

Rotations and quaternions

Jiří Chmelík, Marek Trtík

PA199

Outline

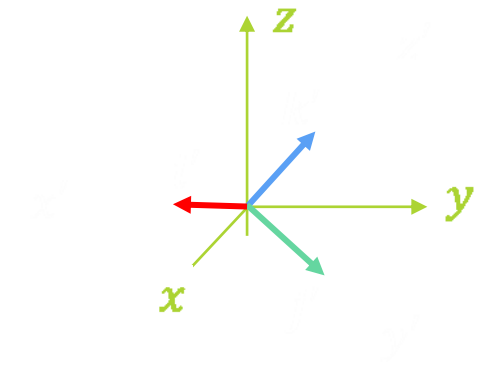
- ▶ Rotation matrix
- ▶ Euler angles
- ▶ Tait-Bryan angles
- ▶ Axis-angle representation
- ▶ Quaternions
 - ▶ Rotations via quaternions
 - ▶ Quaternion derivative

Rotation matrix

- ▶ Well know topic from computer graphics courses.
 - We only discuss relation between basis vectors and rotation matrix.
- ▶ Let i, j, k be orthonormal basis vectors of a coordinate system inside the world coordinate system.
- ▶ Then, the orientation of the coordinate system is represented by the rotation matrix:

$$R = \begin{pmatrix} R_x & R_y & R_z \\ I_x & I_y & I_z \\ K_x & K_y & K_z \end{pmatrix}$$

- ▶ R transforms vectors "to world space".
- ▶ $R^{-1} = R^T$ transforms vectors "from world space".



Euler angles

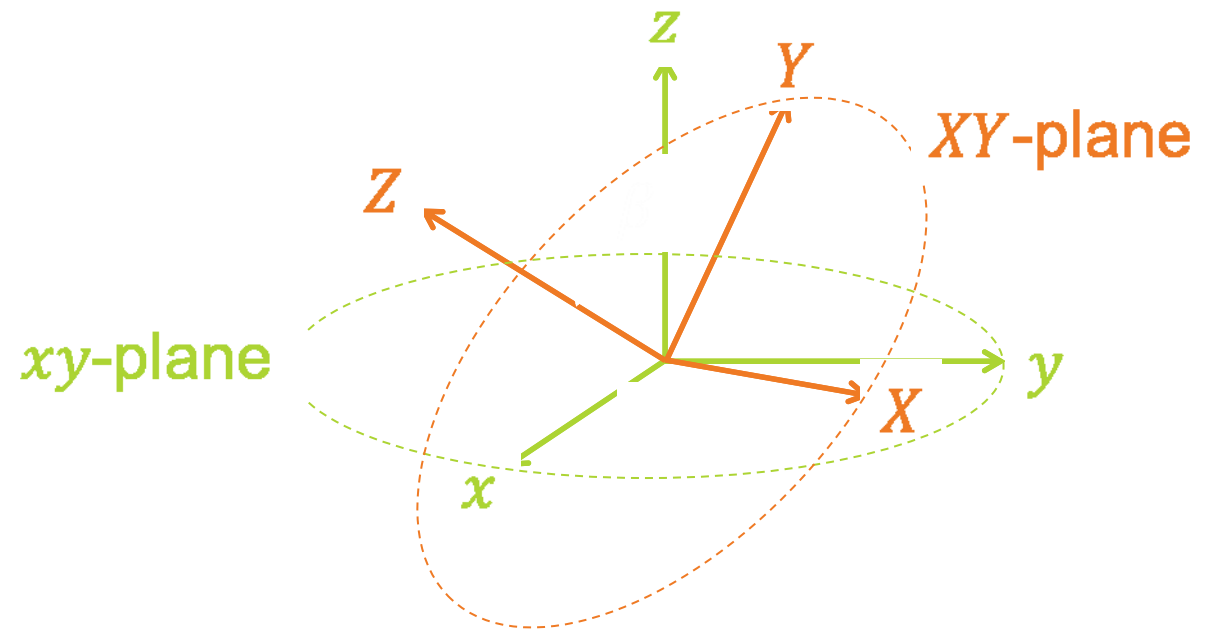
▶ Three rotations are always sufficient to transform a source frame xyz to a target one XYZ

▶ $\alpha \in (0, 2\pi)$

▶ $\beta \in (0, \pi)$

▶ $\gamma \in (0, 2\pi)$

▶ Actual rotations:



▶ α = rotation around z axis

▶ β = rotation around X axis

▶ γ = rotation around Y axis

Euler angles

▶ Three rotations are always sufficient to transform a source frame xyz to a target one XYZ

▶ $\alpha \in (0, 2\pi)$

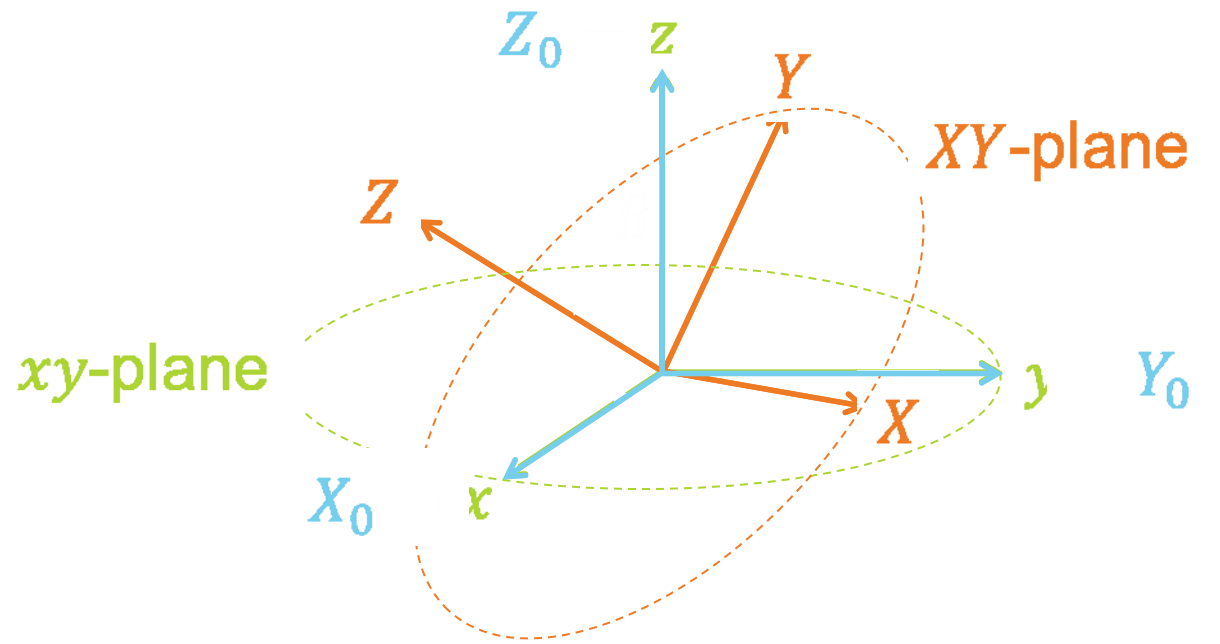
▶ $\beta \in (0, \pi)$

▶ $\gamma \in (0, 2\pi)$

▶ Actual rotations:

▶ Start with frame $X_0Y_0Z_0 = xyz$

▶ Rotate $X_0Y_0Z_0$ about Z_0 by α



▶ α (green) β (red) γ (red)

▶ α (green) β (red) γ (red)

▶ α (green) β (red) γ (red)

Euler angles

▶ Three rotations are always sufficient to transform a source frame xyz to a target one XYZ

