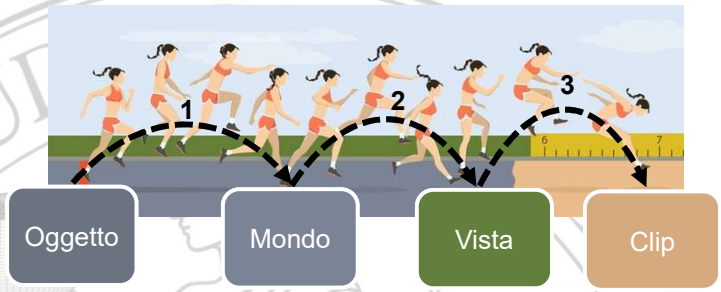
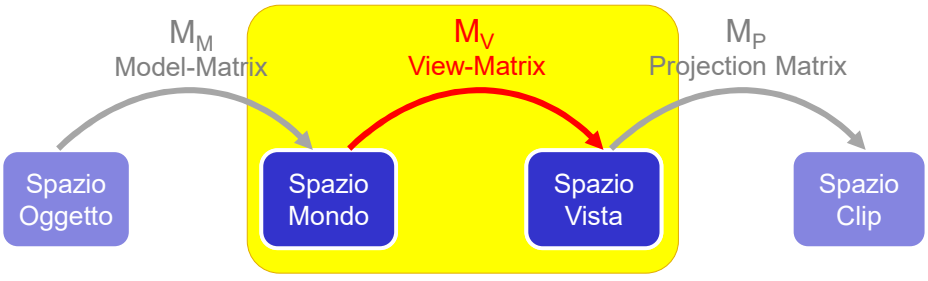
Marco Tarini - Computer Graphics 2024/2025
Università degli Studi di Milano

La sequenza di trasformazioni nel rendering: ancora sulla matrice di vista



1

Ancora sulla trasformazione di vista



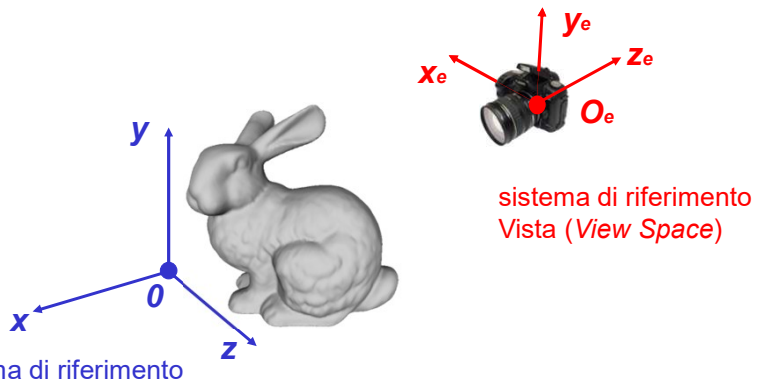
3

Trasformazione di “Vista” (ripasso)

- ✓ Da: **World Space**
  A: **View Space**
- ✓ Definita interamente dai «**parametri estrinseci**» della macchina fotografica (virtuale)
  - ⇒ Cioè da *dove è, e come è orientata* (nel mondo) la nostra pin-hole camera
- ✓ E’ un cambio di sistema di riferimento, cioè una trasformazione affine
  - ⇒ Matrice di Vista = la Matrice che effettua questa trasformazione

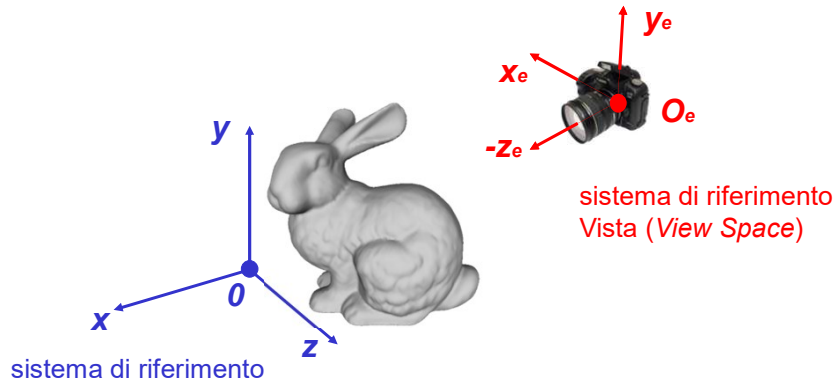
6

Spazio (sistema di riferimento) Vista



7

Spazio (sistema di riferimento) Vista



8

Una modo pratico (e completo) per descrivere i parametri estrinseci

Un modo per esprimere i parametri estrinseci:

