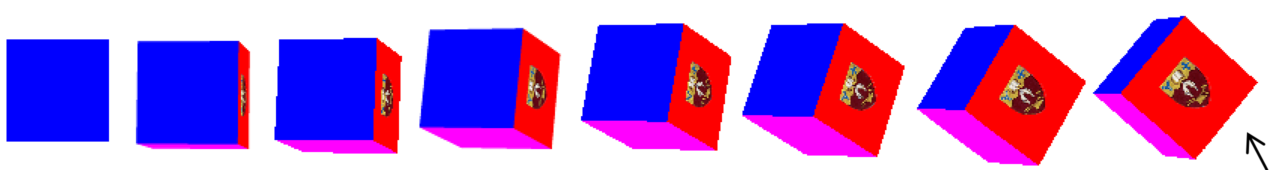
Euler Angle Interpolation

Euler angles are not unique! Different values of Euler angles can produce different interpolation sequences



{0, 0, 0}

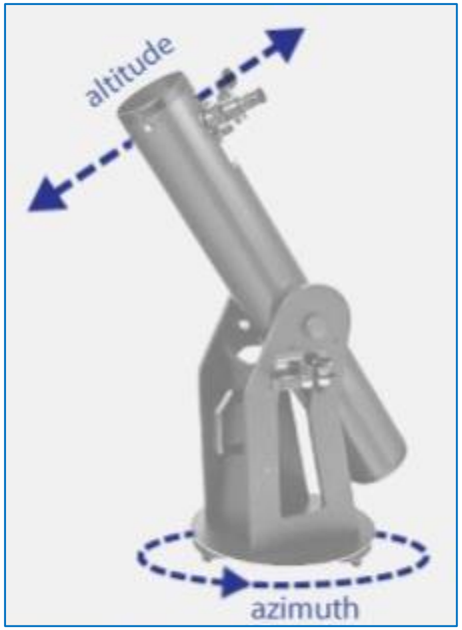
{135,-100,-180}

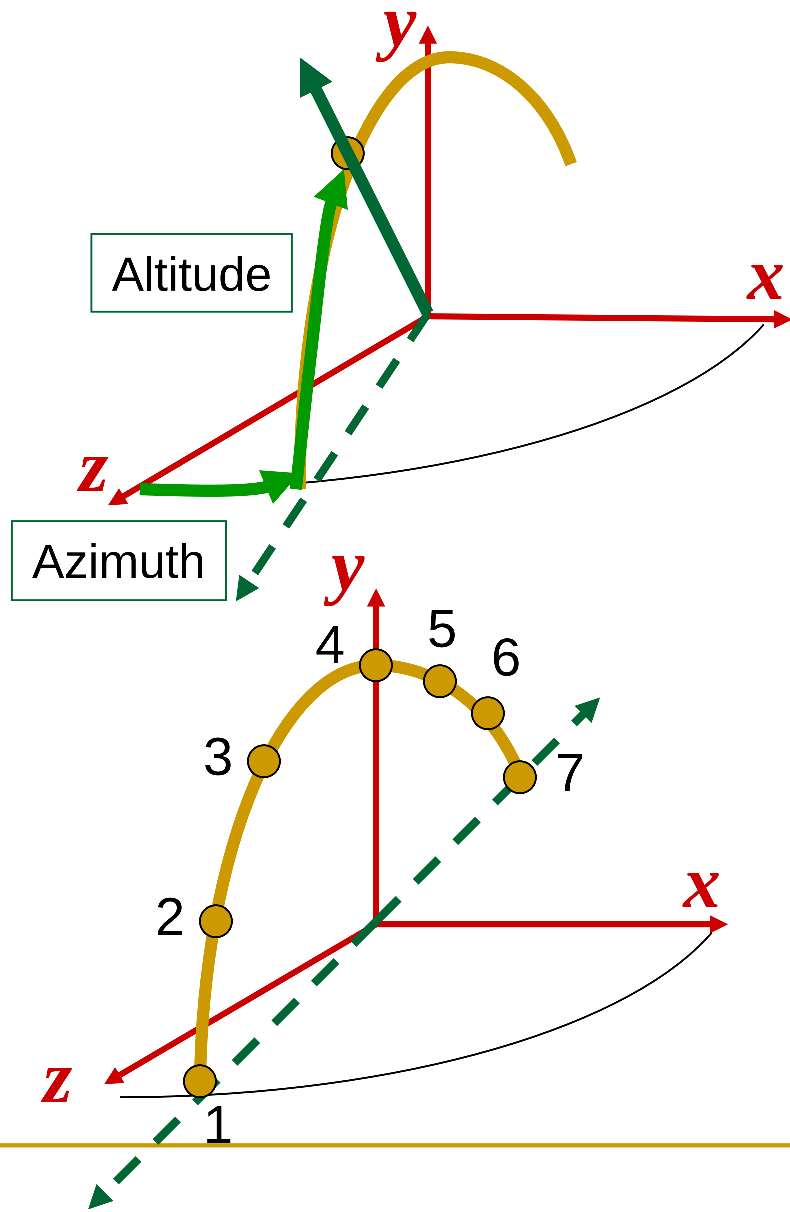


{0, 0, 0}

{-45,-80, 0}

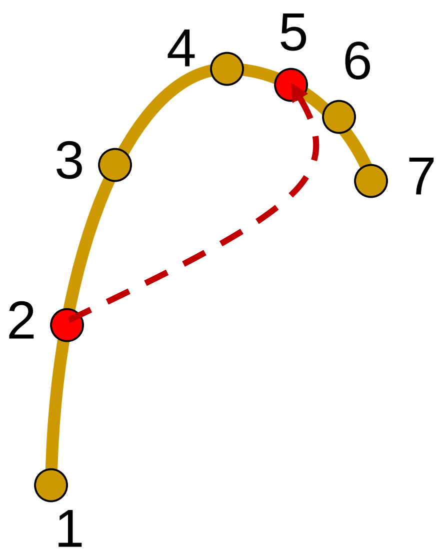
Gimbal Lock: The Tracking Problem





	Azim	Alt
1	15	0
2	15	30
3	15	60
4	15	90
	15	120
5	195	60
6	195	30
7	195	0

Gimbal Lock: The Interpolation Problem



	Azimuth	Altitude
1	15	0
2	15	30
3	15	60
4	15	90
5	195	60
6	195	30
7	195	0

Interpolation

15	30
51	36
87	42
123	48
159	54
195	60

Gimbal Lock in 3D Rotations

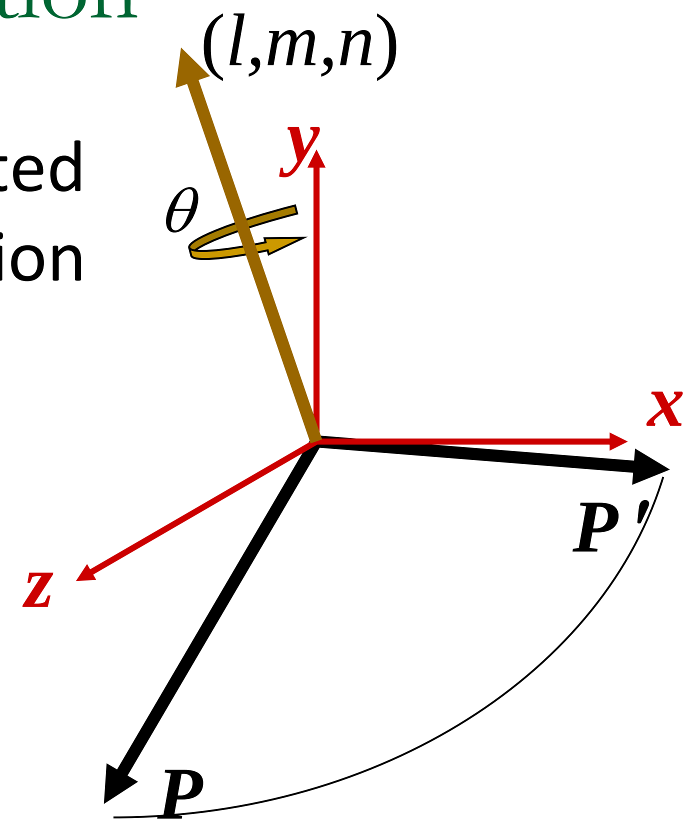
- Obviously, the "Gimbal Lock" in 3D rotational transformations of graphical objects cannot be associated with any physical mounting constraints.
- "Gimbal lock" refers to the possibility of representing the same three-dimensional orientation of an object using multiple sets of values of rotational parameters.