▶ $\alpha \in (0, 2\pi)$

▶ $\beta \in (0, \pi)$

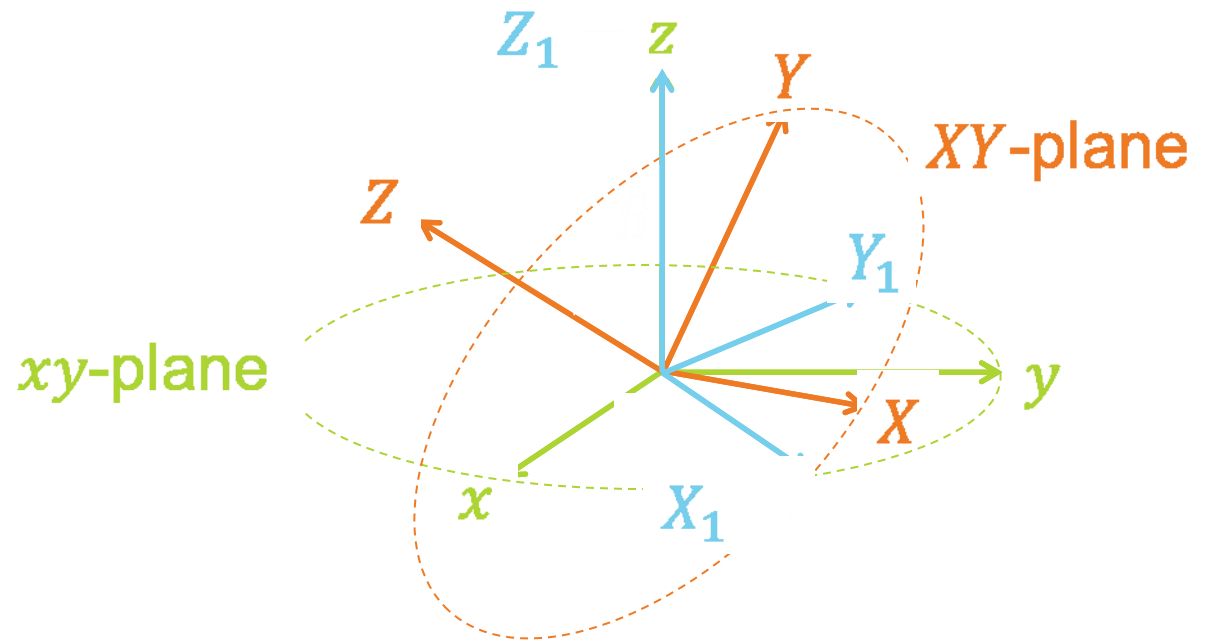
▶ $\gamma \in (0, 2\pi)$

▶ Actual rotations:

▶ Start with frame $X_0Y_0Z_0 = xyz$

▶ Rotate $X_0Y_0Z_0$ about Z_0 by α

▶ Rotate $X_1Y_1Z_1$ about X_1 by β



▶ xy XY

▶ xy XY

▶ xy XY

Euler angles

▶ Three rotations are always sufficient to transform a source frame xyz to a target one XYZ

▶ $\alpha \in (0, 2\pi)$

▶ $\beta \in (0, \pi)$

▶ $\gamma \in (0, 2\pi)$

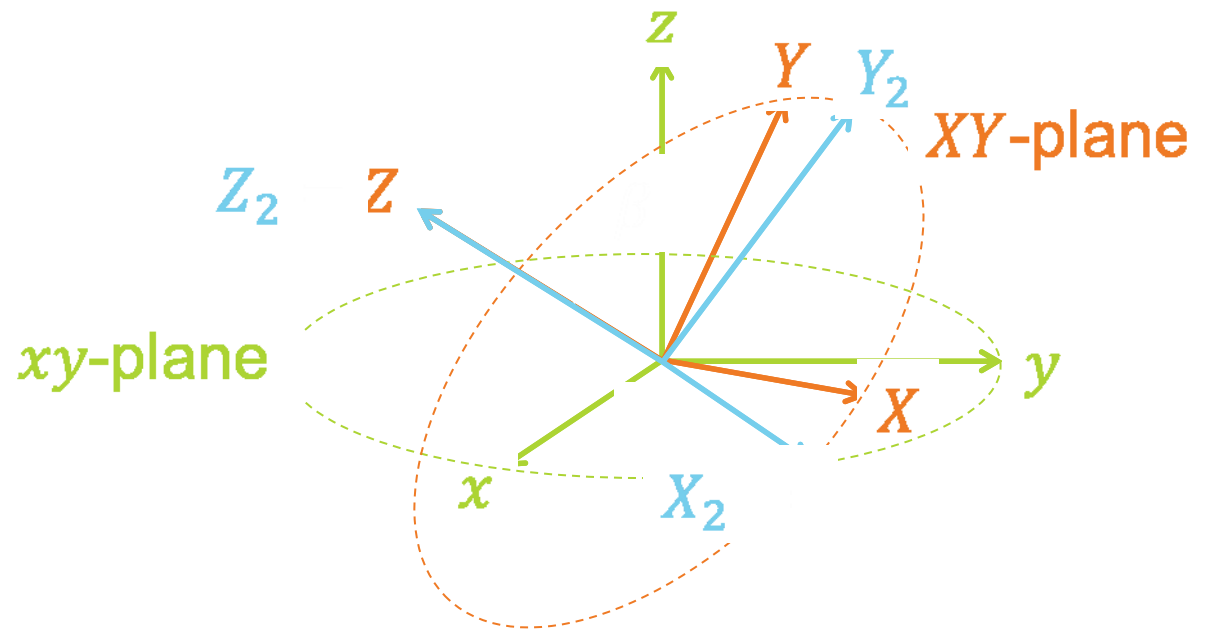
▶ Actual rotations:

▶ Start with frame $X_0Y_0Z_0 = xyz$

▶ Rotate $X_0Y_0Z_0$ about Z_0 by α

▶ Rotate $X_1Y_1Z_1$ about X_1 by β

▶ Rotate $X_2Y_2Z_2$ about Z_2 by γ



▶ xyz XY

▶ $X_1Y_1Z_1$

▶ $X_2Y_2Z_2$

Euler angles

▶ Three rotations are always sufficient to transform a source frame xyz to a target one XYZ

▶ $\alpha \in (0, 2\pi)$

▶ $\beta \in (0, \pi)$

▶ $\gamma \in (0, 2\pi)$

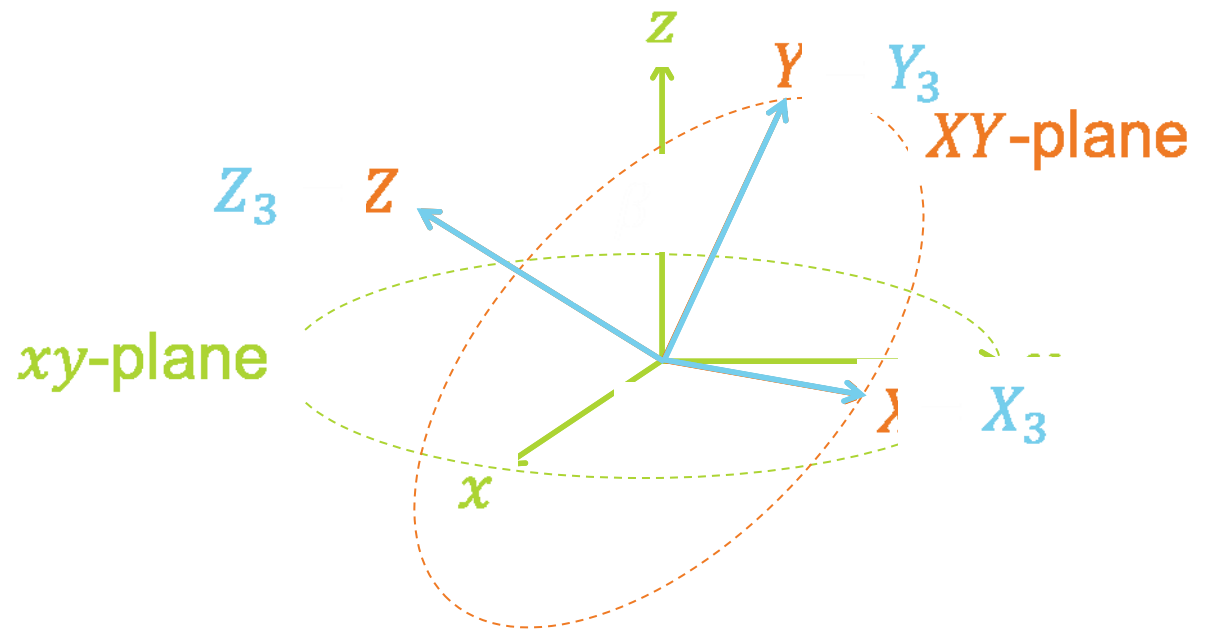
▶ Actual rotations:

