Course Plan

66

Quaternion Product

$\vec{v}$ (over a, b, c); $\vec{w}$ (over e, f, g)

×	a *i*	+	b *j*	+	c *k*	+	d	
e *i*	-1 ae	+	*k* be	+	*-j* ce	+	*i* de	+
+								
f *j*	*-k* af	+	-1 bf	+	*i* cf	+	*j* df	+
+								
g *k*	*j* ag	+	*-i* bg	+	-1 cg	+	*k* dg	+
+								
h	*i* ah	+	*j* bh	+	*k* ch	+	hd	

$$(\vec{w}, h) \cdot (\vec{v}, d) = (\text{some vector}, \text{some scalar})$$

68

Quaternion Product

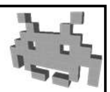
$\vec{v}$ = (a, b, c); $\vec{w}$ = (e, f, g)

×	a i	b j	c k	d
e i	-1 ae	k be	$-j$ ce	i de
f j	$-k$ af	-1 bf	i cf	j df
g k	j ag	$-i$ bg	-1 cg	k dg
h	i ah	j bh	k ch	hd

$$(\vec{w}, h) \cdot (\vec{v}, d) = (\vec{w}\,d + \vec{v}\,h + \vec{w}\times\vec{v},\ \ h\,d - \vec{w}\cdot\vec{v})$$

70

Applying rotations (with quaternions)



quaternion representing a rotation: $(\vec{w}, a)$; quaternion representing 3D point or vector $\vec{v}$: $(\vec{v}, 0)$; conjugate of $(\vec{w}, a)$: $(-\vec{w}, a)$

$$(\vec{w}, a)\ (\vec{v}, 0)\ (-\vec{w}, a) =$$

$$= (\vec{w}, a)\ (a\vec{v} - \vec{v}\times\vec{w}, \vec{v}\cdot\vec{w}) =$$

$$= \left(a\,\vec{w}\times\vec{v} - \vec{w}\times\vec{v}\times\vec{w} + (\vec{v}\cdot\vec{w})\vec{w} + a^2\,\vec{v} - a\,\vec{v}\times\vec{w},\ \ \cancel{a\vec{v}\cdot\vec{w}} - \cancel{a\vec{v}\cdot\vec{w}} + \cancel{\vec{w}\cdot(\vec{v}\times\vec{w})} \right) =$$

$$= (a^2\,\vec{v} + 2a\,\vec{w}\times\vec{v} + (\vec{v}\cdot\vec{w})\,\vec{w} - \vec{w}\times\vec{v}\times\vec{w},\qquad 0)$$

71

Exercise: quaternion norm as a quaternion product

- As you may remember, given a complex number $\mathbf{c} \in \mathbb{C}$, $\mathbf{c} = a + ib$ its magnitude $\|\mathbf{c}\| = \sqrt{a^2 + b^2}$ can be expressed as
$$\|\mathbf{c}\|^2 = \mathbf{c}\,\bar{\mathbf{c}}$$
- Does the same hold for quaternions? Given $\mathbf{q} \in \mathbb{H}$:
$$\|\mathbf{q}\|^2 = \mathbf{q}\,\bar{\mathbf{q}}$$
- Verify, using the multiplication formula we learnt

72

Quaternions as rotations: summary

- Compact to store (4 scalars, almost the minimum)
- Trivial to invert (just conjugate)
- Fast to composite (just multiply: $2^{nd} * 1^{st}$)
- Fast to apply
- Easy to enforce that it stays a rotation (just renormalize)
  - Even after long sequences of cumulations, unlike matrices
- Behaves well under interpolation
  - Just use NLERP – even better with SLERP
  - Remember to take the shortest path (=> flip sign if necessary)
- The favourite representation in 3D games
  - but, other solutions still useful in one context or another

