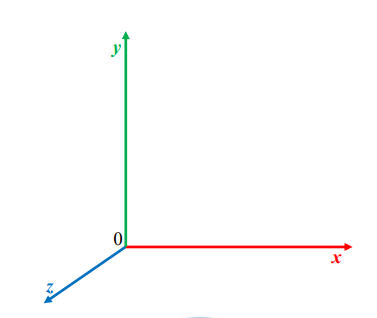
**Matematica per computer-grafica**

Sistema di coordinate

In OpenGL si basa sulla regola della mano destra Oxyz

x: larghezza

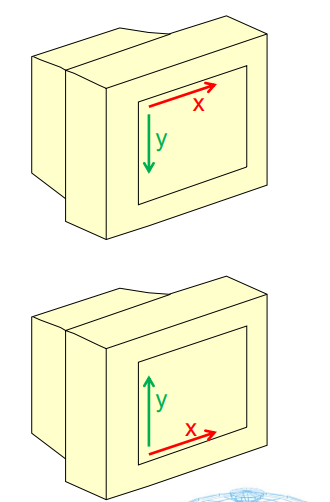
y: altezza

z: profondità, il segno è invertito

più è interno il punto, più la coordinata z è negativa

DirectX usa la legge della mano sinistra.

Il posizionamento dell’origine cambia dal sistema operativo

Windows

MacOS, OpenGL

**Unità di misura**

Non c’è una regola fissa di unità di misura

Un’unità NON è uguale a un pixel, tramite delle matrici di proiezione si può convertire il mondo 3D sullo schermo.

\*consiglio progetto, 1 unità=1 m

**Punto**

Un punto è un oggetto matematico, il pixel è un punto sullo schermo, sono concettualmente diversi

In OpenGL sono la stessa cosa.

Un punto non ha lunghezza spessore e direzione.

Usiamo i punti per costruire i triangoli, sono i vertici.

Possono rappresentare delle fonti di luce puntiformi.

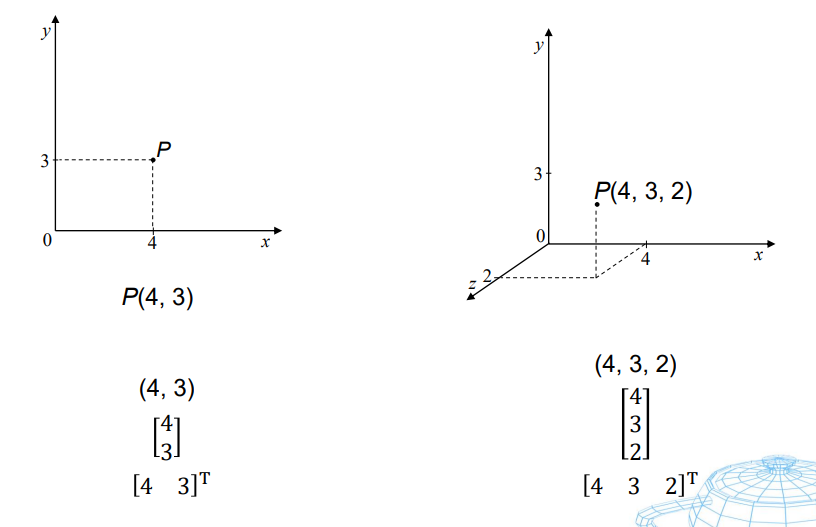


Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

**Vettori**

Il vettore indica un movimento nello spazio, si compone di direzione, verso e norma(modulo)

Il vettore non ha una posizione.

Il vettore può indicare molteplici cose (luci, direzione del vento, proprietà fisiche …)

Vettore 0 è il vettore nullo

Normalizzazione di un vettore

Immagine che contiene testo

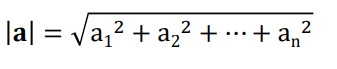
Descrizione generata automaticamente 

Immagine che contiene testo, screenshot, uccello acquatico

Descrizione generata automaticamente

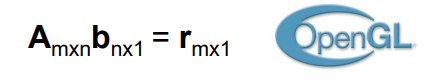
Il prodotto scalare tra due **vettori normalizzati** si ottiene il coseno tra i due vettori

Il prodotto vettoriale (cross) serve per trovare la normale dei triangoli.

