Rendering Pipeline: Geometric Stage

Definita una camera virtuale e una scena tridimensionale, la pipeline di rendering costruisce una serie di **trasformazioni** che proiettano la **scena tridimensionale** in un'immagine in **una finestra contenuta nello schermo bidimensionale**.
Questo stadio della rendering pipeline può essere vista come una **pipeline di trasformazioni geometriche o trasformazioni di sistemi di coordinate.**

Un modello geometrico è infatti trasformato mediante trasformazioni di modellazione, **vista, proiezione e viewport**, ovvero trasformazioni tra vari sistemi di riferimento.

Inizialmente il modello geometrico viene creato nello **spazio locale del modello**.
Viene quindi posizionato ed orientato in scena nel **World Coordinate System** (WCS) mediante **trasformazioni di modellazione** (affini) sui vertici del modello.
Il WCS è unico e dopo che tutti i modelli geometrici necessari per creare una scena sono stati posizionati tutti i modelli sono rappresentati in questo spazio.

Poichè solo i modelli 'visti' dalla camera virtuale sono resi, tutti i modelli sono quindi trasformati mediante una **trasformazione di vista** nel sistema di riferimento della camera o camera frame (**View Coordinate System - VCS**).

Dal centro del sistema della camera solo una porzione di volume, detto **volume di vista**, contiene la scena che è visibile all'osservatore.
Quindi il sistema di rendering elabora la **trasformazione di proiezione** per trasformare il volume di vista contenente la scena visibile dall'osservatore, in un cubo di lato 2 e diagonale di estremi (−1,−1,−1) e (1, 1, 1). Le proiezioni sono in generale ortografica o prospettica, entrambe possono essere costruite con matrici 4×4.
Queste proiezioni trasformano un volume (volume di vista) in un altro volume (cubo).

La vera e propria proiezione dal volume di vista al piano di vista viene infine effettuata **mediante una proiezione ortogonale su una faccia del cubo** che conserva quindi le coordinate x ed y, memorizzando opportunamente la coordinata z, che rappresenta la profondità dei vertici, in una speciale memoria del frame buffer, detta Z-buffer.

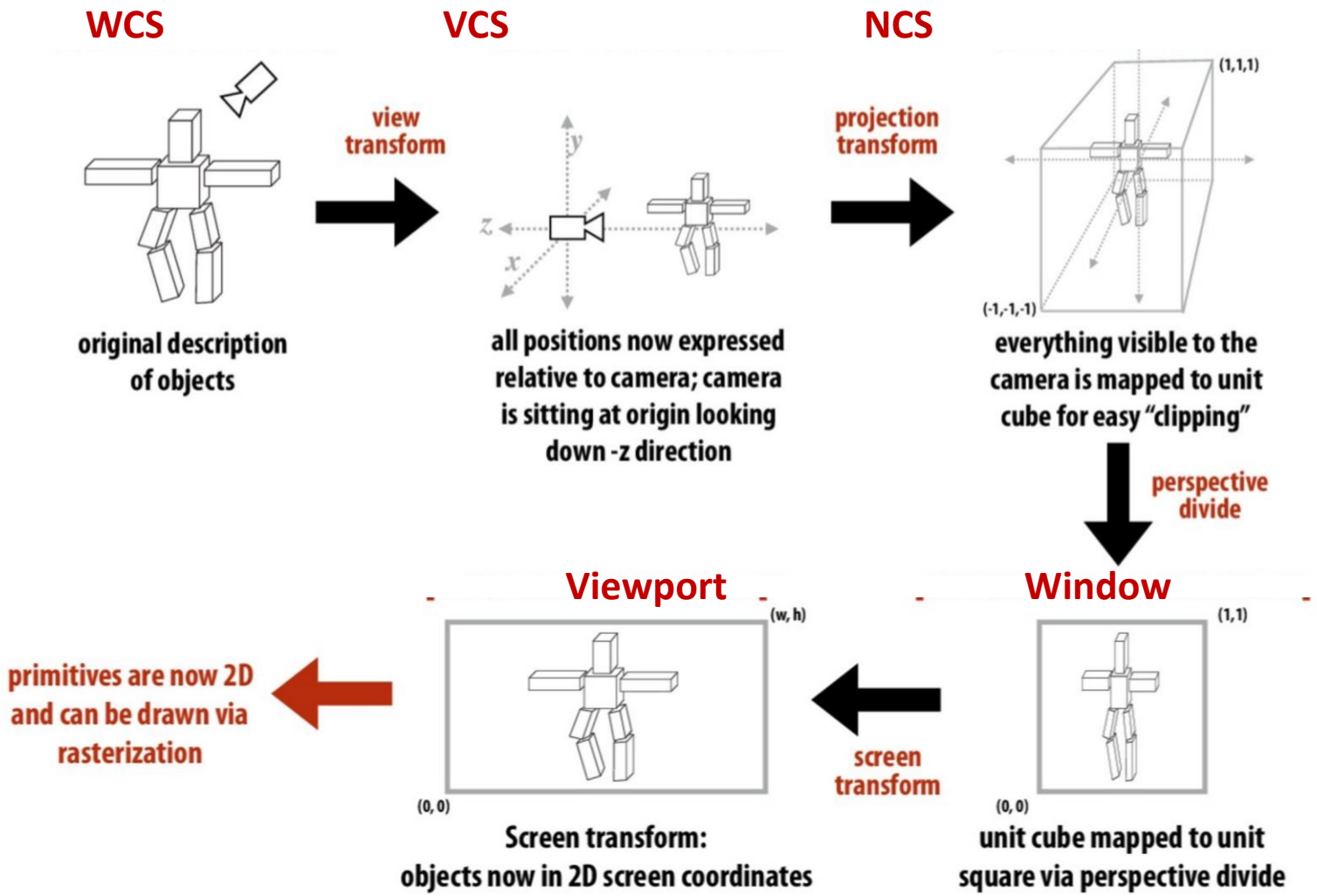
Questo speciale passaggio intermedio da **volume di vista 3D a cubo** (sistema di riferimento normalizzato) **serve essenzialmente per facilitare le operazioni di clipping. In breve**, le primitive che risiedono interamente all'interno del volume di vista verranno disegnate sullo schermo, quelle completamente esterne verranno scartate, mentre quelle parzialmente contenute nel volume di vista verranno eliminate.

