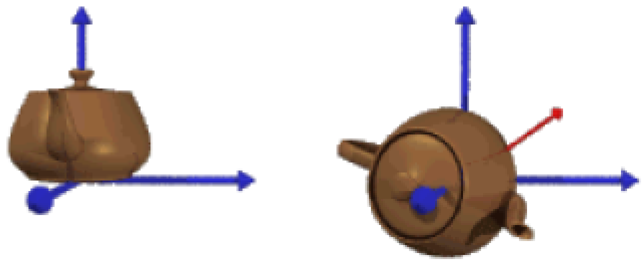
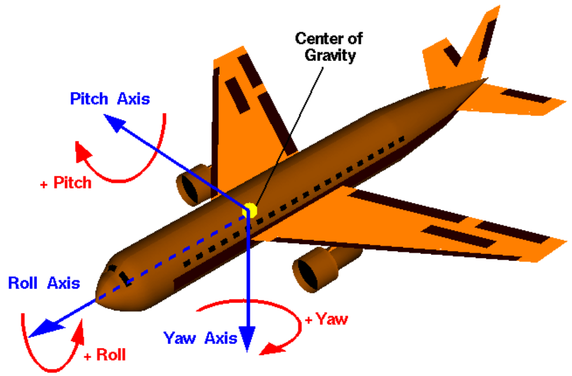
T7 - 3D Rotations

Taesoo Kwon

3D Rotations

- **More complicated than 2D rotations**
  - **Rotate objects along a rotation axis**

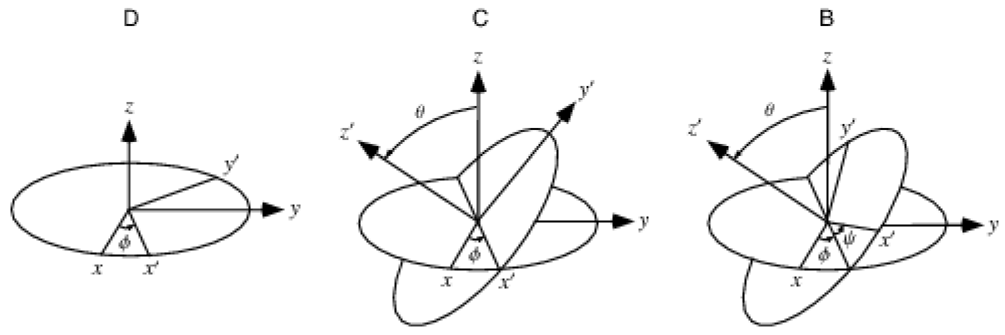




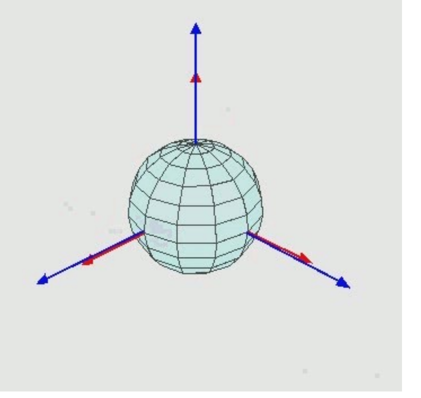
- **Several approaches**

  - Euler angles
  - Rotation matrices
  - Axis-angle (rotation vector)
  - Unit quaternions

Euler Angles



Rot(z,ϕ)   Rot(x',θ)   Rot(z',a)



- ZXZ Euler angle
  1. Rotate along Z-axis
  2. Rotate along X-axis of the new frame
  3. Rotate along Z-axis of the new frame

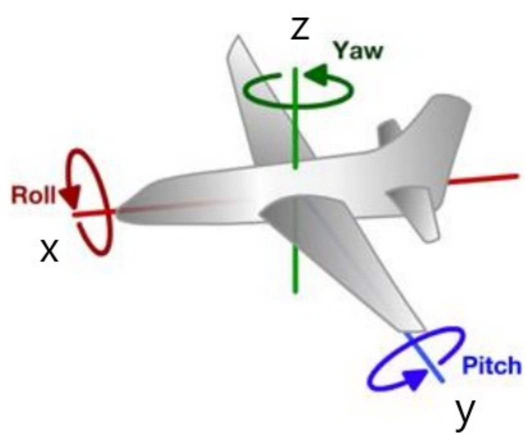
Euler angles:
XYZ XZY XYX XZX
YXZ YZX YXY YZY
ZXY ZYX ZXZ ZYZ

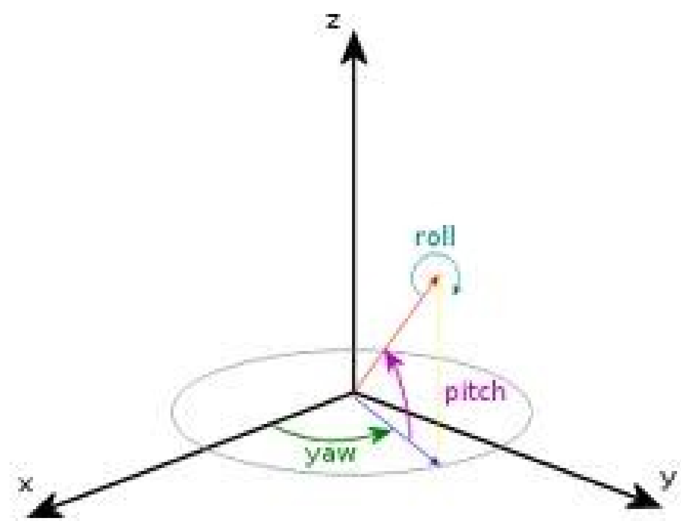
$$R= \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R= \quad R_z(\alpha) \quad R_x(\beta) \quad R_z(\gamma)$$

Vehicle Orientation: Roll-Pitch-Yaw

- Generally, for vehicles, it is most convenient to rotate in roll, pitch, and then yaw
- ZYX Euler angle
  - Rotate about Z-axis (yaw), then about (new) Y-axis (pitch) of body frame, finally about (new) X-axis (roll) of body frame
    (note: textbook is wrong in describing ZYX Euler angle (pg. 32))
  - $R=R_z(\psi)R_Y(\theta)R_x(\phi)$



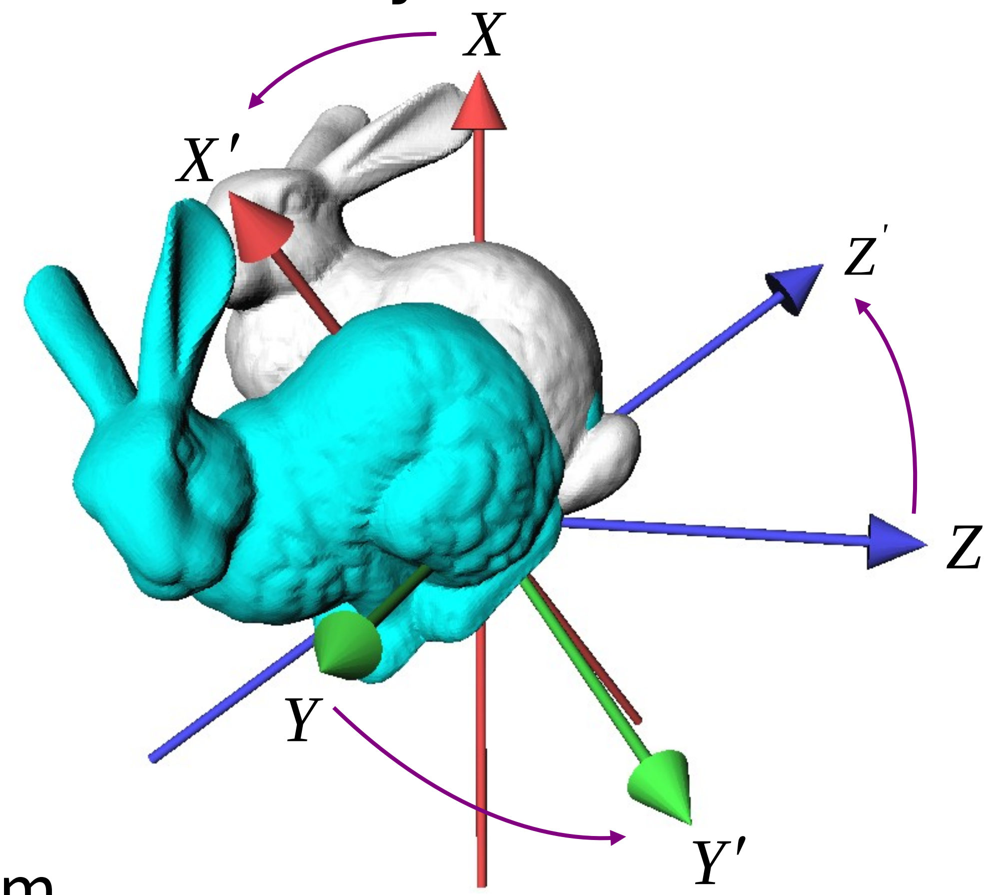


"yaw-pitch-roll"

$$R = R_z(\text{yaw})\, R_y(\text{pitch})\, R_x(\text{roll})$$

Euler's theorem

- Given two arbitrary orientations of a rigid object,



Euler's Theorem

Any two independent orthonormal coordinate frames can be related by a sequence of (not more than three) rotations about coordinate axes.