▶ Start with frame $X_0Y_0Z_0 = xyz$

▶ Rotate $X_0Y_0Z_0$ about Z_0 by α

▶ Rotate $X_1Y_1Z_1$ about X_1 by β

▶ Rotate $X_2Y_2Z_2$ about Z_2 by γ



▶ xyz XY

▶ $X_1Y_1Z_1$

▶ $X_2Y_2Z_2$

Euler angles

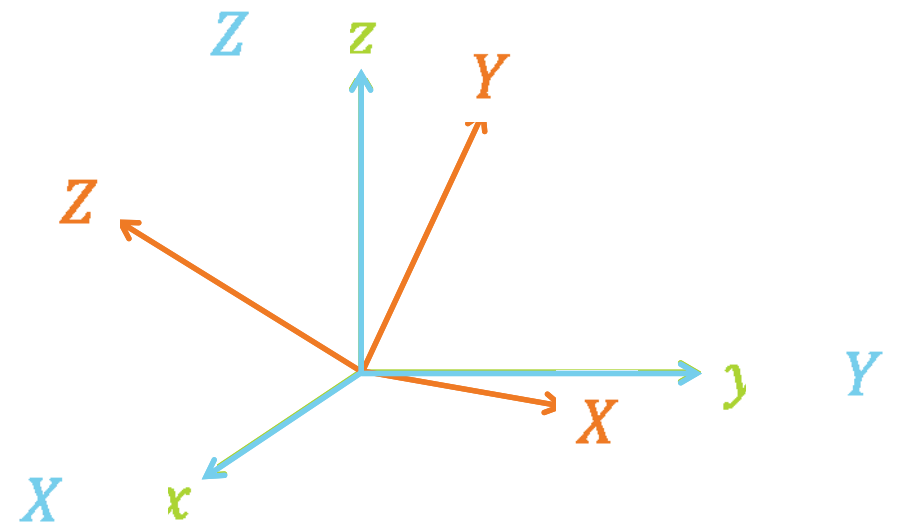
- ▶ Let $R(\varphi, a)$ denotes a rotation matrix about an axis a by an angle φ .
- ▶ So, our rotations can be expressed by matrices:
 - ▶ $R(\alpha, Z_0)$
 - ▶ $R(\beta, X_1)$
 - ▶ $R(\gamma, Z_2)$
- ▶ We compose them by the matrix multiplication:
$$R(\gamma, Z_2) R(\beta, X_1) R(\alpha, Z_0)$$
- ▶ Here we work with **Z X Z** convention. But there are 5 more:
 - ▶ **X Y X X Z X Y X Y Y Z Y** and **Z Y Z**
 - ▶ We can choose any of the conventions we want.
 - ▶ Observation: 1st and 3rd rotation axes are the same.

Euler angles

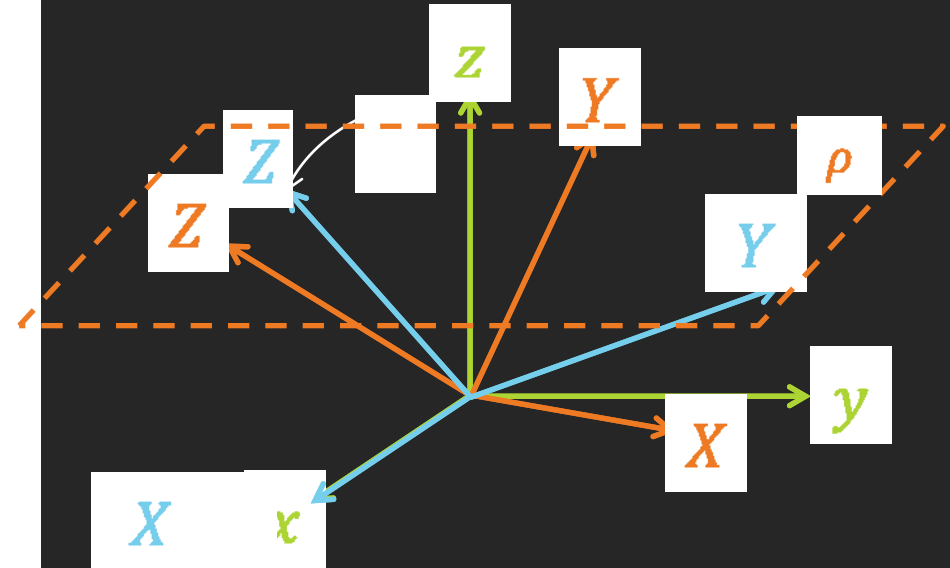
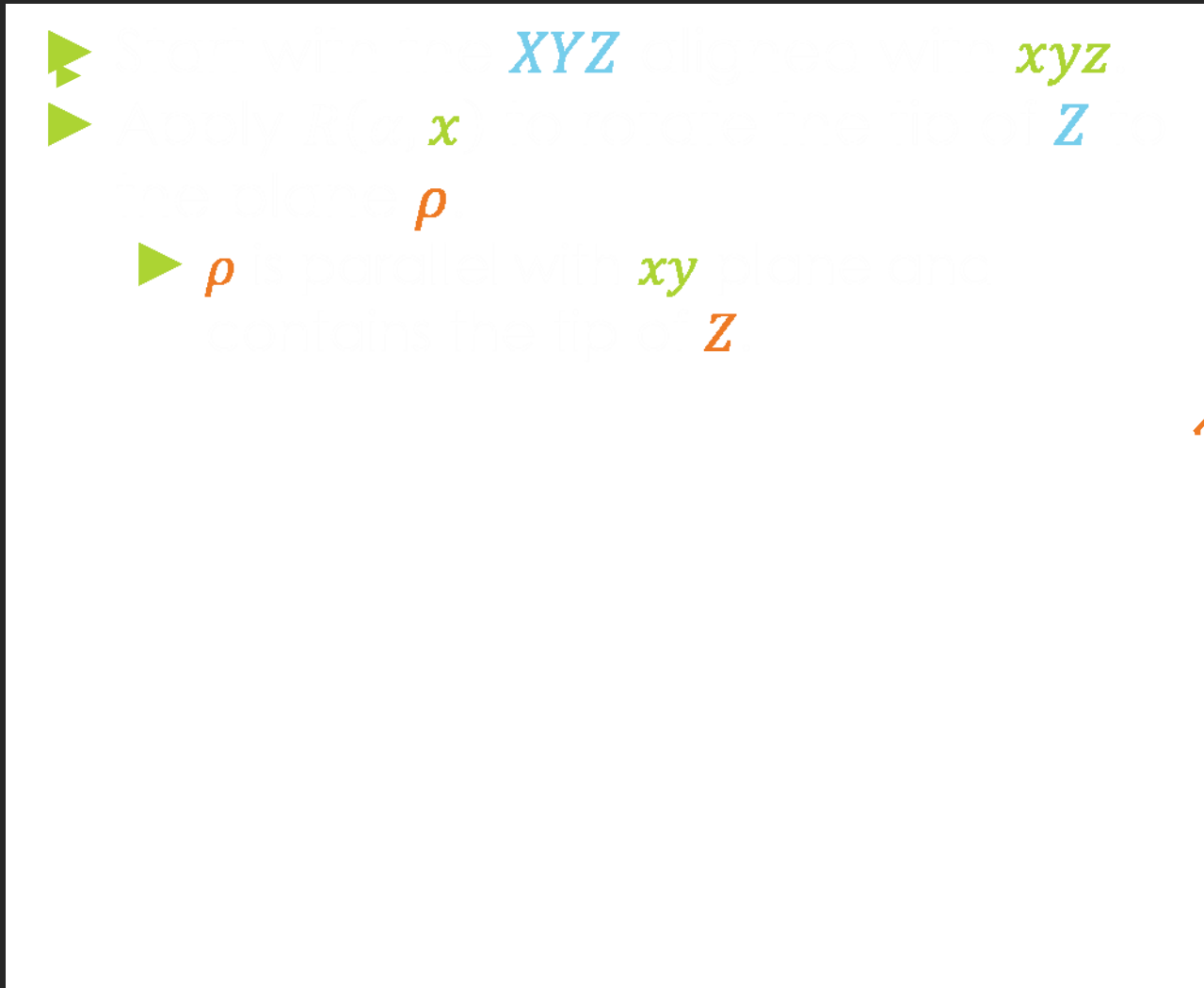
- ▶ The rotations $R(\alpha, Z_0)$, $R(\beta, X_1)$, $R(\gamma, Z_2)$ about the axes of the related (target) frame XYZ are called intrinsic.
- ▶ A practical disadvantage of intrinsic rotations is that some of rotations are about arbitrary oriented axis.
 - ▶ In CG courses we only learned how to build rotation matrices for fixed axes x, y, z .
 - ▶ But axes X_1, Z_2 may be arbitrary (the axis Z_0 is OK, since $Z_0 = z$).
- ▶ Fortunately, we can also transform a source frame xyz to a target one XYZ using extrinsic rotations $R(\alpha, z)$, $R(\beta, x)$, $R(\gamma, z)$.
 - ▶ Let us figure out how to do that.