Solo le primitive interne al volume di vista sono passate all'ultima trasformazione del geometry stage, la **trasformazione di viewport** (o window-viewport **o screen mapping).**
Quest'ultima converte le coordinate (x, y) reali di ogni vertice, in coordinate schermo espresse in pixel (**device coordinate system (NDC)**). Quest'ultime, insieme con la coordinata $-1 \leq z \leq 1$ forniranno l'input per lo stadio successivo della pipeline, la fase di rasterizzazione.

Geometric Stage

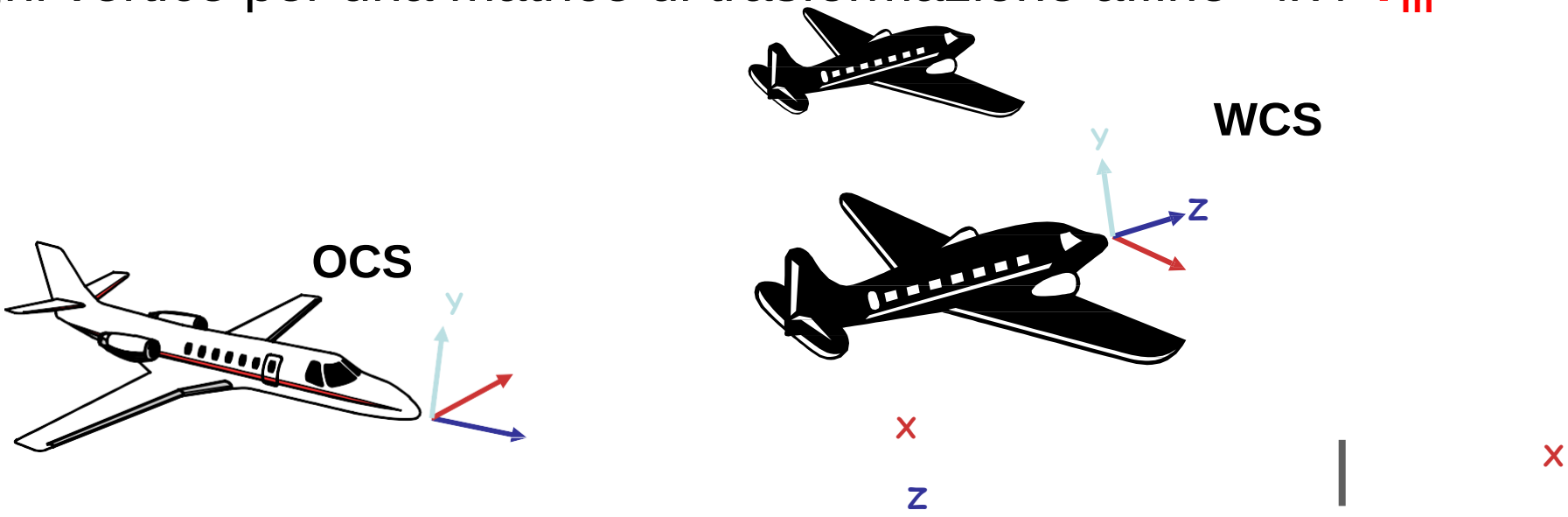
step 1. Composizione degli oggetti in scena in un sistema di riferimento destrorso WCS (sistema di coordinate del mondo);
step 2. *WCS* ⇒ *VCS* Trasformazione delle coordinate della scena da WCS al sistema di coordinate della camera (punto di vista dell'osservatore o camera (VCS)) VCS sinistrorso
step 3. Volume di vista: porzione (volume) dello spazio WCS che risulta visibile all'osservatore. Clipping: tutto quello che si trova esterno al volume risulta tagliato e quindi non visibile;
step 4. Trasformazione spazio immagine: dal volume di vista al cubo (sistema di riferimento normalizzato). Infine una **proiezione ortogonale proietta 3*D* ⇒ 2*D* su un piano immagine 2D** (piano di proiezione)
step 5. Trasformazione di viewport *window* ⇒ *viewport* trasforma l'immagine contenuta nella window in coordinate pixel viewport (corrispondente finestra sullo schermo)

