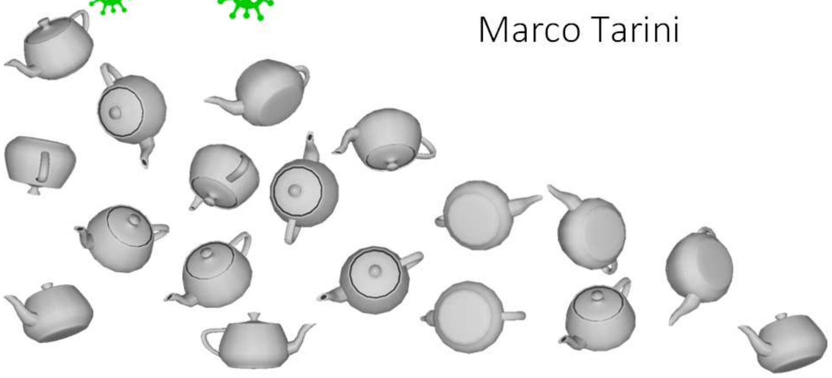
3D VideoGames

Representing Rotations 2/2

REMOTE TEACHING!

Marco Tarini



31

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. 8: Game 3D Animations
lec. 9: Game 3D Audio
lec. 10: Networking for 3D Games
lec. 11: Artificial Intelligence for 3D Games
lec. 12: Game 3D Rendering Techniques

32

Comparing representations (so far)

	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
Efficient / easy to: Invert (produce inverse)	★★★★★ just transpose	★☆☆☆☆
Efficient / easy to: Composite (with another rotation)	★★☆☆☆ Matrix multipl (9 dots) Numerical errors	★☆☆☆☆
Efficient / easy to: 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

33

Rotations as 3x3 matrices exercise: “look-at” rotation

- Given observer position A and observed point B
  - or, directly, a look direction v = (B – A) / || B – A ||

  find the rotation (i.e. the orientation)
  for a character who must be looking in that direction
- Incomplete specification!
  We also need in input: a «target up-vector» u
  - the character wants to keep its up-direction as similar as possible to u, while looking toward B
  - Usually, the (world) up-vector, e.g. (in Unity) (0,1,0)
- Very useful for characters looking at something / facing toward something

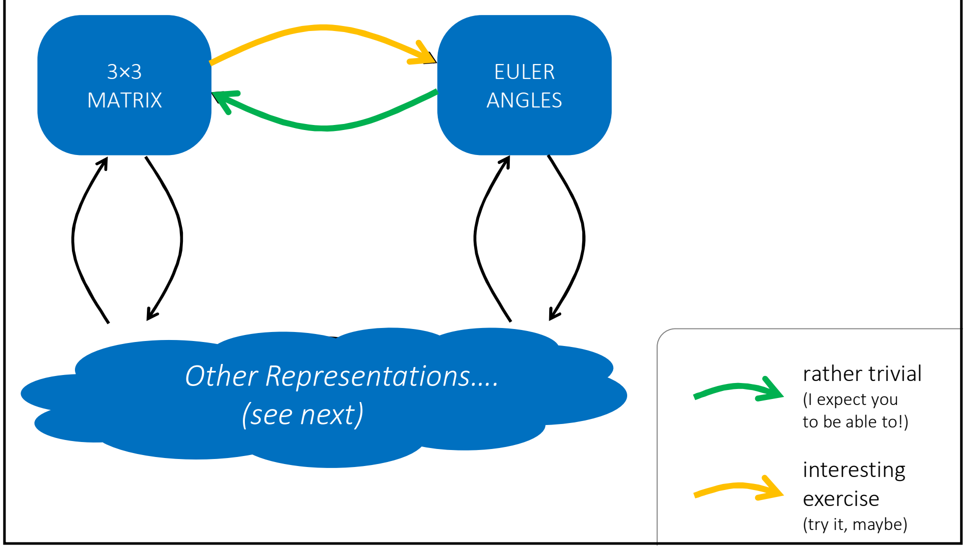
34

Rotations as 3x3 matrices exercise: “look-at” rotation

- Solution:
  - find the *x*, *y*, *z* directions of this local character
  - note: they must be 3 reciprocally orthogonal versors
  - make them the columns of the 3x3 rotation matrix
- for example (using Unity conventional axis names):
  - $z = v$ (easy! the forward direction is exactly v!)
  - $y = u$? NO! it wouldn’t be necessarily orthogonal with z
  - but, $x = u \times z \; / \; || \, u \times z \, ||$ (note the re-normalization)
    i.e. the right vector is orthogonal to both z and u
  - finally, $y = z \times x$

35

Switching between representations



36

from: euler angles
to: 3x3 matrix

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

- Easy to write down!