73

Recap: representing rotations 1/2

	3x3 Matrix	Euler Angles
Space efficient? (in RAM, GPU, storage...)	★☆☆☆☆ 9 scalars	★★★★★ 3 scalars (even as small int!)
Efficient / easy to: Apply (to points/vectors)	★★★★☆ 9 products (3 dot products)	★★☆☆☆ requires trigonometry sin/cos
Invert (produce inverse)	★★★★★ just transpose	★☆☆☆☆
Composite (with another rotation)	★★☆☆☆ Matrix multipl (9 dots) Numerical errors	★☆☆☆☆
Interpolate (with another rotation)	★☆☆☆☆ Introduces shear/scale	★☆☆☆☆ easy to do, unintuitive result (⚠ shortest-path required!
Intuitive? (e.g. to manually set)	★★★☆☆	★★★★★ roll yaw pitch
Notes...	Free extra shear + scale. Useful to extract local axes.	⚠ GIMBAL LOCK

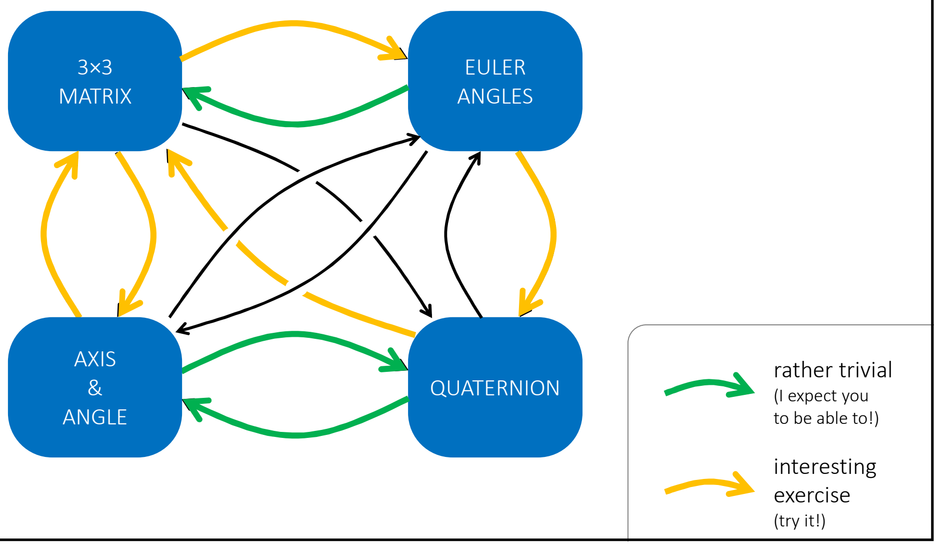
74

Recap: representing rotations 2/2

	axis , angle	(unitary) quaternion
Space efficient? (in RAM, GPU, storage...)	★★★★☆ 4 scalars (or 3) (precision needed)	★★★★☆ 4 scalars (precision needed)
Efficient / easy to: Apply (to points/vectors)	★★★☆☆ Requires trigonometry	★★★★★ Just 2 quat product
Invert (produce inverse)	★★★★★ Just flip axis OR angle	★★★★★ super easy flip imaginary or real part
Composite (with another rotation)	★☆☆☆☆	★★★★★ super easy: 1 quat product
Interpolate (with another rotation)	★★★★★	★★★★☆ easy + good result (NLERP or SLERP)
Intuitive? (e.g. to manually set)	★☆☆☆☆ no	★☆☆☆☆ no
Notes...	⚠ *two* representations for each rotation (flip all → no effect) (for different reasons) Require shortest path!	

75

Switching between representations



78

What defines a rotation, for you?