Geometry stage



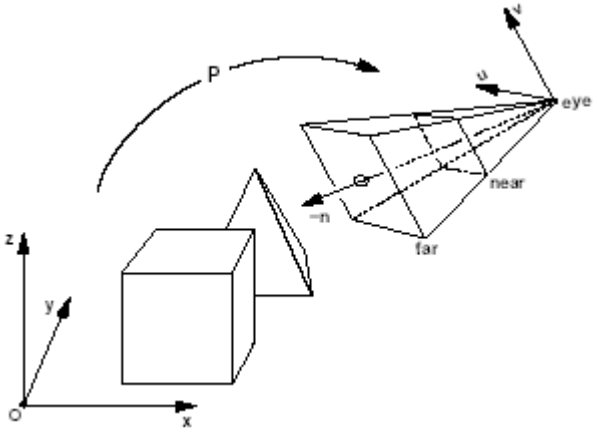
Trasformazioni di Modellazione

- Ogni oggetto è definito in un suo sistema di coordinate, Object Coordinate System”
- Le trasformazioni di modellazione permettono di muovere, orientare, e trasformare modelli geometrici all’interno di un sistema di riferimento comune, sistema di coordinate del Mondo (WCS)
- Moltiplicare ogni vertice per una matrice di trasformazione affine 4x4 $\mathbf{T_m}$



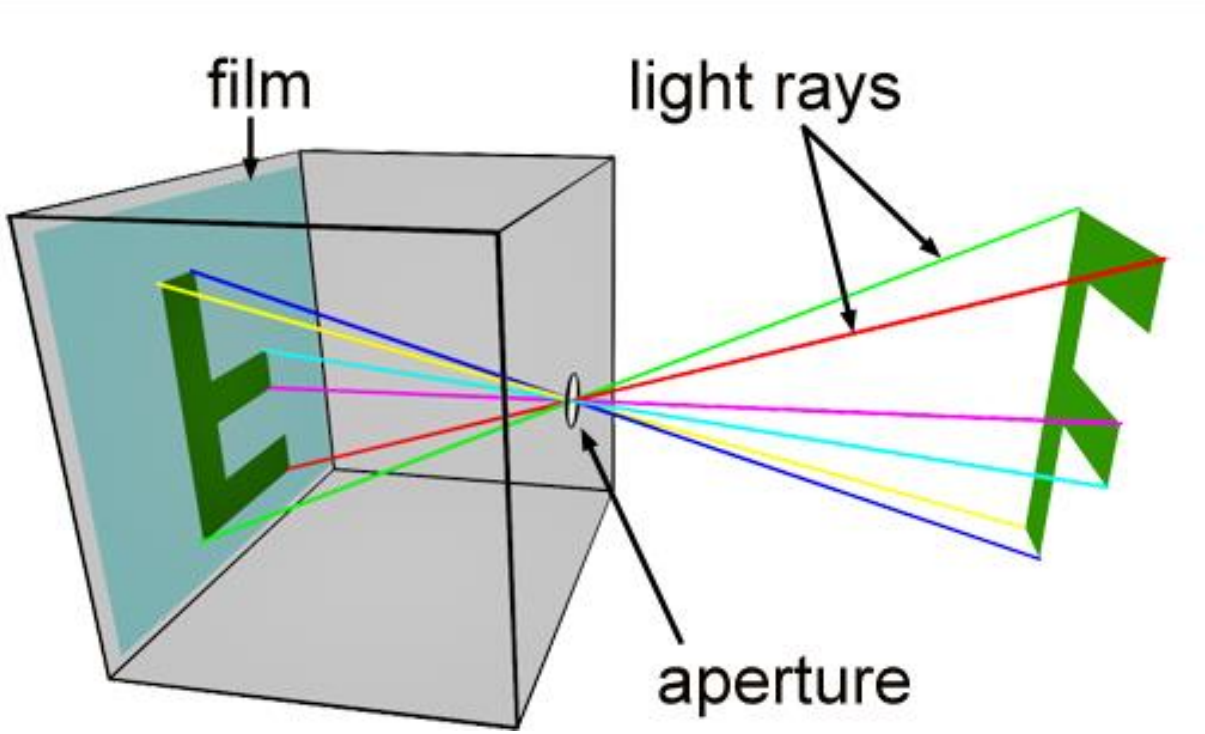
Rendering pipeline:la view transform

INPUT: **vertici in WCS**
OUTPUT:
Vertici in 3-D “**View**” or “camera” **Coordinate System (VCS) (Sistema di Coordinate della Telecamera)**