**Matrici**

In OpenGL servono per le trasformazioni

Post-moltiplicazione:



Il numero di colonne DEVE essere uguale alle righe di vettore b

La post moltiplicazione cambia in base alla libreria, Directx usa un altro sistema, pervia della rappresentazione.

Immagine che contiene testo

Descrizione generata automaticamente

**La moltiplicazione tra matrici NON è commutativa**

Trasposizione di una matrice

Swap di righe con colonne

Con le matrici 0, tutti 0 nella matrice, distrugge ciò che si moltiplica, pericolosa, non usare.

Per inizializzare le matrici usare la matrice identità.

**Inversione di una matrice**

È importante per eseguire il percorso inverso di una matrice.

Per l’inversione si usa Cayley-Hamilton, per il calcolo c’è una libreria apposta.

L’inversione è sempre possibile sulle matrici che si utilizzano.

**RGB**

Colore=Rintensity+Gintensity+Bintensity

Red Green Blue

Per i grigi basta mantenere lo stesso valore per tutti i campi diversi da 0 e F o 255

**RGBA**

Il canale alpha è un canale aggiuntivo che gestisce la trasparenza

Più basso è più sarà trasparente

Il canale alfa è un concetto software, e il risultato è il calcolo del colore tra le due superfici che separa.

**GLM**

È una libreria matematica per openGL

#include <glm/glm.h>

Non ci sono differenze da vettore e punto, glm::vec3

Se si passa un solo parametro si mettono tutte le componenti pari a quel parametro

glm:Mat3 è un matrice 3x3, se passo 1 solo parametro inizializzo una matrice identità

#include <glm/gtx/string\_cast.hpp>

per stampare a schermo: glm:to\_string(…)

**Traslazioni**

**Immagine che contiene testo

Descrizione generata automaticamente**

**Rotazioni**

**Immagine che contiene testo, tavolo

Descrizione generata automaticamente**

Usare post moltiplicazione perché convenzione OpenGL

**Scaling**

**Immagine che contiene testo

Descrizione generata automaticamente**

Simulare un’ombra su un piano (ombra finta)

Immagine che contiene testo

Descrizione generata automaticamente

Riflesso

Immagine che contiene testo

Descrizione generata automaticamente

**Coordinate omogenee**

**senza** coordinate omogenee si possono effettuare traslazioni vettore-vettore, rotazioni-matrice per vettore e scaling-matrice per vettore.

**Le coordinate omogenee consentono di ottenere tutte le operazioni esprimibili matrice \* vettore**

L’altro vantaggio è la pre-moltiplicazione.

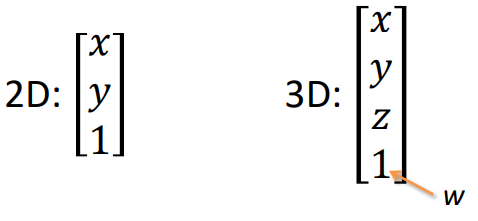
glVertex3f() implicitly sets w = 1)

