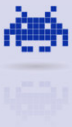
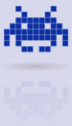
Reppresentazioni principali delle rotazioni



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

Modo 3: asse e angolo

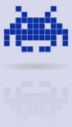


- Qualunque rotazione (*) data può essere espressa come:
  - una rotazione (di un **angolo**) attorno ad un **asse**

opportunamente scelti

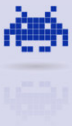
- **Angolo:** uno scalare (1 float)
- **Asse:** un vettore unitario (3 float)
  - (asse passante per l'origine)

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!

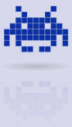
Modo 3: asse e angolo



- 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

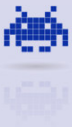


- Cumulare:
  nient'affatto immediato ☹
- Interpolare: ottimo!
  - idea: interpolare asse, intrerpolare angolo
  - Alcuni semplici caveat:
    - ⚠ 1) bisogna prima flippare uno dei due (asse,angolo) se questo avvicina i due assi fra loro
    - ⚠ 2) angolo va interpolato… «modulo 360°» (again!)
    -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

Modo 3: asse e angolo: variante

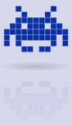
- asse: v (vett normale, $|v| = 1$)
- angolo: α (scalare)
- rappresentarli internamente
  come 1 solo vett: v' (3 float in tutto)
  $v' = \alpha\, v$
  - angolo $\alpha = |v'|$
  - asse $v = v' / |v'|$
  - (nota: se angolo = 0, asse si perde… infatti non conta)
- Più coinciso, ma per il resto equivalente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

da: asse e angolo
a: matrice 3x3



- esercizio!

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.

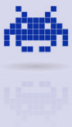
x	i	j	k
i	-1	$+k$	$-j$
j	$-k$	-1	$+i$
k	$+j$	$-i$	-1

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)

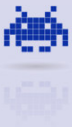
- Applicazione: facillimo ;)
  - ruotare $\mathbf{p} = x\,i + y\,j + z\,k$

    $\mathbf{p}$ ruotato $= \mathbf{q} \cdot \mathbf{p} \cdot \overline{\mathbf{q}}$

    quat che rappresenta il punto di coord (x, y, z)

    Nota: parte reale = 0
  - 2 moltiplicaz quat.
- Cumulare: facillimo ;)
  - 1 moltiplicaz quat
- Invertire: facillimo ;)
  - flippare la parte reale *oppure* quella immaginaria
  - se entrambe: rimane la stessa rotaz!

Modo 4: "quaternioni" (4 float)



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

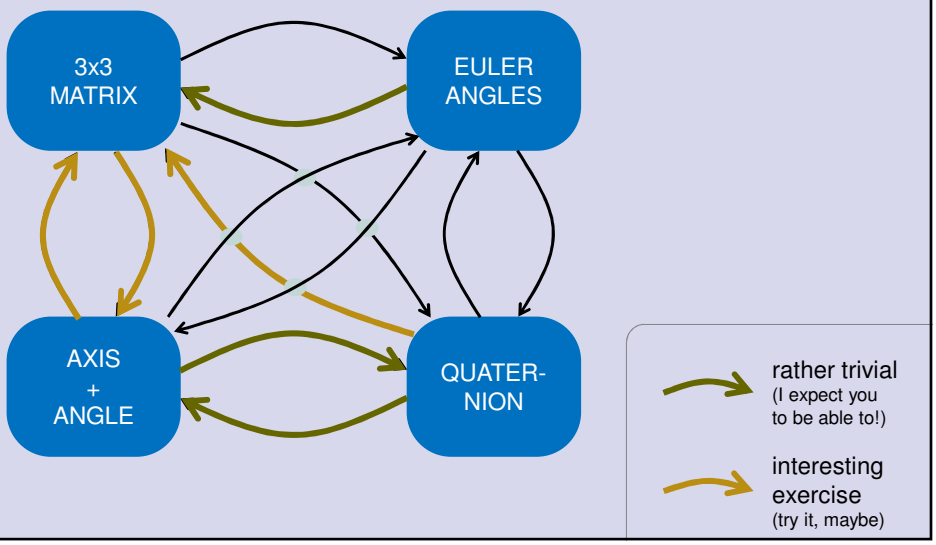
Recap: rappresentaz di rotaz. 2/2

	axis + angle	(unitary) quaternion
Space efficient? (in RAM, GPU, storage…)	4 scalars (or 3) (but precision needed)	4 scalars (but precision needed)
Efficient/easy to: Apply (o points/vectors)	to matrix? + trigonometry	easy: 2 prodotti quat
Efficient/easy to: nvert (produce inverse)	super easy flip di asse o angolo	super easy flip di parte reale o imm.
Efficient/easy to: Cumulate (with another rotation)		super easy: 1 prodotto quat
Efficient/easy to: nterpolate (with another rotation)	easy (best results!)	easy + good result (except speed)
Intutive? (e.g. to manually set)	sometimes	not really
Notes…	*two* representations for each rotation	