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

  - requires several sin / cos evaluations (and matrix mult)

- What about the vice-versa?
  - a medium-difficulty exercise
  - not very convenient:
    many inverse trigonometric functions

37

Representations of 3D rotations

- 3x3 matrices
- Euler Angles
- Axis + angle
  - Most common way in physics
    (and *game* physics)

38

Rotations as axis & angle

- Any rotation can be expressed as:
  - one rotation by some angle around some axis

must be appropriately chosen

- Angle: a scalar
- Axis: a versor (3 scalars)
  - note: the axis is considered to pass around the origin. For the more general case, combine with translations.

39

Rotations as axis & angle

- Compactness: good, 4 scalars
  - Just one more than bare minimum
- Ease of application: not too good ☹
  - Ways include: switch to 3x3 matrix (exercise: how to)
  - Switch to a quaternion: see later
  - “Rodrigues' rotation formula” (look it up)
  - Note they all require trigonometric function (sin, cos)
- Invert: super easy / quick
  - just flip the angle sign *or* the axis vector
  - question: what if both?
    answer: Rotation is inverted twice:
    it’s back to the same rotation again! 🤔

40

Rotations as axis & angle: equivalent representations

- Therefore: $(a_x , a_y , a_z , \alpha)$
  and $(-a_x, -a_y, -a_z , -\alpha)$
  represent the same rotation
- Any rotation has two equivalent representations in this format
  - except the identity, which has infinitely many: angle α = 0, with any axis $\hat{a}$ = (a_x , a_y , a_z)
- This is always a bit inconvenient
  - Complicates interpolation ("shortest path" necessary)
  - Complicates testing for equality/similarity, etc.

41

Rotations as axis & angle

- Compositing rotations:
  not at all immediate or easy to do ☹
- Interpolating rotations: very good!
  - Just interpolate axis and angle separately
  - Some *caveat*:
    - ⚠ 1) *shortest path* for axes: first, flip either rotation (both its axis & angle) when this makes the two axes closer (how to test?)
    - ⚠ 2) *shortest path* for angles: as usual, angles must then be interpolated... «modulo 360°»,
    - ⚠ 3) interpolate between axes requires SLERP or NLERP (when interpolating versors)
    - ⚠ 4) beware degenerate cases (opposite axes); point 1 avoids this
  - best results!
    Usually produces the "expected" intermediate rotation

42

Rotations as axis and angle, variant: as axis angle

- axis: $\hat{a}$ (versor, $|\hat{a}| = 1$)
- angle: α (scalar)
- can be represented as one vector $\vec{a}$ (3 scalars) ← «pseudo-vector»
  $\vec{a} = \alpha\,\hat{a}$
  - angle $\alpha = |\vec{a}|$
  - axis $\hat{a} = \vec{a} / \alpha$
  - note: when $\alpha = 0$, the axis is lost… it's ok, we don't need it!
- more compact, but fairly equivalent
  - actually, better: we now have only 1 representation per rotation (why?)
    … including the identity (why?)

43

Axis and angle - exercise the «from-to» rotation

- Problem: given a pair of versors $\hat{v}$ and $\hat{w}$,
  ($\hat{v}$ = *from* and $\hat{w}$ = *to*)
  find the minimal rotation ← *minimal angle*
  that brings $\hat{v}$ into $\hat{w}$
  - useful problem in several contexts ← *e.g. AI aiming a bazooka, a ball rolling…*
- Solution:
  - the axis a is found as $\hat{v} \times \hat{w}$ (renormalizing it)
  - of the angle α , we know that
    the cosine is ($\hat{v} \cdot \hat{w}$) and the sine is $\| \hat{v} \times \hat{w} \|$.
    so $\alpha = \text{atan2}(\| \hat{v} \times \hat{w} \| \;,\; \hat{v} \cdot \hat{w})$