- The non-uniqueness in Euler angle representation is particularly seen when the middle angle is 90 degs.

$\{\psi, 90, \theta\} = \{\psi+\lambda, 90, \theta+\lambda\}$ for all λ

Angle-Axis Transformation

- Any 3D rotation can be represented in angle-axis form (Euler's rotation theorem)
- Parameters: θ, (l, m, n)

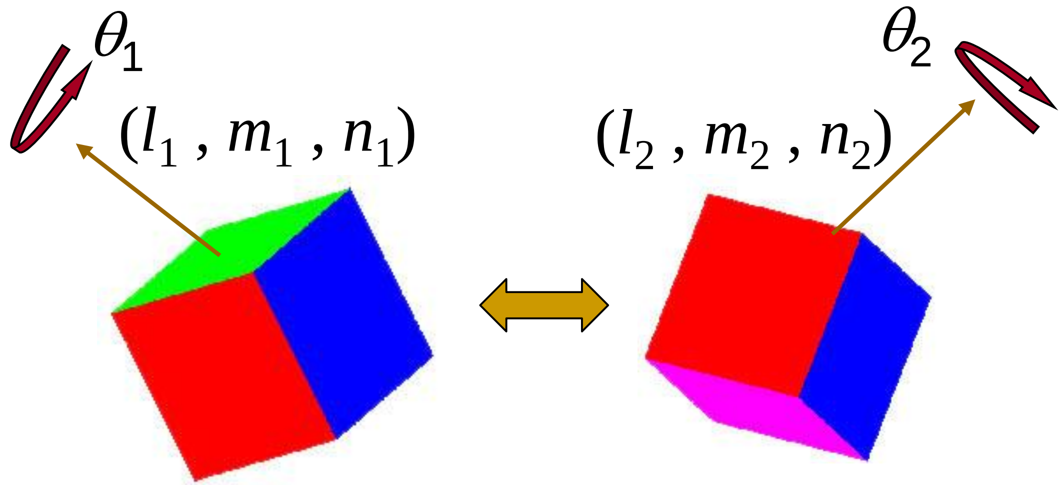
```
glRotatef(angle, l, m, n);
```



Rotation Matrix:

$$\begin{bmatrix} l^2(1-\cos\theta)+\cos\theta & lm(1-\cos\theta)-n\sin\theta & ln(1-\cos\theta)+m\sin\theta & 0 \\ lm(1-\cos\theta)+n\sin\theta & m^2(1-\cos\theta)+\cos\theta & mn(1-\cos\theta)-l\sin\theta & 0 \\ ln(1-\cos\theta)-m\sin\theta & mn(1-\cos\theta)+l\sin\theta & n^2(1-\cos\theta)+\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Angle-Axis Transformation



Can we interpolate between two angle-axis representations for key-frame animation?