« Roll, pitch, and yaw! »

then you are… a pilot, or an astronaut

« X-angle, Y-angle, and Z-angle! »

then you are… a digital artist (an animator, or a scener)

« An angle! »

then you are… a flatland citizen

« A vector! the dir is the axis the magnitude the angle »

then you are… a physicist

« A 3x3 matrix! the submatrix of a 4x4 transform »

then you are… a computer graphicist, or a Graphics API

« A quaternion! »

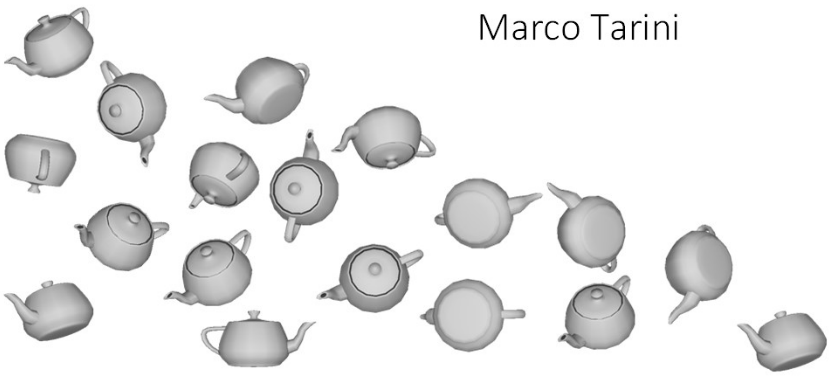
then you are… a game developer 

79

Master Game Dev

Transformations in games: final notes

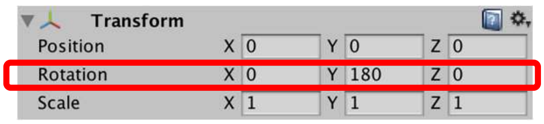
Marco Tarini



80

Notes on rotations in unity (class Quaternion)

- In the GUI :
  - See / set it as Euler Angles (intuitive)



using degrees, not radians
even more intuitive

- Internally:
  - A quaternion (class Quaternion)
- In the C# API:
  - programmer choice: can initialize or use them as a … quaternion, euler angles, axis+angle, or matrix
  - thanks to C# «properties» (setter/getter methods in disguise)
  - gives the illusion to be whichever kind you think they are

82

Notes on Rotations in Unreal Engine

Class `FQuat` : (fields: `W X Y Z`)

- convert from:
  - axis+angle, matrix4x4, Rotator, euler (vec3) (by constructors)
  - Euler angles (`makeFromEuler` method)
  - From-to vector pairs (`FindBetween` method)
- convert to:
  - `ToAxisAndAngle`, `Euler`, `Rotator`,
  - matrix columns `GetAxis(X|Y|Z)`
  - also, with names: `Get(Forward|Right|Up)Vector`,
- methods: invert with `Inverse`,
  blend with `FastSlerp`
  or `FastSlerpFullPath` (no shortest path)
  apply with `RotateVector` / `UnrotateVector`
  composite with `operator *`

Class `FRotator`
for "nautical" Euler angles:
fields: `Pitch Roll Yaw`

83

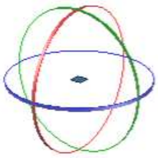
Notes on rotations in OpenGL

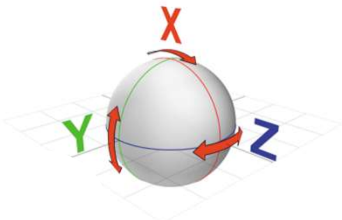
- In the «old school» API:
  (and now in many similar libraries)
  - API: glRotate3f
    - takes: angle & axis
  - Internally:
    - matrices
    - jointly as with any other spatial transform
    - separated in MODEL+VIEW+PROJECT transforms

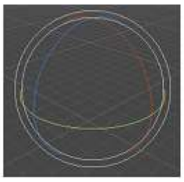
84

GUI: how to author rotations in 3D?

- Typical way: rotation gizmo
  - (also: «arcball» or «trackball»)
  - 3 handles to control the three Euler angles
  - or "free", drag-n-drop mode (trackball metaphor)





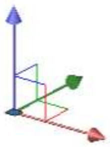


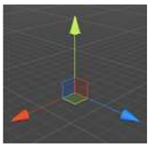
convention: Red = X Green = Y Blue = Z

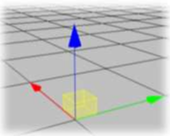
85

GUI: how to author translations in 3D?