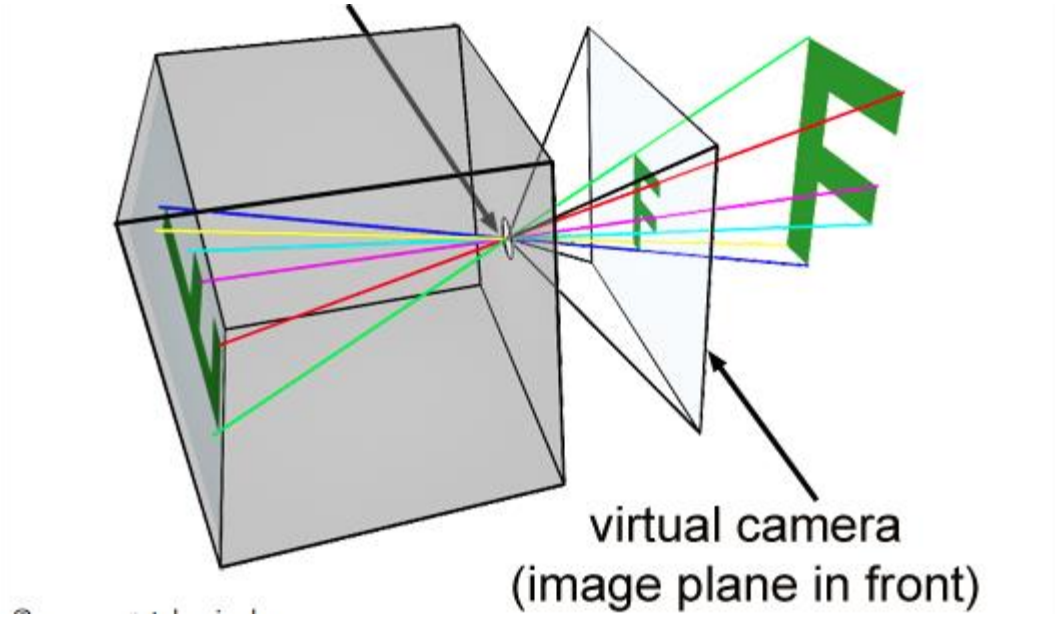


Pinhole camera

- Una pinhole camera è il **modello più semplice di camera virtuale**:
- Una scatola con piccolissimo **foro** su una parete (detto) che permette alla luce di passare **pinhole**
- La luce di una scena passa attraverso l'apertura e proietta un'immagine capovolta sul lato opposto della scatola, dove è posizionata una pellicola fotosensibile.
- La dimensione delle immagini dipende dalla distanza tra l'oggetto e il foro.



- Un’ulteriore semplificazione che si fa e quella di spostare il piano su cui avviene la proiezione dall’altro lato rispetto al pinhole, così da evitare che l’immagine sia generata capovolta. Il pinhole diventa quindi l’occhio dell’osservatore, mentre **la pellicola diventa la finestra** attraverso cui passano i raggi che chiameremo **image plane.**



Per costruire una trasformazione di vista è necessario
posizionare la camera nel WCS e orientarla opportunamente.
Questo permette di generare la **matrice del cambiamento di sistema di riferimento**.
Questa matrice di trasformazione sarà quindi utilizzata **per trasformare ogni vertice degli oggetti in scena dal WCS al VCS.**

I passi da fare sono:

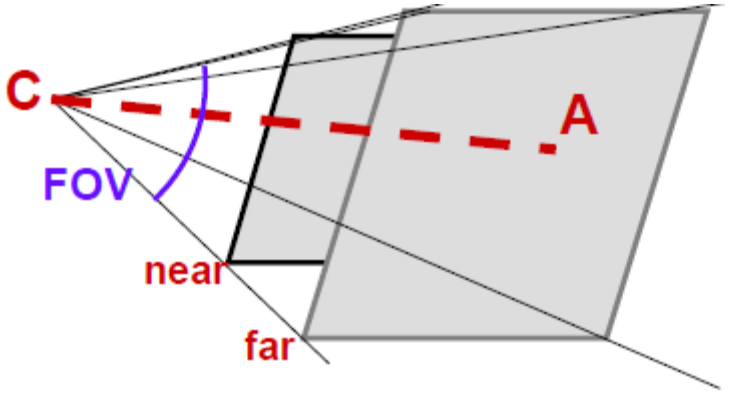
Specifica la vista 3D: impostazione della fotocamera sintetica
- La telecamera è posizionata e orientata in WCS

Costruzione della Trasformazione di Vista a partire dal posizionamento ed orientamento della telecamera.
-Creazione una matrice di trasformazione T_v (un cambio di sistema di riferimento)