- Angle-axis interpolation will not always yield a smooth rotation between two key frames.
- We need to consider the singularity when θ = 180 degs

Introduction to Quaternions

- Quaternions are hypercomplex numbers of rank 4 (usually denoted by **H**).
- Quaternions are powerful tools for representing arbitrary rotations in three-dimensional space, and provide several advantages over Euler angle representation.
- Quaternion interpolation can be used to compute intermediate frames between two rotations or camera orientations.

The Discoverer

William Rowan Hamilton (1805-1865)

- 1823: Entered Trinity College, Dublin.
- 1827: Appointed as Professor of Astronomy.
- 1835: Received the Royal Medal of the Royal Society of London.
- 1843: Invented Quaternions.

Plaque commemorating discovery of quaternions:

Brougham Bridge, Dublin.

Image source: Wikipedia

Ref: https://en.wikipedia.org/wiki/William_Rowan_Hamilton

Complex vs Hyper-Complex

$$(a, b) \rightarrow a + i\,b$$

a: Real

b: Imaginary

$$i = \sqrt{-1}$$

$$(q_0, q_1, q_2, q_3) \rightarrow \underbrace{q_0}_{\text{Scalar}} + \underbrace{i\,q_1 + j\,q_2 + k\,q_3}_{\text{Vector}}$$

$$i\,,\, j\,,\, k = ??$$

Confusion of Rank 4!

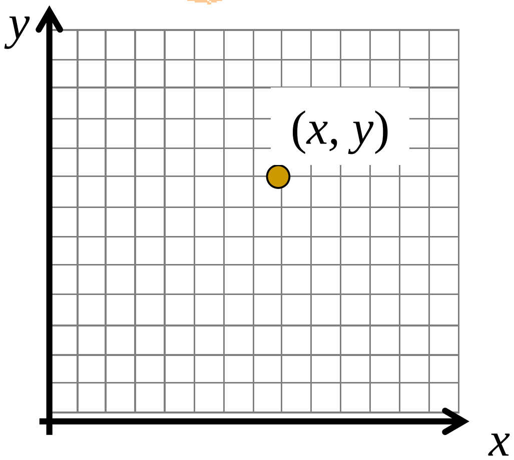
$$i^2 = j^2 = k^2 = ijk = -1$$

$$i \neq j \neq k$$

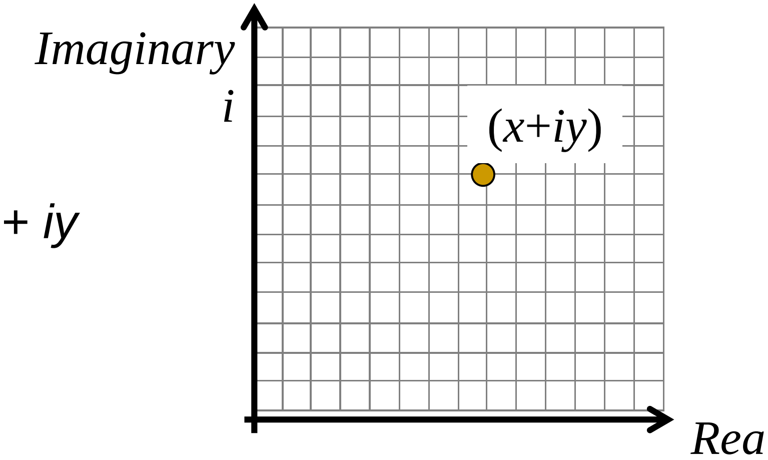
$$ij = k, \quad jk = i, \quad ki = j$$

$$ji = -k, \quad kj = -i, \quad ik = -j$$

Review of Complex Numbers



$(x, y) \Leftrightarrow x + iy$



Complex addition (Translation in 2D space):

$(x_1 + i\, y_1) + (x_2 + i\, y_2) = (x_1 + x_2) + i\,(y_1 + y_2)$

Scalar multiplication (Scaling in 2D space) by a constant *c*:

$c\,(x + i\, y) = cx + i\, cy$

Complex Multiplication

$(x_1 + i\, y_1)\,(x_2 + i\, y_2) = (x_1 x_2 - y_1 y_2) + i\,(x_1 y_2 + x_2 y_1)$

Review of Complex Numbers

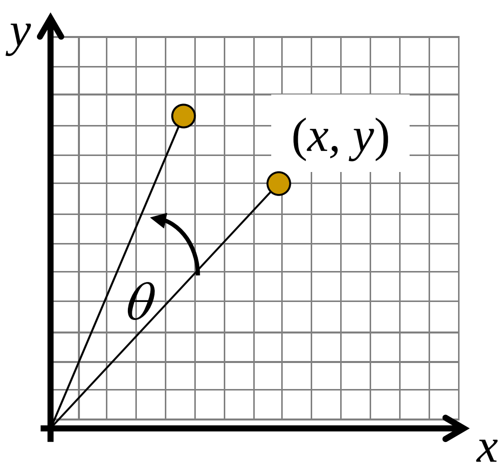
Using the multiplication formula on the previous slide,

$(\cos\theta + i\sin\theta)\,(x + i\,y) = (\cos\theta\, x - \sin\theta\, y) + i(\sin\theta\, x + \cos\theta\, y)$

In 2D domain, this operation corresponds to:

$(\cos\theta,\ \sin\theta)\,(x, y) \rightarrow (\cos\theta\, x - \sin\theta\, y,\ \ \sin\theta\, x + \cos\theta\, y)$

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}$$



The complex number $(\cos\theta + \mathrm{i}\sin\theta)$ represents a 2D rotation operator