1. Posizione dell'osservatore (POV)
   - ⇒ Cioè, dell'occhio (la pupilla), della macchina fotografica (il punto di fuoco) etc
2. Posizione di un punto target osservato
   - ⇒ Si richiede che questo punto compaia in mezzo alla foto
   - ⇒ Oppure, equivalentemente, una **direzione** di vista (un vettore)
3. Vettore "alto" ("up-vector")
   - ⇒ Descrive una direzione che, nella foto, deve apparire come direzione verticale, verso l'alto
   - ⇒ Distingue, ad es, una foto "*portrait*" da una "*landscape*" (o da un "campo obliquo")

✓ Nota: si tratta di punti e vettori tutti espressi nel Sistema di coordinate Mondo

- ⇒ Descrivono la posizione / orientamento della camera NELLA SCENA

9

Settare la matrice di vista: vediamo due strategie

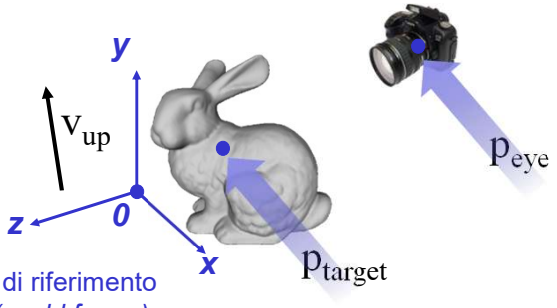
- ✓ Modo 1: calcolarla a partire da (una descrizione de) i parametri estrinseci della macchina fotografica
  - ⇒ come faremo in lab01 creando un'apposita funzione JavaScript
- ✓ Modo 2: concatenando trasformazioni ciascuna delle quali sposa il mondo rispetto alla camera
  - ⇒ quindi, nell'effetto, la camera rispetto al mondo
  - ⇒ come sappiamo, «spostare la camera in avanti di un metro è equivalente a spostare il mondo indietro di un metro»
  - ⇒ come metteremo in pratica nella prossima lezione

10

Una modo pratico (e completo) per descrivere i parametri estrinseci

1) camera position: p_{eye}
2) target position: p_{target}
3) vettore di alto: v_{up}

nb: punti e vettori espressi in spazio mondo!



sistema di riferimento mondo *(world frame)*

12

Osservazione

Matrice di modellazione M di un oggetto qualsiasi:
da spazio (di quel dato) oggetto, a spazio mondo

Matrice di Vista V:
da spazio mondo a spazio Vista
(cioè lo spazio oggetto… dell'oggetto *camera*)

Quindi, se l'oggetto in questione è la *camera*: V è l'inversa di M

la trasformazione di **modellazione** che localizza
un **oggetto** in una certa posizione & orientamento

è l'inversa

della trasformazione di **vista** necessaria per piazzare
la **camera** nella stessa posizione & orientamento

13

Esercizio: come calcolare la matrice di vista V a partire dai parametri estrinseci $\mathbf{p}_{eye}$, $\mathbf{p}_{target}$ e $\vec{\mathbf{v}}_{up}$?

✓ Traccia della soluzione:

attraverso l'algebra di punti e vettori

1. Usare i parametri per trovare il sistema di rifermento { punto origine + tre vettori asse }, dello spazio vista (cioè lo spazio della camera) espressi come punti e vettori in spazio mondo
2. La matrice M, le cui colonne sono i vettori e punto trovati, rappresenta la trasformazione che porta lo spazio vista nello spazio mondo

nota: se la telecamera fosse un qualsiasi oggetto da renderizzare nella scena, allora M sarebbe la sua matrice di modellazione