- **Applicazione di questa trasformazione** a ogni vertice dell'oggetto

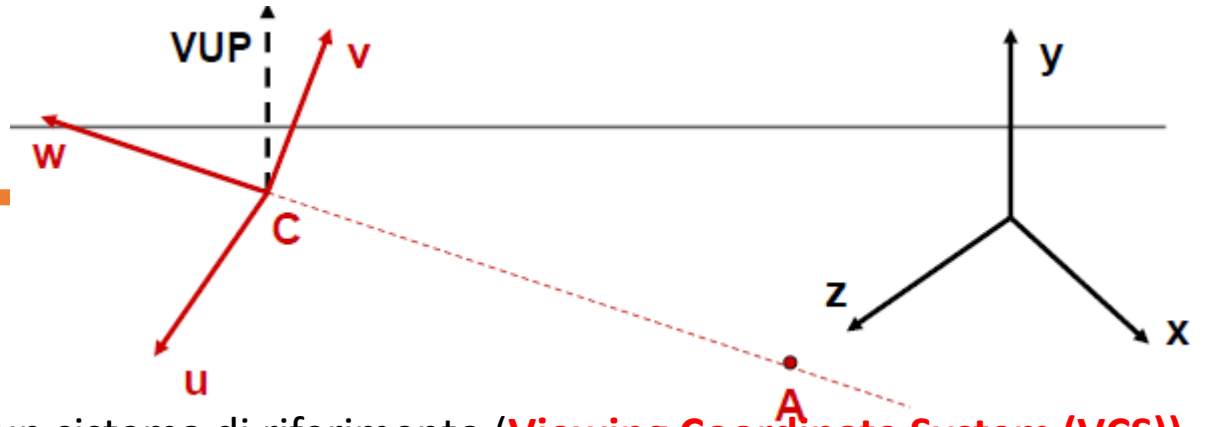
Definire la Vista : Camera “look at”

- Abbiamo bisogno di sapere quattro informazioni sul modello di fotocamera sintetica per costruire la trasformazione della vista
- **Punto C**: posizione della telecamera in WCS (da dove si sta guardando)
- **Punto A**: Il vettore A specifica in quale direzione sta puntando la telecamera (direzione di vista C-A, A = centro della scena)
- -field of view FOV (grandangolo, normale ...)
- -depth of field (piano vicino, piano lontano)



FOV: indica l’angolo di visibilità verticale nella scena espresso in radianti. Maggiore è questo valore, maggiore sarà la porzione di scena visualizzata. Di conseguenza, è possibile regolare lo zoom della camera attraverso questo parametro

Depth of field : La distanza tra l’oggetto Più vicino, e quello più lontano che appaiono perfettamente a fuoco
.



Definiamo un sistema di riferimento (**Viewing Coordinate System (VCS)**) associato all'osservatore (camera) di origine *C*, che stabilisce la posizione della camera **F(C,u,v,w)** :

Posizione della telecamera **C** (punto di vista)

asse **w** (direzione di vista) la direzione unitaria verso cui punta la fotocamera. Per convenzione, la telecamera guarda in direzione –w

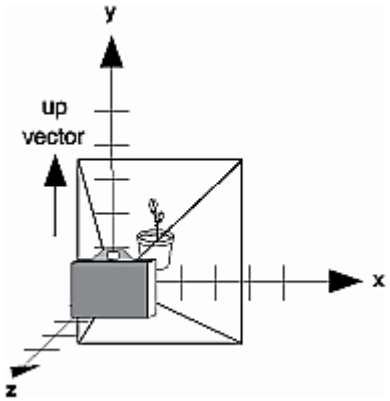
$$w = \frac{C - A}{\|C - A\|}$$

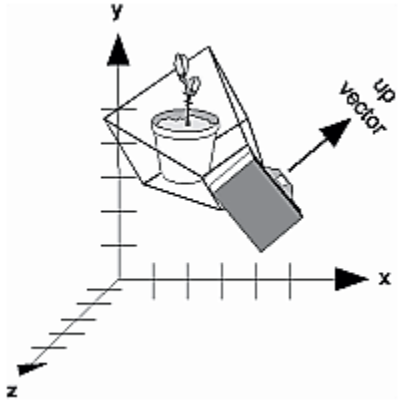
- **View Up Vector (VUP)** - determina il modo in cui la fotocamera viene ruotata attorno alla direzione di vista

.

- **Asse u**: asse unitario *u* che punta alla destra dell'osservatore, è perpendicolare sia all'asse *w* che al vettore up VUP:

$$u = \frac{VUP \times w}{\|VUP \times w\|}$$





Nel caso **VUP || all'asse y** il prodotto vettoriale può essere ottimizzato come segue:

$$[\,0\;1\;0\,] \times w = [\,w_z \;\; 0 - w_x\,]$$

L' **asse *v*** è ortogonale sia all' asse *w* che all'asse *u*, quindi

$$v = w \times u.$$

Poiché **w** ed **u** sono normalizzati, anche **v** risulta normalizzato.

$$|v| = |u||v|\cos(90°)=1$$

- Questo modo che abbiamo presentato per specificar la vista prende il nome di **Camera “look at”**
- Ci sono altri modi per specificare il tipo di vista: come simulazione di volo, telecamera rotante, ...

