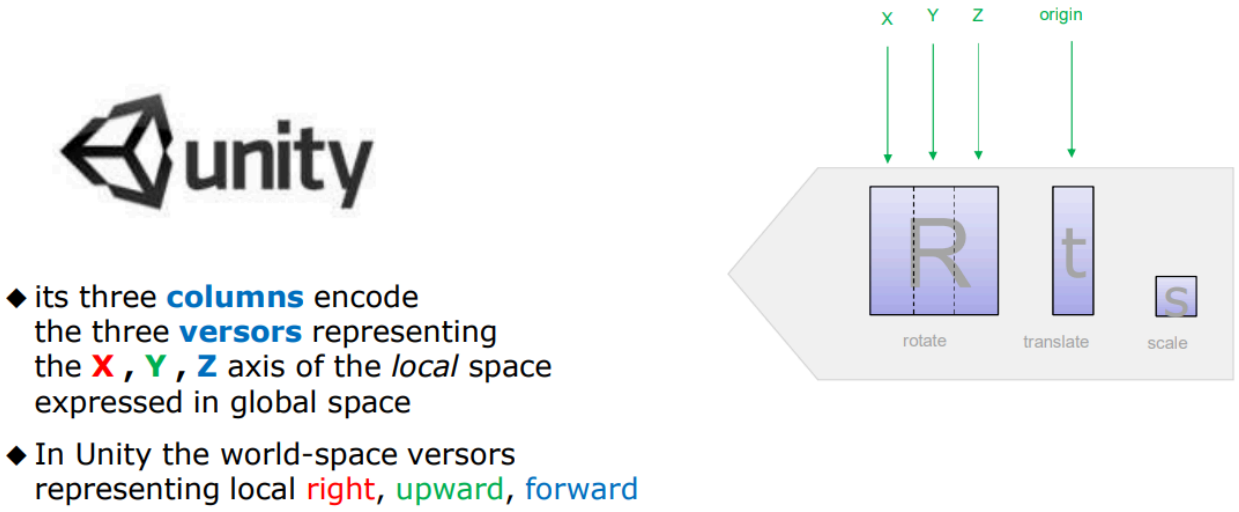
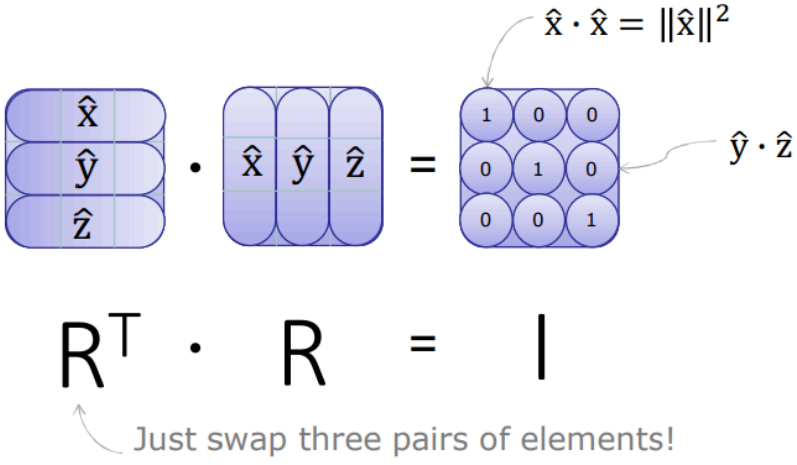
Lezione 4 19/03/2025

Poi tutte le rotazioni rappresentate come matrici hanno per inversa la matrice trasposta.