3. L'inversa di M è quindi la matrice di vista V

14

Costruzione transformazione di vista dai parametri estrinseci

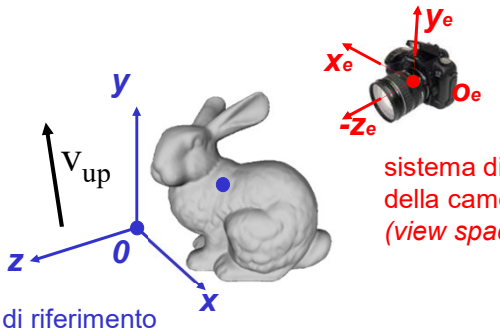
Input:

1) camera position: p_{eye}
2) target position: p_{target}
3) vettore di alto: v_{up}

Output:

Matrice di Trasformazione

world space ⟶ *view space*



sistema di riferimento della camera *(view space)*

sistema di riferimento mondo *(world frame)*

15

Costruzione transformazione di vista dai parametri estrinseci

Input:

1) camera position: p_{eye}
2) target position: p_{target}
3) vettore di alto: v_{up}

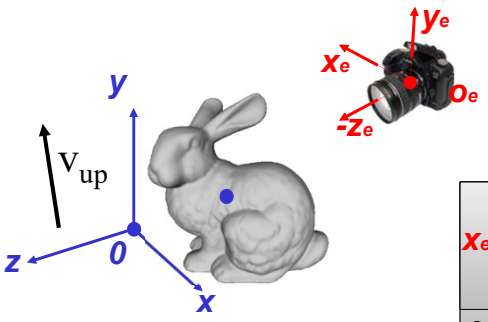
PSEUDOCODICE (v. anche versione in JS)

```
vec3 oe = p_eye;
vec3 xe, ye, ze;

ze = p_target - p_eye;
ze = -ze;
ze = normlize( ze );

xe = cross( vup , ze );
xe = normalize( xe );

ye = cross(ze, xa);
```



x_e	y_e	z_e	o_e
0	0	0	1

matrice che va da spazio vista a spazio mondo.
E' l'inversa di quella che volevamo.
Ergo, va invertita.

16

Costruzione transformazione di vista dai parametri estrinseci

Origine e assi del sistema vista espressi nelle coord del sistema mondo

Per def, l'asse zeta della vista va verso l'osservatore

Deve essere reso unitario

Operazioni dette "completamento di base"

Normalizzaz non necessaria (perché?)

Osservaz:
è possibile fallire? le due normalizzazioni possono essere divisioni per by 0? Quando succede?

```
vec3 oe = p_eye;
vec3 xe, ye, ze;

ze = p_eye - p_target;
ze = normlize( ze );

xe = cross( vup , ze );
xe = normalize( xe );

ye = cross(ze, xa);
```

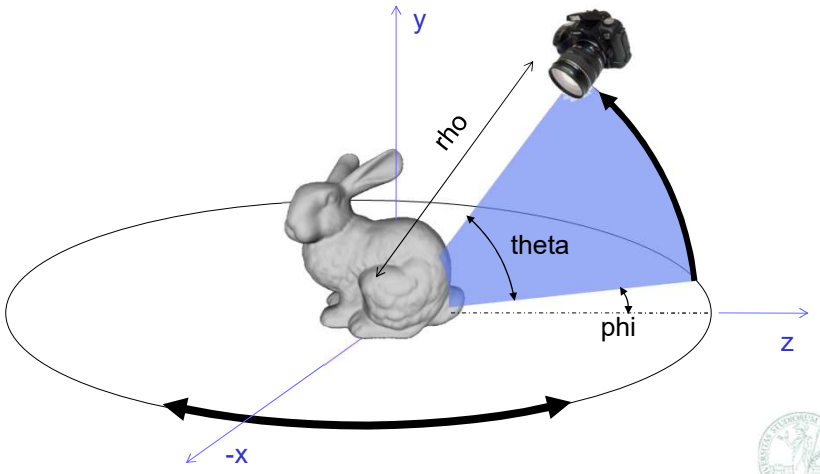
x_e	y_e	z_e	o_e
0	0	0	1

matrice che va da **spazio vista** a **spazio mondo**. E' l'inversa di quella che cerchiamo. Ergo, va invertita.