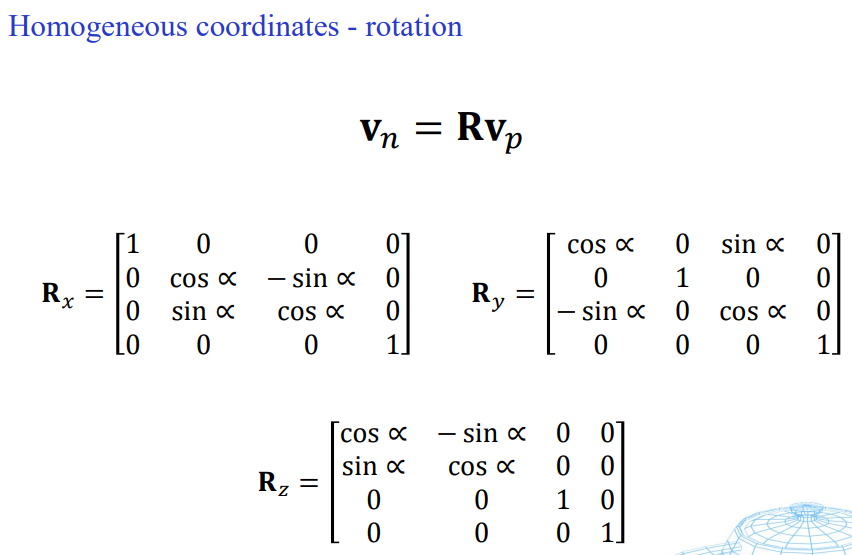
Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo, orologio

Descrizione generata automaticamente



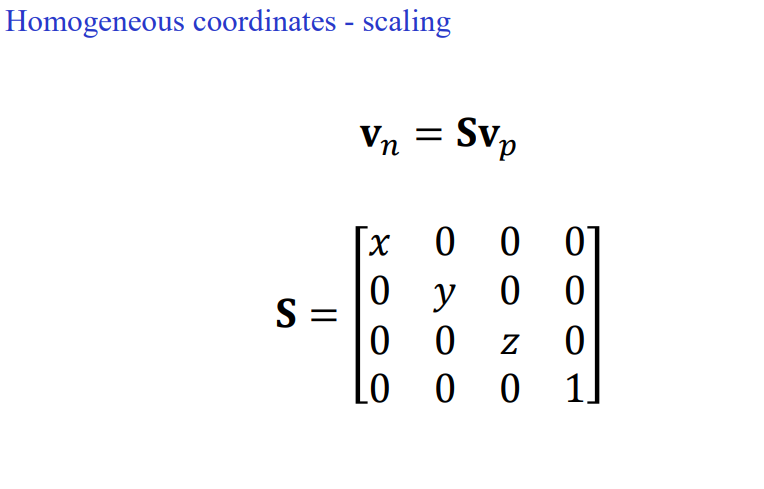
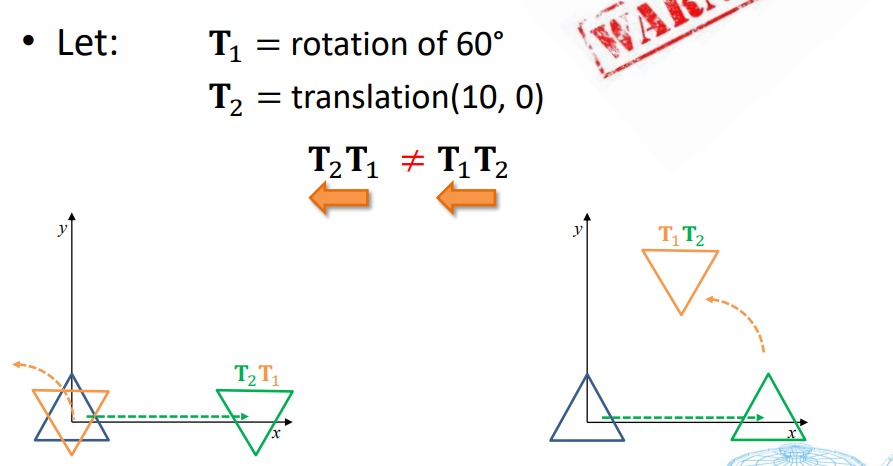


Immagine che contiene testo

Descrizione generata automaticamente



La rotazione intorno all’origine nell’oggetto non si usa la traslazione degli assi all’origine, rotazione e

ri-traslazione, va evitata come operazione. (Risoluzione della CPU limitata)

