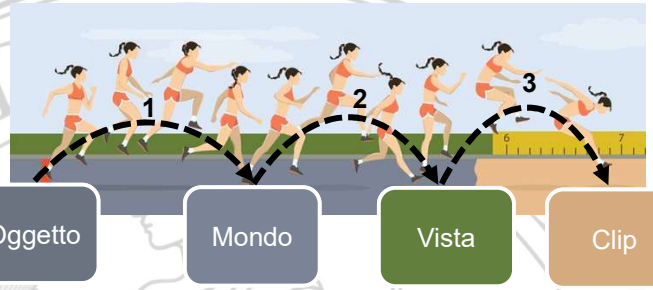
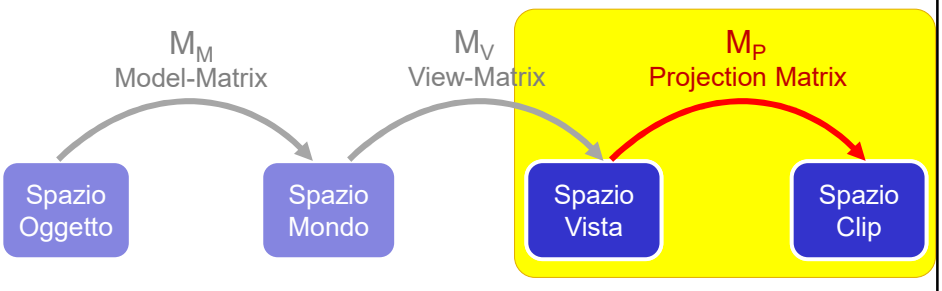
Marco Tarini - Computer Graphics 2025/2026
Università degli Studi di Milano

## La sequenza di trasformazioni nel rendering: trasformazione di proiezione




1

## Trasformazione di proiezione



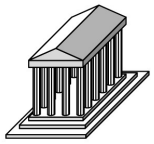

2

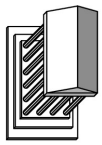
## Trasformazione di proiezione

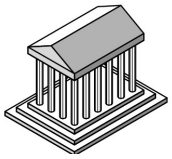
- ✓Problema classico:
  - ⇒in arte, architettura, …
- ✓come riportare
  - ⇒oggetti 3D
  - ⇒su un piano 2D

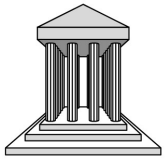
Alcune delle soluzioni classiche hanno un nome come…

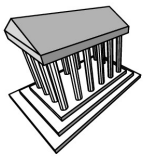

Prospetto Frontale


Assonometria cavaliera


Pianta Obliqua


Assonometria Isometrica


Prospettiva a punto di fuga singolo


Prospettiva a punto di fuga triplo

3

## Trasformazione di proiezione “ortogonale” (oppure “ortografica”)

- ✓Un modo banale per proiettare da 3D a 2D? (“proiettare” = perdere una dimesione)
  - ⇒facile: azzerare la *z*
  - ⇒matrice corrisponente:

$$P_Z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

4

## Trasformazione di proiezione “ortogonale” (o “ortografica”)

✓Abbiamo ottenuto una *proiezione ortogonale*

⇒non c’è alcuna prospettiva:
la dimensione (a schermo) degli oggetti
*non* dipende dalla loro distanza dall’osservatore

⇒linee parallele (in spazio vista)
rimangono parallele anche a schermo

⇒la direzione di vista è costante in ogni pixel

⇒Approssima bene una situazioni reali come:

- un cannocchiale molto potente
che inquadri la scena da molto lontano
- una vista satellitare di un paesaggio urbano
- Insomma, lunghezza focale molto lunga