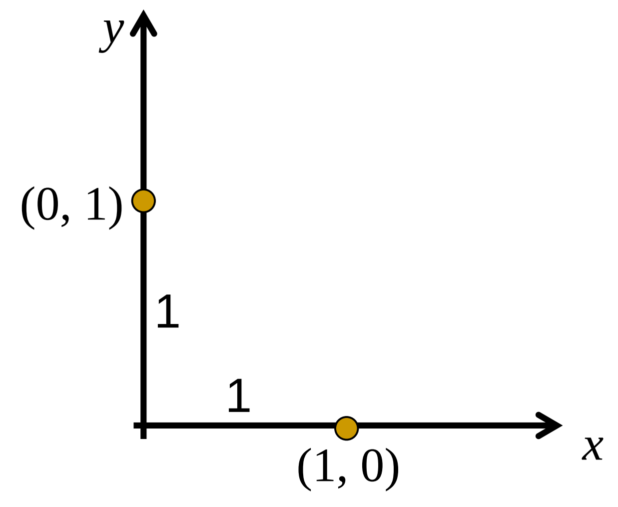
Complex Number i

Complex Multiplication Rule (Slide 24):

$$(x_1,\ y_1)\,(x_2,\ y_2) = (x_1x_2 - y_1y_2\,,\ \ x_1y_2 + x_2y_1)$$

$$i\ =\ (0,\ 1)$$

$$i^2 = (0, 1)\,(0, 1) = (-1,\ 0)$$



<u>Side note:</u>

Dual number algebra:

$a + \varepsilon\, b$ where, $\varepsilon = \sqrt{0} \neq 0$

Quaternion Basis

$$\boldsymbol{i} = (0, 1, 0, 0)$$

$$\boldsymbol{j} = (0, 0, 1, 0)$$

$$\boldsymbol{k} = (0, 0, 0, 1)$$

$$\mathbf{1} = (1, 0, 0, 0)$$

Quaternion Operations

Let $P = (p_0, p_1, p_2, p_3)$, $Q = (q_0, q_1, q_2, q_3)$. Then,

- $P+Q = (p_0+q_0, p_1+q_1, p_2+q_2, p_3+q_3)$
- $P-Q = (p_0-q_0, p_1-q_1, p_2-q_2, p_3-q_3)$
- $Q^* = (q_0, -q_1, -q_2, -q_3)$.
- $|Q| = \mathrm{sqrt}(q_0^2 + q_1^2 + q_2^2 + q_3^2)$.
- Q is a unit quaternion $\Rightarrow q_0^2+q_1^2+q_2^2+q_3^2 = 1$.
- Q is a unit quaternion $\Rightarrow QQ^* = 1$.

Quaternion Product

- $$PQ = (\, p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3\,,$$
  $$p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2\,,$$
  $$p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1\,,$$
  $$p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0\,)$$
  $$PQ \neq QP.$$

- Scalar-vector representation of quaternions:
  - $P = (p_0, \boldsymbol{v})$, where $\boldsymbol{v} = (p_1, p_2, p_3)$
  - $Q = (q_0, \boldsymbol{w})$, where $\boldsymbol{w} = (q_1, q_2, q_3)$

$$PQ = (\, p_0q_0 - \boldsymbol{v}.\boldsymbol{w},\ \ p_0\boldsymbol{w} + q_0\boldsymbol{v} + \boldsymbol{v}\times\boldsymbol{w})$$

Exercise: Prove the statements on slide 23.

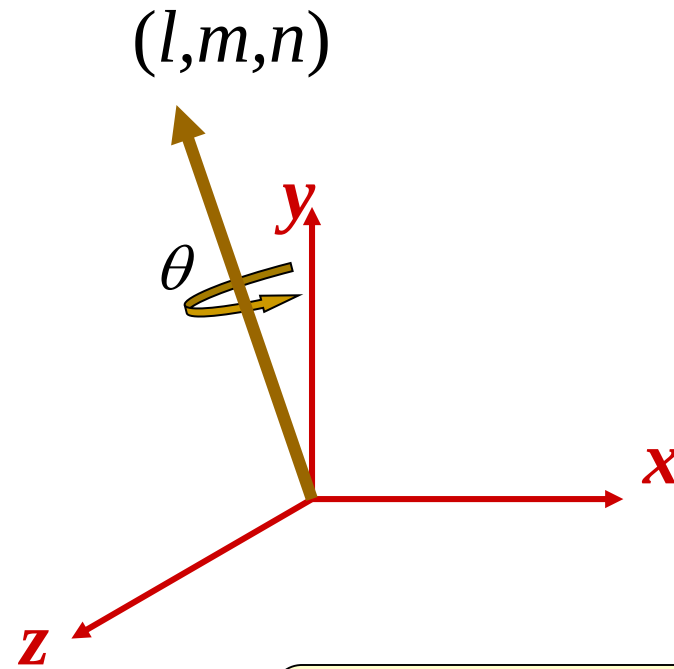
Quaternion Representations

- A real number r: $(r, 0, 0, 0)$.
- A complex number (a, b): $(a, b, 0, 0)$
- A 3D vector (a, b, c): $(0, a, b, c)$
- A 3D point (x, y, z): $(0, x, y, z)$
- A unit quaternion representation a rotation in three-dimensional space. Given a point $P = (x, y, z)$, the transformed point P' can be obtained as

$$P' = QPQ^*$$

Quaternions

An angle-axis rotation about a **unit** vector (*l,m,n*), and the angle of rotation θ can be represented by a <u>unit</u> quaternion:



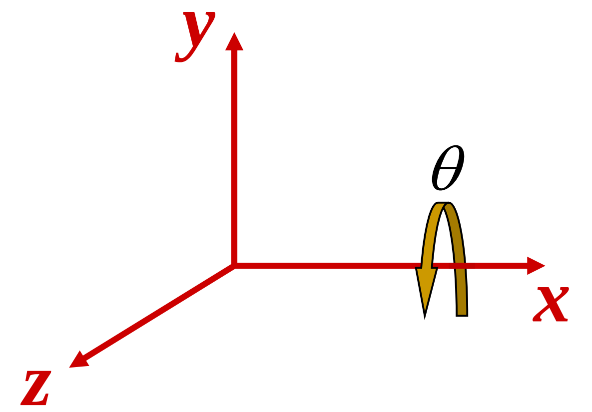
A very important formula!

$$Q = \left(\cos\frac{\theta}{2},\ l\sin\frac{\theta}{2},\ m\sin\frac{\theta}{2},\ n\sin\frac{\theta}{2} \right)$$

Note: Q is a unit vector.