- translation gizmo
  - handles to traslate along axes or planes





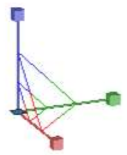
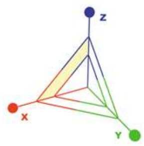


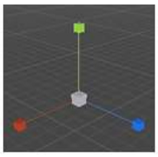
convention: Red = X Green = Y Blue = Z

86

GUI: how to author scalings in 3D?

- scale gizmo
  - 3 handles for anisotropic scalings
    1 handle (middle) for uniform scalings



convention: Red = X Green = Y Blue = Z

87

Next: representations for roto-rotations (notes)

- So far, we assumed that
  the **rotation** and **translation** components
  of a transformation
  are stored *separatedly*
  - We have seen reasons why this is convenient
- Mathematical representations exist, that store rotation + translation (aka *roto-translations*, aka *rigid* transformations) jointly:
  - 4x4 matrices (we have seen them, their pros and cons)
  - Dual quaternsions

88

Representations for ~~rotations~~ **roto-translations**

aka "rigid" transforms

- 3×3 Normal Matrices
- Euler Angles
- Angle & Axis
- Quaternions

+ Translation (displacement vector)

OR:

- 4×4 Matrices (or 3×4)
- Dual Quaternions

As there's no need to store the last row, it's (0,0,0,1)

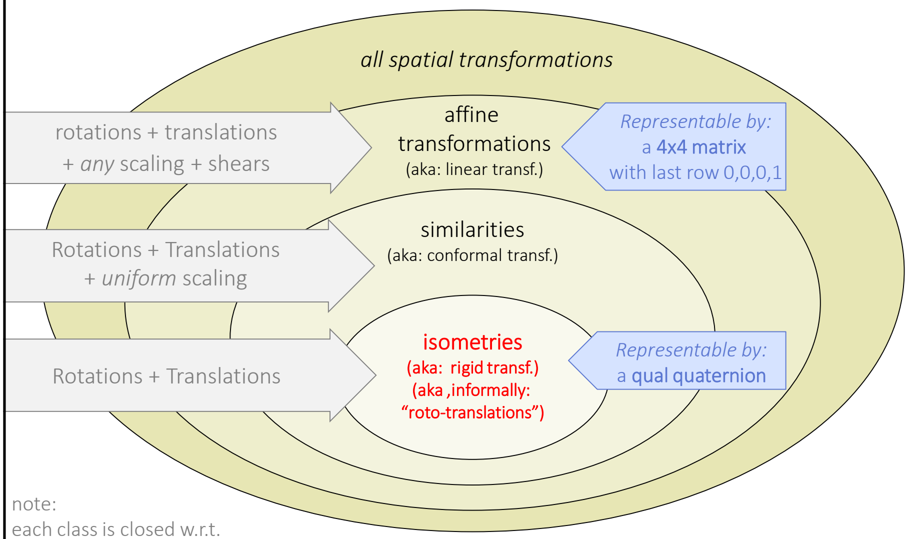
89

Q: why dual-quaternions?
A: better interpolation of rigid motions

- Problem with interpolating rotations and translations separately:
  - must choose "which one goes first" (R then T, or, T then R)?
  - Different choices → very different interpolation results
  - Often, neither is what you had in mind
- Dual quaternions = a better* math abstraction to model roto-translations
  - * better interpolation of roto-translations

90

A math classification of spatial transformations



note:
each class is closed w.r.t.
cumulation, and (when they are invertible) inversion

91

The math of Dual Quaternions in a nutshell 1/3

- Dual quaternions are a mathematical way to represent a roto-translation (aka, a rigid motion)
- They result in very good interpolation between 2 (or more) roto-translations
- They are sometimes used in animation techniques
  - See lecture about skeletal animations later