6

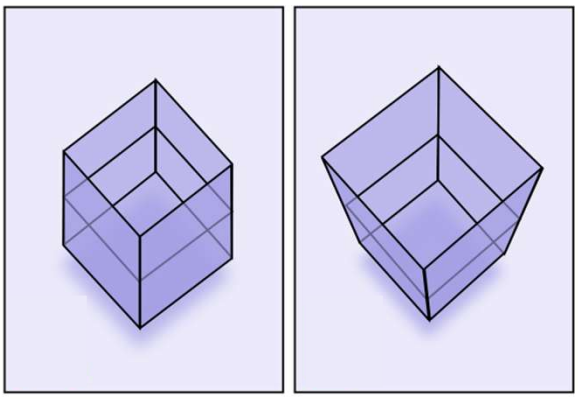
## Proiezioni

Proiezione
ortografica

Proiezione
prospettica

7

## Differenze



Proiezione ortografica

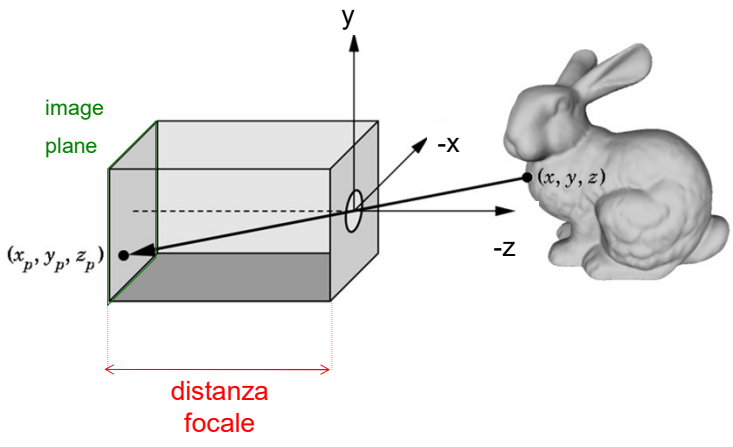
Proiezione prospettica

Vediamo ore come ottenere una **matrice di proiezione prospettica**

8

## Il nostro modello semplificato di camera (richiamo)

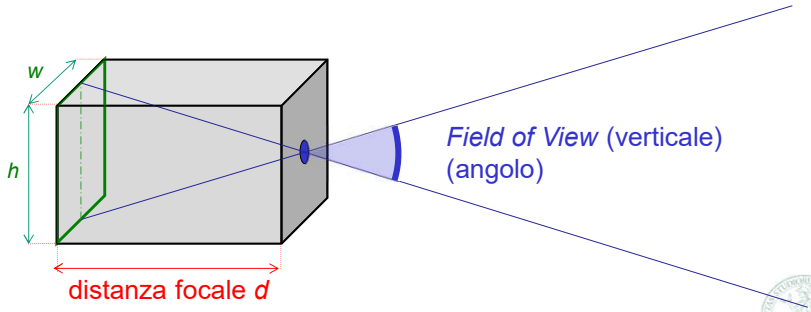
✓ pin-hole camera




11

## Parametri *intrinseci* della camera: (solo i principali)

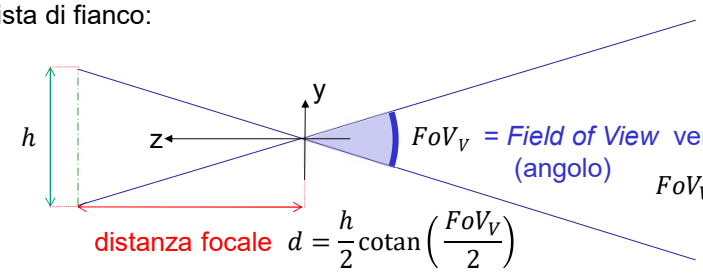
- ✓ dimensioni image plane (*w* , *h* ) (in spazio vista)
- ✓ distanza focale (*d* ) (in spazio vista) *oppure* Field of View (FoV)
  - ⇒ si possono ottenere uno dall'altro, sapendo *h* (come?)
  - ⇒ il valore *d* è facile da usare nei conti, ma FoV è intuitivo da determinare:
    - FoV grande >60° (dist foc. piccola): «grandangolo»
    - FoV piccolo <45° (dist foc. grande): «teleobiettivo»




13

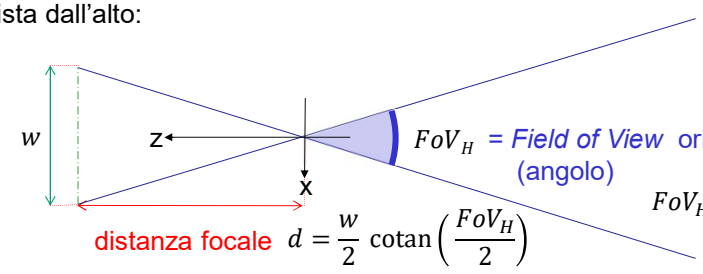
## Da **Field of View** (verticale o orrizontale) a **Distanza focale**, o viceversa

Vista di fianco:




$$FoV_V = 2\arctan\left(\frac{h}{2d}\right)$$

Vista dall'alto:

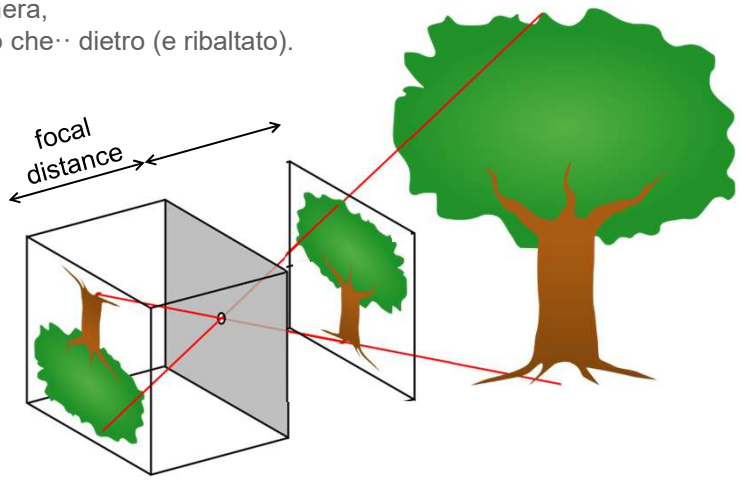



$$FoV_H = 2\arctan\left(\frac{w}{2d}\right)$$

16

## Pin Hole camera (richiamo)

per semplicità, immaginiamo
il piano immagine *davanti*
alla camera,
piuttosto che·· dietro (e ribaltato).