Gimbal

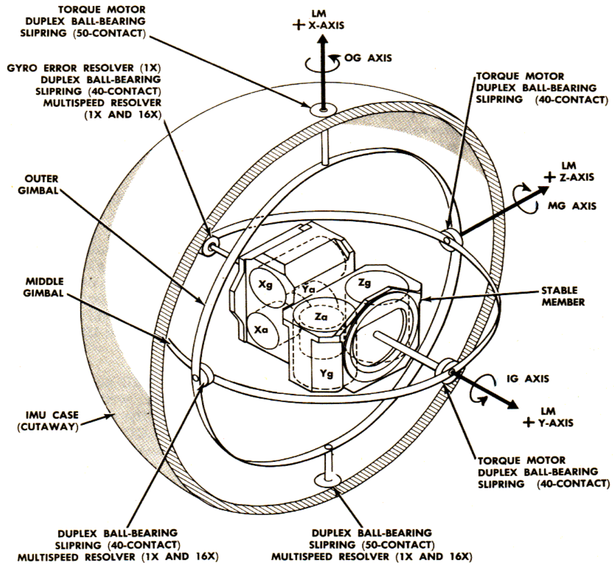
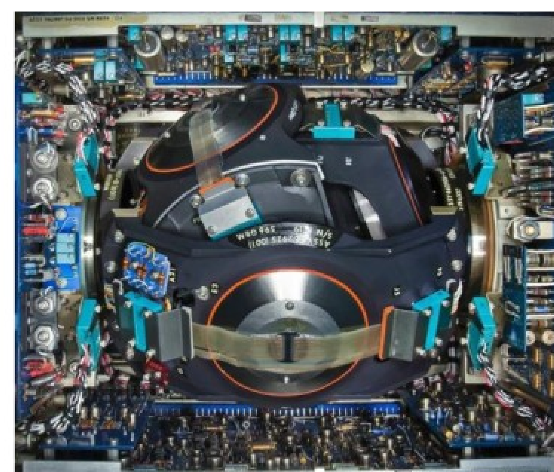


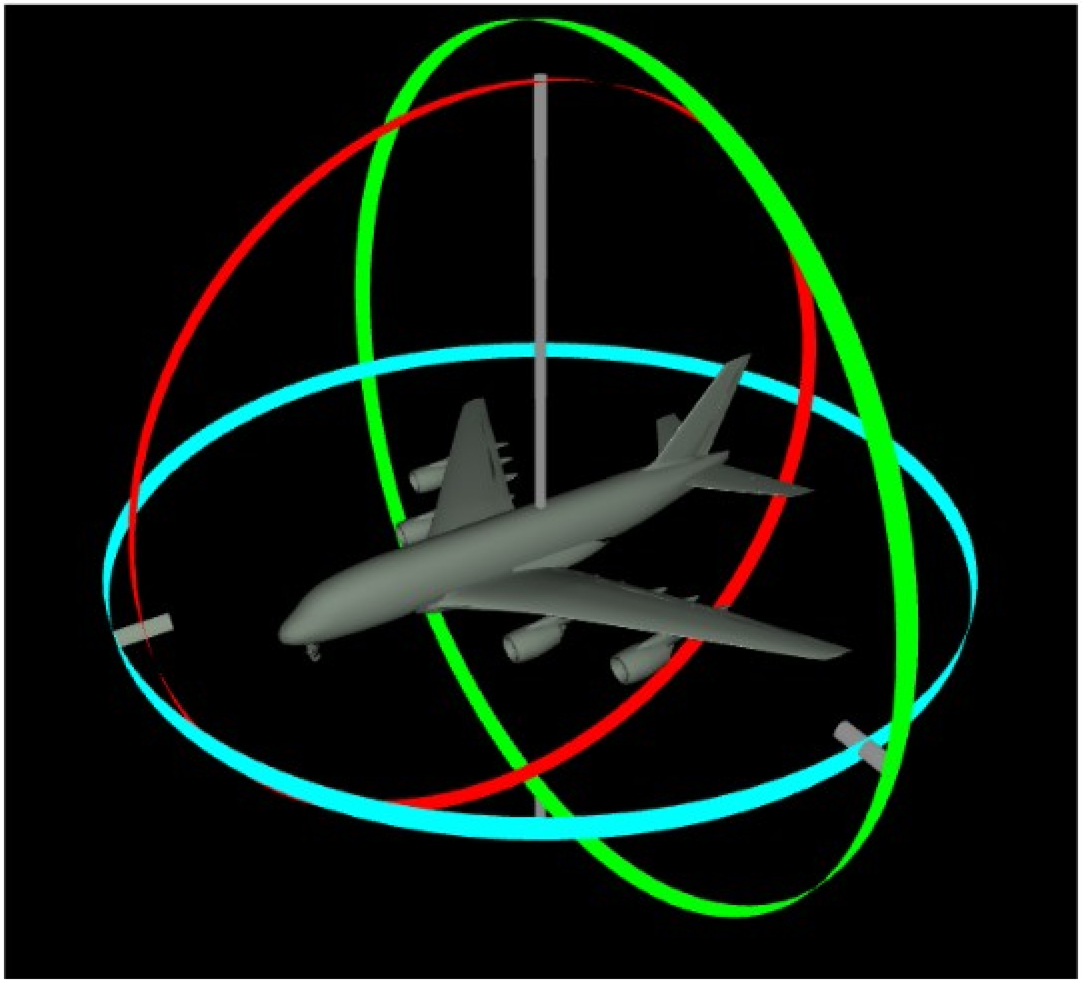
Figure 2.1-24. IMU Gimbal Assembly

Hardware implementation of Euler angles

Used for cameras, Inertial navigation systems for aircrafts and ships



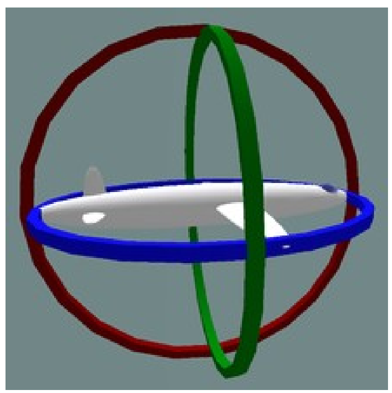
[Practice] Euler Angles Online Demo



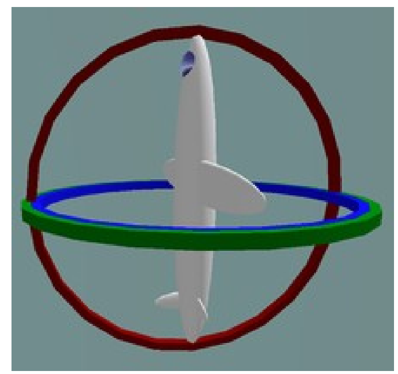
http://www.ctralie.com/Teaching/COMPSCI290/Materials/EulerAnglesViz/

Gimbal Lock

- One potential problem that Euler angles can suffer from is ‘gim bal lock’
- This results when two axes effectively line up, resulting in a tem porary loss of a degree of freedom



Normal situation.
The plane can rotate
in any directions



Gimbal lock:
two out of the three
gimbals are in the same
plane, one DoF is lost

- Euler angles have **singularities**, i.e., it loses DoFs (can’t move in a certain direction) at some configurations

Axis-angle Representations

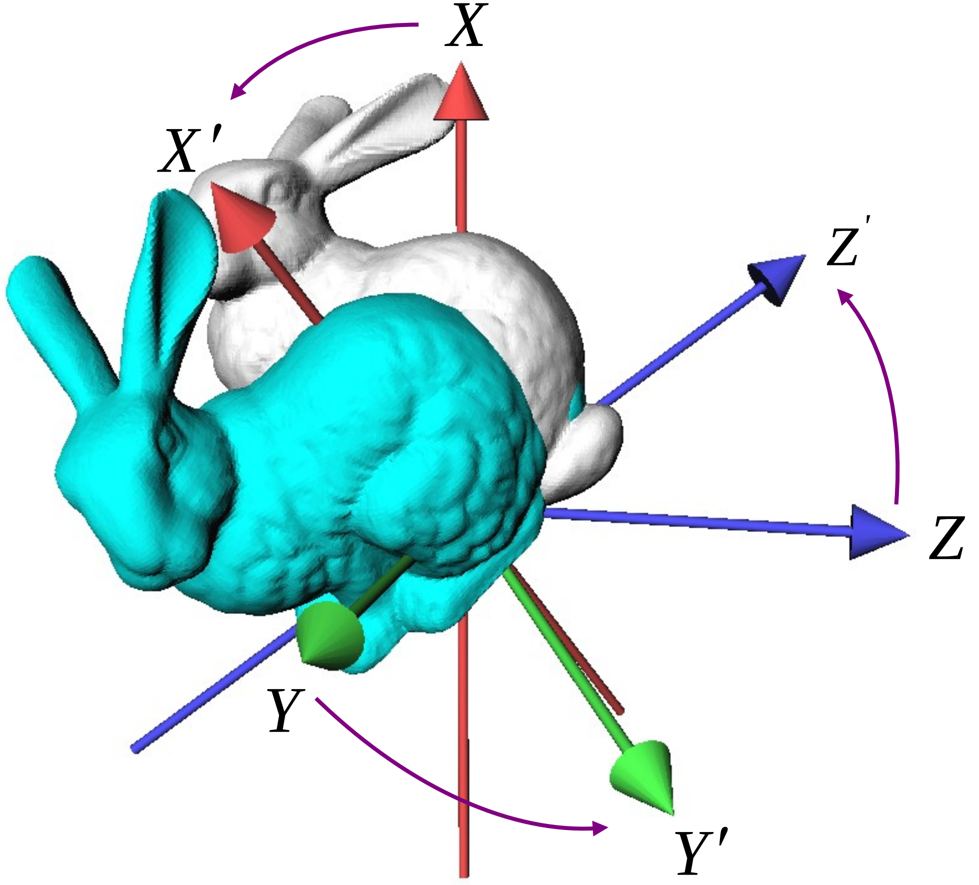
Axis angle: (angle, axis)

Rotation vector : axis*angle

Quaternion
- a compact representation of a 3x3 rotation matrix
  (can be converted back and forth)
- 4D vector (w,x,y,z) having unit length
- angle, axis representation
- definition : (cos(angle/2), sin(angle/2)*axis)

