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 (ie. the facing) for a character which must be facing toward this look.
- Incomplete specification!
  We also need: a target «up» vector u
  - the character wants to keep its up-direction as similar as possible to u, while looking toward B
- Very useful for characters looking at something / facing toward something

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, in Unity:
  - $z = v$ (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, $y = z \times x$

Representations of 3D rotations

- 3x3 matrices
- Euler angles
  - the most intuitive way to express a rotation
  - e.g., well understood by digital artists!

Way 2: Euler angles (3 floats)

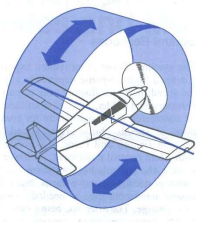
- Any 3D rotation can be expressed as:
  - a rotation around X axis (by α degrees), followed by:
  - a rotation around Y axis (by β degrees), followed by :
  - a rotation around X axis (by γ degrees):
- Angles α β γ :
  "Euler angles" of a specific
  - (therefore: its "coordinates")

this order (X-Y-Z) is chosen arbitrary but once and for all! (in a given game engine / lib)

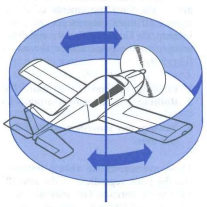
Way 2: Euler angles (3 floats)

- In nautical / aeronautical language, the three angles have names:

roll
(*rollio*)



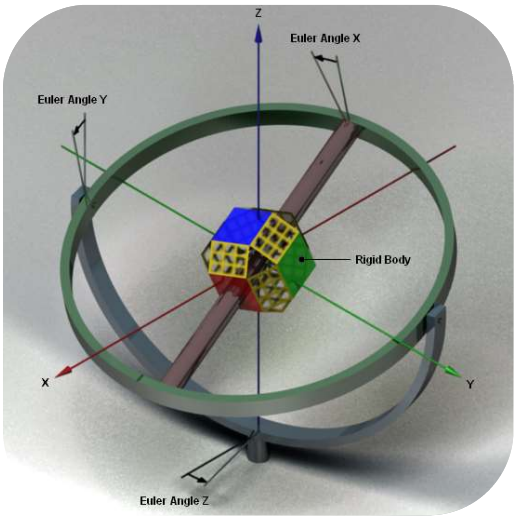
pitch
(beccheggio)



yaw
(imbardata)

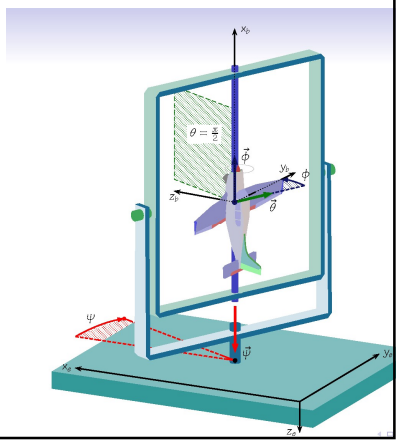
Way 2: Euler angles (3 floats)

- A physical implementation: “three axes globe”



Way 2: Euler angles (3 floats)

- Is it 1:1 ?
  - 1 rotation ⇔ 1 euler angle triplet ?
- Almost
  - assuming angles are properly bounded (exercise: how?)
- Ugly exception: "GIMBAL LOCK"
  - when 1st rotation makes the axes of the next two axes *coincide*
  - this cannot be disallowed, no matter how axes are chosen



Way 2: Euler angles (3 floats)

- Conciseness: perfect!
- Application : a bit fatiguing
  - (three rotations in succession)
- Interpolation : you can do that…
  - just interpolate the three angles

  ⚠ (remember to pay attention whenever interpolating angles: take in account $\alpha = \alpha + 360\,(k)$ equivalence)

  …but results won't always be intuitive / nice !
- Cumulate / invert: not easy nor immediate…

exercise: why just summing / flipping the three angles **won't work**?

from: euler angles to: 3x3 matrix

- Easy to write down!

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

  - but requires several sin / cos evaluations

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

Recap: representations of 3D rot. 1/2

	3x3 Matrix	Euler Angles
Space efficient? (in RAM, GPU, storage...)	9 scalars	3 scalars (even small int!)
Efficient / easy to: Apply (to points/vectors)	9 products (3 dot products)	trigonometry sin/cos
Efficient / easy to: Invert (produce inverse)	just transpose	
Efficient / easy to: Cumulate (with another rotation)	matrix multiplication (9 dots)	
Efficient / easy to: Interpolate (with another rotation)		easy to do... unintuitive result
Intuitive? (e.g. to manually set)	?!?	roll & yaw & pitch
Notes...	Free extra skew + scale!	GIMBAL LOCK

Reppresentazioni principali delle rotazioni

- Matrici 3x3
- Angoli di Eulero
- Asse + angolo
  - il metodo comunemente usato per es in fisica

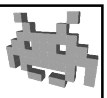
Modo 3: asse e angolo

- Qualunque rotazione data può essere espressa come:
  - una rotazione di un angolo attorno ad un asse

opportunamente scelti

- Angolo: uno scalare (1 float)
- Asse: un versore (3 float)
  - nota: l'asse passa per l'origine
    Per il caso opposto, usare anche traslazioni.

