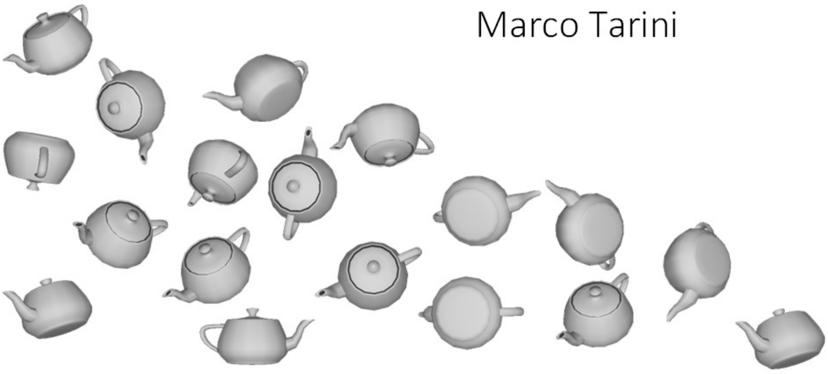
Master Game Dev

Rotations: exercises

Marco Tarini



74

Course Plan

- lec. 1: Introduction
- lec. 2: Mathematics for 3D Games
- lec. 3: Scene Graph
- lec. 4: Game 3D Physics +
- lec. 5: Game Particle Systems
- lec. 6: Game 3D Models
- lec. 7: Game Textures
- lec. 9: Game Materials
- lec. 8: Game 3D Animations
- lec. 10: Networking for 3D Games
- lec. 11: 3D Audio for 3D Games
- lec. 12: Rendering Techniques for 3D Games
- lec. 13: Artificial Intelligence for 3D Games

75

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:

76

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: $\left[0,0,\sin\left(\frac{\alpha}{2}\right),\cos\left(\frac{\alpha}{2}\right)\right]$

77

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?

78

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 $\widehat{w}$ ← minimal angle
  - useful problem in several contexts ← 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 $\alpha = \text{atan2}(\| \hat{v} \times \widehat{w} \| \,,\; \hat{v} \cdot \widehat{w})$

79

Marco Tarini
Unviersità degli studi di Milano

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…

80

Exercise: find the «look-at» rotation

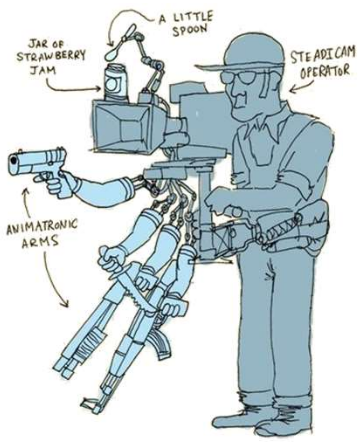
- Solution: as a 3×3 matrix
  - find the $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{z}}$ directions of this local character
  - they must be 3 reciprocally orthogonal versors
  - they are the columns of the sought matrix
- that is (assuming Unity conventional axis names):
  - $\hat{\boldsymbol{z}} = (\mathbf{t} - \mathbf{e}) / ||\mathbf{t} - \mathbf{e}||$
  - $\hat{\boldsymbol{y}} = \hat{\boldsymbol{u}}$? 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, $\hat{\boldsymbol{y}} = \hat{\boldsymbol{z}} \times \hat{\boldsymbol{x}}$

81

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

83

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

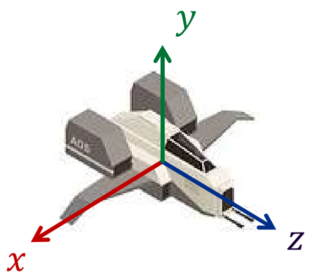
Marble Madness, Atari, 1986

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

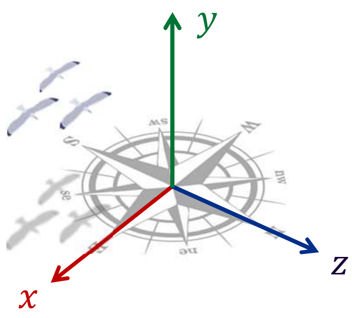
84

Exercise: find the orientation of a spaceship/airplane "character"

Local Space



Global Space



85

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?)