92

The math of Dual Quaternions in a nutshell 2/3

- New “fantasy” assumption: there is a ε such that $\varepsilon \neq 0$, $\varepsilon^2 = 0$
- A dual quaternion: p + ε q , with p,q ∈ $\mathbb{H}$ ← quaternion set
- So, eight scalars (*a*,*b*,*c*,*d*,*e*,*f*,*g*,*h*)
  - weights for: $1, i, j, k, \varepsilon, \varepsilon i, \varepsilon j, \varepsilon k$

$$\underbrace{\overbrace{a}^{\text{real part of p}} + \overbrace{b\,i + c\,j + d\,k}^{\text{imaginary part of p}}}_{\text{p (the “primal” quaternion)}} + \underbrace{\overbrace{e\,\varepsilon}^{\text{real part of q}} + \overbrace{f\,\varepsilon\,i + g\,\varepsilon\,j + h\,\varepsilon\,k}^{\text{imaginary part of q}}}_{\varepsilon\text{ q (the “dual” quaternion)}}$$

93

The math of Dual Quaternions in a nutshell 3/3

- A dual quaternion p + ε q can represent: (p = $a + b\,i + c\,j + d\,k$, q = $e + f\,i + g\,j + h\,k$)
  - a point / vector in 3D , when p = 1 and Real(q) = *e* = 0
    then Im(q) = (*f*,*g*,*h*) = (*x*,*y*,*z*)
  - a roto-translation, when ||p|| = 1 and p·q = 0 (4D dot product)
    then p encodes the rotational part
    and q encodes the translational part
  - (nothing, otherwise)
- To roto-translate a point a with roto-trans b
  just “conjugate” their representations a’ ← b ∗ a ∗ $\overline{\text{b}}$

dual-quaternion conjugate: $\overline{\text{p}} - \varepsilon\,\overline{\text{q}}$

∗ : dual quaternion multiplication

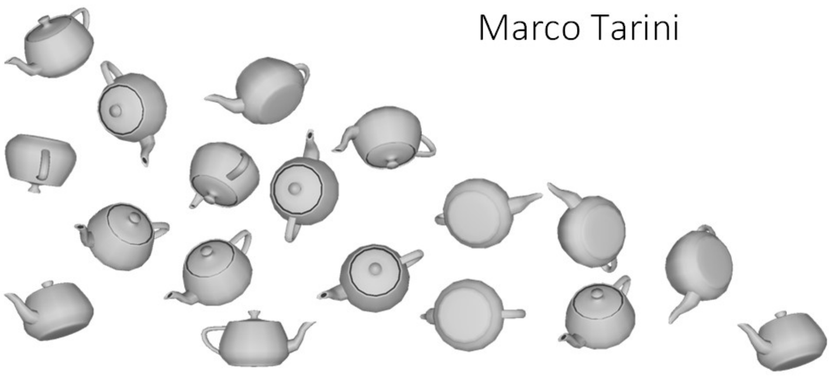
94

Marco Tarini
Unviersità degli studi di Milano

Master Game Dev

Rotations: exercises

Marco Tarini



95

Exercise: 2D rotations as 3D rotations

- A 2D rotation (of an angle α , around the origin) can be seen as the *restriction* of a 3D rotation in the X-Y plane (of an angle α , around the... Z axis!)
- Find this 3D rotation in *all* representations:
  - as... a 3x3 Matrix:
  - as... Axis-times-Angle:
  - as... Euler angles (Roll=Z, Pitch=X, Yaw=Y):
  - as... a quaternion:

96

Exercise:
2D rotations as 3D rotations

- A 2D rotation (of an angle α , around the origin) can be seen as the *restriction* of a 3D rotation in the X-Y plane (of an angle α , around the... Z axis!)
- Find this 3D rotation in *all* representations:
  - as... a 3x3 Matrix: $\begin{bmatrix} +\cos(\alpha) & -\sin(\alpha) & 0 \\ +\sin(\alpha) & +\cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  - as... Axis-times-Angle: $[0,0,\alpha]$
  - as... Euler angles (Roll=Z, Pitch=X, Yaw=Y): $[\alpha, 0,0]$
  - as... a quaternion: $\left[0, 0, \sin\left(\frac{\alpha}{2}\right), \cos\left(\frac{\alpha}{2}\right)\right]$

97

Exercises:
find the rotation that...