Quaternions (Examples)

A rotation θ about the *x*-axis:



$$Q = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}, 0, 0\right)$$

A 90 deg rotation about the z-axis:

$Q = (0.707,\ 0,\ 0,\ 0.707)$

A 60 deg rotation about the y-axis:

$Q = (0.866,\ 0,\ 0.5,\ 0)$

Quaternions and Rotations

- Every unit quaternion with $q_0 \neq 1$ represents a rotation in 3D space.
- If $Q = (q_0, q_1, q_2, q_3)$ is a unit quaternion, then

$$q_0 = \cos\frac{\theta}{2}, \qquad \sqrt{q_1^2 + q_2^2 + q_3^2} = \sin\frac{\theta}{2}$$

The angle and axis of rotation can be obtained from the above equations:

$$\theta = 2\tan^{-1}\left(\frac{\sqrt{q_1^2 + q_2^2 + q_3^2}}{q_0}\right)$$

$$l = \frac{q_1}{\sqrt{q_1^2 + q_2^2 + q_3^2}}, \quad m = \frac{q_2}{\sqrt{q_1^2 + q_2^2 + q_3^2}}, \quad n = \frac{q_3}{\sqrt{q_1^2 + q_2^2 + q_3^2}},$$

Rotation Matrix from Quaternions

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) & 0 \\ 2(q_1q_2+q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) & 0 \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

For a unit quaternion, we can rewrite the diagonal terms in the above equation as shown below:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1-2q_2^2-2q_3^2 & 2q_1q_2-2q_0q_3 & 2q_1q_3+2q_0q_2 & 0 \\ 2q_1q_2+2q_0q_3 & 1-2q_1^2-2q_3^2 & 2q_2q_3-2q_0q_1 & 0 \\ 2q_1q_3-2q_0q_2 & 2q_2q_3+2q_0q_1 & 1-2q_1^2-2q_2^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Quaternions from Rotation Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} m_0 & m_4 & m_8 & 0 \\ m_1 & m_5 & m_9 & 0 \\ m_2 & m_6 & m_{10} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$q_0 = \frac{\sqrt{1 + m_0 + m_5 + m_{10}}}{2}$$

$$q_3 = \frac{m_1 - m_4}{4q_0}$$

$$q_1 = \frac{m_6 - m_9}{4q_0}$$

$$q_2 = \frac{m_8 - m_2}{4q_0}$$

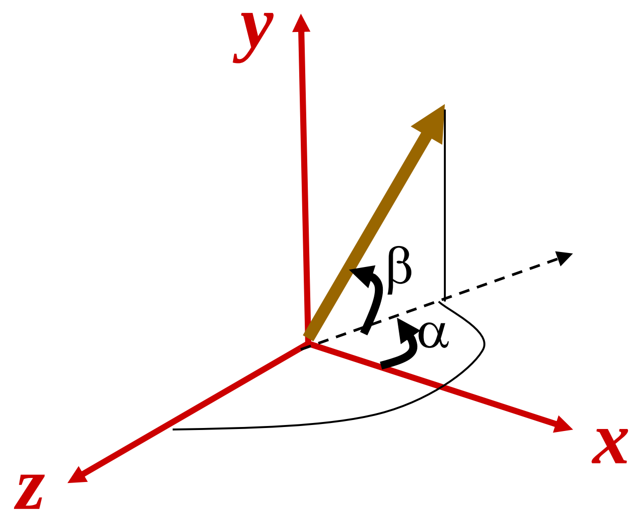
Singularity-free extraction of quaternions!

If $q_0 = 0$, the matrix corresponds to a rotation of 180 degs
$(q_0, q_1, q_2, q_3) = (0, l, m, n)$.