Euler angles

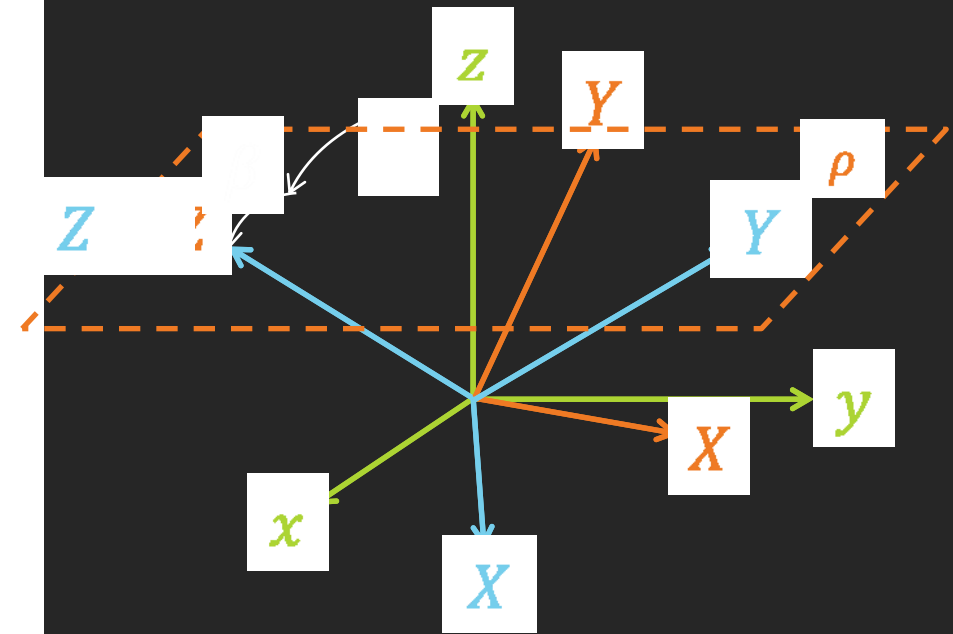
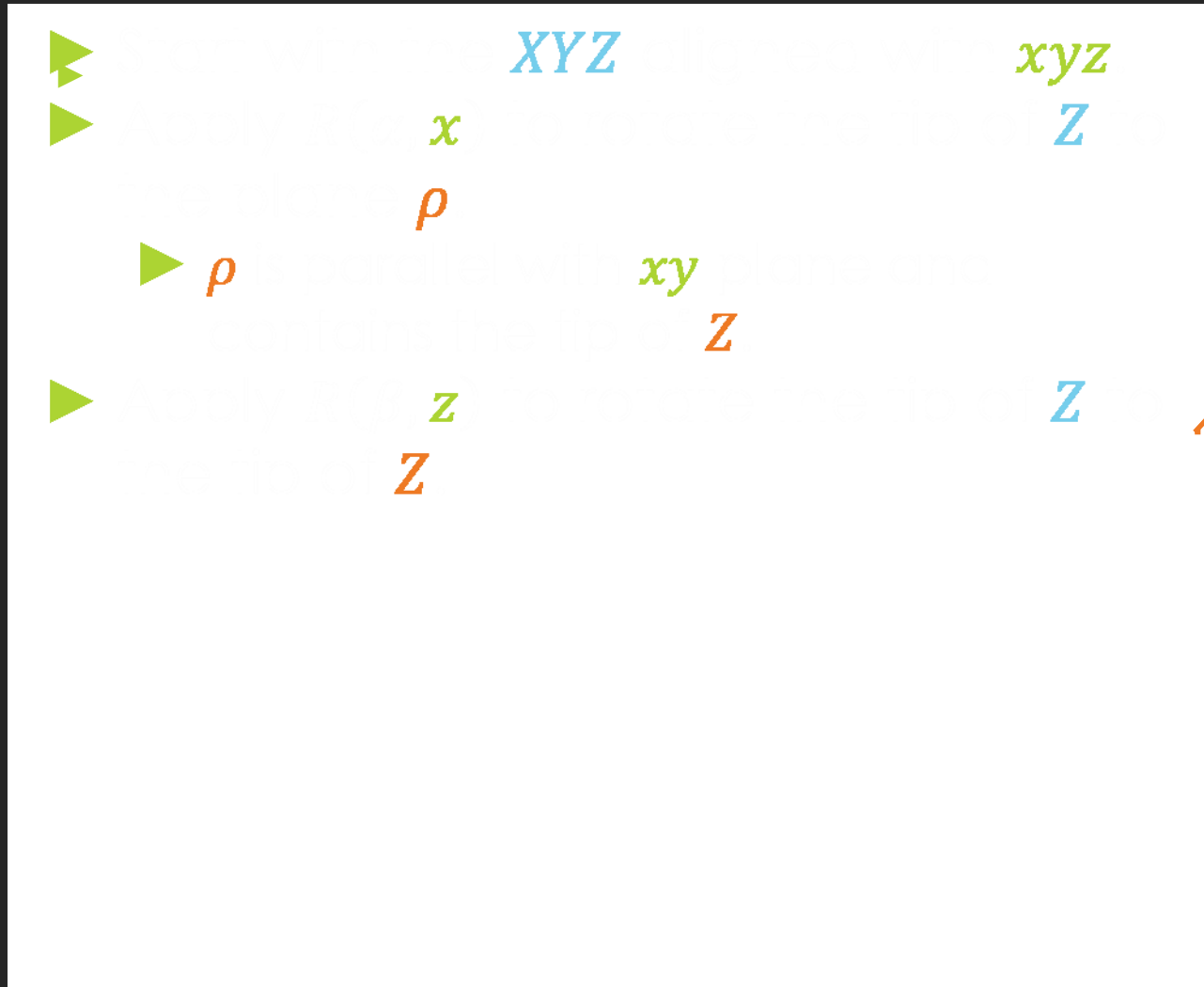
▶ Start with the XYZ aligned with xyz