Modo 3: asse e angolo



- Compattezza:
  abb buono: 4 float
- Efficienza di applicazione: maluccio ☹
  - modo migliore: passare a matrice 3x3 (come?)
    (o a quaternione: vedi poi)
- Invertire: facilissimo
  - (ribaltare angolo *oppure* asse)
  - nota: se si invertono entrambi?
    stessa rotazione!

Modo 3: asse e angolo

- Nota:
  (ax, ay, az, alpha)
  (-ax, -ay, -az, -alpha)
  sono la stessa rotazione!
- → Ogni rotazione ha
  due rappresentaz equivalenti
  come asse e angolo!
  - (eccetto l'identità, che ne ha infinite:
    alpha = 0, asse qualunque)

Modo 3: asse e angolo

- Cumulare:
  nient'affatto immediato ☹
- Interpolare: ottimo!
  - idea: interpolare asse, intrerpolare angolo
  - Alcuni semplici caveat:
    - 1) bisogna prima flippare uno dei due (asse,angolo) se questo avvicina i due assi fra loro
    - 2) angolo va interpolato… «modulo 360°» (again!)
    - 3) l'asse va rinormalizzato post interpolazoine (come tutti i versori)
    - 4) occhio ai casi degeneri (assi opposti)
  - best results! la rotazione giusta
    - eccetto per: velocità di rotaz

Modo 3: asse e angolo: variante

- asse: v (vett normale, $|v| = 1$)
- angolo: α (scalare)
- rappresentarli internamente
  come 1 solo vett: v' (3 float in tutto)
  $v' = \alpha\, v$
  - angolo $\alpha = |v'|$
  - asse $v = v' / |v'|$
  - (nota: se angolo = 0, asse si perde… infatti non conta)
- più coinciso, ma per il resto equivalente
  - anzi, meglio: una sola rappresentaz per ogni rotaz (perchè?)
    … compresa l'identità (perchè?)

da: asse e angolo
a: matrice 3x3

- esercizio!

Axis and angole - esercize
«from-to» rotation

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

Reppresentazioni principali delle rotazioni

- Matrici 3x3
- Angoli di Eulero
- Asse + angolo
- Quaternioni

Ripasso: numeri complessi

Assunzione "fantasiosa":

c'è un i t.c.

$$i^2 = -1$$

- *Conseguenze:*
  - "Num complesso": $(a + b\,i)$
    - *interpretaz geom:* punti 2D $(a\,,\,b)$
  - Moltiplicaz fra complessi: ...
    - *interpretaz geom:* ...
  - Dunque:
    - moltiplicare per numero complesso (a norma 1) → ruotare in 2D (attorno all'origine)
    - numeri complessi (a norma 1) → rappresentaz rotazioni in 2D

Passare ai quaternioni

Assunzione "fantasiosa":

ci sono i, j, k t.c.

x	i	j	k
i	-1	$+k$	$-j$
j	$-k$	-1	$+i$
k	$+j$	$-i$	-1

cioè:

$i^2 = j^2 = k^2 = -1$
$ij = k \quad ji = -k$
$jk = i \quad kj = -i$
$ki = j \quad ik = -j$

- *Conseguenze:*
  - "Quaternione": $(a\,i + b\,j + c\,k + d)$
    - *interpr. geom:* punti 3D (a,b,c), quando $d=0$
  - Molitplicare due quat: ...
  - Invertire un quat: ...
  - Coniungare due quat q e p (fare $q\,p\,\bar{q}$): ...
    - *interpretaz geom:* ruotare p con la rotaz def da q
  - Dunque:
    - coniugare con un quat (con norma 1) → ruotare in 3D (asse pass. x ori)
    - quaternioni (a norma 1) → rappresentaz rotazioni in 3D

Da asse+angolo a quaternione

- rotaz: di α attorno all'asse (a_x, a_y, a_z)
- quaternione:

  $q = s\,a_x\,i + s\,a_y\,j + s\,a_z\,k + c$

  con

  $c = \cos(\alpha / 2)$

  $s = \sin(\alpha / 2)$

  vett. unitario

- cioè $q = (s\,a_x, s\,a_y, s\,a_z, c)$
- nota: $||q|| = 1$

  verificare!

Modo 4: "quaternioni" (4 float)

- Applicazione: facillimo ;)
  - ruotare $\mathbf{p} = x\,i + y\,j + z\,k$
    $\mathbf{p}$ ruotato = $\mathbf{q} \cdot \mathbf{p} \cdot \bar{\mathbf{q}}$
  - 2 moltiplicaz quat.
- Cumulare: facillimo ;)
  - 1 moltiplicaz quat
- Invertire: facillimo ;)
  - flippare la parte reale *oppure* quella immaginaria
  - se entrambe: rimane la stessa rotaz!

quat che rappresenta il punto di coord (x , y , z)
Nota: parte reale = 0

Modo 4: "quaternioni" (4 float)

- Interpolare: facillimo ;) …e good results!
  - simili caveats di asse e angolo:
    - ⚠ 1) flippare un quaternione prima, se accorcia la distanza fra i due
    - ⚠ 2) ri-normalizzare il quaternione dopo
  - velocità: non corretta
    - c'è modo di correggerla
    - vedi: SLERP vs LERP