45

Representations of 3D rotations

- 3x3 matrices
- Euler angles
- Axis + Angle
- Quaternions

46

A flashback: Complex Numbers in a nutshell 1/3

- It all starts with a «fantasy» assumption, which is: there is an imaginary number i such that $i^2 = -1$
  - And for any other purpose, i behaves just like a (non-zero) Real number
- Consequences:
  - We now have number of the form $a + b\ i$, with $a, b \in \mathbb{R}$, called complex numbers (the set is $\mathbb{C}$)
    - *real part* → a; *imaginary part* → b
  - The algebra of complex numbers (how to sum, multiply, invert them…) is simply determined by the «fantasy» assumption above

47

A flashback: Complex Numbers in a nutshell 2/3

- For example, sum: *(real part, imaginary part)*

$$(a + b\,i) + (c + d\,i) = (a + c) + (b + d)i$$

- For example, product (remembering $i^2 = -1$):

$$(a + b\,i) * (c + d\,i) = (ac - bd) + (ad + bc)i$$

- For example, inverse (check):

$$(a + b\,i)^{-1} = \frac{(a \; - b\,i)}{a^2 + b^2}$$

the «coniugate» of $(a + b\,i)$

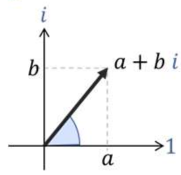
the squared «magnitude» of $(a + b\,i)$

- What is interesting to us is the geometric interpretation of these objects & operations

48

A flashback: Complex Numbers in a nutshell 3/3

- Geometric interpretation:
  - $a + b\,i$ represents the vector/point (a, b)
  - Complex sum = vector sum
  - Complex conjugate = mirroring with the Real axis (horizontal)
  - Product is = add angles (with Real axis), multiply magnitudes



- Therefore,
  - product with a unitary (magnitude = 1) complex number is a 2D rotation around origin
  - A complex number $\mathrm{r} \in \mathbb{C}$ with $\|\mathrm{r}\| = 1$ represents a 2D rot; multiply a vector $(x + y\,i)$ with r means to rotate it

Wouldn't it be nice to have the same for 3D rotations?

49

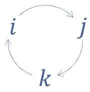
Quaternions

as a table:

×	i	j	k
i	-1	$+k$	$-j$
j	$-k$	-1	$+i$
k	$+j$	$-i$	-1

- New «fantasy» assumption: there are three different "imaginary" numbers i , j , k such that:

$$\begin{cases} i^2 = k^2 = j^2 = -1 \\ ij = k \ , \quad ji = -k \\ jk = i \ , \quad jk = -i \\ ki = j \ , \quad kj = -j \end{cases}$$



  - for any other purpose, i, j, k behave like real numbers
- Consequences:
  - We now have number of the form $a\ i + b\ j + c\ k + d$, with $a, b, c, d \in \mathbb{R}$, called Quaternions (their set is $\mathbb{H}$)
    (imaginary parts: $a\ i$, $b\ j$, $c\ k$; real part: d)
  - The algebra of quaternions (how to sum, multiply, invert them…) is simply determined by the «fantasy» assumption
  - Again, what is interesting to us is the geometric interpretation…

51

Quaternions: how to write them (equivalently)

- Algebraic form: $a\ i + b\ j + c\ k + d$
  - often, omitting the zeros, e.g. $i + 2\ k$ is a quaternion
- As vectors of $\mathbb{R}^4$: $(\ a\ ,\ b\ ,\ c\ ,\ d\ )$
- As vector & scalar pair: $(\ \vec{v}\ ,\ d)$

  imaginary part, a vector: $\vec{v} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$ — real part, a scalar: d

- Conjugate of a quaternion: invert the sign of the imaginary part

52

Quaternions: operations how-to

$q \in \mathbb{H}$ $\qquad$ $q = ai + bj + ck + d$

- Sum, Scale, Interpolate , etc.: trivial
  - same as 4D vectors
- Magnitude

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}$$
$$\|q\|^2 = a^2 + b^2 + c^2 + d^2$$

  - «unitary» if it's 1
  - same as 4D vectors

53

Quaternions: operations how-to

$q \in \mathbb{H}$ $\qquad$ $q = ai + bj + ck + d$

- Product: just apply «fantasy» assumptions
  - Observe: product is not commutative (nor anticommut.)
  - (see next 3 slides for the math)
- «Coniugate»:
  - like for complex numbers: $\bar{q} = -ai - bj - ck + d$

Flip imaginary parts

- Inverse: (like for complex numbers) $q^{-1} = \bar{q} \,/\, \|q\|^2$
  - For unitary quat, it's just the coniugate

54

Quaternions: Geometric Interpretation!