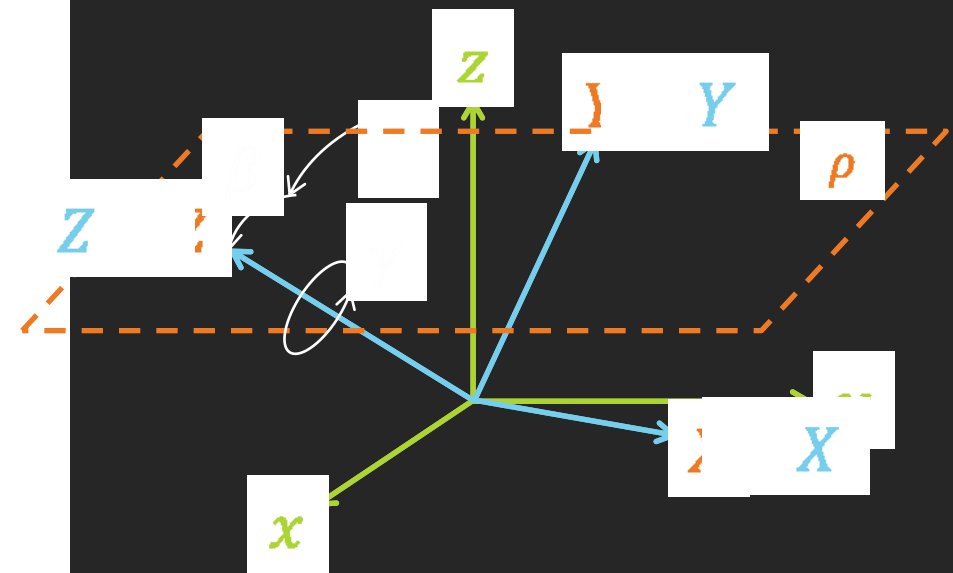
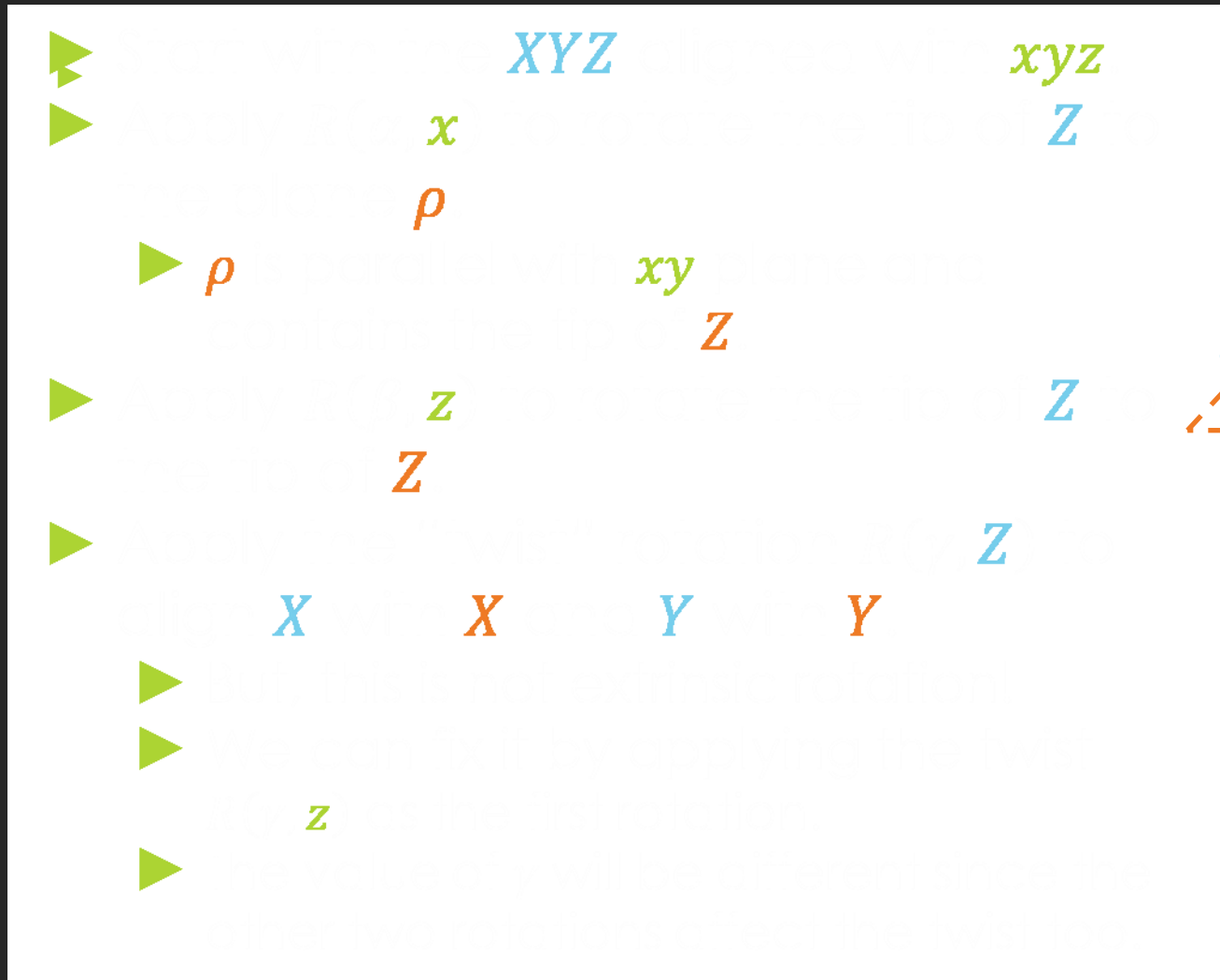
Euler angles



Euler angles



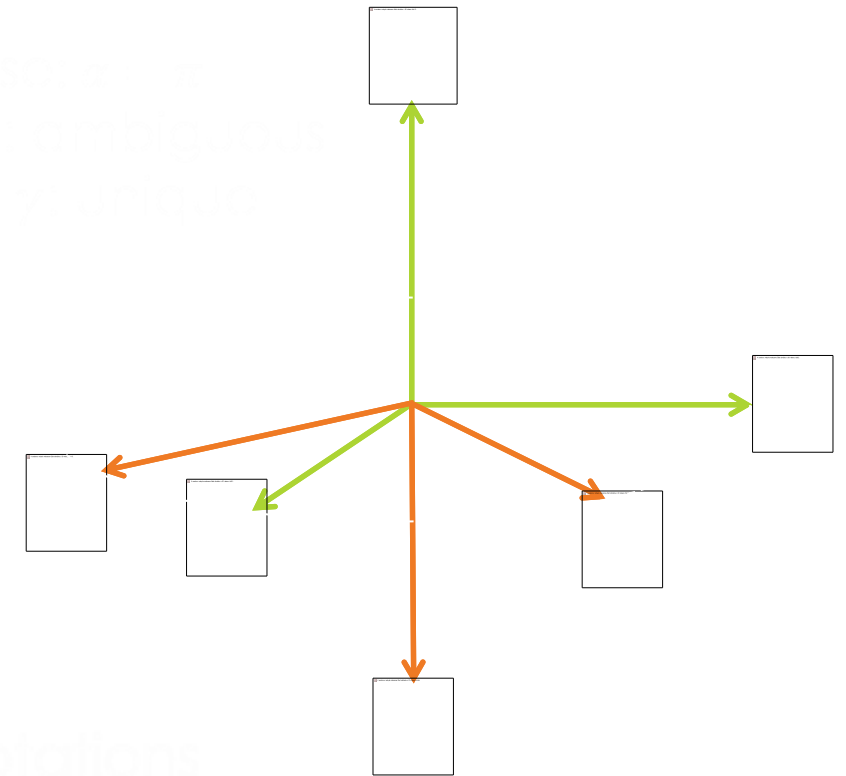
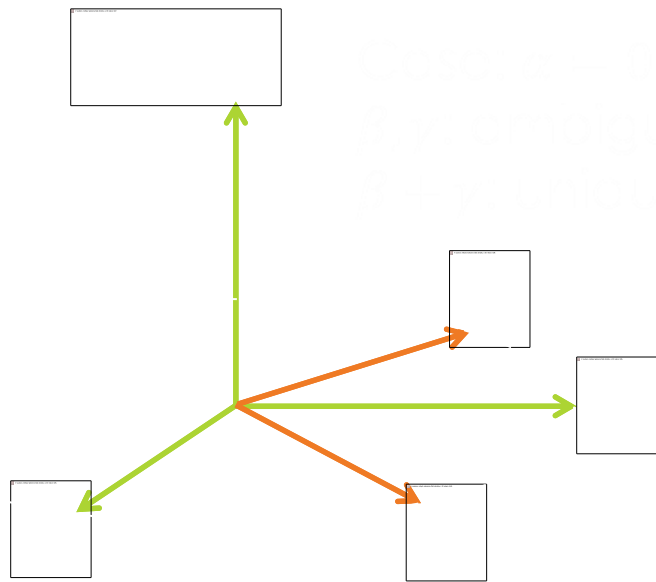
Euler angles



More intuition is in video: [5]

Euler angles: gimbal lock

- ▶ Planes xy and XY are parallel \Rightarrow 1 degree of freedom is lost = gimbal lock.



- ▶ The special value of α "locks" the other two rotations into the same plane (although they can rotate freely in that plane).