Costruzione della matrice di Trasformazione di Vista

- Dati i frame WCS e VCS,
- Calcolare la matrice di trasformazione di Vista T_v
- Per ogni punto P_w in coordinate WCS, convertire le sue coordinate in VCS

$$P_v = T_v P_w$$

Esprimiamo prima di tutto il sistema di riferimento della camera VCS (C,*u*, *v*,*w*) in termini del sistema WCS (O,*x*, *y*, *z*):

$$u = u_x x + u_y y + u_z z + 0 \cdot O$$
$$v = v_x x + v_y y + v_z z + 0 \cdot O$$
$$w = w_x x + w_y y + w_z z + 0 \cdot O$$
$$C = C_x x + C_y y + C_z z + 1 \cdot O$$

La matrice $M = \begin{bmatrix} u_x & v_x & w_x & C_x \\ u_y & v_y & w_y & C_y \\ u_z & v_z & w_z & C_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$ **mappa le coordinate di un punto nel sistema VCS nelle coordinate del sistema WCS**

Se un punto nel frame WCS ha coordinate omogenee $P_w = (x_P, y_P, z_P, 1)$ e nel frame VCS ha coordinate omogenee $P_v = (u_P, v_P, w_P, 1)$, allora

$$\mathbf{P_w = M\, P_v}$$

La matrice *M* ci fornisce il cambio di coordinate da frame VCS a sistema WCS, a noi serve la matrice inversa M^{-1} da WCS a VCS che chiameremo *Tv*:

$$\boldsymbol{P_v = M^{-1} P_w = T_v P_w}$$

che determina la rappresentazione di un punto P_v in coordinate omogenee in VCS data la sua rappresentazione in WCS.

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_w & x_C \\ y_u & y_v & y_w & y_C \\ z_u & z_v & z_w & z_C \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix}$$

$$P_{xyz} = \begin{bmatrix} u & v & w & C \\ 0 & 0 & 0 & 1 \end{bmatrix} P_{uvw}$$

$$P_{xyz} = M \; P_{uvw}$$

$$\begin{bmatrix} u_p \\ v_p \\ w_p \\ 1 \end{bmatrix} = \begin{bmatrix} x_u & x_v & x_w & x_C \\ y_u & y_v & y_w & y_C \\ z_u & z_v & z_w & z_C \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix}$$

$$P_{uvw} = \begin{bmatrix} u & v & w & C \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} P_{xyz}$$

$$P_{uvw} = M^{-1} P_{xyz}$$

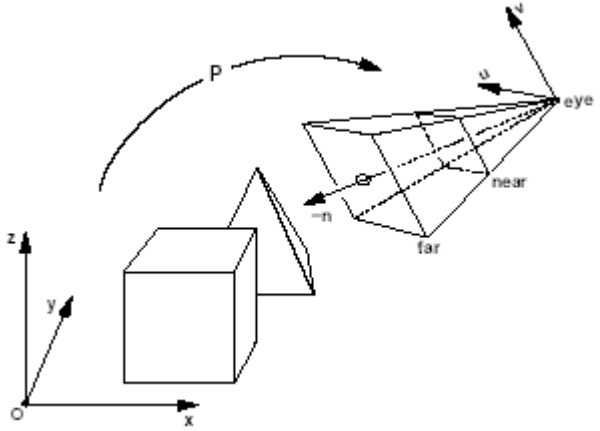
Supponiamo di voler rendere una scena con un oggetto da un certo punto di vista (camera).

- L'oggetto viene posizionato in coordinate mondo con matrice T_M, la camera è posizionata in coordinate mondo con matrice T_v.
- La seguente trasformazione prende ogni vertice dell' oggetto dal sistema di coord. locali,P, in coordinate mondo e da coordinate mondo WCS a coord. della camera VCS: $P_v = T_v T_M P$.

Stiamo "vedendo" gli oggetti in scena dall'origine del frame camera e la scena è stata trasformata per stare nell' asse negativo di *z*. Per poterli finalmente visualizzare i punti dovranno poi essere proiettati nello spazio 2D.

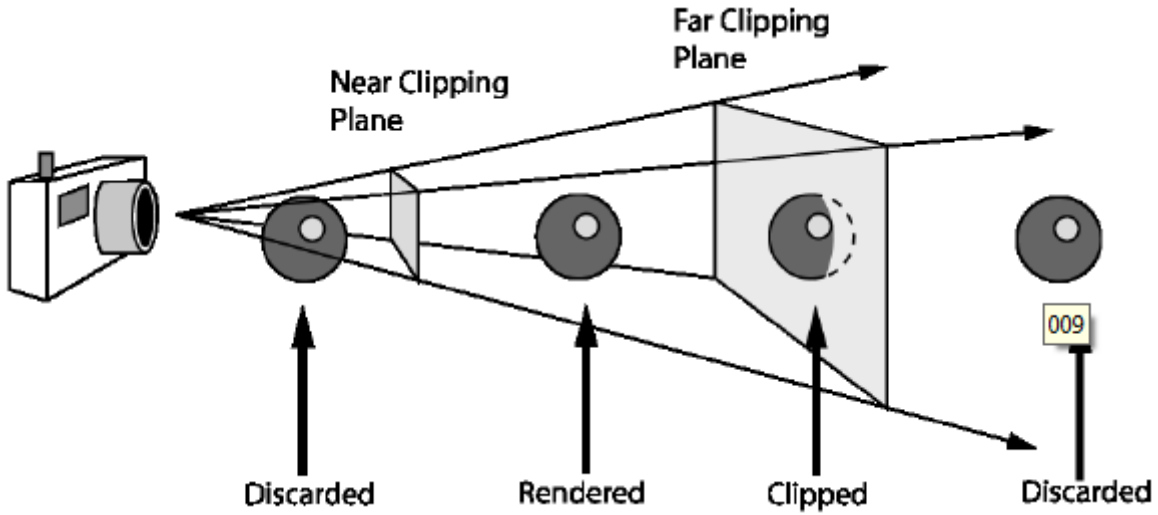
Rendering pipeline: projection transform