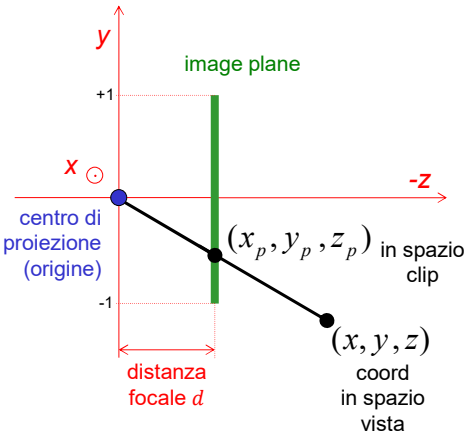



18

## Matematicamente…

(ipotizzando per ora che le dimensioni della camera siano w=2 e h=2)

$$\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = k \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{con } k \text{ t.c. } z_p = -d$$




quindi… $k = -d / z$

e $\begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix} = \begin{pmatrix} -d \cdot x / z \\ -d \cdot y / z \\ -d \end{pmatrix}$

19

## Trasformazione di proiezione (prospettica) da applicare ai punti

✓ Dunque la nostra **trasformazione di proiezione**, sui **punti**, è questa:

$$f_P\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -d \cdot x/z \\ -d \cdot y/z \\ -d \end{pmatrix}$$

coord cartesiane in spazio vista (di un punto)

coord cartesiane in spazio clip (di quel punto)

✓ Posso esprimerla in forma matriciale? Cioè informa…

$$\mathbf{M_P}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -d \cdot x/z \\ -d \cdot x/z \\ -d \\ 1 \end{bmatrix}$$

matrice 4x4 di "proiezione"

coordinate omogenee in spazio vista (di un punto)

coordinate omogenee in spazio clip (di quel punto)

(nota: non serve applicare questa trasformazione a **vettori**)

20

## Estendiamo la rappresentazione di punti e vettori in coordinate **omogenee**

Punti:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ * \end{bmatrix}$$

$\neq 0$ →

Vettori:

$$\vec{v} = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

$0$ →

22

## Estendiamo la notazione

✓ Posso esprimere i *punti* anche con la notazione

$$\mathbf{p} = \begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix} \quad \text{con} \quad w \neq 0$$

$$\begin{bmatrix} wx \\ wy \\ wz \\ w \end{bmatrix} \xrightarrow{\text{divisione per 4ta comp}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

anche detta normalizzazione affine

23

## Estendiamo la notazione

✓ Per es:

$$\begin{bmatrix} 20 \\ 10 \\ 30 \\ 10 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 1 \\ 3 \\ 1 \end{bmatrix} \quad \begin{bmatrix} 0.2 \\ 0.1 \\ 0.3 \\ 0.1 \end{bmatrix} \quad \begin{bmatrix} 4 \\ 2 \\ 6 \\ 2 \end{bmatrix} \quad \dots$$

questa è in "forma canonica" ( $w = 1$ )

sono alcune delle **coordinate omogenee** del **punto** che ha queste **coordinate cartesiane**

$$\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$ **punto**

$$\begin{bmatrix} 2 \\ 1 \\ 3 \\ 0 \end{bmatrix}$$

sono le uniche **coordinate omogenee** del **vettore** che ha queste **coordinate cartesiane**

$$\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$$ **vettore**

24

## Coordinate omogenee di punti (recap)

- ✓ Il punto **p** di coordinate cartesiane $(x,y,z)$ è rappresentato in coordinate omogenee come $(xw,yw,zw,w)$, con $w$ qualsiasi, eccetto 0
- ✓ Coordinate omogenee diverse $(x, y, z, w)$ e $(x', y', z', w')$ possono rappresentare lo stesso punto;
  - ⇒ Per es, (2,4,6,2) e (1,2,3,1)
- ✓ Quando $w = 1$ (forma *canonica*) le coord. cartesiane del punto coincidono con le prime tre coord. omogenee.
- ✓ Con $(x, y, z, w \neq 0)$ si rappresentano *punti*, con $(x, y, z, 0)$ si rappresentano *vettori.*

25

## Coordinate omogenee (o affini)

- ✓ Tutte le **matrici di trasformazione** che abbiamo visto fin'ora continuano a "funzionare" anche su **punti** espressi in coordinate omogenee non canoniche (cioè con $w \neq 1$)
- ✓ Per esempio, traslazione:

$$\begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 20 \\ 15 \\ 50 \\ 10 \end{pmatrix} = \begin{pmatrix} 70 \\ 35 \\ 80 \\ 10 \end{pmatrix}$$

Matrice di traslazione di (+5 , +2 , +3 )

Punto di coord cartesiane (2 , 1.5 , 5)

Punto di coord cartesiane (7 , 3.5 , 8)