Rotation Interpolation: Example

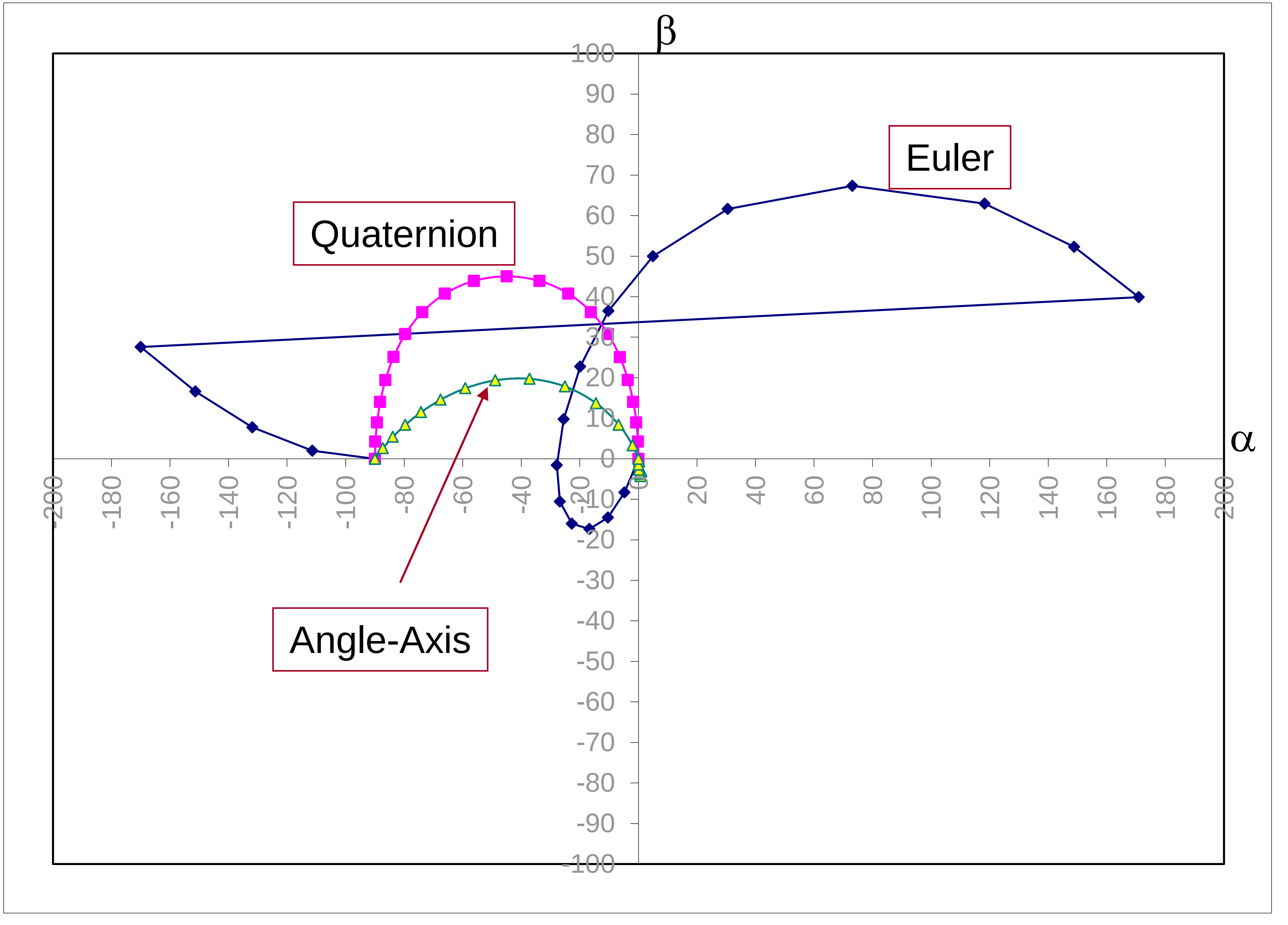
Demo program:

Orientation defined using azimuth and elevation angles: α, β.



Initial orientation: –90, 0

Final orientation: 0, 0



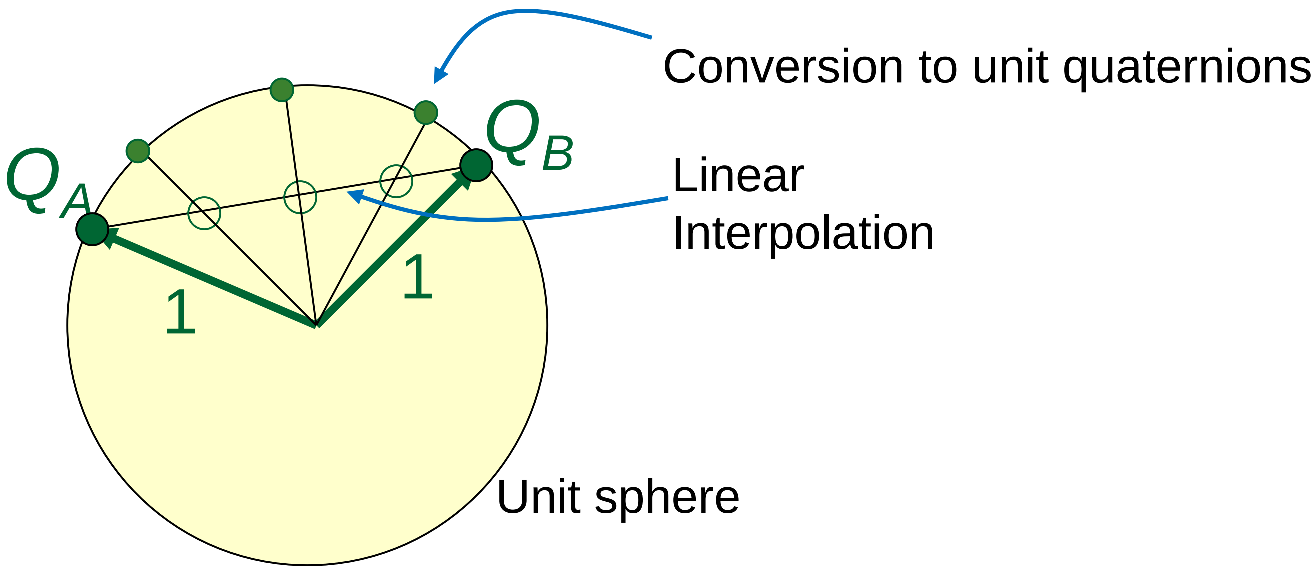
β
α
Euler
Quaternion
Angle-Axis

Quaternion Interpolation (Linear)