Angoli di eulero

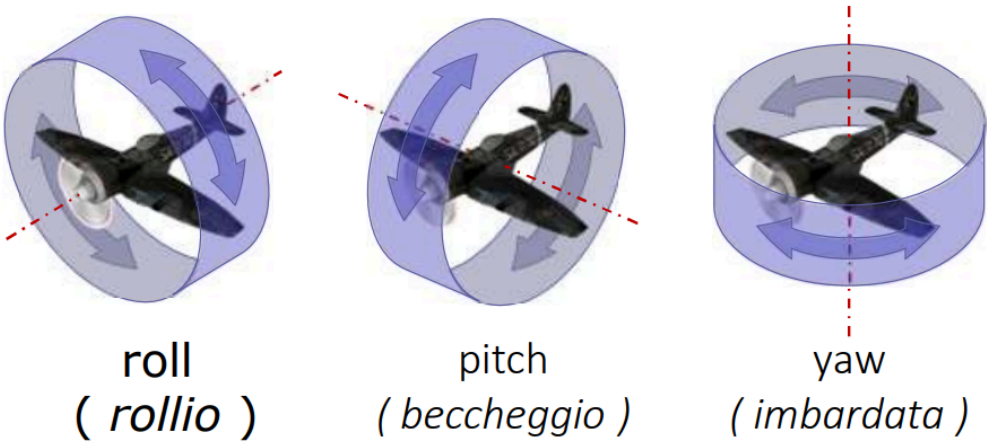
◆ Euler angles
- the most intuitive way to express a rotation
- e.g., well understood by digital artists!

◆ 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 Z axis (by γ degrees):

◆ Angles α β γ : "Euler angles" of a specific rotation
- therefore: the "coordinates" of that rotation

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

Possiamo rappresentare una rotazione 3D come una rotazione in termini di **angoli su ciascuno dei 3 assi**.



In questo modo dobbiamo salvare solamente 3 scalari. Applicarle all'oggetto però è poco comodo rispetto alle matrici, dobbiamo applicare una rotazione 3 volte (su ogni asse). L'interpolazione si può fare. Fare composizioni e inversioni non è facile, bisogna stare attenti all'ordine.

Angoli + assi

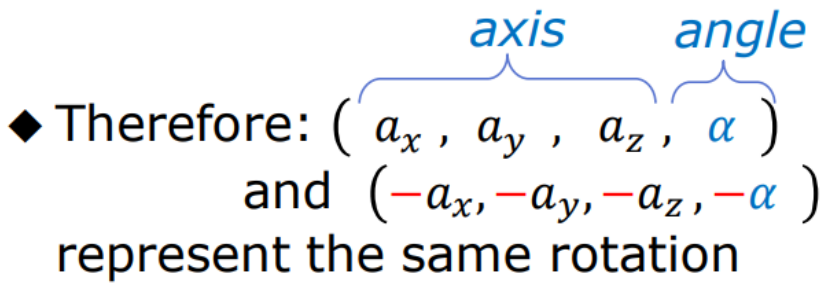
Questo è il metodo che viene usato in fisica (anche fisica di gioco).

Se sono nello spazio 3D, ogni rotazione può essere espressa come una **rotazione lungo un angolo**, quindi ci serve **l'angolo** (scalare) e il versore che rappresenta

l'asse. Quindi abbiamo uno scalare (angolo) e tre scalari (versori).

Per fare questo l'asse deve essere un vettore che passa per l'origine.

C'è un problema. Se consideriamo il senso antiorario (regola della mano destra, pollice nel senso del vettore, il senso in cui si chiude la mano è il senso in cui applicare la rotazione. Se prendiamo alpha e un angolo o -alpha e -angolo abbiamo la stessa trasformazione. Quindi il problema è che non è 1 a 1 la rappresentazione di una rotazione, ogni rotazione ha 2 rappresentazioni. L'identità ha infinite rappresentazioni.



Poi è complicato fare interpolazione perchè abbiamo di nuovo il problema dello shortest path.

è **difficile confrontarle** tra di loro perchè se cambio asse devo prima riuscire a trasformare l'informazione tra che differenza c'è tra gli assi, non posso confrontare gli angoli.

Comporre rotazioni non è semplice.

Per la concatenazione abbiamo un metodo che può portare a buoni risultati:

- 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

Funziona molto bene, il risultato è quello atteso, però dobbiamo fare interpolazione praticamente 2 volte.

Con un trucco si possono usare 3 valori al posto di 4, perchè posso prendere un vettore e far diventare il modulo di quel vettore lo scalare dell'angolo, perchè tanto l'asse è sempre un versore. Quindi tolgo il peso di uno scalare nella rappresentazione a 4 valori, e quando lo voglio sapere vado a calcolare il modulo del versore. Il modulo mi da l'angolo e il versore mi da l'asse.