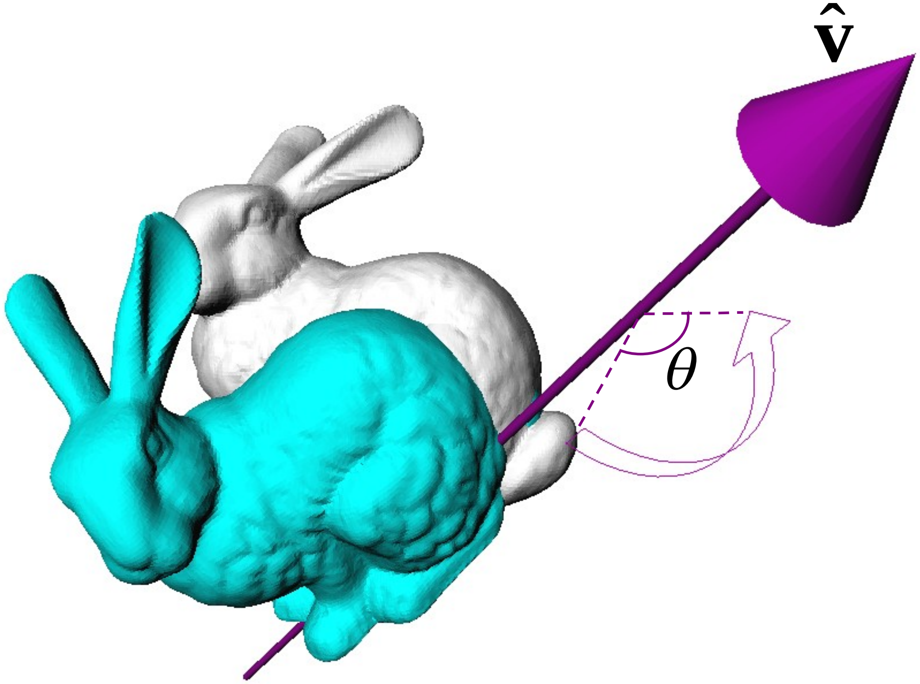
3D Rotation

- Given two arbitrary orientations of a rigid object,



3D Rotation

- We can always find a fixed axis of rotation and an angle about the axis



Euler’s Rotation Theorem

The general displacement of a rigid body with one point fixed is a rotation about some axis

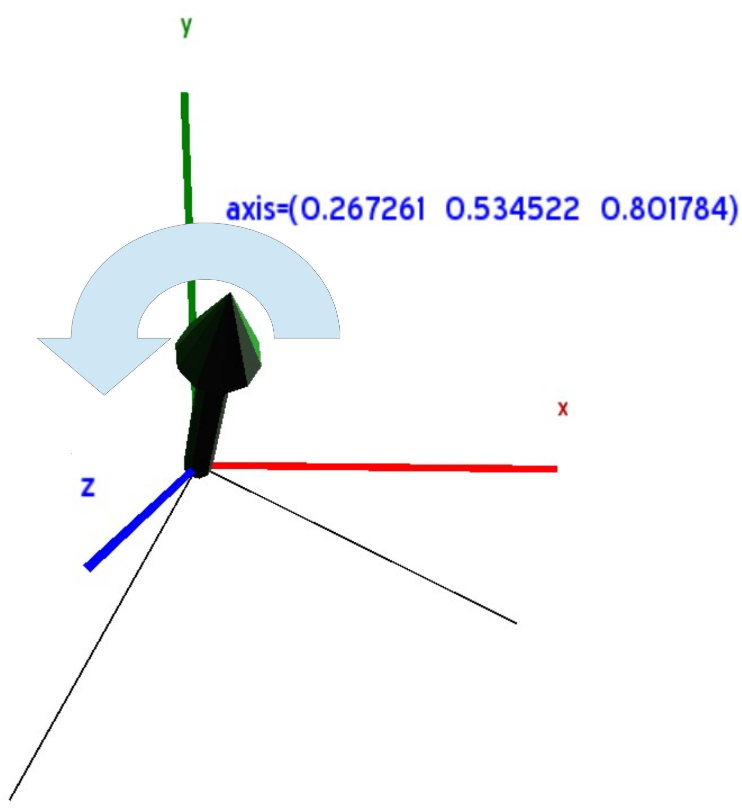
Leonhard Euler (1707-1783)

In other words,

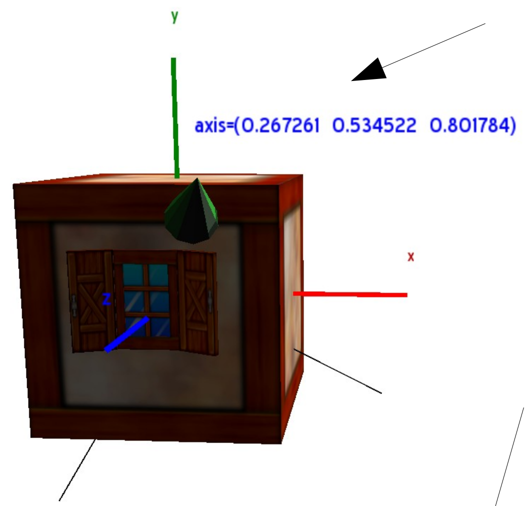
- Arbitrary 3D rotation equals to one rotation around an axis
- Any 3D rotation leaves one vector unchanged

Let's compute the rotation matrix R

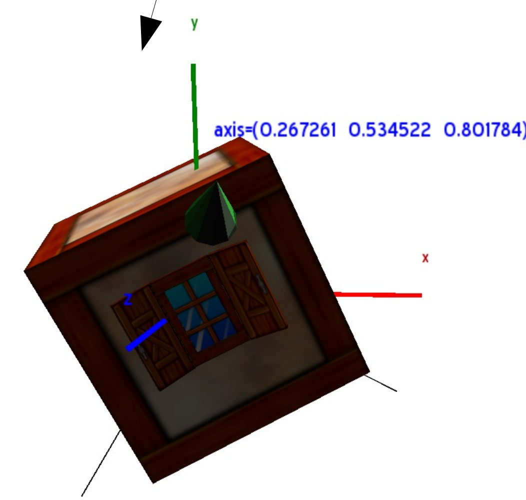
Rotation about $\boldsymbol{axis}=(1,2,3)$ by $\theta=30$ degrees



Before rotation



After rotation



Let's compute the rotation matrix R

Rotation about $\boldsymbol{axis} = (1,2,3)$ by $\theta = 30$ degrees

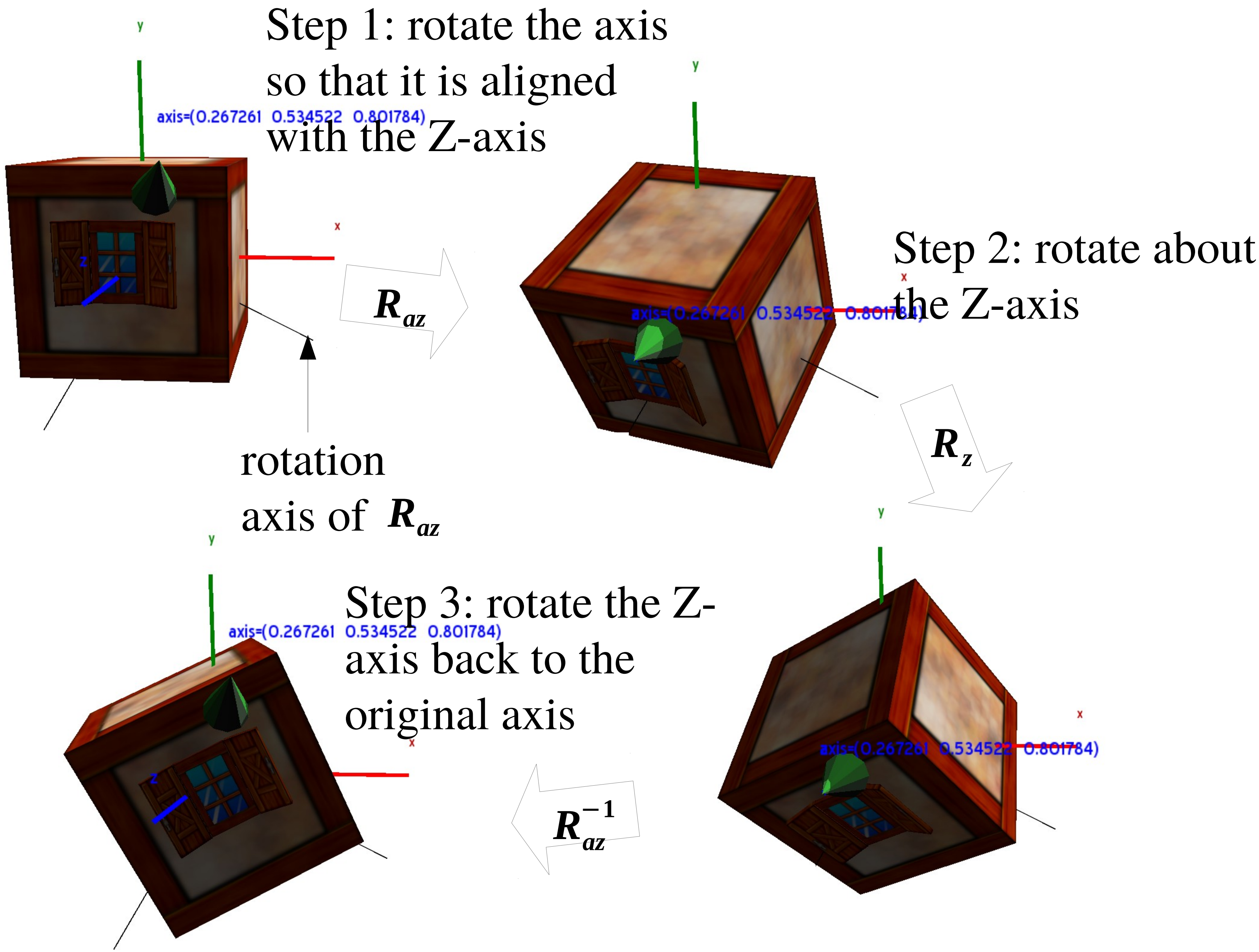
1. We know the rotation matrix about Z-axis

$$\boldsymbol{R_z} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} & -0.5 & 0 \\ 0.5 & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