17

Oppure: possiamo definire la *matrice di vista* combinando rotazioni e traslazioni

Una semplice "Trackball":



19

Trackball
come elemento di un'interfaccia grafica (una GUI)

- ✓ Interfaccia usata per consentire all'utente di selezionare posizione e orientamento di un oggetto
  - ⇒attraverso un pointer device (come mouse o touch screen)
  - ⇒quando l'oggetto è la camera, come nel nostro caso: la trackball determina la matrice di vista
- ✓ Una semplice trackball: posizione & orientamento della camera controllata da soli tre parametri:
  - ⇒due angoli (`phi` e `theta`) e una distanza (`rho`)
  - ⇒es mappati su assi X Y mouse + mousewheel
  - ⇒detta spesso «orbit control»
- ✓ Utile per visualizzare un piccolo oggetto 3D
  - ⇒permettere alla camera di orbitare intorno all'origine
  - ⇒controllo semplice e intuitivo per l'utente

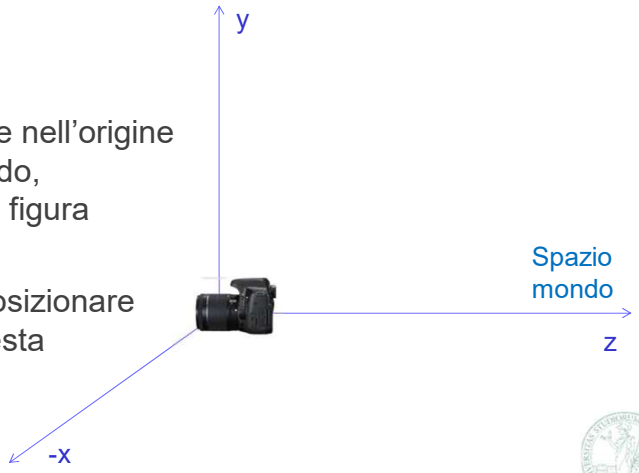
21

Passo 0

Se la matrice di vista è l'identità, allora lo spazio vista coincide con lo spazio mondo

La camera è dunque nell'origine dello spazio mondo, orientate come in figura

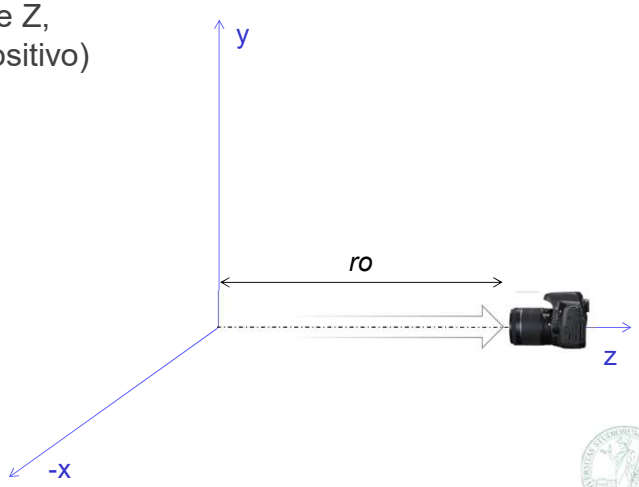
Immaginiamo di riposizionare la camera da questa situazione.



22

Passo 1

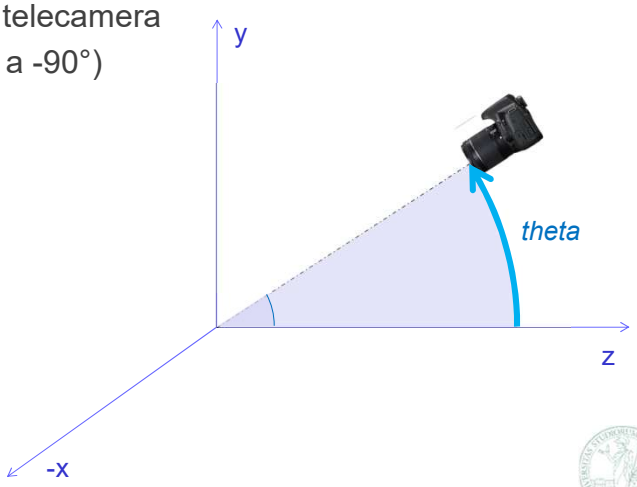
Spostare la macchina all'indietro
di *ro* unità all'indietro
(dunque, sull'asse Z,
con *ro* numero positivo)