- axis: $\hat{a}$ (versor, $\|\hat{a}\| = 1$)
- angle: α (scalar)
- can be represented as one vector $\vec{a}$ (3 scalars)
  $\vec{a} = \alpha \hat{a}$
  - 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 otherwise equivalent
  - actually, better:
    we now have only 1 representation per rotation (why?)
    … including the identity (why?)

Sometimes called «pseudo-vector» because it flips sign if the world is mirrored

Numeri complessi

- 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 any (non-zero) real number
- Consequences:
  - We now have number of the form $a+b\,i$, (real part: a; imaginary part: b)
    with $a,b\in\mathbb{R}$, called complex numbers (the set is called $\mathbb{C}$)
  - The algebra of complex numbers (how to sum, multiply, invert them…) is simply determined by the «fantasy» assumption above

Definiamo un'algebra che sfrutta il fatto che i numeri complessi hanno una parte reale e una immaginaria, rappresentata con la lettera i.

Questa i ha la proprietà che al quadrato fa -1, cosa che i numeri reali non fanno.

- For example, sum:

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

(real part: $a+c$; imaginary part: $b+d$)

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

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

- For example, inverse (verify it!):

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

($a - b\,i$: *the «coniugate» of* $(a + b\,i)$; $a^2 + b^2$: *the squared «magnitude» of* $(a + b\,i)$)

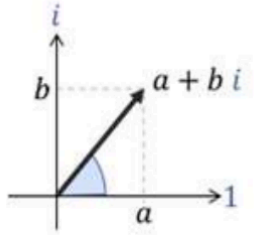
- What is interesting to us is the **geometric interpretation** of these objects & operations

La parte reale si somma con la parte reale, e quella immaginaria con quella immaginaria.

Il prodotto è più complicato, teniamo il risultato reale da una parte e quello moltiplicato per i dall'altra.

L'inversa è quella scritta, andrebbe dimostrata. Il coniugato del numero complesso è lo stesso numero, con il segno tra la parte reale e quella complessa invertito. La magnitudine (valore assoluto?) è la norma dei coefficienti della parte reale e di quella immaginaria.

- ◆ Geometric interpretation:
  - $a+b\,i$ represents the vector/point (a,b)
  - Complex sum = vector sum
  - Conjugate = mirroring with the imaginary axis (vertical)
  - Product = 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 $r\in\mathbb{C}$ with $\|r\|=1$ represents a 2D rot; multiply a vector $(x+y\,i)$ with r means to rotate it



I numeri complessi si possono rappresentare sul piano cartesiano, mettiamo quelli reali sulla x e quelli immaginari sulla y, quindi rappresentiamo un numero complesso come un **punto sul piano**.

Quando facciamo il la **somma** in questo spazio, stiamo facendo la somma tra vettori.

Il **coniugato**, che è di fatto l'inversione del segno della y, riflette il punto rispetto all'asse delle x.

Il **prodotto** addiziona gli angoli e moltiplica i valori assoluti, ovvero la lunghezza dei vettori.

Quindi operazioni che erano molto costose con i vettori normali diventano semplici con i vettori complessi.

I numeri complessi rappresentano i **vettori nel piano cartesiano** ma rappresentano anche le **rotazioni nel piano 2D**.

Possiamo estendere questa cosa al 3D con i quaternioni.

Quaternioni

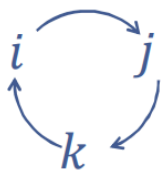
I quaternioni sono una versione più complicata dei numeri complessi.

Sui numeri complessi avevamo solo la regola i^2 = -1, nei quaternioni abbiamo **3 differenti parti immaginarie i j k** che si comportano in questo modo:

$$i^2 = k^2 = j^2 = -1$$

$$ij = k \qquad ji = -k$$
$$jk = i \qquad jk = -i$$
$$ki = j \qquad kj = -j$$

to remember it:



(errore: la penultima dovrebbe essere jk = -i e l'ultima dovrebbe essere ik = -j)

Tutti e 3 i numeri immaginari quadrati fanno -1.

Poi abbiamo altre regole, che dicono cosa succede quando li moltiplichi tra di loro. Se l'ordine non è corretto, viene il terzo ma negativo.