2. We know how to composite rotations

$$\boldsymbol{R} = \boldsymbol{R_{az}^{-1}} \boldsymbol{R_z} \boldsymbol{R_{az}}$$

Where $\boldsymbol{R_{az}}$ is the matrix rotates the $\boldsymbol{axis} = (1,2,3)$ to Z-axis


Step 1: rotate the axis so that it is aligned with the Z-axis
axis=(0.267261 0.534522 0.801784)
y
x
z
R_{az}
rotation axis of R_{az}
Step 2: rotate about the Z-axis
R_z
Step 3: rotate the Z-axis back to the original axis
R_{az}^{-1}

How to compute R_{az}

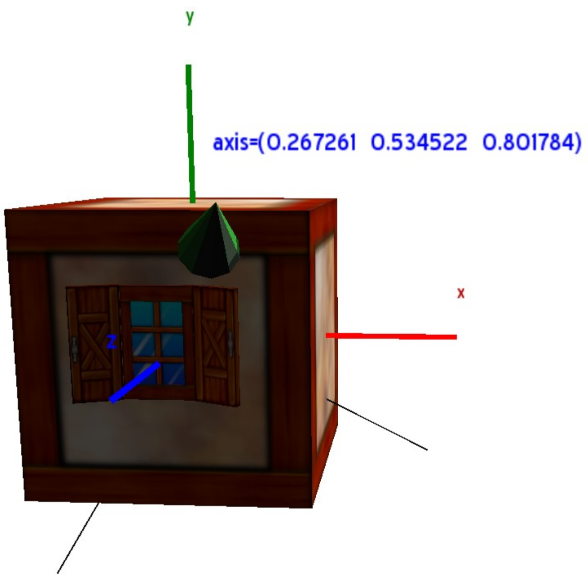
- This is an inefficient way to do this.
- A better method will be explained later

$$\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v}\cdot\boldsymbol{v}} = x^2 + y^2 + z^2 \quad \text{where} \quad \boldsymbol{v} = (x, y, z)$$

1. Calculate the unit-length rotation axis

$$\boldsymbol{axis} = (1,2,3)$$

$$\boldsymbol{a} = \frac{axis}{\|axis\|} \approx (0.27, 0.53, 0.80)$$



Matlab codes:

```
> axis=[1 2 3]'
> a=axis/norm(axis)
```

How to compute R_{az} (Axis a to axis z)

- (This is an inefficient way, but ...)

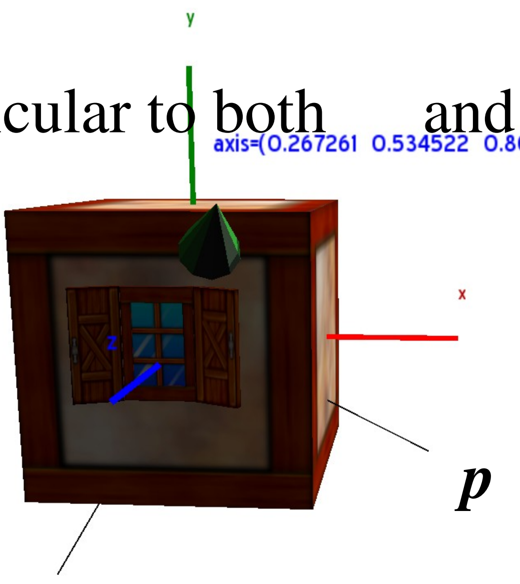
$$\|\boldsymbol{v}\| = \sqrt{\boldsymbol{v}\cdot\boldsymbol{v}} = x^2 + y^2 + z^2 \quad \text{where} \quad \boldsymbol{v} = (x, y, z)$$

1. Let the normalize axis

$$\boldsymbol{a} = \frac{axis}{\|axis\|} \approx (0.27, 0.53, 0.80)$$

2. Calculate vector $\boldsymbol{p}$ that is perpendicular to both and Z-axis

$$\boldsymbol{p} = \frac{\boldsymbol{a}\times(0,0,1)}{\|\boldsymbol{a}\times(0,0,1)\|}$$



Matlab codes:

```
> p=cross(a,[0;0;1])
> p=p/norm(p)
```

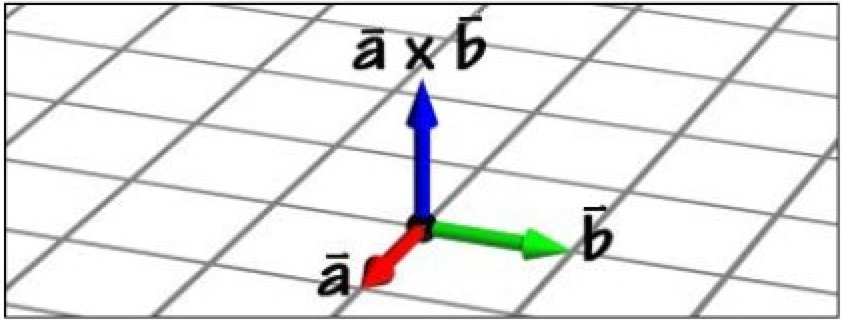
Cross Product (×)

$$\vec{a}\times\vec{b}\equiv\begin{bmatrix}0 & -a_z & a_y & 0\\ a_z & 0 & -a_x & 0\\ -a_y & a_x & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}b_x\\ b_y\\ b_z\\ 0\end{bmatrix}=\vec{c}$$

$$\vec{a}\cdot\vec{c}=0$$
$$\vec{b}\cdot\vec{c}=0$$

$$\vec{c}=\begin{bmatrix}a_y b_z - a_z b_y & a_z b_x - a_x b_z & a_x b_y - a_y b_x\end{bmatrix}$$

- **Return a vector $\vec{c}$ that is perpendicular to both $\vec{a}$ and $\vec{b}$, oriented according to the right-hand rule**
- **The matrix is called the skew-symmetric matrix of $\vec{a}$**



KAIST

Cross Product (×)

- **A mnemonic device for remembering the cross-product**

$$\vec{a} \times \vec{b} \equiv det \begin{bmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{bmatrix}$$

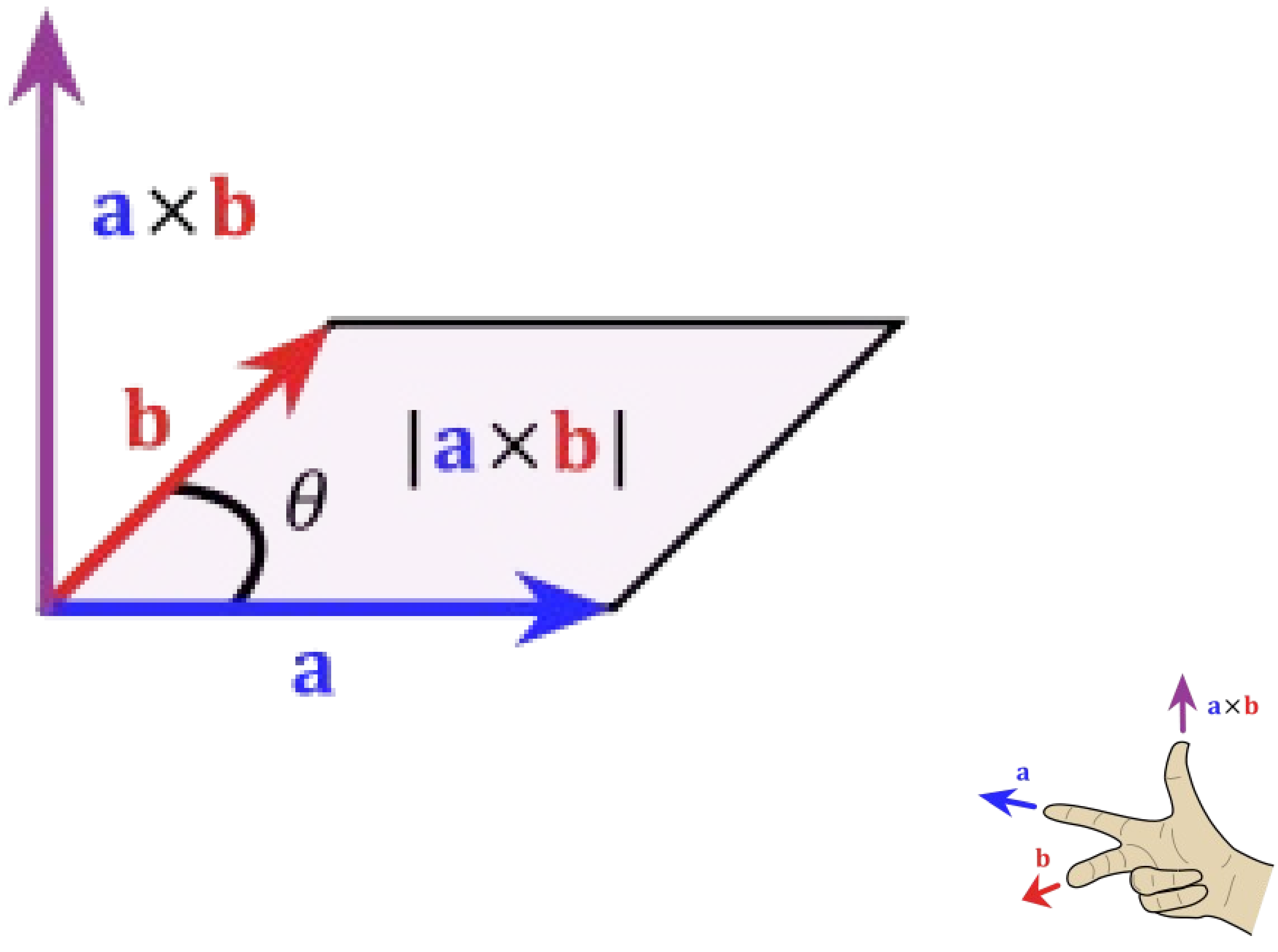
$$= (a_y b_z - a_z b_y)\vec{i} + (a_z b_x - a_x b_z)\vec{j} + (a_x b_y - a_y b_x)\vec{k}$$