- For all the following exercises: we can pick any rotation representation! (unless otherwise specified)
  - As long as we have algorithms to translate one representation into another
  - Try to understand which one is the most convenient format, for a given task?

98

Exercise: find the «from-to» rotation

- Problem: given a pair of versors $\hat{v}$ and $\widehat{w}$, ($\hat{v}$ = *from* and $\widehat{w}$ = *to*) find the minimal rotation that brings $\hat{v}$ into $\widehat{w}$
  - useful problem in several contexts

minimal angle

e.g. AI aiming a bazooka

- A solution: as axis-and-angle
  - the axis *a* is found as $\hat{v} \times \widehat{w}$ (renormalizing it)
  - of the angle α , we know that the cosine is ($\hat{v} \cdot \widehat{w}$) and the sine is $|| \hat{v} \times \widehat{w} ||$. so $\alpha = \text{atan2}(|| \hat{v} \times \widehat{w} || \,,\, \hat{v} \cdot \widehat{w})$

99

Exercise: find the «look-at» rotation

- Given observer's position $\mathbf{e}$ and observed point $\mathbf{t}$ find the rotation (i.e., the orientation) for a character who must be looking in that direction
- That specification is incomplete: we also need another input: a «target up-vector» $\hat{\boldsymbol{u}}$
  - the character wants to keep its up-direction as similar as possible to $\hat{\boldsymbol{u}}$, while looking toward $\mathbf{t}$
  - Usually, the (world) up-vector, e.g. (in Unity) (0,1,0)
- Useful for… characters heads looking at something / facing toward something, setting up the camera…

101

Exercise: find the «look-at» rotation

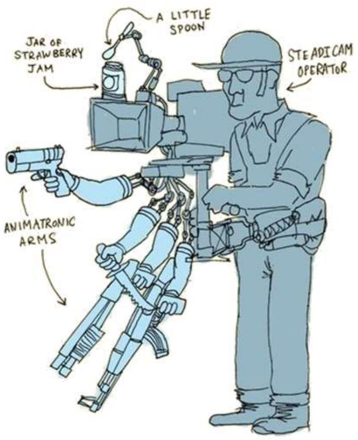
- Solution: as a 3×3 matrix
  - find the $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{z}}$ directions of this local character
  - they must be 3 reciprocally orthogonal versors
  - they are the columns of the sought matrix
- that is (assuming Unity conventional axis names):
  - $\hat{\boldsymbol{z}} = (\mathbf{t} - \mathbf{e}) / ||\mathbf{t} - \mathbf{e}||$
  - $\hat{\boldsymbol{y}} = \hat{\boldsymbol{u}}$? Wrong: it wouldn't be necessarily orthogonal with $\hat{\boldsymbol{z}}$
  - but, $\hat{\boldsymbol{x}} = \hat{\boldsymbol{u}} \times \hat{\boldsymbol{z}} \;/\; ||\, \hat{\boldsymbol{u}} \times \hat{\boldsymbol{z}} \,||$ (note the re-normalization) because the right direction is orthogonal to both $\hat{\boldsymbol{z}}$ and $\hat{\boldsymbol{u}}$
  - finally, $\hat{\boldsymbol{y}} = \hat{\boldsymbol{z}} \times \hat{\boldsymbol{x}}$

102

What about the "look-at" complete transform

- Setting up the complete transform of a camera (from the same data):
  - Camera position: is the translation component
  - the "look-at" rotation : is the rotation component
  - (scale component = 1)

In Computer Vision the set of these parameters are defined as the camera extrinsic parameters



"Camera-man in videogame logic"
unknown artist, circa 2010

104

Exercise: update the orientation of a rolling ball *

- A ball with radius r stands on a flat plane (with plane normal $\hat{\mathrm{n}}$), currently oriented with rotation R_0 and positioned (center position) in $\mathbf{p}_0$
- It then rolls in position $\mathbf{p}_1$ (staying on the plane)
- Find its new orientation R_1