Per tutto il resto, i j k si comportano come numeri reali.

- 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 *real part*

- 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 a pair (vector, scalar): $(\vec{v}\ ,\ d)$

the imaginary part, a vector $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ *the real part, a scalar*

Per quanto riguarda le operazioni, basta considerarli come vettori 4D:

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

◆ **Sum**, **Scale**, **Interpolate** , etc.:
- Trivial

Consider them as 4D versors

◆ **Magnitude**

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

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

- «unitary» if $\|\mathrm{q}\|$ is 1

◆ **Product**: just apply the «fantasy» assumptions
- Observe: product is not commutative (nor anticommut.)
- (see later slides to see what the product turns out to be)

◆ «**Conjugate**»:
- like for complex numbers:

Flip the imaginary parts

$$\bar{\mathrm{q}} = -ai - bj - ck + d$$

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

Se abbiamo un **quaternione unitario** (magnitudine = 1), l'inverso è il suo coniugato.

Noi li rappresentiamo come un vettore 3D e uno scalare perchè ci è molto comodo.

- A quaternion $\mathrm{q} = (\vec{\mathrm{v}}, d)$ represents :
  - the **3D point / vector / versor** $\vec{\mathrm{v}}$, when $d = 0$
  - a **3D rotation**, when q is unit, i.e. $\|\mathrm{q}\|^2 = \|\vec{\mathrm{v}}\|^2 + d^2 = 1$
  - neither, otherwise

I quaternioni che hanno solo parte immaginaria sono **vettori nello spazio 3D**. Quelli che hanno solo parte immaginaria e hanno magnitudine = 1 sono **versori nello spazio 3D**.

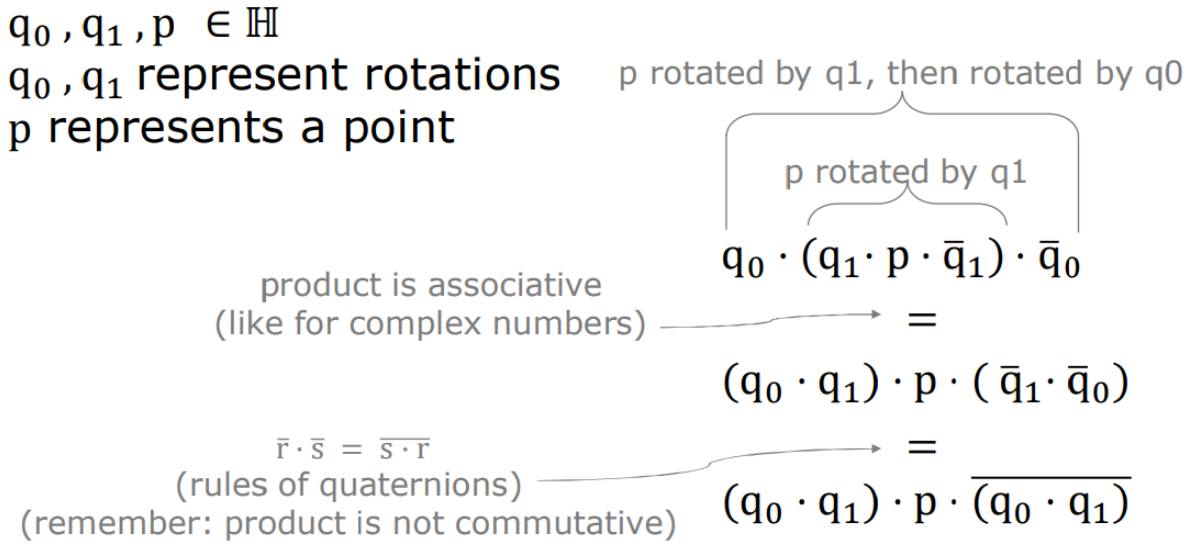
Una **rotazione** è rappresentabile come un **quaternione unitario**.

- 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
  - (so, $\bar{\mathrm{q}} \cdot \mathrm{p} \cdot \mathrm{q}$ is point p rotated… in the *other* direction)
  - $\mathrm{q}_0 \cdot \mathrm{q}_1$ is the composited rotation (first q_1 then q_0)

Se q è una rotazione e p è un punto, allora entrambi possono essere rappresentati come quaternioni. Il punto avrà parte reale = 0, la rotazione avrà magnitudine = 1.