INPUT: vertici 3d in **VCS**
OUTPUT:
Coordinate di schermo 2D dei vertici visibili nello **Screen space**



Clipping

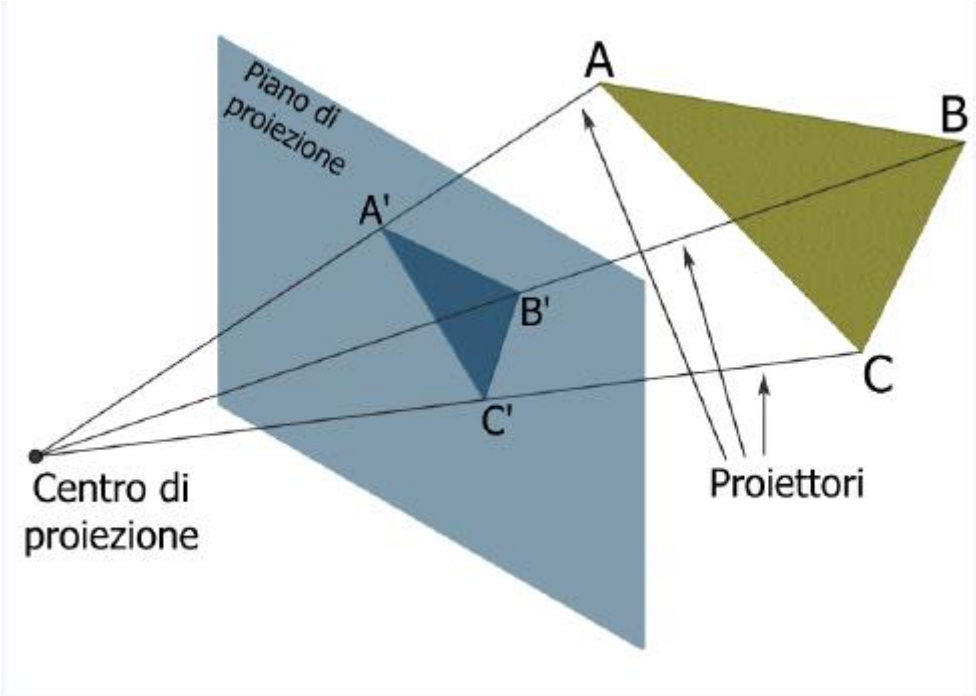
- Il volume di spazio tra i piani di clipping anteriore e posteriore definisce cosa può vedere la telecamera
- • Posizione dei piani definita dalla distanza lungo la direzione di vista
  • Gli oggetti che appaiono al di fuori del volume della vista non vengono disegnati
  • Gli oggetti che intersecano il volume della vista vengono tagliati



Le proiezioni

- Si dice *proiezione* una trasformazione geometrica con il dominio in uno spazio di dimensione *n* ed il codominio in uno spazio di dimensione *n*–1 (o minore)
- A noi interessano solo le proiezioni da 3 a 2 dimensioni

- La proiezione di un oggetto 3D è definita da un insieme di **rette di proiezione** (dette *proiettori*)
- aventi origine comune da un centro di proiezione,



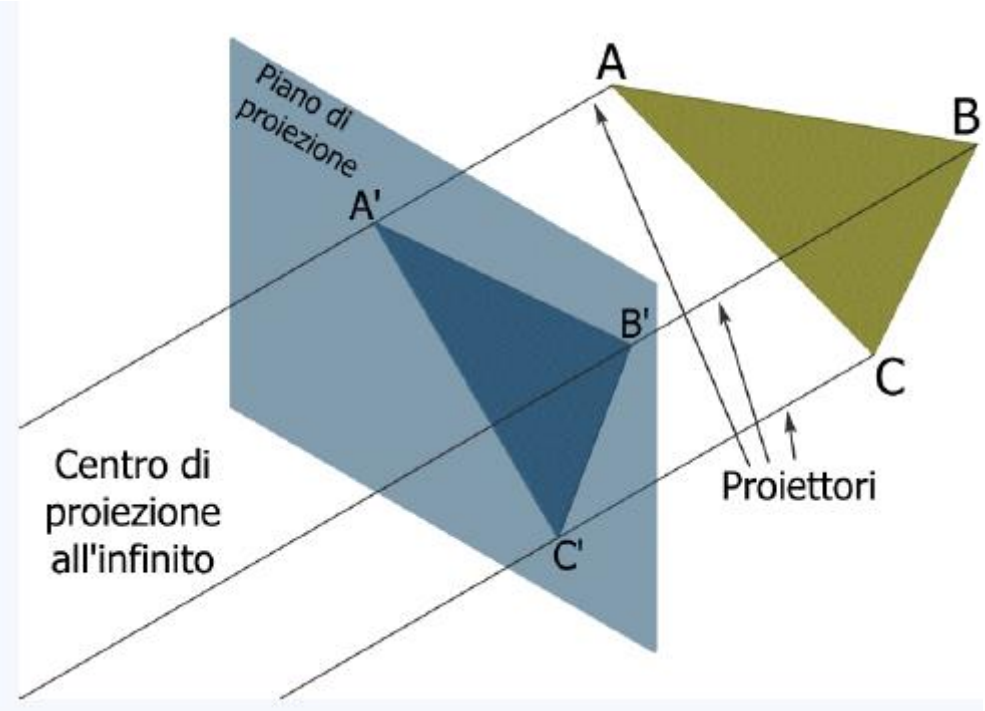
che passano per tutti i punti dell'oggetto e intersecano un piano di proiezione per formare la proiezione vera e propria.