- A quaternion $\mathrm{q} = (\vec{\mathrm{v}}\ ,\ d\ )$ represents :
  - the 3D point or vector $\vec{\mathrm{v}}$, when $d = 0$
  - a 3D rotation, when q is unit, i.e. $\|\mathrm{q}\|^2 = \|\vec{\mathrm{v}}\|^2 + d^2 = 1$
  - (neither, otherwise)
- If q is a rotation and p is a point ($\mathrm{q}, \mathrm{p} \in \mathbb{H}$) then…
  - $\mathrm{q} \cdot \mathrm{p} \cdot \bar{\mathrm{q}}$ is the rotated point / vector
  - $\bar{\mathrm{q}}$ is the inverse rotation
  - $\mathrm{q}_0 \cdot \mathrm{q}_1$ is the composited rotation (first q_1 then q_0)
  - (so, $\bar{\mathrm{q}} \cdot \mathrm{p} \cdot \mathrm{q}$ is the pt rotated… in the *other* direction)

56

Compositing Quaternions: why it works

$\mathrm{q}_0 , \mathrm{q}_1 , \mathrm{p} \in \mathbb{H}$
$\mathrm{q}_0 , \mathrm{q}_1$ represent rotations
p represents a point

p rotated by q1, rotated by q0

p rotated by q1

$$\mathrm{q}_0 \cdot (\mathrm{q}_1 \cdot \mathrm{p} \cdot \bar{\mathrm{q}}_1) \cdot \bar{\mathrm{q}}_0$$

product is associative
(like for complex numbers)

$$= (\mathrm{q}_0 \cdot \mathrm{q}_1) \cdot \mathrm{p} \cdot (\bar{\mathrm{q}}_1 \cdot \bar{\mathrm{q}}_0)$$

$\bar{\mathrm{r}} \cdot \bar{\mathrm{s}} = \overline{\mathrm{s} \cdot \mathrm{r}}$
(rules of quaternions)
(remember: product is not commutative)

$$= (\mathrm{q}_0 \cdot \mathrm{q}_1) \cdot \mathrm{p} \cdot \overline{(\mathrm{q}_0 \cdot \mathrm{q}_1)}$$

57

3D Rotations as Quaternions

- quaternion $\mathbf{q}$ representing the 3D rotation of angle α around axis $\hat{a}$:
  - $q = \left(\sin\left(\frac{\alpha}{2}\right) \hat{a} \,,\ \cos\left(\frac{\alpha}{2}\right) \right)$

  that is
  - $q = \sin\left(\frac{\alpha}{2}\right) \hat{a}_x i + \sin\left(\frac{\alpha}{2}\right) \hat{a}_y j + \sin\left(\frac{\alpha}{2}\right) \hat{a}_z k + \cos\left(\frac{\alpha}{2}\right)$
- Observe that $\|q\|^2 = 1$

verify

58

Example: turn-around rotation

- Find the quaternion $\mathbf{r}$ representing the rotation by 180° (π radiants) around axis Y
  - $\hat{a} = (0,1,0)$
  - $\alpha = \pi \,, \sin\left(\frac{\alpha}{2}\right) = 1,\ \cos\left(\frac{\alpha}{2}\right) = 0$
  - $\mathbf{r} = (1\ \hat{a} \,, 0) = 0i + 1j + 0k + 0 = j$

imaginary vector — real scalar

- Find the quaternion $\mathbf{q}$ representiong point $\begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$
  - $\mathbf{q} = 2i + 3j + 4k$
- Rotate that point with that rotation.
  - $\mathbf{q}' = \ \mathbf{r}\,\mathbf{q}\,\bar{\mathbf{r}} = j\ (2i + 3j + 4k)(-j) = \ ...$ *(finish me!)*

59