$$\vec{i} = [1 \quad 0 \quad 0]$$

$$\vec{j} = [0 \quad 1 \quad 0]$$

$$\vec{k} = [0 \quad 0 \quad 1]$$

KAIST

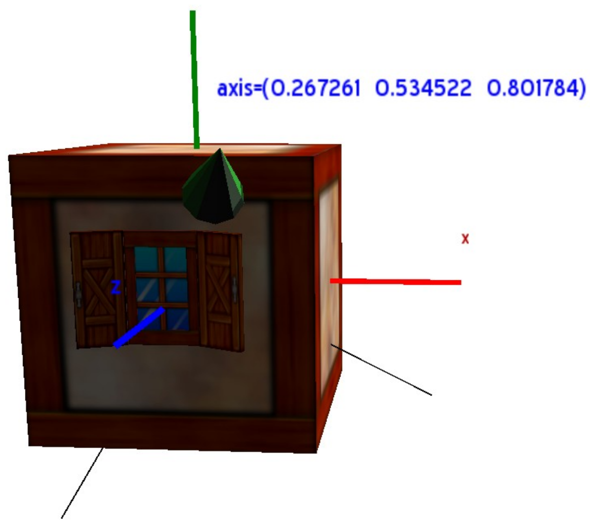


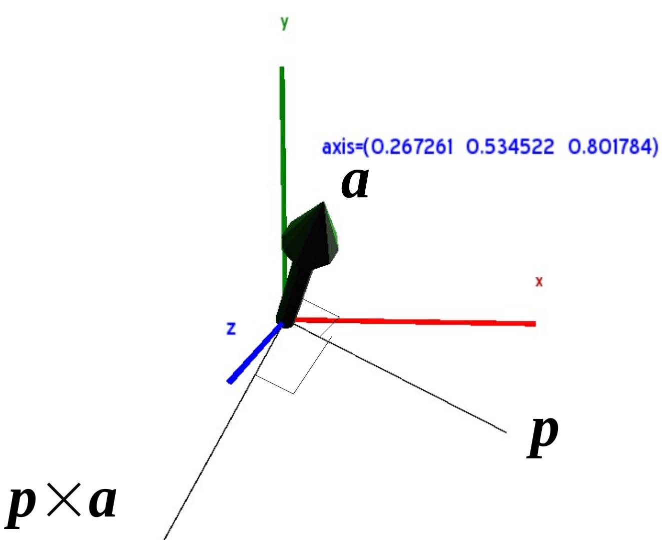
a × b
b
θ
|a × b|
a
a × b
a
b

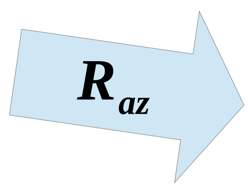
How to compute R_{az}

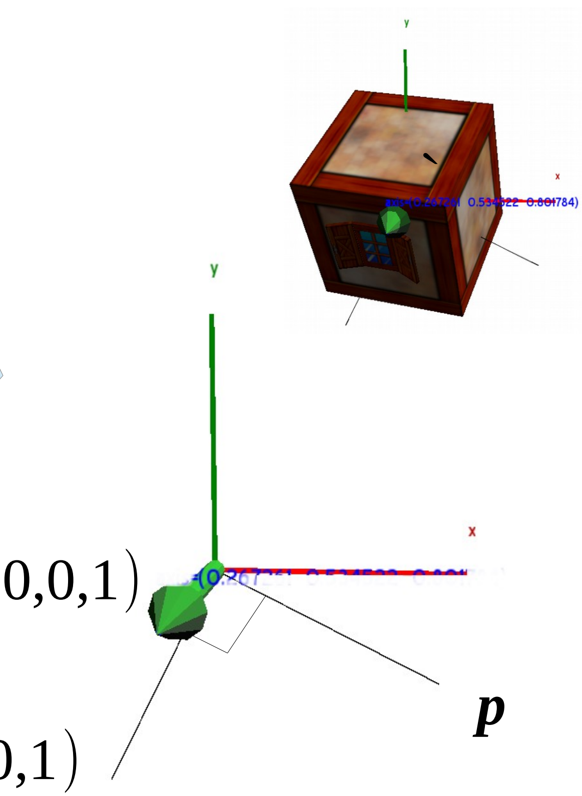
3. after rotation R_{az}

$R_{az}\,a$	becomes	$(0,0,1)$
$R_{az}\,p$	becomes	p
$R_{az}(p \times a)$	becomes	$p \times (0,0,1)$









How to compute $\boldsymbol{R}_{az}$

3. Then after the rotation $\boldsymbol{R}_{az}$

$$\boldsymbol{R}_{az}\,\boldsymbol{a} \quad \text{becomes} \quad (0,0,1)$$

$$\boldsymbol{R}_{az}\,\boldsymbol{p} \quad \text{becomes} \quad \boldsymbol{p}$$

$$\boldsymbol{R}_{az}(\boldsymbol{p}\times\boldsymbol{a}) \quad \text{becomes} \quad \boldsymbol{p}\times(0,0,1)$$

Therefore,

$$\boldsymbol{R}_{az}([\boldsymbol{a}][\boldsymbol{p}][\boldsymbol{p}\times\boldsymbol{a}])=\begin{pmatrix} 0 & & \\ 0 & [\boldsymbol{p}] & [\boldsymbol{p}\times(0,0,1)] \\ 1 & & \end{pmatrix}$$

Finally,

$$\boldsymbol{R}_{az}=\begin{pmatrix} 0 & & \\ 0 & [\boldsymbol{p}] & [\boldsymbol{p}\times(0,0,1)] \\ 1 & & \end{pmatrix}([\boldsymbol{a}][\boldsymbol{p}][\boldsymbol{p}\times\boldsymbol{a}])^{-1}$$

Matlab codes:

```
> z=[0;0;1]
> Raz=[z p cross(p,z)] *inv([a p cross(p,a)])
```

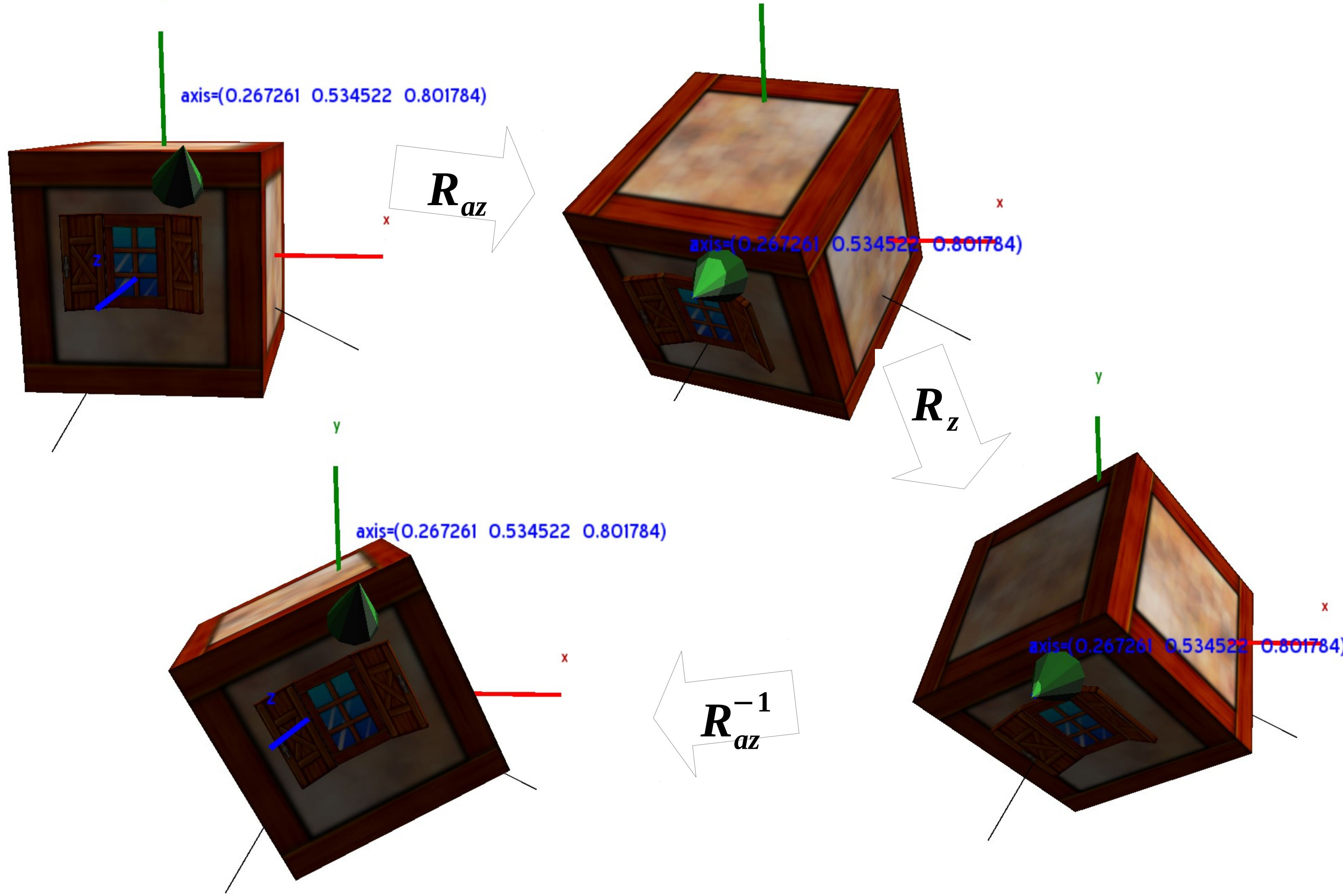
Test orthonormality of R_az

```
> Raz'*Raz
→ identity

> Raz*Raz'
→ identity

> Raz(:,1)
> Raz(:,2)
> Raz(:,1)'*Raz(:,2)
> Raz(:,2)'*Raz(:,3)
> Raz(:,3)'*Raz(:,1)
```