Quindi se prendiamo q e p, possiamo **applicare la rotazione** al punto usando l'equazione q * p * coniugato di q. **Il coniugato di q è la rotazione inversa**, stiamo solo cambiando il segno delle parti immaginarie.

Composizione:



Sopra abbiamo il punto p a cui applichiamo la rotazione q1, al risultato di questo applichiamo la rotazione di q0. Siccome vale la proprietà associativa, possiamo raggruppare q0 * q1 e i due coniugati. **Il prodotto dei coniugati è il coniugato del prodotto dei due componenti invertiti** (la riga sotto). In questo modo la composizione è facilissima, e rimane una rotazione.

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

  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 $\|q\|^2 = 1$

La **rotazione di un angolo alpha rispetto ad un asse a** (versore in 3D), cioè fissiamo un asse (rappresentato da un versore) e un angolo alpha (scalare), allora il quaternione è dato dalla formula sopra. La parte reale è il coseno.

La norma di questo quaternione è sempre 1, quindi i quaternioni sono sempre unitari.

◆ Around axis $\hat{a}$ by angle α :

$$q = \left(\sin\left(\frac{\alpha}{2}\right)\hat{a}\,, \cos\left(\frac{\alpha}{2}\right) \right)$$

◆ Around axis $-\hat{a}$ by angle $(-\alpha)$ (it's the **same rotation**!) :

$$q' = \left(-\sin\left(\frac{-\alpha}{2}\right)\hat{a}\,, \cos\left(\frac{-\alpha}{2}\right) \right) = q$$

the **same** quaternion :-)

Nice! But:

◆ Around axis $\hat{a}$ by angle $(\alpha + 2\pi)$ (again, it's the **same rotation**!) :

$$q'' = \left(\sin\left(\frac{\alpha}{2} + \pi\right)\hat{a}\,, \cos\left(\frac{\alpha}{2} + \pi\right) \right) = \left(-\sin\left(\frac{\alpha}{2}\right)\hat{a}\,, -\cos\left(\frac{\alpha}{2}\right) \right) = -q$$

a **different** quaternion :-(

◆ Conclusion: quaternion q and quaternion $-q$ encode the same rotation

Se ora invece faccio una rotazione attorno all'asse -a (versore con verso opposto) di un angolo -alpha, abbiamo la **stessa rotazione** in output.

Poi **ci sono quaternioni diversi che rappresentano le stesse rotazion**i

---

Given a quaternion q representing a rotation:

◆ Flip its imaginary part (getting $\bar{q}$): **invert** rotation

◆ Flip its real part (getting $-\bar{q}$): **invert** rotation

◆ Flip everything (getting $-q$): **same** rotation

Every single rotation is encoded
by **two** different quaternions: q and $-q$.

(also, the inverse rotation is encoded by two different quaternion as well: $\bar{q}$ and $-\bar{q}$

Dato un quaternione q, se prendiamo il coniugato di q, stiamo prendendo **l'inversa della rotazione**.

Se prendiamo il quaternione che ha solo il segno della parte reale cambiata (- il coniugato di q è la stessa cosa), questo **inverte la rotazione**.

Invece -q è la **stessa rotazione**.

Quindi **ogni singola rotazione nello spazio 3D può essere rappresentata da una coppia di quaternioni** (perchè esistono due rotazioni uguali q e -q).

L’inversa di una rotazione è anch’essa una rotazione, ed è rappresentata dai due quaternioni (coniugato di q e -coniugato di q).

è anche **facile interpolare**, basta separare le componenti. Abbiamo sempre il problema dello shortest path e dobbiamo stare attenti al fatto che bisogna sempre assicurarsi che i quaternioni abbiano magnitudine 1, altrimenti non sono più rotazioni. è quindi sempre necessario **normalizzare**. Possiamo farlo facendo un’interpolazione normalizzata, ovvero normalizzando alla fine dell’interpolazione, oppure usando spherical interpolation considerando i quaternioni come vettori 4D (come avevamo visto in una lezione precedente).