Euler angles representation

- ▶ We can use 3 angles to express any orientation of an object in 3D space:

```
public class Orientation {  
    float alpha;  
    float beta;  
    float gamma;  
};
```

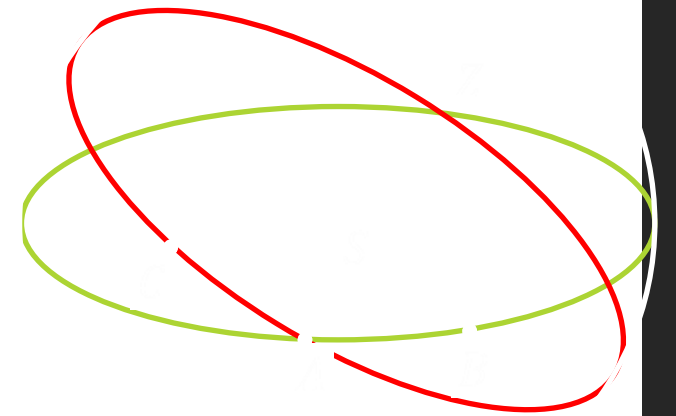
- ▶ Pros:
 - ▶ Low memory footprint.
 - ▶ Easy to understand.
- ▶ Cons:
 - ▶ Suffers from the gimbal lock.
 - ▶ Slow conversion to matrix representation (sin and cos for each angle).

Tait-Bryan angles

- ▶ Same as Euler angles, except that all three axes are different.
- ▶ There are 6 possible conventions:
 - ▶ **XYZ XZY YXZ YZX ZXY and ZYX**
- ▶ The line of nodes is different: it is an intersection of the **xy** plane and the plane orthogonal to the 3rd rotation axis of the convention.
- ▶ The angles α , β , γ are often called yaw, pitch, roll, respectively.

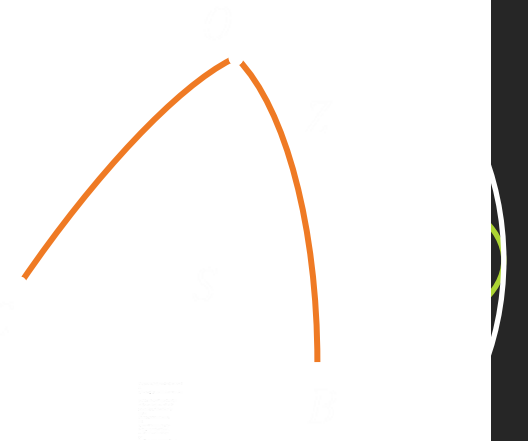
Axis-angle rotation

- ▶ Euler's rotation theorem (one of the versions): Any reconfiguration of an object in 3D space with one of its points fixed is equivalent to its single rotation about an axis passing through the fixed point.
- ▶ Proof:
 - ▶ We look for a rotation axis passing through S .
 - ▶ We "paint" a great circle (green) on the sphere in the initial position.
 - ▶ We rotate the sphere \Rightarrow We get the rotated green circle, which is depicted as red circle.
 - ▶ If the circles coincide, then the axis clearly exists. Otherwise, the circles intersect - two points A, Z .
 - ▶ A is on red circle \Rightarrow its pre-image B is on green one.
 - ▶ A is on green circle \Rightarrow its post-image C is on red one.



Axis-angle rotation

- ▶ Construct a great circle (blue) passing through A, Z and bisecting the angle BAC .
- ▶ Find a point O on the blue circle s.t. the length of arcs AO and BO is the same.
- ▶ The length of the arc AO must be equal to the length of the arc CO , because lengths of arcs AB and AC are the same and the blue circle is the bisector of the angle CAB .
 - ⇒ Triangles CAO and ABO on the sphere must be the same.
Actually, ABO becomes CAO after the rotation.
 - ⇒ The point O lies on the searched rotation axis, because it does not move when rotating the triangles.
 - ⇒ O is the rotation axis and the arc length AB is the angle.



Axis-angle rotation

► Rodrigues' rotation formula: A vector $v \in \mathbb{R}^3$ rotated about a unit axis $a \in \mathbb{R}^3$ by an angle $\theta \in (0, 2\pi)$ is the vector

$$v' = \cos \theta v + (1 - \cos \theta)(a \cdot v)a + \sin \theta a \times v.$$

► Proof:

$$v_a = (a \cdot v)a,$$

$$v_{\perp} = v - v_a = v - (a \cdot v)a,$$

$$v_x = a \times v_{\perp} = a \times (v - v_a) = a \times v.$$

Note: $v_x = v_{\perp} \times a$

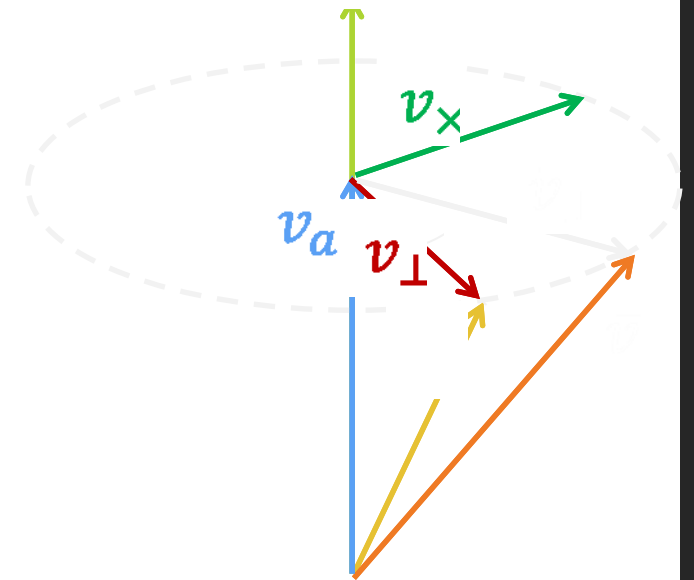
$$v_{\perp}' = \cos \theta v_{\perp} + \sin \theta v_x$$

$$v' = v_a + v_{\perp}'$$

$$= v_a + \cos \theta v_{\perp} + \sin \theta v_x$$

$$= (a \cdot v)a + \cos \theta (v - (a \cdot v)a) + \sin \theta a \times v$$

$$= \cos \theta v + (1 - \cos \theta)(a \cdot v)a + \sin \theta a \times v.$$



Axis-Angle to rotation Matrix

- ▶ Vector triple product: $u \times (v \times w) = (u \cdot w)v - (u \cdot v)w$
- ▶ If $|a| = 1$, then $a \times (a \times v) = (a \cdot v)a - (a \cdot a)v = (a \cdot v)a - v$
- ▶ Matrix representation of the cross product:

$$u \times v = \begin{bmatrix} 0 & -u_x & -u_y \\ u_y & 0 & -u_z \\ -u_y & u_x & 0 \end{bmatrix} v = [[u]]v$$

- ▶ Matrix representation of the axis-angle:

$$\begin{aligned} v &= \cos \theta v + (1 - \cos \theta) (a \cdot v)a + \sin \theta a \times v \\ &= \cos \theta v + (1 - \cos \theta) (a \times (a \times v) + v) + \sin \theta a \times v \\ &= v + (1 - \cos \theta) a \times (a \times v) + \sin \theta a \times v \\ &= v + (1 - \cos \theta) [[a]]^2 v + \sin \theta [[a]]v \\ &= (I + (1 - \cos \theta) [[a]]^2 + \sin \theta [[a]])v \\ &= R(\theta, a) \end{aligned}$$

Linear interpolation (lerp)

- ▶ Given two axis-angle rotations $R(\varphi, a)$ and $R(\psi, b)$, the linearly interpolated rotation is then $R\left((1-t)\varphi + t\psi, \frac{(1-t)a + tb}{|(1-t)a + tb|}\right)$, $t \in (0,1)$.
- ▶ Technical issues related to $|(1-t)a + tb|$:
 - ▶ slow – we must compute the square root.
 - ▶ The result is not defined when $=0$.
- ▶ Problem: The velocity is not constant (increases and decreases).
 - ⇒ visible visual artefact – we prefer uniform blending between rotations.
- ▶ Can we do better?
 - ▶ Yes, use spherical linear interpolation.