(+5 , +2 , +3 )

- ✓ Ora possiamo esprimere anche la **proiezione prospettica**…

26

## Matrice di proiezione prospettica: alternativa (scegliamo arbitariamente questa)

matrice di trasformazione per la **proiezione prospettica**

Nota: non è una trasformazione *affine*. Guarda l'ultima riga!

$$\mathbf{P} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

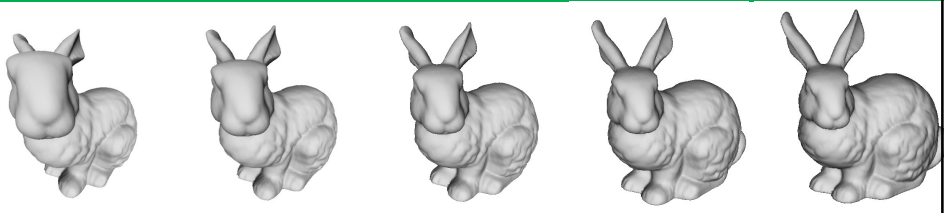
$$\mathbf{P}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} d \cdot x \\ d \cdot y \\ d \cdot z \\ -z \end{pmatrix}$$

…che esprime il punto di coordinate *cartesiane*… (dividendo per w)

$$\begin{pmatrix} -x \cdot d/z \\ -y \cdot d/z \\ -d \end{pmatrix}$$

28

## Proiezione Prospettica: effetto della la distanza focale



$d$ piccolo

$$\mathbf{P} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$d$ grande

Più distorsione prospettica.

Effetto "fish-eye" (grandangolo)

Proporzioni più mantenute

Effetto "zoom" (eg. vista dal satellite)

30

## Non è una buona idea perdere l'informazione sulla z (cioè la distanza del punto dal POV)

La lunghezza focale, cioè la profondità della pin-hole camera (ipotizzando che la sua altezza e larghezza siano 2)

normalizzazione affine (per l'estrazione di coordinate Cartesiane)

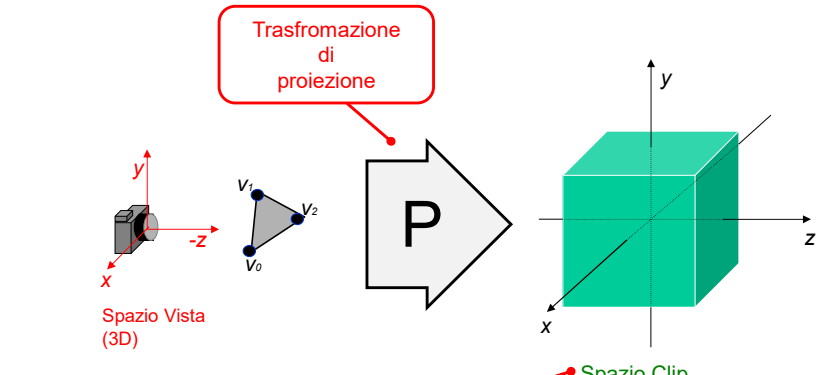
$$\begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} d \cdot x \\ d \cdot y \\ d \cdot z \\ -z \end{pmatrix} \rightarrow \begin{pmatrix} -x \cdot d/z \\ -y \cdot d/z \\ -d \\ 1 \end{pmatrix}$$

perspective matrix — view coords — clip coords

PROBLEMA: il valore di *z* risulta costante .
Formalmente è corretto
(dato che il punto finisce sempre sul piano immagine)
ma perdiamo una dimensione inutilmente

31

## In realtà la trasf. di Proiezione non deve perdere la terza dimensione (la z): ci sarà utile

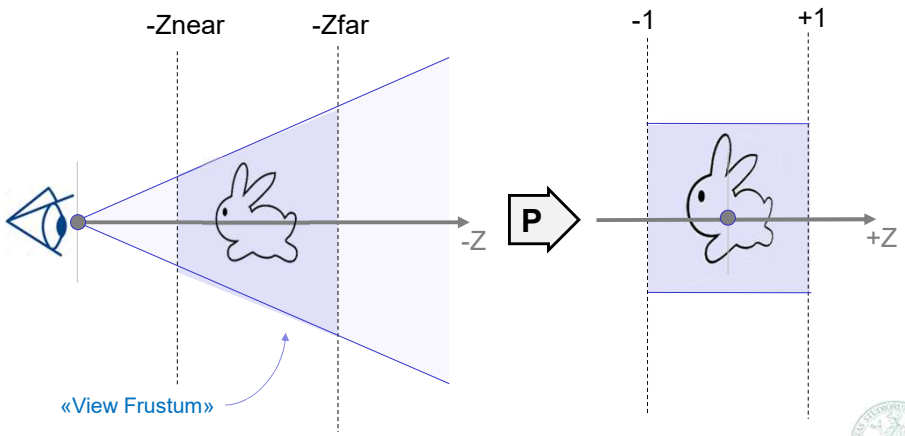



La parte visibile casca per definizione in
**[-1,+1] × [-1,+1] × [-1,+1]**
(*indipendentemente* dalla dimensione, aspect ratio, o risoluzione dello schermo).
(per questo le coord in spazio clip sono dette anche "Normalized Device Coordinates" , NDC )