Marble Madness, Atari, 1986

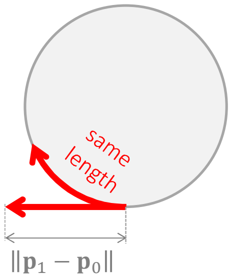
* a classic of many 3D games! Including early ones

105

Exercise: update the orientation of a rolling ball *

Solution (trace): as axis-angle…

- The axis must be:
  - parallel to the ground; therefore, orthogonal to $\hat{\mathrm{n}}$!
  - orthogonal to the direction of motion $(\mathbf{p}_1 - \mathbf{p}_0)$
  - (also, it must be expressed as a unit vector)
- The angle α must satisfy…
  full-circumference : length-of-arch = full-circle : α

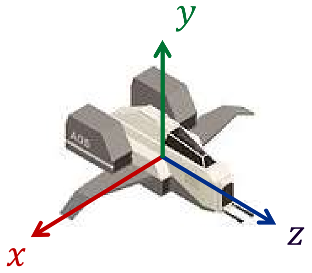


$2\,\pi\,r$ ↑ full-circumference; $\|\mathbf{p}_1 - \mathbf{p}_0\|$ ↑ length-of-arch; $2\,\pi$ radiants, or 360° ↑ full-circle

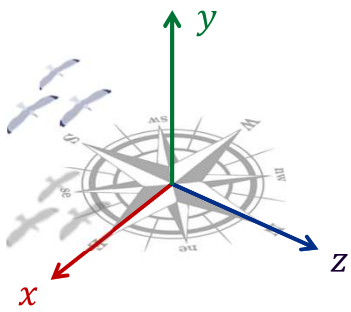
106

Exercise: find the orientation of a spaceship/airplane “character”

Local Space



Global Space



107

Exercise: find the orientation of a spaceship/airplane “character”

- Find the orientation $\mathrm{R_P}$ of an airplane at spawn time
  - The airplane is going NNE, and climbing up at 30° angle.
  - Its wings are parallel to the ground.
- Local space of airplane:
  - X-axis: left-right (the direction of the wings)
  - Y-axis: below to above
  - Z-axis: engine-to-propeller
- World space:
  - X-axis: west to east
  - Y-axis: ground to sky
  - Z-axis: south to north

NNE = halfway between North and NE

(which handedness is world and local spaces?)

108

Exercise: find the orientation of the head of the pilot of previous exercise

- The head of the pilot inside that plane is tilted 20° to the left, and 10° degrees above
- What it is its orientation $\mathrm{R_H}$?
- Local space of the head:
  - X-axis: left-eye to right-eye
  - Y-axis: chin to top of the head
  - Z-axis: view direction

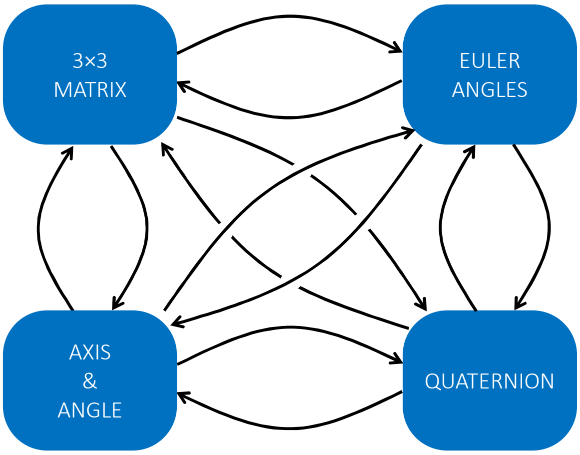
109

Exercise: find the angle of a turning head

- The pilot inside a plane is looking in direction $\hat{\mathrm{v}}$,
  - no tilt of the head: that is, the eye-to-eye vector is parallel to the ground
  - Axes : same as previous exercise
- What it is the orientation $\mathrm{R_H}$ of the head?
- Given that the plane is oriented as $\mathrm{R_P}$, what is the angle his neck is turning, with respect to the body?
  - Always assume you can turn any rotation representation into another

110

Converting rotation representations: Notes and exercises



Every representation is useful in some circumstance.