- Represent the orientations by unit quaternions Q_A , Q_B.
- Use linear (element-wise) interpolation LERP(Q_A, Q_B, t):

$$\boldsymbol{Q = (1-t)\, Q_A + t\, Q_B}, \qquad 0 \le t \le 1.$$

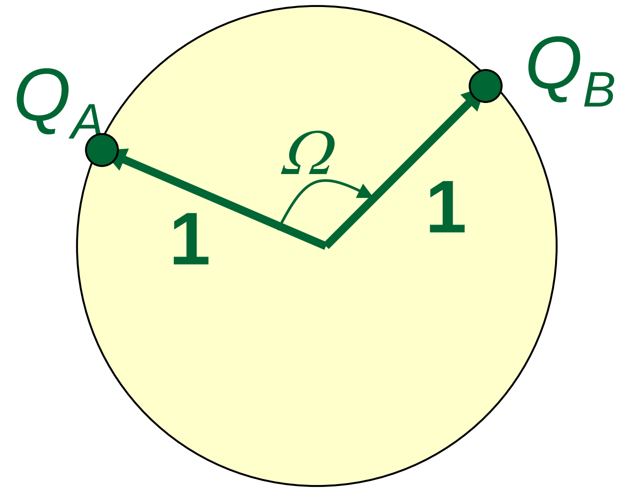
- Normalize Q to get a unit quaternion.
- A constant step size will not produce a uniform rotation sequence.



Quaternion Interpolation (SLERP)

Spherical Linear Interpolation

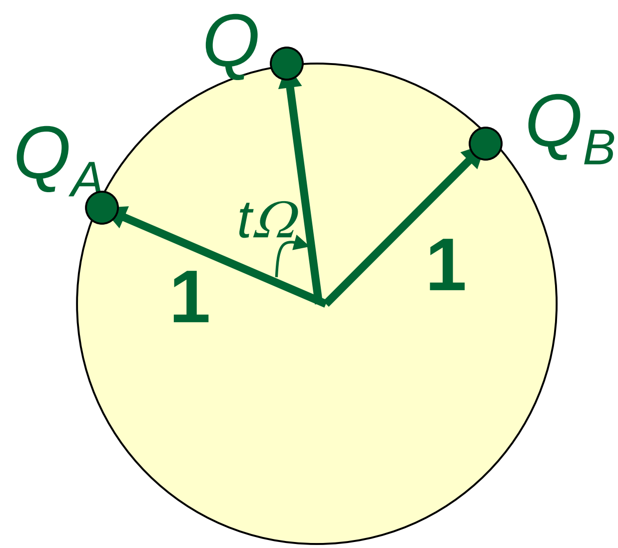
- Compute the angle Ω between Q_A , Q_B : $\cos \Omega = Q_A \bullet Q_B$



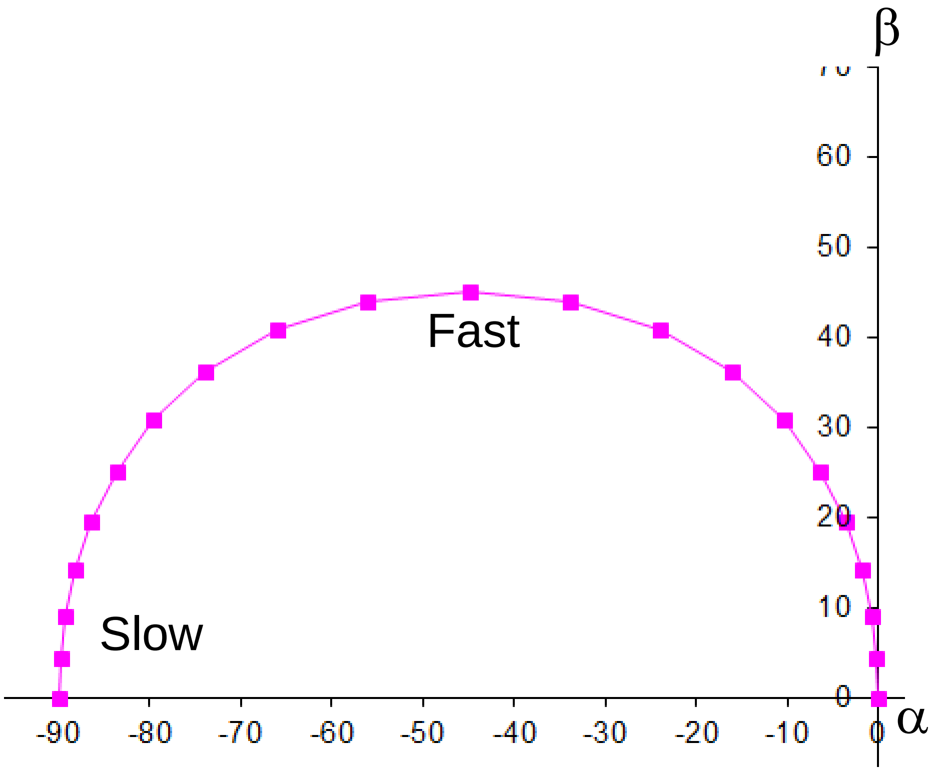
- The spherical linear interpolation formula SLERP(Q_A, Q_B, Ω, t) is

$$Q = Q_A \frac{\sin(1-t)\Omega}{\sin \Omega} + Q_B \frac{\sin t\Omega}{\sin \Omega}$$