33

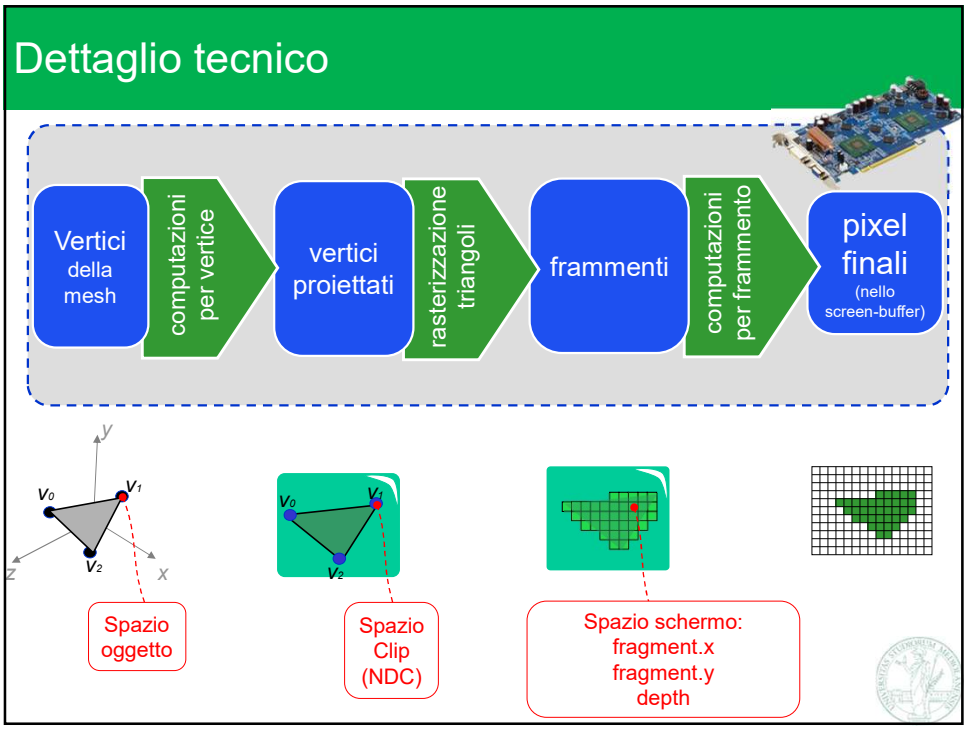
## Modificare la matrice di proiezione per produrre una z in spazio clip fra -1 e +1

In Spazio Vista: In spazio Clip:




34

## Dettaglio tecnico




35

## Dettaglio tecnico

Nell'algoritmo di rendering che stiamo descrivendo,
la fase di **processing per vertice**
ha il compito di produrre …

- ✓ (a partire dalle coordinate in **spazio oggetto**)

le **coordinate clip** del vertice
sotto forma di **coordinate omogenee**
$(x \cdot w, y \cdot w, z \cdot w, w)$

- ✓ Se le coord $x, y, z$ sono in $[-1..+1]$, il vertice è inquadrato
- ✓ quando $x, y$, **oppure** $z$ fuori da $[-1..+1]$ → fuori da immagine!
- ✓ cioè: se $z > +1$ , troppo lontano: non disegno
  cioè: se $z < -1$ , troppo vicino: non disegno
- ✓ La matrice di proiezione deve quindi occuparsi
  anche di produrre anche la $z$ corretta in spazio clip

36

## Modificare la matrice di proiezione per rimappare la Z correttamente (senza perdita di informazione)

- ✓ Scelgo arbitrariamente…
  - ⇒z_near : la distanza dalla camera
    della cosa più *vicina*
    da inquadrare nell'immagine
  - ⇒z_far : la distanza dalla camera
    della cosa più *lontana*
    da inquadrare nell'immagine
  - ⇒nota: entrambe sono distanze definite in spazio vista
    - sono due numeri reali > 0
- ✓ La trasf di proiezione deve portare
  l'intervallo Z (in spazio vista) [-z_near .. –z_far]
  nell'intervallo Z (in spazio clip) [-1..+1]

37

## Modificare la matrice di proiezione per rimappare la Z correttametne

✓ Voglio modificare P in modo che la $z$ in output non sia costante (ma dipenda della $z$ in input)

MATRICE PROIEZIONE — COORDINATE VISTA OMOGENEE — COORDINATE CLIP OMOGENEE — COORD. CLIP OMOGENEE, MA IN FORMA CANONICA

$$\begin{pmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} d \cdot x \\ d \cdot y \\ Az + B \\ -z \end{pmatrix} \rightarrow \begin{pmatrix} -x \cdot d/z \\ -y \cdot d/z \\ -A - B/z \\ 1 \end{pmatrix}$$

quali valori devo usare per $A$ e $B$ ?