Good results, but two *caveats*:

- Take the “shortest path” (as usual):
  flip 2nd quaternion first, if this makes them closer
  - Distance defined as dot product in 4D
    (consider quaternions as 4D unit vectors for this)
    (remember: dot product between unitary vectors is a measure of similarity!)
- Loss of normality
  - Needs re-normalization (NLERP),
  - Or SLERP
    (again, just consider quaternions as 4D unit vectors)

- Let $\mathbf{p}$ and $\mathbf{q}$ be two rotations
- $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation.
  - Which one to choose?
- Which one is closer to $\mathbf{p}$?
  - Distance between $\mathbf{p}$ and $\mathbf{q}$ $= \mathrm{dot}(\mathbf{p}, \mathbf{q})$
  - Distance between $\mathbf{p}$ and $-\mathbf{q}$ $= \mathrm{dot}(\mathbf{p}, -\mathbf{q}) = -\mathrm{dot}(\mathbf{p}, \mathbf{q})$
- Conclusion:
  - If $\mathrm{dot}(\mathbf{p}, \mathbf{q})$ is positive, interpolate with $\mathbf{q}$
  - Otherwise, interpolate with $-\mathbf{q}$

Il dot product ci dice la distanza tra i quaternioni, perchè questi sono sostanzialmente versori.

La distanza tra p e -q è -dot product tra p e q. Quindi io posso calcolare il dot product tra p e q, se è positivo allora significa che l'angolo è più piccolo, perchè l'altro sarà negativo e quindi maggiore. Quindi usa q per interpolare, altrimenti usa -q.

Prodotto tra quaternioni

Prendiamo un quaternione *a b c d*, e un quaternione *e f g h*. Vogliamo fare il **prodotto**. Ciascuno di questi è il coefficiente rispettivamente di *i j k* e la parte reale. Noi sappiamo come si comportano i j k quando fanno il prodotto tra di loro. Per esempio quando si moltiplicano tra di loro fanno -1.

Se fissiamo la riga i, prendiamo j, sappiamo che i*j visto che andiamo nell'ordine orario, fa k (il terzo). E il suo coefficiente sarà e * b. Nello stesso modo, possiamo riempire la tabella.

×	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

$\vec{w}$

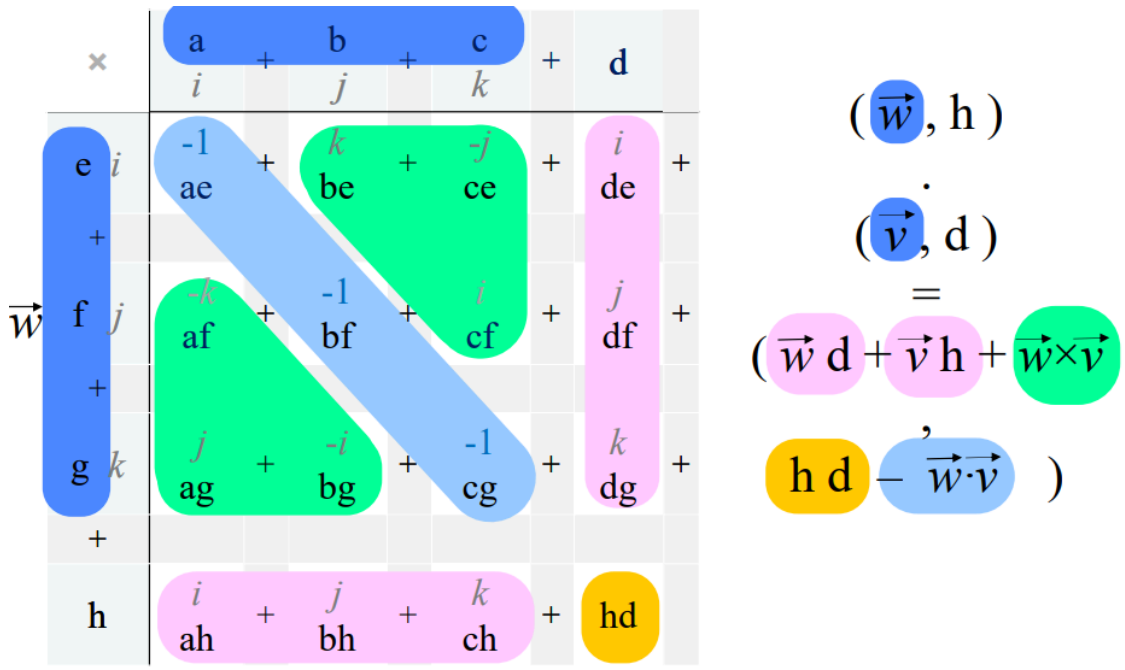
$$(\vec{w}, h) \cdot (\vec{v}, d) = (\text{some vector}, \text{some scalar})$$