All combinations leads to 0

Finally,

- The rotation matrix $R = R_{az}^{-1} R_z R_{az}$



Verity the solution using matlab or octave (a free alternative to matlab)

```
> R= [  0.875595  -0.381753   0.295900; 0.42   0.9043  -0.076; -0.23 0.19   0.9521]
> R
R =

   0.875595  -0.381753   0.295900
   0.420031   0.904300  -0.076000
  -0.230000   0.190000   0.952100
```

The resulting matrix R should be similar to this

```
> R*R'
ans =

   9.9996e-01   6.9406e-05   7.8065e-03
   6.9406e-05   9.9996e-01   2.8503e-03
   7.8065e-03   2.8503e-03   9.9549e-01
```

m' means the transpose of m

Orthonormality
→ ans = Identity

Eigen values and vectors

> eig(R)

ans =

0.86528 + 0.49752i
0.86528 - 0.49752i
1.00143 + 0.00000i

Eigen value == 1

An **eigenvector** of a square matrix A is a non-zero vector v that, when the matrix is multiplied by v, yields a constant multiple of v, the multiplier being commonly denoted by λ. That is:

$$Av = \lambda v$$

> [v,d]=eig(R)

v =

0.68249 + 0.00000i	0.68249 - 0.00000i	0.26848 + 0.00000i
-0.10502 - 0.59228i	-0.10502 + 0.59228i	0.53329 + 0.00000i
-0.15928 + 0.38341i	-0.15928 - 0.38341i	0.80220 + 0.00000i

Eigen vector :

$$\boldsymbol{a} = \frac{axis}{\|axis\|} \approx (0.27, 0.53, 0.80$$

D = ...

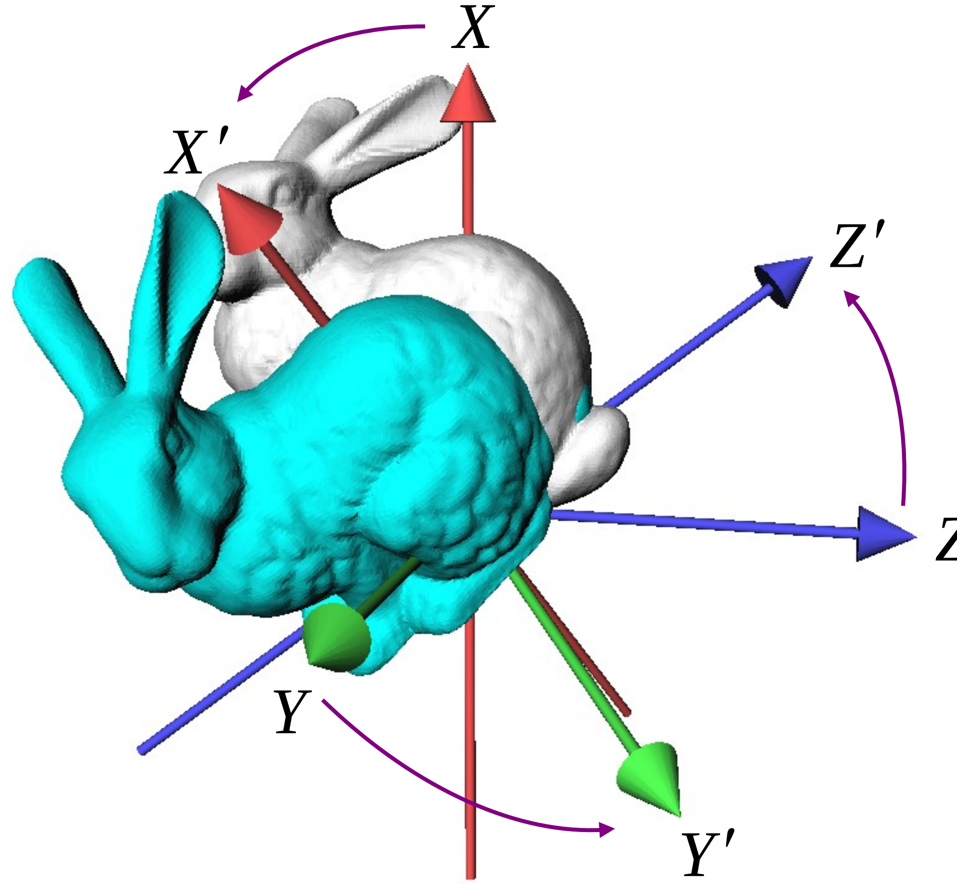
More efficient solution:

- Use the following Rodrigues' rotation formula
  - The previous solution is slow because it involves a matrix inversion (or transpose)

Homogeneous coordinates in 3D (Affine transformation)

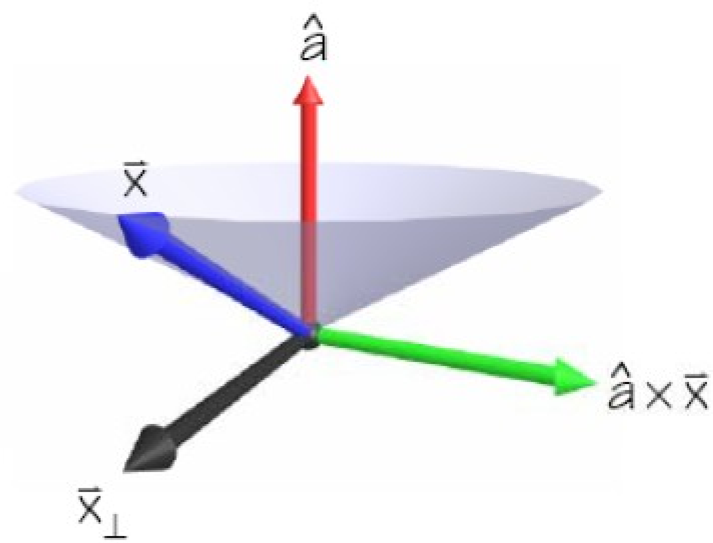
$$\left(\begin{array}{cccc} v_{1x} & v_{2x} & v_{3x} & o_x \\ v_{1y} & v_{2y} & v_{3y} & o_y \\ v_{1z} & v_{2z} & v_{3z} & o_z \\ 0 & 0 & 0 & 1 \end{array}\right)$$

$$\begin{array}{cccc} \uparrow & \uparrow & \uparrow & \uparrow \\ \vec{X'} & \vec{Y'} & \vec{Z'} & \dot{O} \end{array}$$

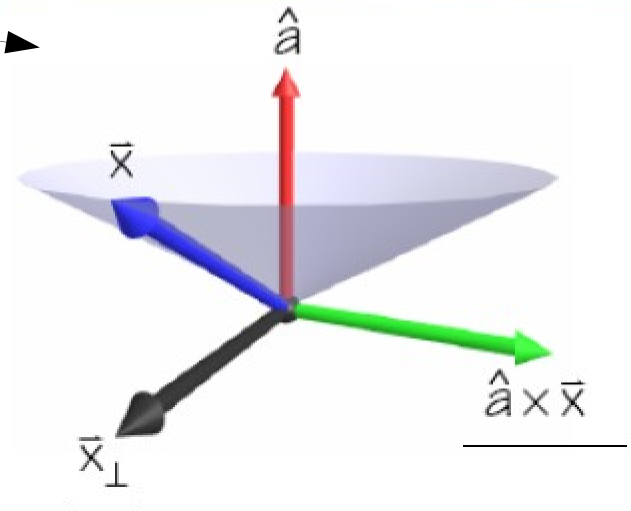


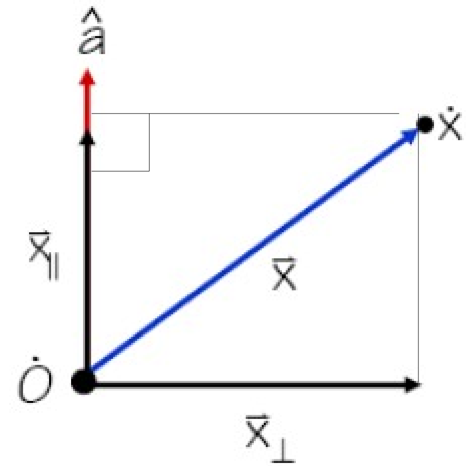
Geometry of a Rotation

- **Natural basis for rotation of a vector about a specified axis:**
  - $\hat{a}$ - rotation axis (normalized)
  - $\hat{a} \times \vec{x}$ - vector perpendicular to
  - $\vec{x}_{\perp}$ - perpendicular component of $\vec{x}$ relative to $\hat{a}$