view coords

clip coords

38

## Modificare la matrice di proiezione per produrre una z in spazio clip fra -1 e +1

✓ Troviamo i valori per $A$ e $B$

✓ Vogliamo che:

⇒ se $z$ vista = $-zNear$, allora $z$ clip deve essere $-1$

⇒ se $z$ vista = $-zFar$, allora $z$ clip deve essere $+1$

⇒ Quindi

$$-A + \frac{B}{zNear} = -1 \qquad -A + \frac{B}{zFar} = +1$$

✓ Risolvo per $A$ e $B$ ottenendo… (verificare!)

$$A = \frac{zNear + zFa}{zNear - zFar} \qquad B = \frac{2 \cdot zNear \cdot zFar}{zNear - zFar}$$

39

## Matrice di Proiezione Prospettica ottenuta

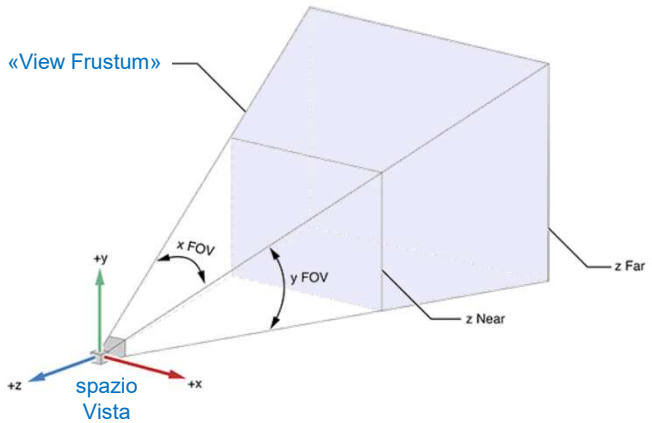
$$P = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & \frac{(z_N + z_F)}{(z_N - z_F)} & \frac{2 \cdot z_N \cdot z_F}{(z_N - z_F)} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

$d$ lunghezza focale

(assmendo che H e W della pin-hole camera siano 2 e 2)

40

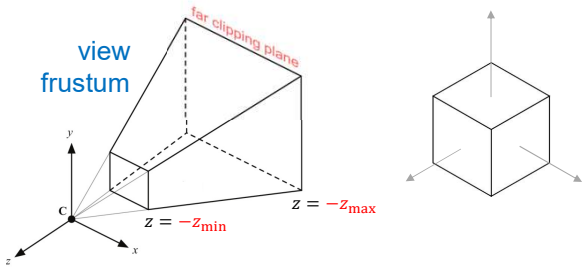
## View Frustum: la zona dello spazio vista che contiene ciò che viene visto dalla camera




42

## Limiti dello spazio visibile (cioè quello inquadrato dalla camera)

| | in Spazio Vista | in Spazio Clip |
|---|---|---|
| $x$ | «left clipping plane» | $> -1$ |
| | «right clipping plane» | $< +1$ |
| $y$ | «bottom clipping plane» | $> -1$ |
| | «top clipping plane» | $< +1$ |
| $z$ | $< -z_{\min}$ | $> -1$ |
| | $> -z_{\max}$ | $< +1$ |