Ogni quaternione possiamo pensarlo anche come un vettore 3D e uno scalare (sopra).

Questo può aiutare a semplificare i conti.

Il risultato sarà sempre un quaternione, quindi avrà un campo vettore e uno scalare.

Innanzitutto sappiamo che il prodotto delle parti reali (h*d) sarà reale. Poi abbiamo il dot product con segno negativo tra i due vettori: il segno negativo è dato dai -1 sulla diagonale. I coefficienti della diagonale sono gli elementi che moltiplichiamo nel dot product dei due vettori.



Per la parte immaginaria abbiamo: la parte reale del secondo (d) che moltiplica il vettore 3D del primo. Sommata alla parte reale del primo che moltiplica il vettore 3D del secondo.

Infine manca il cross product tra i due vettori 3D.

Applicazione di rotazioni con quaternioni

quaternion representing a rotation

Quaternion representing 3D point or vector $\vec{v}$

conjugate of $(\vec{w}, a)$

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

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

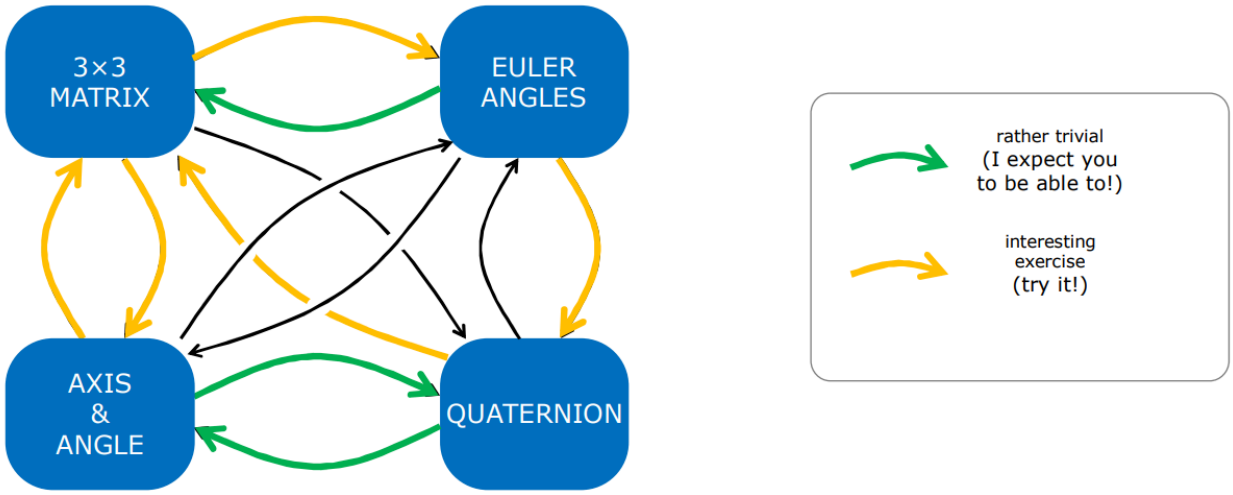
$$= \begin{pmatrix} a\,\vec{w} \times \vec{v} - \vec{w} \times \vec{v} \times \vec{w} + (\vec{v} \cdot \vec{w})\vec{w} + a^2\,\vec{v} - a\,\vec{v} \times \vec{w}, \\ \cancel{a\vec{v} \cdot \vec{w}} - \cancel{a\vec{v} \cdot \vec{w}} + \cancel{\vec{w} \cdot (\vec{v} \times \vec{w})} \end{pmatrix} =$$

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

Sommario quaternioni per le rotazioni

- Comodi da tenere in memoria perchè sono 4 scalari
- Facili da invertire → basta fare il coniugato
- Facili da comporre → basta fare il prodotto
- Facile da applicare → basta fare il prodotto
- Basta normalizzare il quaternione perchè abbia magnitudine unitaria per forzare il risultato a stare nell'insieme delle rotazioni
- Si comporta bene per le interpolazioni → basta fare interpolazioni fra vettori 4D e ricordarsi di normalizzare
- è ufficialmente la rappresentazione favorita nei 3D game. Non è l'unica usata, ma in generale è la più usata.