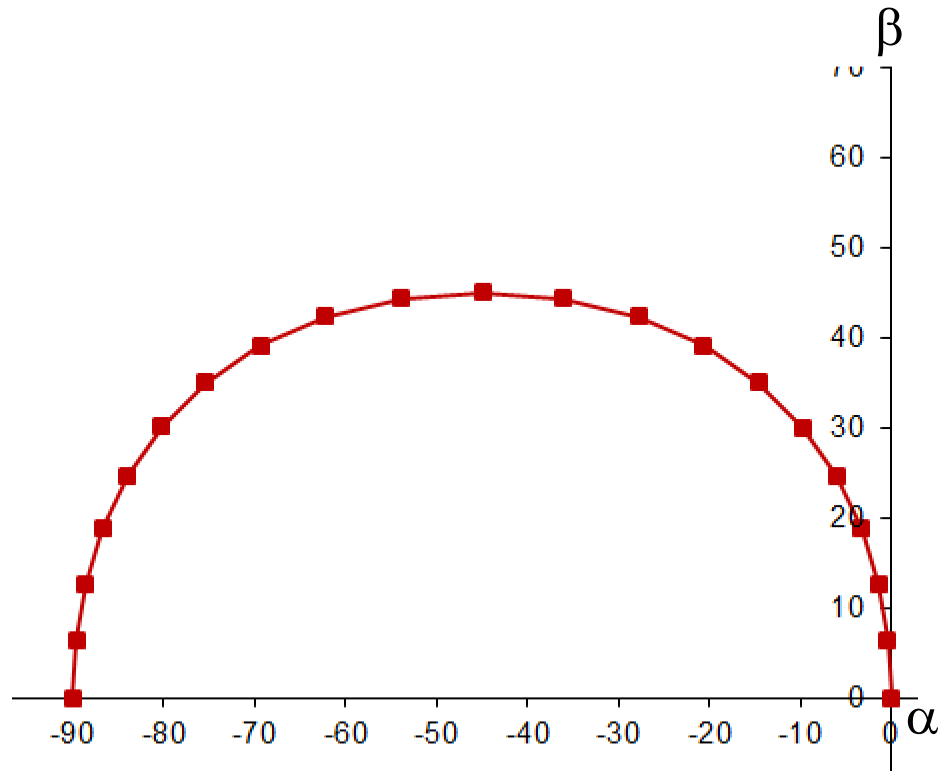
$0 \leq t \leq 1.$



Quaternion Linear Interpolation (LERP)



Quaternion Spherical Linear Interpolation (SLERP)



Q , –Q Represent Same Rotation!

Let $Q = \left(\cos\frac{\theta}{2},\ \ l\sin\frac{\theta}{2},\ m\sin\frac{\theta}{2},\ n\sin\frac{\theta}{2}\right)$

Then, $-Q = \left(-\cos\frac{\theta}{2}, -l\sin\frac{\theta}{2}, -m\sin\frac{\theta}{2}, -n\sin\frac{\theta}{2}\right)$

$$\boxed{\begin{aligned} cos(180+\Theta) &= -cos\Theta \\ sin(180+\Theta) &= -sin\Theta \end{aligned}}$$

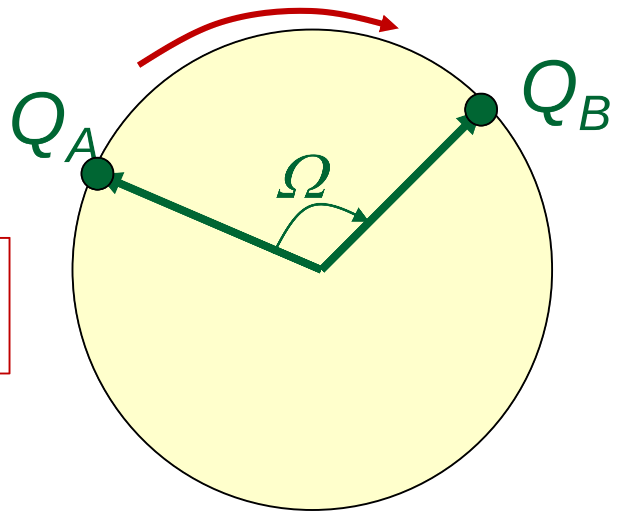
$$= \left(\cos 180 + \frac{\theta}{2},\ \ l\sin 180 + \frac{\theta}{2},\ m\sin 180 + \frac{\theta}{2},\ n\sin 180 + \frac{\theta}{2}\right)$$

$$= \left(\cos\frac{360+\theta}{2},\qquad l\sin\frac{360+\theta}{2},\ m\sin\frac{360+\theta}{2},\ n\sin\frac{360+\theta}{2}\right)$$

= a rotation about the vector (*l*,*m*,*n*) by an angle 360+θ degs.

Optimizing SLERP

Angle optimal rotation:



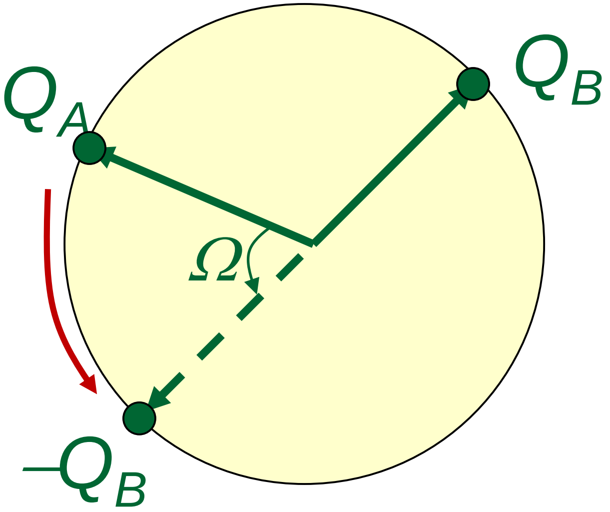
Non-optimal interpolation
(Ω > 90 degs)

If Ω > 90 degs:

$Q_B = -Q_B$

$\Omega = 180 - \Omega$

$Q = \text{SLERP}(Q_A, Q_B, \Omega, t)$



Quaternions vs. Euler Angles

	Quaternions	**Euler Angles**
Components	Algebraic expressions	Trigonometric functions
Composition of Rotation	16 Mults, 12 Additions	27 Mults 18 Additions
Inverse	Sign flip	Matrix transpose
Uniqueness	Q and $-Q$ represent the same orientation	Multiple representations of the same orientation
Singularities	No	Yes