Spherical linear interpolation (slerp)

- ▶ Given two linearly independent unit vectors u, v and a parameter $t \in (0, 1)$, find a unit vector $w = \alpha u + \beta v$ s.t. $\alpha, \beta > 0$ and angle between u, w is $t\theta$, where θ is the angle between u, v .

- ▶
$$\vec{w} = \frac{v - \cos \theta u}{\sqrt{(v - \cos \theta u)(v - \cos \theta u)}} = \frac{v - \cos \theta u}{\sqrt{1 - \cos^2 \theta}} = \frac{v - \cos \theta u}{\sin \theta}$$

- ▶
$$w = \cos t\theta u + \sin t\theta \frac{v - \cos \theta u}{\sin \theta}$$

$$= \left(\cos t\theta - \frac{\sin t\theta \cos \theta}{\sin \theta} \right) u + \frac{\sin t\theta}{\sin \theta} v$$

$$= \frac{\cos t\theta \sin \theta - \sin t\theta \cos \theta}{\sin \theta} u + \frac{\sin t\theta}{\sin \theta} v$$

$$= \frac{\sin(1-t)\theta}{\sin \theta} u + \frac{\sin t\theta}{\sin \theta} v$$

- ▶ Given two axis-angle rotations $R(\phi, a)$ and $R(\psi, b)$, the interpolated rotation is then $R\left((1-t)\phi + t\psi, \frac{\sin(1-t)\theta}{\sin \theta} a + \frac{\sin t\theta}{\sin \theta} b\right)$.

Axis-angle representation

- ▶ We can use axis-angle to express any orientation of object in 3D space.

```
public class Orientation {  
    float angle;  
    Vector3 unitAxis;  
};
```

- ▶ Pros:
 - ▶ Fast conversion to matrix representation (sine and cosine for one angle).
 - ▶ We can use lerp and slerp.
 - ▶ Easy to understand.
 - ▶ Low memory footprint.
- ▶ Cons:
 - ▶ Complicated composition of rotations (often solved via other rep.).

Quaternions

- ▶ Let a, b are real numbers and $i^2 = -1$ be an imaginary unit. Then

$$a + bi$$

is a complex number (constructed by the pairing process).

- ▶ Let $a + bi, c + di$ are complex numbers and $j^2 = -1$ be an imaginary unit, $i \neq j$. Then

$$\begin{aligned}(a + bi) + (c + di)j &= \\ a + bi + cj + dij &= \\ a + bi + cj + dk &\end{aligned}$$

where $k = ij$, is a quaternion.

- ▶ $k^2 = -1$ is another unique imaginary unit, i.e., $ik \neq i, k \neq j$.

- ▶ Relations between imaginary units:

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

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

} How to remember those? Think of the cross product of basis vectors of \mathbb{R}^3 , e.g., $\mathbf{i} \times \mathbf{j} = \mathbf{k}$.

Quaternions

- ▶ A quaternion $q = s + u_x i + u_y j + u_z k$ can be written in a scalar-vector notation as a pair $q = (s, u)$, where the vector $u = (u_x, u_y, u_z)$.
- ▶ Let $q = (s, u), p = (t, v)$ be quaternions and c a real number. Then
 - ▶ $q + p = (s, u) + (t, v) = s + u_x i + u_y j + u_z k + t + v_x i + v_y j + v_z k = (s+t, u+v)$.
 - ▶ $cq = (cs, cu)$. $(-1)q = -q = (-s, -u)$.
 - ▶ $qp = (st - u \cdot v, sv + tu + u \times v)$. $c(sp) = (cq)p$.
 - ▶ Conjugation $q^* = s - u_x i - u_y j - u_z k = (s, -u)$. $cq^* = (cq)^*$.
 - ▶ Length $|q| = \sqrt{qq^*} = \sqrt{s^2 + u \cdot u}$. If $|q| = |p| = 1$, then $|qp| = 1$.
 - ▶ Additive unit quaternion $(0, 0)$, multiplicative unit quaternion $(1, 0)$.
 - ▶ $q^{-1} = \frac{1}{|q|^2} q^*$. If $|q| = 1$, then $q^{-1} = q^*$.
 - ▶ $(q + p)^* = q^* + p^*$.
 - ▶ $(qp)^* = p^* q^*$.
 - ▶ Dot product: $q \cdot p = st + u \cdot v$.
 - ▶ Addition and multiplication are associative. Only addition is commutative.

Rotation via quaternion

- ▶ Let $q = (s, u)$ be a quaternion s.t. $|q| = 1$. Then there exists an angle $\alpha \in (0, 2\pi)$ s.t. $q = (\cos \alpha, \sin \alpha v)$, where $v = 0$ if $|s| = 1$, else $v = u/|u|$.
Proof:

If $|s| = 1 \Rightarrow v = 0$. Otherwise,

$|q| = 1 \Rightarrow |s| < 1 \Rightarrow \alpha = \cos^{-1} s$ (choose α s.t. $\sin \alpha > 0$).

$$|q|^2 = |s|^2 + u \cdot u = \cos^2 \alpha + |u|^2 \Rightarrow |u|^2 = 1 - \cos^2 \alpha = \sin^2 \alpha \Rightarrow |u| = |\sin \alpha|$$

TADY →

- ▶ Let $q = (s, u), p = (0, v)$ be quaternions s.t. $|q| = 1$. Then we define $R_q(p) = qpq^{-1} = qpq^* = \dots = (0, (s^2 - u \cdot u)v + 2(u \cdot v)u + 2s(u \times v))$.

- ▶ Let us compare the Rodrigues' rotation formula with $R_q(p)$:
 $\cos \theta v + (1 - \cos \theta)(a \cdot v)a + \sin \theta a \times v$ ← Rodrigues' formula
 $(s^2 - u \cdot u)v + 2(u \cdot v)u + 2s u \times v$ ← vector of $R_q(p)$

- ▶ Question: Is the vector part of $R_q(p)$ equal to Rodrigues' formula?

Rotation via quaternion

- ▶ $q = (s, u)$ can be expressed as $q = (\cos \alpha, \sin \alpha a)$, $s = \cos \alpha$, $u = \sin \alpha a$.
- ▶ Therefore, the vector part of $R_q(p)$ is:

$$(s^2 - u \cdot u)v + 2(u \cdot v)u + 2su \times v =$$

$$(\cos^2 \alpha - \sin^2 \alpha)v + 2 \sin^2 \alpha (a \cdot v)a + 2 \cos \alpha \sin \alpha a \times v.$$

So, the angle α must satisfy these three equalities (relevant trigonometry identities are on the right).

$$\cos \theta = \cos^2 \alpha - \sin^2 \alpha \quad // \quad \cos 2\alpha = \cos^2 \alpha - \sin^2 \alpha$$

$$1 - \cos \theta = 2 \sin^2 \alpha \quad // \quad \cos 2\alpha = 1 - 2 \sin^2 \alpha$$

$$\sin \theta = 2 \cos \alpha \sin \alpha \quad // \quad \sin 2\alpha = 2 \cos \alpha \sin \alpha$$

A solution exists: **$\theta = 2\alpha$** so **$R_q(p)$ rotates p about axis a by 2α .**

- ▶ Observations, for quaternions q, q', p s.t. $|q| = |q'| = 1$:
 - ▶ $R_q(p)q = q'(qpq^*)q = p$ $\Rightarrow q'$ is the inverse rotation to q .
 - ▶ $R_{-q}(p) = (-q)p(-q)^* = qpq^* = R_q(p)$. $\Rightarrow -q$ is the same rotation as q .
 - ▶ $R_{q'}(R_q(p)) = R_{q'}(qpq^*) = q'(qpq^*)q'^* = (q'q)p(q'q)^* = R_{q'q}(p)$.