Diverse rappresentazioni

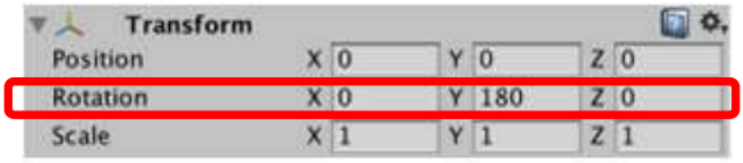


Da angoli di eulero a rotazioni c'è tutta quella roba del coseno seno di... con tutti i prodotti.

Quelli sotto sono "banali".

Le rotazioni su Unity internamente sono processate come quaternioni.

Usa gli angoli, non i radianti.



Quando programmiamo, possiamo lavorare direttamente con i quaternioni.

Cosa succede se voglio rappresentare le **roto-traslazioni**, ovvero tutte quelle trasformazioni ottenute componendo rotazioni e traslazioni. Sono le "trasformazioni rigide" perchè non cambio l'oggetto ma cambio solo la sua posizione.

Un modo è quello di **salvarle separatamente**, rotazioni e traslazioni. (**ricordiamoci l'ordine scala → rotazione → traslazione**)

Altrimenti esiste una rappresentazione in **matrici 4×4** (le abbiamo viste), oppure con **dual quaternions**.

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

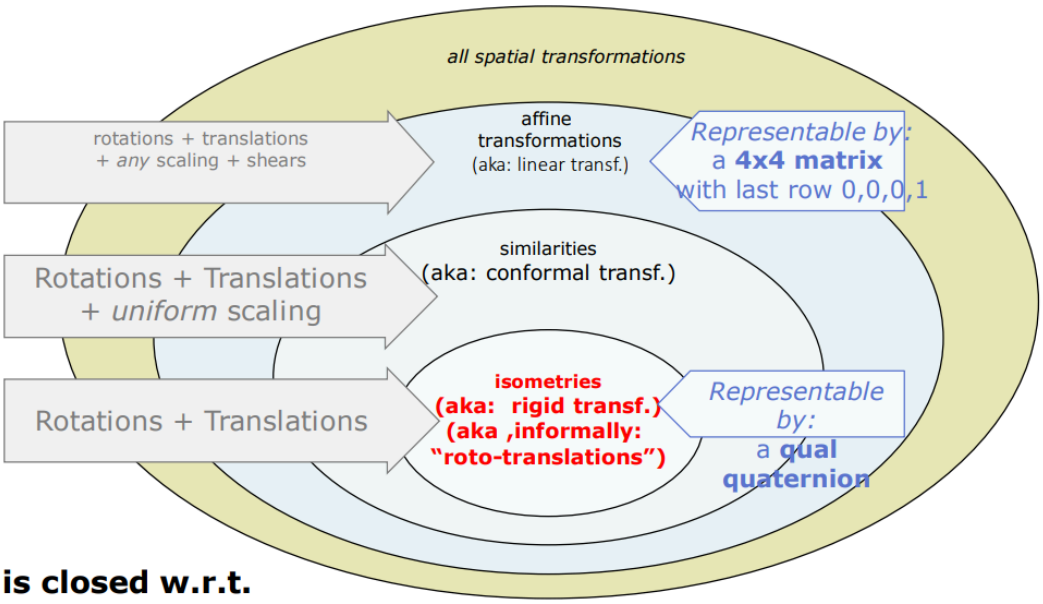
+ Translation

(displacement vector)

OR:

- 4×4 Matrices (or 3×4) ← Why?
- Dual Quaternions ← Not in this course but interesting topic

Possiamo anche usare 3×4 perchè l'ultima riga non va ricordata.



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