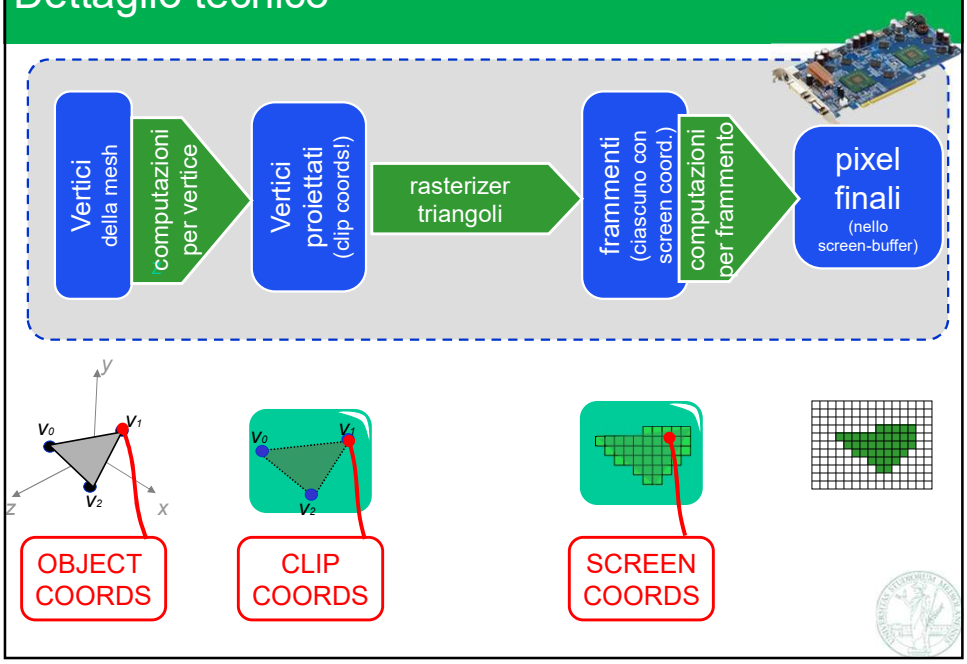

43

## Spazio screen (screen space) (anche detto spazio viewport)

- ✓ **Viewport**: area dello schermo riempita dal rendering
  - ⇒ per es: nelle applicazioni grafiche “a schermo intero”, il viewport è l’intero schermo.
  - ⇒ Nelle applicazioni a finestra, è il rettangolo della finestra
- ✓ **Spazio «screen»**
  - ⇒ Uno spazio in cui i pixel del viewport hanno coordinate intere
  - ⇒ Asse $x$: verso la destra dello schermo va da 0 a risoluzione orizzontale del *viewport*
  - ⇒ Asse $y$: verso l’alto dello schermo va da 0 a risoluzione verticale del *viewport*
  - ⇒ Punto 0,0: il pixel in basso a sx dello schermo
  - ⇒ Asse $z$: direzione profondità del pixel la coordinata $z$ è un valore da 0 a 1 detto ***depth*** (profondità)

45

## Dettaglio tecnico




46

## Trasformazione di viewport (compito del rasterizzatore)

La fase di **rasterizzatore** del pipeline

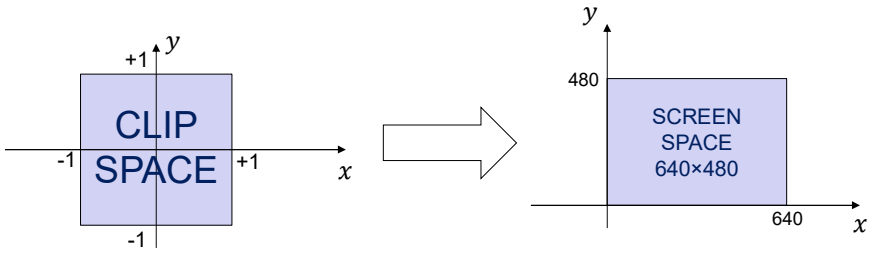
- ✓ riceve i vertici dalla fase precedente
  in coordinate *clip* omogenee (4 numeri *reali*)
- ✓ come preliminare: converte qundi queste coordinate…
  - ⇒ da: omogenee in *clip space*
  - ⇒ passando da: cartesiane in *clip space* (dividendo per *w*)
  - ⇒ a: cartesiane in *screen space* (mapping lineare, vedi dopo)
- ✓ questa è quindi un'ulteriore trasformazione spaziale
  detta di viewport
  - ⇒ eseguita automaticamente (in modo hard-wired) dal rasterizzatore
  - ⇒ avviene dopo la trasf. di proiezione
- ✓ produce poi frammenti a coordinate *screen* cartesiane (e intere)
  - ⇒ e.s. in [0..639] x [0..479], se lo schermo è 640 x 480
  - ⇒ vedi lezione sulla rasterizzazione

47

## Da Clip Space a Screen Space (o “Viewport”)

- ✓ In **coordinate screen**, le posizioni dei pixel (e dei frammenti) sono coordinate intere: sono gli indici della matrice di pixel

il rettangolo di schermo che contiene il rendering (nelle applicazioni che non sono a schermo intero)




48

## Trasformazione di viewport (automatica): da Clip Space a Screen Space (o Viewport)

$$f_{VIEWPORT}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} (x+1)/2 \cdot RES_X \\ (y+1)/2 \cdot RES_Y \end{pmatrix}$$

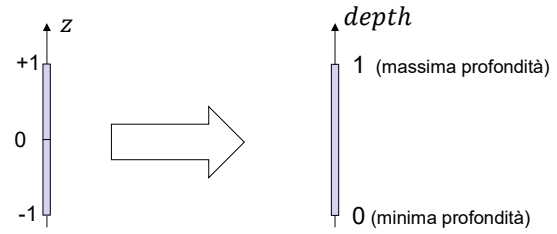
in $[-1,+1] \times [-1,+1]$

in $[0,1] \times [0,1]$

50

## La Z in spazio Screen (convenzione)

✓ La coordinata cartesiana z in spazio clip (che va da -1 a +1 per i punti nel quadro) viene anch'essa trasformata in spazio screen (cioè in coordinate pixel), dove prende in nome di *depth* (profondità del pixel), e va nell'intervallo 0,1




51

## Da Clip Space a Screen Space (trasformazione di viewport)

$$f_{to\_screen}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (x+1)/2 \cdot RES_X \\ (y+1)/2 \cdot RES_Y \\ (z+1)/2 \end{pmatrix}$$

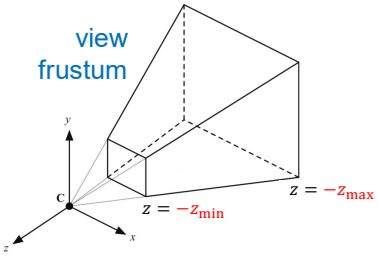
in $[-1, +1]^3$ — in $[0, 1]^3$

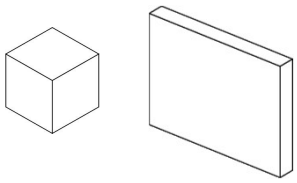
Quando $RES_X \neq RES_Y$ questo passaggio corrisponde ad una scalatura non uniforme della *x* e *y*