And the winner is…

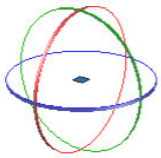
- Ovviamente, i quaternioni,
  - perché sono più efficienti su tutte le op
- Ovviamente, gli angoli di Eulero
  - perché sono più intuitivi (e pure più compatti)
- Ovviamente, asse+angolo
  - perché hanno l'interpolazione più naturale (in molti casi, speed giusta di default)
- Ovviamente, le matrici 3x3
  - perché sono lo standard in graphics
  - perché non solo rot ma qualsiasi affine (eccetto la traslaz, ma tanto è storata separatamente)

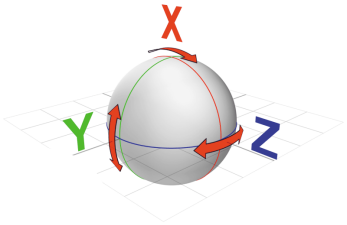
Switching between representations

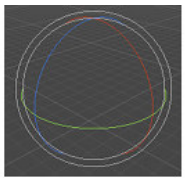


GUI: come *specifica* un utente le rotazioni in 3D?

- Metodo ricorrente: **rotation gizmo**
  - (a volte: «arcball» o «trackball»)
  - tre handles per controllare i tre angoli di Eulero
  - o "free", drag-n-drop "intuitivo" (metafora trackball)







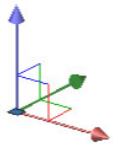
convenzione: Rosso = X Verde = Y Blu = Z

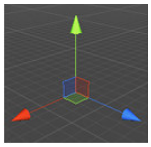
GUI: come *specifica* un untente le traslazioni in 3D?

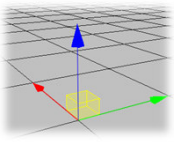
- "free": drag-and-drop 2D in spazio immagine, ← es: x spostare un oggetto "qui, in questo punto dello schermo"
  - mantendo distanza da osservatore

oppure

- **translation gizmo**
  - orientato nello spazio mondo, oppure ← es: x spostare un oggetto "in basso"
  - orientato nello spazio oggetto
  - handles per trasl lungo assi e/o piani ← es: x spostare un oggetto lungo il "suo" basso



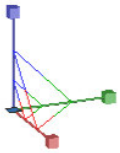


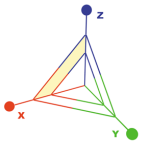


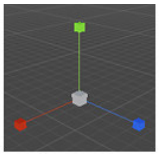
convenzione: Rosso = X Verde = Y Blu = Z

GUI: come *specifica* un untente le scalature in 3D?

- **scaling gizmo**
  - (tipic. orientato in spazio oggetto)
  - tre handles per le scalature anisotropichje
    + un handle centrale per scalature uniformi







convenzione: Rosso = X Verde = Y Blu = Z

Rotazioni in unity

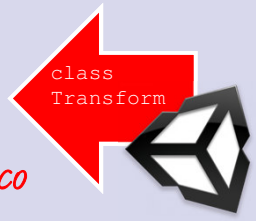
- Nella GUI del game tools:
  - Euler Angles
- Internamente:
  - Quaternions
- Nell'interfaccia degli scripts:
  - a scelta, (quat, euler, axis+angle…)
    con metodi setter/getter

Trasformazioni in unity

- Lineari (trasformaz. affini)

$$f(\alpha v_0 + \beta v_1) = \alpha f(v_0) + \beta f(v_1)$$

- Similitudini (trasformaz. conformali)
  - "Mantengono gli angoli"
  - Rotaz + Traslaz + Scaling ~~uniforme~~ anisotropico
- Isometrie (rototraslazione)
  - "Mantengono la magnitudine"
  - Rotaz + Traslaz



Rotazioni in OpenGL

- Nelle API «old school»: glRotate3f
  - Asse e angolo
- Internamente:
  - Matrici
  - (come tutte le altre trasformazioni spaziali)

Reppresentare
~~rotazioni~~ roto-traslazioni

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

} + Traslazione (displ. vec)

- Matrici 4x4 (o 3x4)
- Dual Quaternions