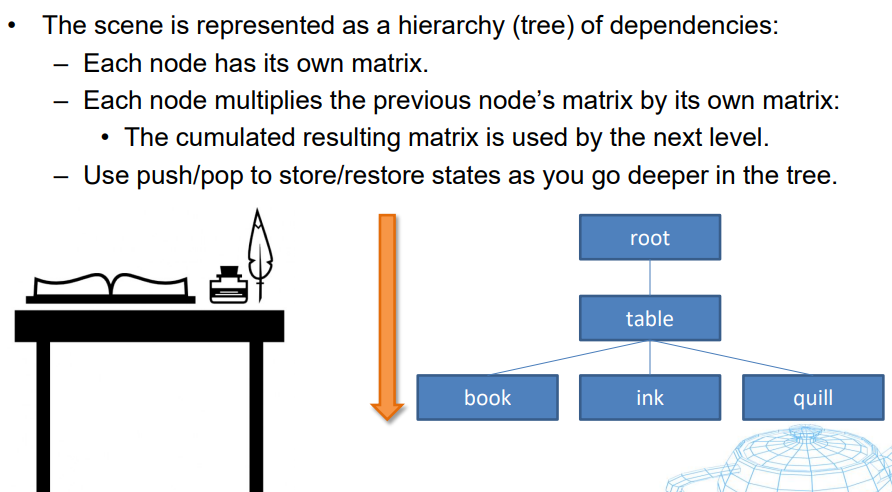
alternativa, far ripartire tutti gli oggetti dall’origine, ruotarli e riposizionarsi, per ogni frame.

Si effettuano meno calcoli.

**Grafico di scena**

Si può esprimere come una gerarchia (albero) di dipendenze.

Ogni nodo ha una sua matrice, che moltiplicherà anche i nodi figli



**Spazi di coordinate**

* Coordinate di modello: sono le coordinate rispetto all’origine relativo del modello (il suo centro)
* Coordinate globali: sono le coordinate della posizione assoluta nel mondo
  + Centro del mondo (0,0,0)
  + Matrice Oggetto \* coordinate Oggetto = coordinate globali
* Coordinate della camera: i vertici 3D relativi alla posizione della camera sono:
  + Centro è la posizione della camera
  + Matrice Inversa Camera \* tutte le coordinate globali = coordinate sulla camera (immagine renderizzata), **(0,0,0) si trova al centro dello schermo**

Immagine che contiene testo

Descrizione generata automaticamente

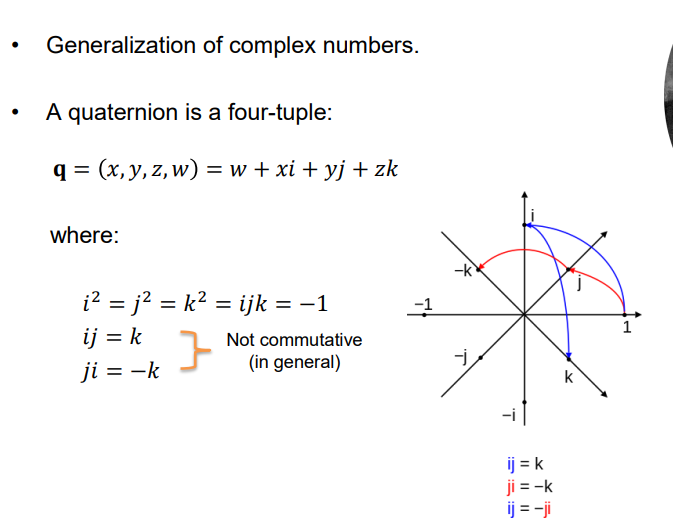
**Interpolazione matrici**

Le metrici non possono essere interpolate, 2 soluzioni::

* Decomposizione delle matrici, applicando le trasformazioni e concatenarle in una matrice:
  + Utilizzare la scomposizione della matrice su qualsiasi matrice di trasformazione arbitraria per ottenere i suoi parametri di scala, rotazione e traslazione, che puoi linearmente

interpolare e concatenare nuovamente in una matrice.