86

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 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

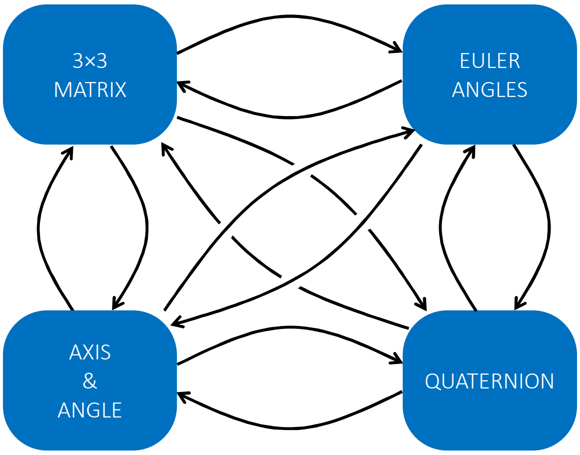
87

Exercise: find the angle of a turning head

- The pilot inside a plane is looking in direction $\hat{v}$,
  - 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 R_H of the head?
- Given that the plane is oriented as 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

88

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.

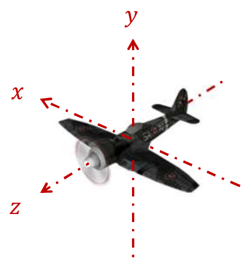
89

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!

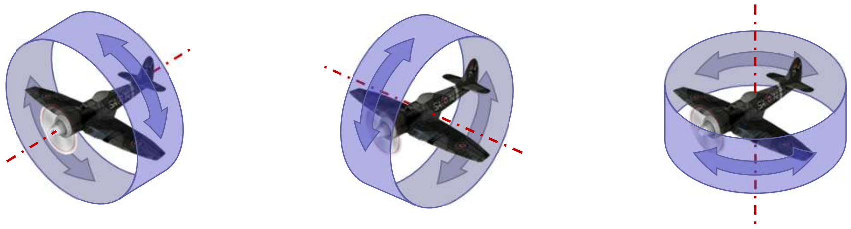
90

from: Euler angles
to: 3x3 matrix



Assume "Unity" axis definitions

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



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

91

from: Euler angles
to: 3x3 matrix

← the order is prescribed by the choice of Euler Angles

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

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

See: rotations in 2D

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

92

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

- Question:
  - Which matrix R rotates by α degrees around axis $\hat{\mathrm{a}}$?
- Trace:
  1. Find a rotation matrix R_A mapping $\hat{\mathrm{a}}$ the axis into the X axis (hint: find three orthogonal versors to use as columns of R_A , one of them being $\hat{\mathrm{a}}$)
  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)

93

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

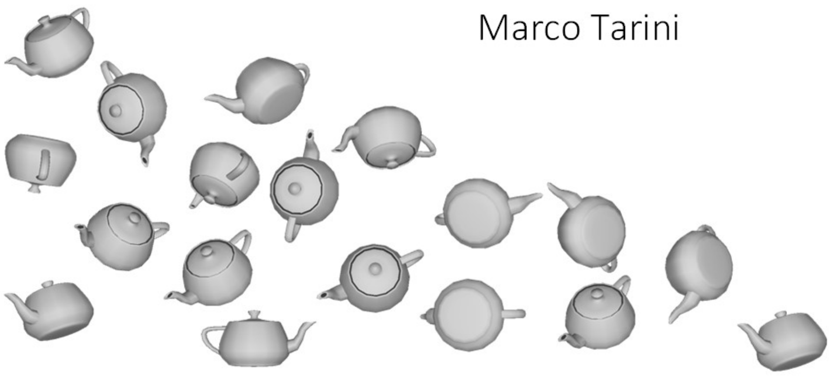
- Question:
  - Given a rotation matrix R , find axis $\hat{\mathrm{a}}$ and rotation angle α
  - Assumption: R is actually a rotation matrix
- Trace:
  1. Obeservation: for the given matrix R , $\mathrm{R}\,\hat{\mathrm{a}} = \hat{\mathrm{a}}$ (why?)
  2. In other words, $\hat{\mathrm{a}}$ is an eigenvector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

94

Master Game Dev

Transformations in games: final notes

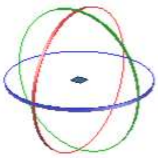
Marco Tarini

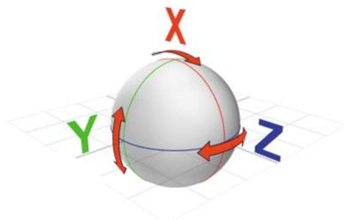


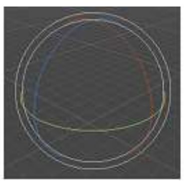
95

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)





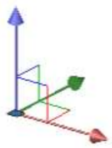
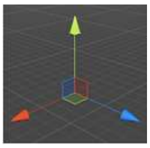
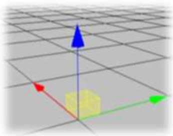


convention: Red = X Green = Y Blue = Z

96

GUI: how to author translations in 3D?