52

## Limiti dello spazio visibile (sommario)

| | Spazio Vista | Spazio Clip | Spazio Schermo | |
|---|---|---|---|---|
| x | «left clipping plane» | $> -1$ | $\geq 0$ | fragment coordinates |
| | «right clipping plane» | $< +1$ | $< res_x$ | |
| y | «bottom clipping plane» | $> -1$ | $\geq 0$ | |
| | «top clipping plane» | $< +1$ | $< res_y$ | |
| z | $< -z_{min}$ | $> -1$ | $\geq 0$ | fragment depth |
| | $> -z_{max}$ | $< +1$ | $\leq 1$ | |



53

## Trasformazione di viewport : include uno scaling NON uniforme (distorce)

- ✓ Il quadrato in Spazio Clip: $[-1, +1] \times [-1, +1]$ …
  - ⇒un quadrato (di lato 2)
- ✓ … diventa il Viewport: $[0, resX] \times [0, resY]$
  - ⇒è quadrato solo se $resX = resY$
- ✓ Il passaggio da Clip Space a Screen Space introduce uno scaling NON uniforme (se $resX \neq resY$)

**quindi:**

- ✓ nella Trasformazione di Proiezione deve compensare questo effetto «in anticipo» applicando lo scaling non-uniforme inverso

54

## Pre-correzione dell'aspect ratio

✓Matrice di proiezione prospettica con correzione dell'aspect ratio $a = \frac{RE_X}{RES_Y}$

$$\mathrm{P} = \begin{pmatrix} d & 0 & 0 & 0 \\ 0 & da & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \text{oppure} \quad \mathrm{P} = \begin{pmatrix} d/a & 0 & 0 & 0 \\ 0 & d & 0 & 0 \\ 0 & 0 & A & B \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

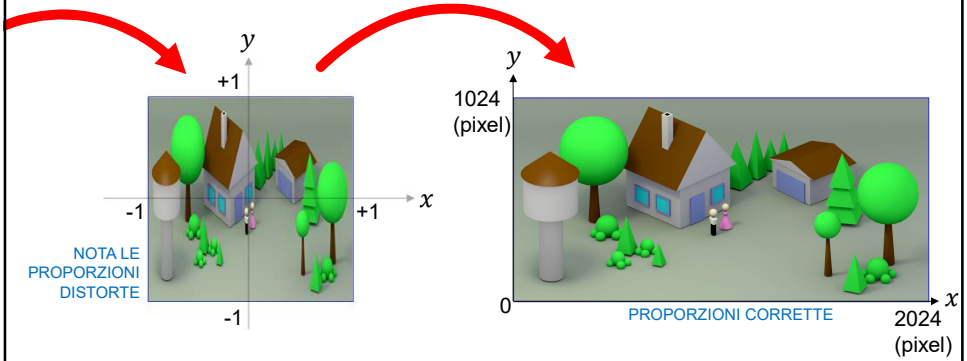
Nota: la trasformaz. di proiezione dipende dai parametri INTRINSECI della macchina fotografica… compreso l'*aspect ratio* del suo frame!

55

## Pre-deformazione della scena da parte della trasformazione di proiezione

**Matrice di Proiezione**
con scalatura non-uniforme
qui: dimezza le $x$
(distorcente!)

**Trasf di Viewport**
con scalatura non-uniforme.
qui: raddoppia le $x$ rispetto alle $y$
(ripristina le correzioni corrette)




**CLIP SPACE**
aspect ratio: 1
(sempre)

**SCREEN SPACE**
aspect ratio: 2
(in questo caso)

56

## Matrice di Proiezione Prospettica finale

$$\mathbf{P} = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & a \cdot d & 0 & 0 \\ 0 & 0 & \frac{(z_N + z_F)}{(z_N - z_F)} & \frac{2 \cdot z_N \cdot z_F}{(z_N - z_F)} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

tutto in spazio vista:
- $a$ aspect ratio
- $d$ lunghezza focale
- $z_N$ z-near
- $z_F$ z-far

} detti i parametri intrinseci della camera fotografica

57

## In totale: la trasformazione di proiezione modella la macchina fotografica virtuale usata

- ✓ La matrice di proiezione che abbiamo visto riflette i parametri (detti «**intriseci**») della pin-hole camera:
  - ⇒La lunghezza focale (espresso nella stessa misura in cu
  - ⇒o, equivalentemente, il FOV
  - ⇒L'aspect ratio (descrive la forma dek il rettangolo sul piano immagine
- ✓ E inoltre due parametri artificiali
  - ⇒z-min e z-far
- ✓ I parametri estrinseci della camera rappresentano invece la sua collocazione nella scena, e sono riflessi nella trasformazione (e matrice) di vista

58