* Usare i quaternioni (solo per le rotazioni), sono numeri complessi e 3 parti immaginarie



La matrice di rotazione può essere convertita in un quaternione, i quaternioni possono essere interpolati.

**Clip coordinates**

è uno spazio di coordinate temporaneo

clip coordinate= matrice di proiezione \* eye coordinates

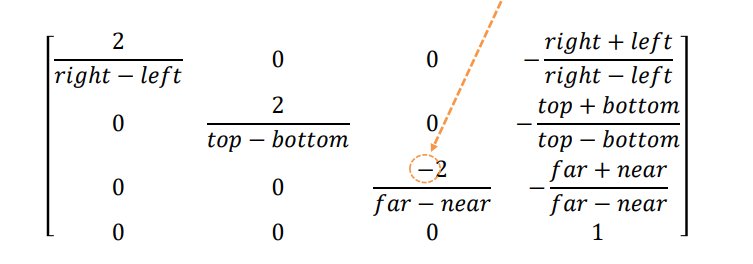
l’obbiettivo è preparare la matrice per la divisione per w, e per la normalizzazione

normalizzare la scena significa avere un elemento compreso tra -1 e +1

**Tipi di proiezione**

* Proiezione ortografica: all’allontanarsi/avvicinarsi dall’oggetto la dimensione è costante
* Proiezione prospettica: effetto prospettico

Matrice di proiezione

****

Right, left: clipping su x

Top,botton: clipping y

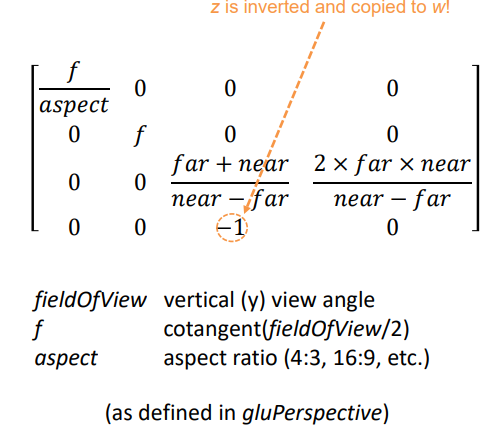
Far,near: clipping su z

Z è invertito, cambia il sistema in mano sinistra, all’allontanarsi dallo schermo (verso dentro) si ha z positivo

**Matrice prospettica**

Più un vertice è lontano dall’osservatore maggiore sarà la distanza, maggiore sarà lo scaling da applicare al vertice.

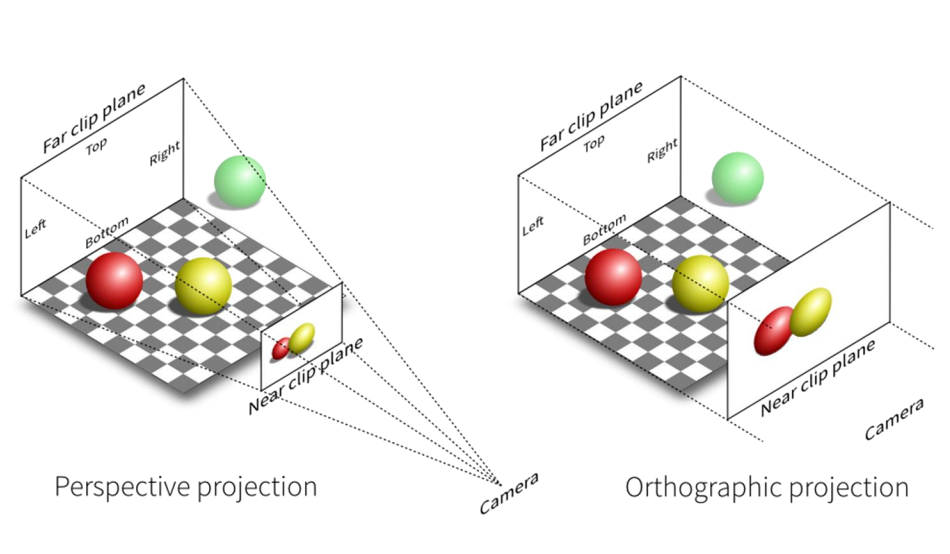
Il rendering non è più un rettangolo, ma un tronco di piramide, che è l’inverso di un punto di fuga.