In a game engine or lib, conversions are useful.

Not all arrows need to be implemented!

Just ensure there is a path.

111

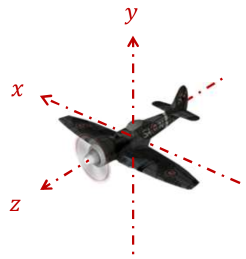
From: axis-&-angle To: quaternion, or viceversa

- Trivial exercise. Observation:
  - When going from an angle-based representation (*Euler angles*, *Axis-&-Angle*) to a non-angle-based representation (*Matrix*, *Quaternion*) you'll need trigonometric functions (sin , cos , ...)
  - When going from a non-angle-based representation (*Euler angles*, *Axis-&-Angle*) to an angle-based representation (*Matrix*, *quaternion*) you'll need inverse trigonometric functions (asin , acos , atan2) ← Remember this convenient one exists!

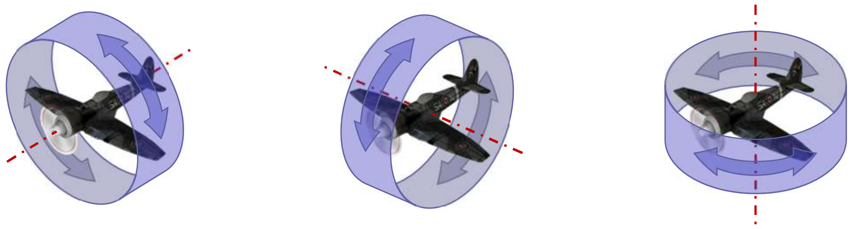
112

from: Euler angles to: 3x3 matrix

- Question:
  - Which matrix $\mathbf{R}$ does this?



Assume "Unity" axis definitions



Roll α (1st) | Pitch β (2nd) | Yaw γ (3rd)

113

from: Euler angles to: 3x3 matrix

← the order is prescribed by the choice of Euler Angles

$$R = R_y(\gamma) \cdot R_x(\beta) \cdot R_z(\alpha)$$

$$\begin{bmatrix} +\cos(\beta) & 0 & +\sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & +\cos(\beta) \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & +\cos(\alpha) & -\sin(\alpha) \\ 0 & +\sin(\alpha) & +\cos(\alpha) \end{bmatrix} \quad \begin{bmatrix} +\cos(\alpha) & -\sin(\alpha) & 0 \\ +\sin(\alpha) & +\cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

See: rotations in 2D

- What about the vice-versa?
  - a more difficult exercise
  - requires inverse trigonometric functions (of course)

114

from: axis-&-angle to: 3x3 matrix (exercise)

- Question:
  - Which matrix R rotates by α degrees around axis $\hat{a}$?
- Trace:
  1. Find a rotation matrix R_A mapping $\hat{a}$ the axis into the X axis (hint: find three orthogonal versors to use as columns of R_A , one of them being $\hat{a}$)
  2. Define a rotation matrix R_x rotating by α around X axis
  3. Then: $\mathrm{R} = \mathrm{R}_A^{-1} \cdot \mathrm{R}_x \cdot \mathrm{R}_A$ (understand why)

115

from: 3x3 matrix to axis-&-angle (exercise)

- Question:
  - Given a rotation matrix R , find axis $\hat{a}$ and rotation angle α
  - Assumption: R is actually a rotation matrix
- Trace:
  1. Obeservation: for the given matrix R , $\mathrm{R}\,\hat{a} = \hat{a}$ (why?)
  2. In other words, $\hat{a}$ is an eigenvector of R of eigenvalue 1
  3. Find α : remember atan2 exists

116