Geometry of a Rotation





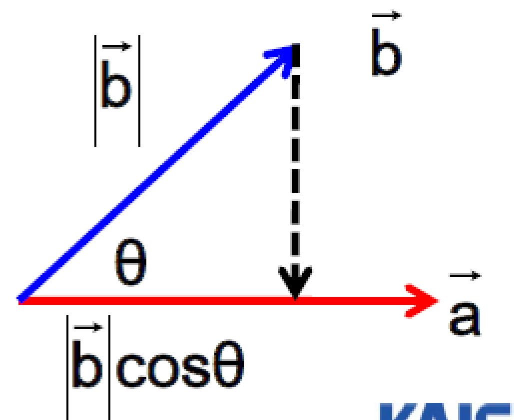
$\vec{x}_{\parallel}$ Parallel to the axis

$\vec{x}_{\perp}$ Perpendicular to the axis

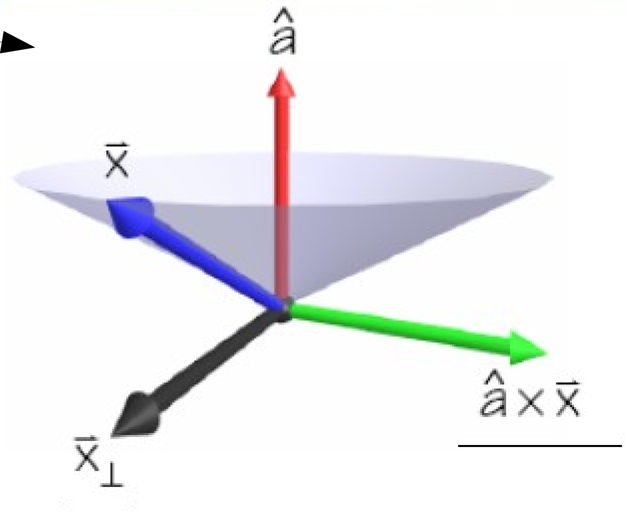
Dot Product ($\cdot$)

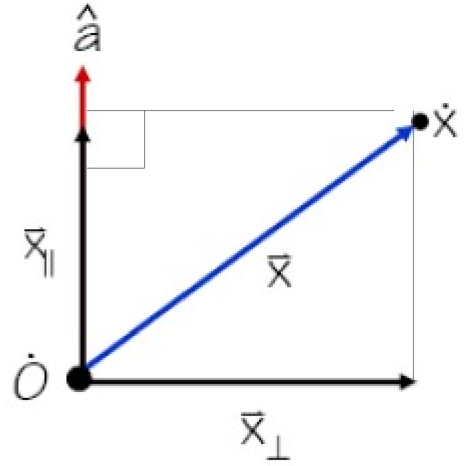
$$\vec{a} \cdot \vec{b} \equiv \vec{a}^T \vec{b} = \begin{bmatrix} a_x & a_y & a_z & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \\ 0 \end{bmatrix} = s$$

- **Returns a scalar s**
- **Geometric interpretations s:**
  - $\vec{a} \cdot \vec{b} = |a||b|\cos\theta$
  - **Length of** $\vec{b}$ **projected onto and** $\vec{a}$ **or vice versa**



$\vec{x}_{\parallel}$ Parallel to the axis

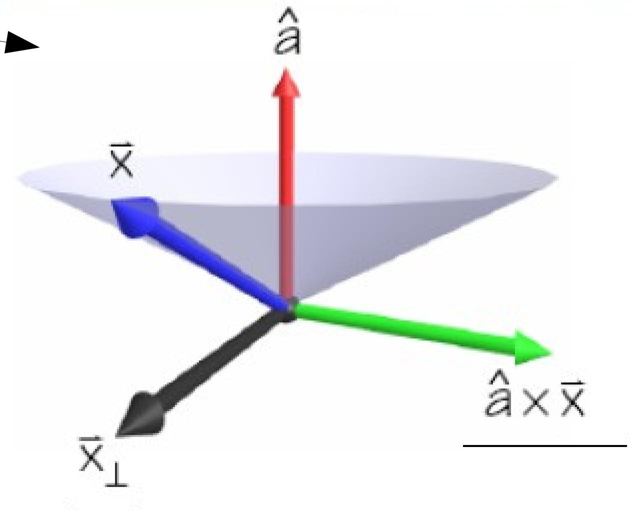


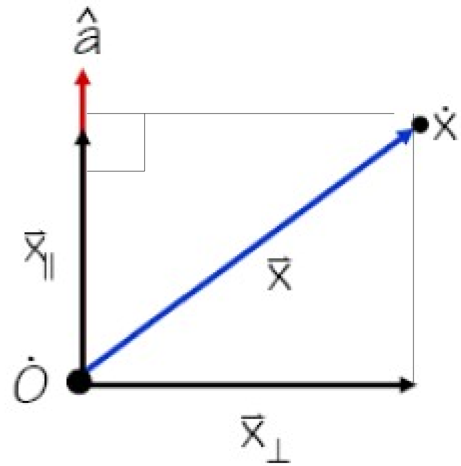


$$\left\| \vec{x}_{\parallel} \right\| = (\hat{a} \cdot \vec{x})$$

$$\vec{x}_{\parallel} = \hat{a}(\hat{a} \cdot \vec{x})$$

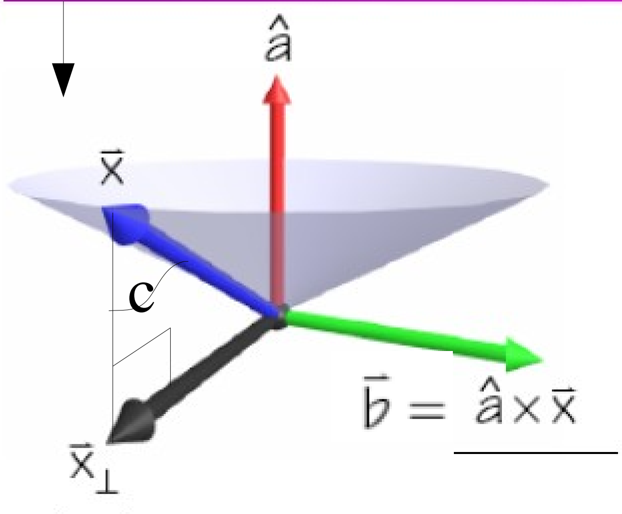
$\vec{x}_\perp$ Perpendicular to the axis





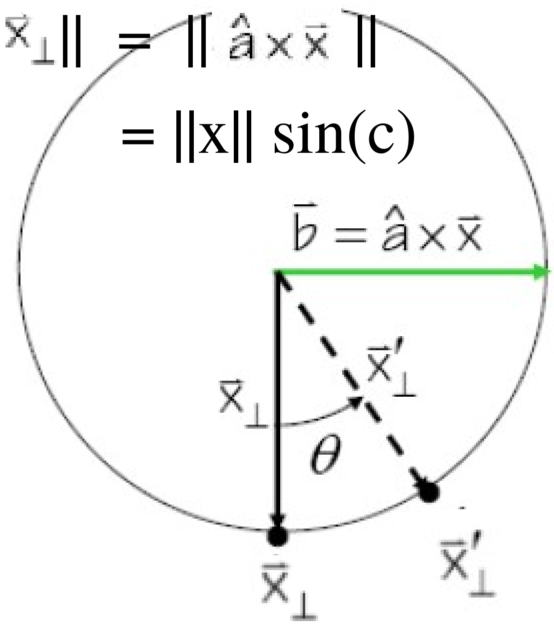
$$\vec{x}_\perp = \vec{x} - \vec{x}_\parallel$$