23

Passo 2

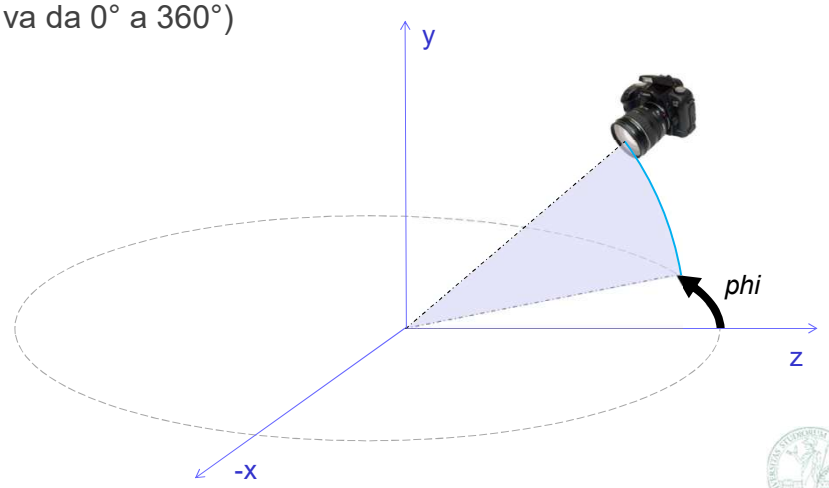
Ruotare di theta gradi
attorno all'asse delle X,
quindi alzando la telecamera
(theta va da +90° a -90°)



24

Passo 3

Ruotare la camera di phi
gradi attorno all'asse delle Y
(phi va da 0° a 360°)



25

La matrice di Vista controllata dalla Trackball

✓ Matrice che sposta la telecamera nel posto voluto:

$$\mathbf{R}_{\mathrm{Y}}(phi) \cdot \mathbf{R}_{\mathrm{X}}(theta) \cdot \mathbf{T}(0,0,rho)$$

(matrice di) **rotazione** attorno all'asse delle Y di angolo phi

seguita da ← ← ←

(matrice di) **rotazione** attorno all'asse delle X di angolo theta

seguita da ← ← ←

(matrice di) **traslazione** di ro sull'asse delle z

✓ Questa è la matrice di modellazione dell'oggetto camera

⇒ va da spazio camera a spazio mondo!

✓ La **matrice di vista** è dunque la sua *inversa*, e cioè:

$$\mathbf{T}(0,0,-rho) \cdot \mathbf{R}_{\mathrm{X}}(-theta) \cdot \mathbf{R}_{\mathrm{Y}}(-phi)$$

Le inverse, in ordine inverso

30

Esercizi pratici implementativi (che puoi provare, e che faremo la prossima volta in aula)

- ✓ Trova il bug nell'esercizio fatto in classe
  - ⇒ la riga in cui si trova il bug è evidenziata da un commento
  - ⇒ Risolto il bug, puoi usare una matrice di proiezione prospettica
- ✓ Prova a costruire una oggetto "trackball".
  1. Inserisci i suoi tre parametri come campi
  2. Inserisci funzioni che calcolano e restituiscono la matrice di vista a parire dai suoi parametri.
     Usa questa funzione per settare la matrice di vista
     (un campo dell'oggetto telecamera, come sappiamo)
  3. Bonus: scrivi delle callback di risposta al mouse che modificano queste variabili col drag e l'uso della rotella

31

Traccia della implementazione di una trackball

- ✓ La trackball come oggetto JavaScript (o JSON):

```
var unaTrackball = {
      rho: 3,     // distanza dall'origine
      phi: 15,    // in gradi (una scelta)
      theta: -35, // in gradi
}
```

- ✓ Schema per costruire la matrice di vista (da mettere in un apposito metodo):

```
var V = new THREE.Matrix4();
V.identity();
V.multiply( matriceDiTraslazione( 0, 0, -this.rho ) );
V.multiply( matriceDiRotazioneX( -this.theta ) );
V.multiply( matriceDiRotazioneY( -this.phi ) );
```

- ✓ (multiply POST moltiplica la matrice, quindi l'ultima riga di codice è la prima trasformazione che viene eseguita)

32