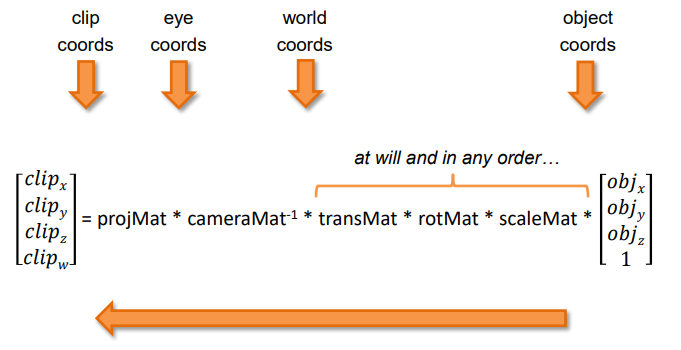


Z è invertito e vale -1 quindi più un oggetto è lontano, maggiore sarà il valore per cui si andrà a dividere.

Più si allontana un oggetto più si avvicina al centro dello schermo

L’output dividendo per w si ottiene un vettore di 3 dimensioni





**Clipping**

Si introduce un ulteriore spazio di coordinate ossia le coordinate normalizzate.

All’atto pratico ci pensa openGL.

Qui si effettua la divisione per w

In questo spazio tutti i vertici sono nel range (-1,-1,-1) e (1,1,1) i vertici che non sono in questo range vengono clippati. OpenGL andrà a ricalcolare i vertici clippati renderizzando solo il risultato, non renderizza casi in cui non ci siano dentro al volume di visualizzazione

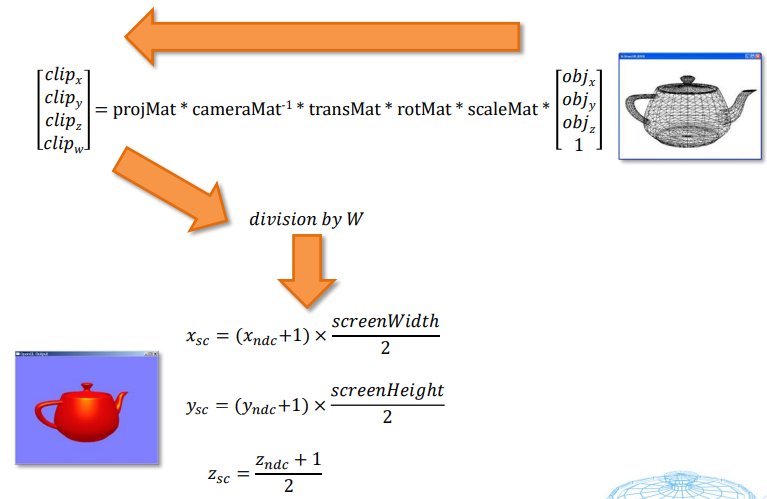
**Screen Coordinates**

Sono le coordinate dello schermo (pixel)

Screen pixels= Viewport transformation\* nomalized device coordinates

Immagine che contiene testo

Descrizione generata automaticamente



La coordinata viene salvata in un range tra (0,1) ed indica la posizione rispetto al nearplane

0 è sul nearplane; 1 è sul far plane

La rasterizzazione unirà i vertici calcolati

**GLM**per le matrici di rotazione richiedono l’angolo in radianti (glm::radians(angolo)) trasforma un angolo in gradi in radianti