Geometry of a Rotation



Note that $\|\vec{x}_\perp\| = \|\hat{a} \times \vec{x}\|$

$= \|x\| \sin(c)$



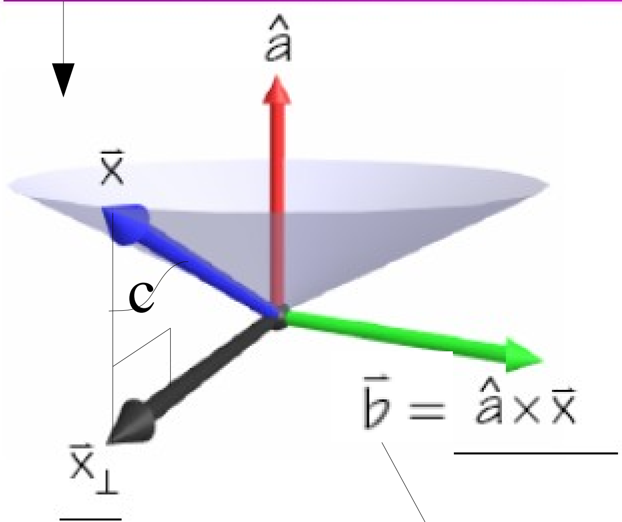
$x_{\|}$

$$\dot{x}' = \dot{O} + x_{\|} + \vec{x}'_\perp$$

$$\vec{x}'_\perp = \cos\theta\,\vec{x}_\perp + \sin\theta\,\vec{b}$$

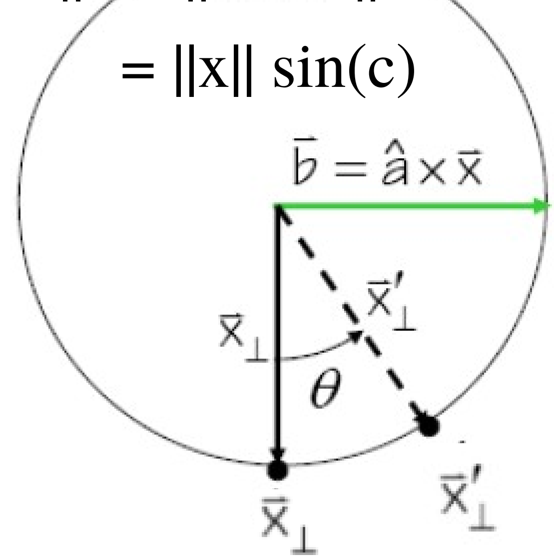
$x_{\|}$ is preserved after rotation

Geometry of a Rotation



Note that $\|\vec{x}_\perp\| = \|\hat{a} \times \vec{x}\|$

$= \|x\| \sin(c)$



$$\dot{x}' = \dot{O} + x_{\|} + \vec{x}'_\perp$$

$$\vec{x}'_\perp = \cos\theta\, \vec{x}_\perp + \sin\theta\, \vec{b}$$

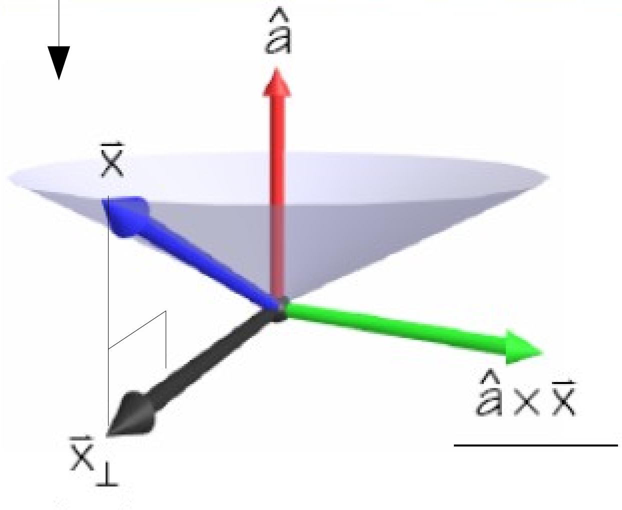
$$\vec{x}_{\|} = \hat{a}(\hat{a} \cdot \vec{x})$$

$$\vec{x}_\perp = \vec{x} - \vec{x}_{\|}$$

$$\boxed{\dot{x}' = \dot{O} + \cos\theta\, \vec{x} + (1 - \cos\theta)(\hat{a}(\hat{a} \cdot \vec{x})) + \sin\theta(\hat{a} \times \vec{x})}$$

The above equation is obtained after substituting equations

Geometry of a Rotation



$$c_{\dot{x}} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

$$c_{\dot{x}'} = M = \left[\begin{matrix} f\begin{pmatrix}1\\0\\0\end{pmatrix} & f\begin{pmatrix}0\\1\\0\end{pmatrix} & f\begin{pmatrix}0\\0\\1\end{pmatrix} & \dot{o} \end{matrix}\right]$$

$\dot{O}$

$$\dot{x}' = \dot{O} + x_{\|} + \vec{x}'_{\perp}$$

$$\vec{x}'_{\perp} = \cos\theta\,\vec{x}_{\perp} + \sin\theta\,\vec{b}$$

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

$$\vec{x}_{\perp} = \vec{x} - \vec{x}_{\|}$$

where $\dot{x}' = f(\vec{x})$ denotes the equation below

$$\boxed{\dot{x}' = \dot{O} + \cos\theta\,\vec{x} + (1-\cos\theta)(\hat{a}(\hat{a}\cdot\vec{x})) + \sin\theta(\hat{a}\times\vec{x})}$$

$$\mathbf{c}_{\dot{x}'} = \mathbf{M}\mathbf{c}_{\dot{x}}$$

$$\mathbf{M} = \mathrm{diag}(\dot{O}) + \cos\theta\,\mathrm{diag}([1 \quad 1 \quad 1 \quad O]^t)$$
$$+ (1-\cos\theta)\mathbf{A}_{\otimes} + \sin\theta\,\mathbf{A}_{\times}$$

Let R be the upper-left 3x3 submatrix of $\mathbf{M}$, and

$$\hat{\boldsymbol{a}} = \begin{bmatrix} u_x \\ u_y \\ u_z \\ 0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ 0 \end{bmatrix}$$

$$R = \begin{bmatrix} \cos\theta + u_x^2\left(1-\cos\theta\right) & u_x u_y\left(1-\cos\theta\right) - u_z\sin\theta & u_x u_z\left(1-\cos\theta\right) + u_y\sin\theta \\ u_y u_x\left(1-\cos\theta\right) + u_z\sin\theta & \cos\theta + u_y^2\left(1-\cos\theta\right) & u_y u_z\left(1-\cos\theta\right) - u_x\sin\theta \\ u_z u_x\left(1-\cos\theta\right) - u_y\sin\theta & u_z u_y\left(1-\cos\theta\right) + u_x\sin\theta & \cos\theta + u_z^2\left(1-\cos\theta\right) \end{bmatrix}.$$ [4]

This can be written more concisely as

$$R = I\cos\theta + \sin\theta[\mathbf{u}]_\times + (1-\cos\theta)\mathbf{u}\otimes\mathbf{u},$$

where

$$\mathbf{u}\otimes\mathbf{u} = \begin{bmatrix} u_x^2 & u_x u_y & u_x u_z \\ u_x u_y & u_y^2 & u_y u_z \\ u_x u_z & u_y u_z & u_z^2 \end{bmatrix}, \qquad [\mathbf{u}]_\times = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}.$$

$$= \boldsymbol{u}\boldsymbol{u}^T = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}$$

Tensor Product ($\otimes$)

$$\vec{a}\otimes\vec{b} \equiv \vec{a}\vec{b}^t = \begin{bmatrix} a_x \\ a_y \\ a_z \\ 0 \end{bmatrix} \begin{bmatrix} b_x & b_y & b_z & 0 \end{bmatrix} = \begin{bmatrix} a_x b_x & a_x b_y & a_x b_z & 0 \\ a_y b_x & a_y b_y & a_y b_z & 0 \\ a_z b_x & a_z b_y & a_z b_z & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$(\vec{a}\otimes\vec{b})\vec{c} = \begin{bmatrix} (b_x c_x + b_y c_y + b_z c_z) a_x \\ (b_x c_x + b_y c_y + b_z c_z) a_y \\ (b_x c_x + b_y c_y + b_z c_z) a_z \end{bmatrix} = \vec{a}(\vec{b}\cdot\vec{c})$$

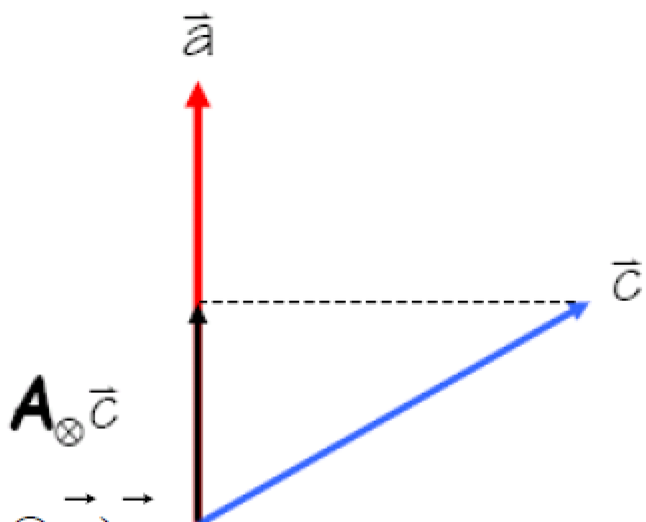
- **Creates a matrix that when applied to a vector $\vec{c}$ return $\vec{a}$ scaled by the project of $\vec{c}$ onto $\vec{b}$**

KAIST

Tensor Product ($\otimes$)

- **Useful when** $\vec{b} = \vec{a}$
- **The matrix** $\vec{a} \otimes \vec{a}$ **is called the symmetric matrix of** $\vec{a}$
  - **We shall denote this** $A_\otimes$

$$A_\otimes = \vec{a} \otimes \vec{a} = \begin{bmatrix} a_x a_x & a_x a_y & a_x a_z & 0 \\ a_y a_x & a_y a_y & a_y a_z & 0 \\ a_z a_x & a_z a_y & a_z a_z & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



$$= (\vec{a} \otimes \vec{a})\vec{c}$$

$$= \vec{a}(\vec{a} \cdot \vec{c})$$

Sanity Check

- **Consider a rotation by about the x-axis**

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

$$=\begin{bmatrix}1&0&0&0\\0&\cos\theta&-\sin\theta&0\\0&\sin\theta&\cos\theta&0\\0&0&0&1\end{bmatrix}$$

- **You can check it in any computer graphics book, but you don't need to memorize it**

```
void matrix::fromAxisAngle(AxisAngle4d const & a1) {
   double c = cos(a1.angle);
   double s = sin(a1.angle);
   double t = 1.0 - c;
       //  if axis is not already normalised then uncomment this
       // double magnitude = Math.sqrt(a1.x*a1.x + a1.y*a1.y +
a1.z*a1.z);
       // if (magnitude==0) throw std::runtime_error(“”);
       // a1.x /= magnitude;
       // a1.y /= magnitude;
       // a1.z /= magnitude;
   m00 = c + a1.x*a1.x*t;
   m11 = c + a1.y*a1.y*t;
   m22 = c + a1.z*a1.z*t;
   double tmp1 = a1.x*a1.y*t;
   double tmp2 = a1.z*s;
   m10 = tmp1 + tmp2;
   m01 = tmp1 - tmp2;
   tmp1 = a1.x*a1.z*t;
   tmp2 = a1.y*s;
   m20 = tmp1 - tmp2;
   m02 = tmp1 + tmp2;
   tmp1 = a1.y*a1.z*t;
   tmp2 = a1.x*s;
   m21 = tmp1 + tmp2;
   m12 = tmp1 - tmp2;
}
```

$$\mathbf{M} = diag(\dot{O}) + \cos\theta \, diag([1 \quad 1 \quad 1 \quad 0]^t) + (1 - \cos\theta)\mathbf{A}_{\otimes} + \sin\theta \, \mathbf{A}_{\times}$$

http://www.euclideanspace.com/maths/geometry/rotations/conversions/angleToMatrix/