Quaternions and other representations

- ▶ Conversion axis-angle to quaternion: A vector $v \in \mathbb{R}^3$ rotated about a unit axis $a \in \mathbb{R}^3$ by an angle $\theta \in (0, 2\pi)$ can be computed using quaternions as $R_q(p) = qpq^*$, where $p = (0, v)$, $q = (\cos \theta/2, \sin \theta/2 a)$.
- ▶ Conversion quaternion to axis-angle: Let q be a quaternion s.t. $|q| = 1$ expressed in the form $q = (\cos \alpha, \sin \alpha a)$, $|a| = 1$. Then q represents rotation about the axis vector a by the angle 2α .
- ▶ Conversion quaternion to rotation matrix: Let $q = (s, u)$ be a quaternion s.t. $|q| = 1$. Then the vector part of $R_q((0, v))$ is:

$$(s^2 - u \cdot u)v + 2(u \cdot v)u + 2su \times v =$$

$$(s^2 - (1 - s^2))v + 2(u \times (u \times v)) + (1 - s^2)v + 2su \times v =$$

$$v + 2[u]_x^2 v + 2su \times v =$$

$$[u]_x^2 = 1 - s^2$$

$$I + 2[u]_x^2 + 2s[u]_x$$

$$[u]_y^2 = 1 - s^2$$

$$[u]_z^2 = 1 - s^2$$

$$[u]_x \times [u]_y = -[u]_z$$

$$[u]_y \times [u]_z = [u]_x$$

$$[u]_z \times [u]_x = [u]_y$$

Linear interpolation (lerp)

- ▶ Given quaternions q, p s.t. $|q| = |p| = 1$ we can linearly interpolate between them by $t \in (0,1)$:

$$q(t) = \frac{(1-t)q + tp}{|(1-t)q + tp|}$$

- ▶ Technical issues related to $|(1-t)q + tp|$:
 - ▶ Slow – we must compute the square root.
 - ▶ The result is not defined when $=0$.
- ▶ Problem: The velocity is not constant (increases and decreases).
 - ⇒ visible visual artefact – we prefer uniform blending between rotations.
- ▶ Can we do better?
 - ▶ Yes, use spherical linear interpolation.

Spherical linear interpolation (slerp)

▶ let us first compute a quaternion Δq representing a rotation from a quaternion $q_0 = (s, u)$ to $q_1 = (h, v)$. We assume $|\Delta q| = |q_0| = |q_1| = 1$.

$$\Rightarrow \Delta q q_0 = q_1$$

$$\Delta q = q_1 q_0^{-1} = q_1 q_0^* = (sh + u \cdot v, sv - hu + u \times v) = (\cos \alpha, \sin \alpha a).$$

where,

$$\alpha = \cos^{-1}(sh + u \cdot v), a = (sv - hu + u \times v) / \sin \alpha \quad (\alpha = 0 \text{ for } \alpha = 0, \pi).$$

▶ For $t \in (0, 1)$ we define $\Delta q(t) = (\cos t\alpha, \sin t\alpha a)$. So, we get:

$$\text{slerp}(q_0, q_1, t) = \Delta q(t) q_0 = (\cos t\alpha, \sin t\alpha a) q_0.$$

Quaternion derivative

► Let $q(t) = s(t) + u_x(t)i + u_y(t)j + u_z(t)k = (s(t), u(t))$, be a quaternion where $s(t), u_x(t), u_y(t), u_z(t)$ are functions of $t \in \mathbb{R}$. Then we define

$$\begin{aligned}\frac{dq(t)}{dt} &= \lim_{\Delta t \rightarrow 0} \frac{q(t+\Delta t) - q(t)}{\Delta t} \\ &= \lim_{\Delta t \rightarrow 0} \frac{s(t+\Delta t) - s(t)}{\Delta t} + i \lim_{\Delta t \rightarrow 0} \frac{u_x(t+\Delta t) - u_x(t)}{\Delta t} + j \lim_{\Delta t \rightarrow 0} \frac{u_y(t+\Delta t) - u_y(t)}{\Delta t} \\ &\quad + k \lim_{\Delta t \rightarrow 0} \frac{u_z(t+\Delta t) - u_z(t)}{\Delta t} \\ &= \frac{ds(t)}{dt} + i \frac{du_x(t)}{dt} + j \frac{du_y(t)}{dt} + k \frac{du_z(t)}{dt} \\ &= \begin{pmatrix} \frac{ds(t)}{dt} \\ \frac{du(t)}{dt} \end{pmatrix}\end{aligned}$$

► Note: $\int q(t) dt = \int s(t) dt + i \int u_x(t) dt + j \int u_y(t) dt + k \int u_z(t) dt$.

Quaternion derivative

- ▶ Example: Let's compute a derivative of the orientation of a frame of reference (orange) rotating a constant angular speed $|\omega|$ about the unit axis vector $\hat{\omega} = \omega/|\omega|$.

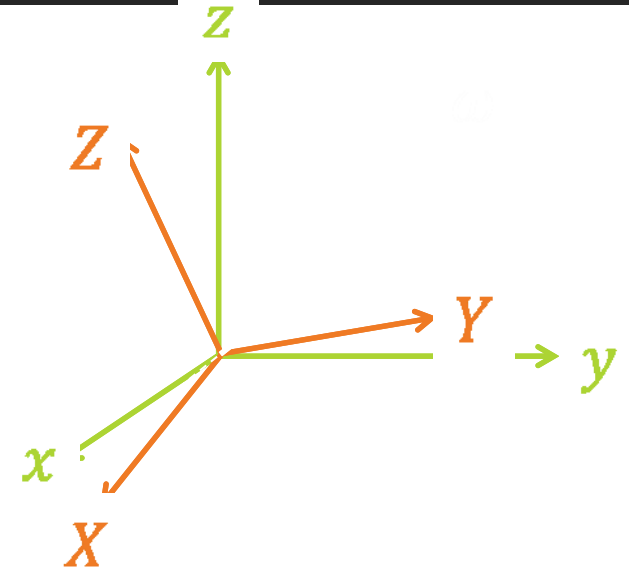
▶ Solution:

- ▶ Let $q(t)$ be an orientation (rotation) of the orange frame in the green one (world).

- ▶ We express the rotation about $\hat{\omega}$:

$$\Delta q(\Delta t) = \left(\cos \frac{|\omega|\Delta t}{2}, \sin \frac{|\omega|\Delta t}{2} \hat{\omega} \right).$$

- ▶ $\dot{q}(t) = \lim_{\Delta t \rightarrow 0} \frac{q(t+\Delta t) - q(t)}{\Delta t} = \lim_{\Delta t \rightarrow 0} \frac{\Delta q(\Delta t)q(t) + q(t)}{\Delta t} = \left(\lim_{\Delta t \rightarrow 0} \frac{\Delta q(\Delta t)q(t)}{\Delta t} + (1, 0) \right) \dot{q}(t)$



Quaternion derivative



Quaternion representation

- ▶ We can use quaternions to express any orientation of object in 3D space.

```
public class Orientation { // Equals to a quaternion  $q=(s,u)$ ,  $|q|=1$ .
    float s; // the scalar part
    Vector3 u; // the vector part
};
```

- ▶ Pros:
 - ▶ Low memory footprint.
 - ▶ Fast conversion to rotation matrix (no need to compute cosine & sine).
 - ▶ Fast composition of rotations (just multiply the quaternions).
 - ▶ We can use lerp and slerp.
- ▶ Cons:
 - ▶ Less human readable.

What representation to use?

- ▶ There is no single winner – each representation has pros and cons.
- ▶ Examples:
 - ▶ When specifying a rotation along a **coordinate** axis (e.g., world Z), then Euler angles are a good choice.
 - ▶ When specifying a rotation along **non-coordinate** axis, then axis-angle representation is a good choice.
 - ▶ When composing rotations of some joint of a skeleton, then quaternions can do it quickly.
 - ▶ When rotations must be composed with other transformations, then use matrix representation.
- ▶ Game engine should provide **all** representations and conversions between them.

References

- ▶ [1] D.M.Mount; Lecture notes CMSC 425 Game Programming; University of Maryland, 2016.
- ▶ [2] E.B.Dam,M.Koch,M.Lillholm; Quaternions, Interpolation and Animation; TechnicalReport DIKU-TR-98/5, University of Copenhagen, 1998.
- ▶ [3] M.C.Nechyba; Lecture notes EEL6667: Kinematics, Dynamics and Control of Robot Manipulators - Introduction to quaternions; 2003, https://mil.ufl.edu/nechyba/www/__eel6667.f2003/course_materials.html.
- ▶ [4] Y.-B. Jia; Lecture notes (Com S 477/577 Notes) Quaternions; 2018.
- ▶ [5] Euler angles: <https://www.youtube.com/watch?v=A6lf8t9WXn8>