- translation gizmo
  - handles to traslate along axes or planes

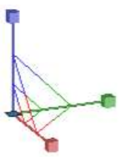
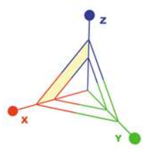
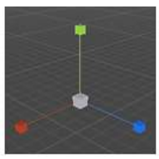




convention: Red = X Green = Y Blue = Z

97

GUI: how to author scalings in 3D?

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

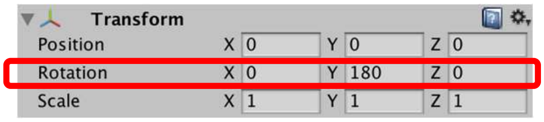




convention: Red = X Green = Y Blue = Z

98

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

99

Notes on Rotations in UNREAL ENGINE

fields: `W X Y Z`

Class **`FQuat`** :

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

- 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 *`

100

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 transform

101

A representations for roto-rotations (notes)

- So far, we assumed that the rotation and translation component of a transformations are stored separatedly
  - We have seen reasons why this is convenient
- There're mathematical representations which store rotation and translation (roto-translations, aka "rigid" transformations) jointly:
  - 4x4 matrices (we have seen them, their pros and cons)
  - Dual quaternsions

102

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)

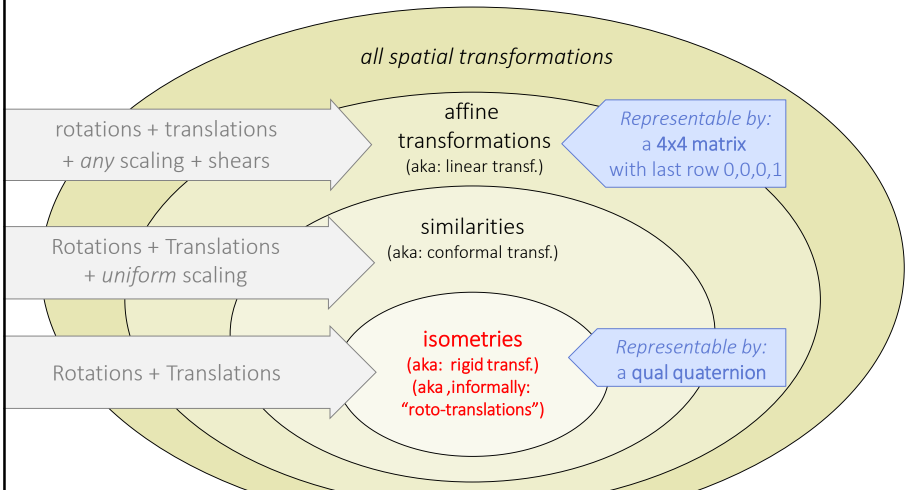
103

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
  - Usually, neither is what you have in mind in all cases
- Dual quaternions = a better* math abstraction to model roto-translations
  - * better interpolation of roto-translations

104

A math classification of spatial transformations



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

105

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 results in very good interpolation between 2 (or more) roto-translations
- They are used in animation techniques
  - See lecture about skeletal animations later

106

The math of Dual Quaternions in a nutshell 2/3

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

real part of p | imaginary part of p | real part of q | imaginary part of q

$$a + b\,i + c\,j + d\,k + e\,\varepsilon + f\,\varepsilon\,i + g\,\varepsilon\,j + h\,\varepsilon\,k$$

p + ε q

the “primal” quaternion | the “dual” quaternion

107

The math of Dual Quaternions in a nutshell 3/3

$a + b\,i + c\,j + d\,k$ | $e + f\,i + g\,j + h\,k$

- A dual quaternion p + ε q can represent:
  - 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: $\overline{p}$ - ε $\overline{q}$

dual quaternion multiplication

108