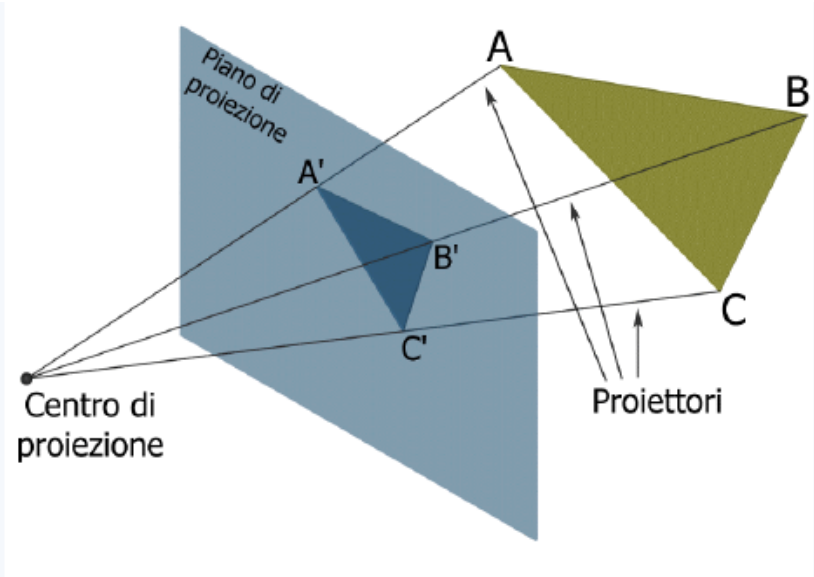
- Questo tipo di proiezioni, caratterizzate dal fatto che:
  - I proiettori sono rette (potrebbero essere curve generiche)
  - La proiezione è su di un piano (potrebbe essere su una superficie generica)
- si chiamano **proiezioni geometriche piane**.

- Ci sono due classi di base in cui si possono suddividere le proiezioni geometriche piane:
  - prospettiche
  - parallele

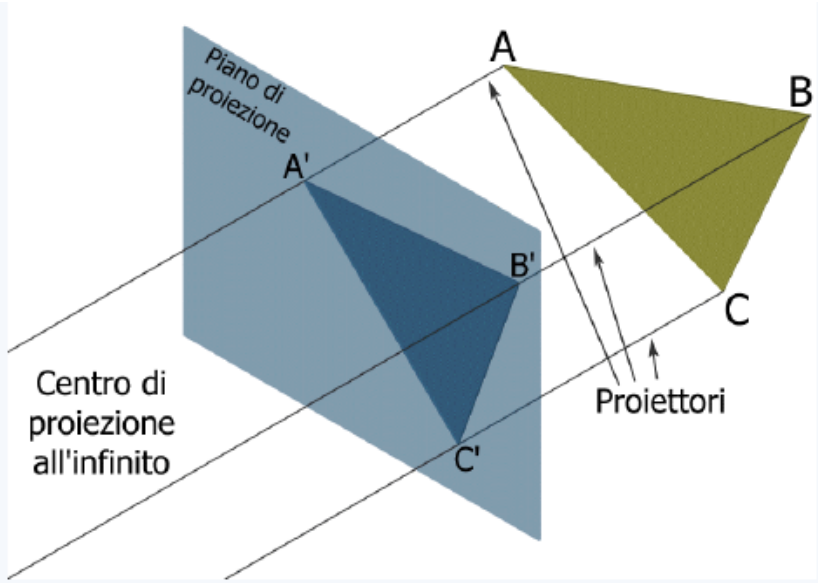
- La differenza tra le due classi è data dalla distanza tra il centro di proiezione ed il piano di proiezione: se tale distanza è finita la proiezione è prospettica, se è infinita è parallela

- Le proiezioni parallele devono il loro nome ai proiettori (che sono paralleli)
- Mentre per una proiezione prospettica si specifica un *centro di proiezione*, nel caso delle proiezioni parallele si parla di una *direzione di proiezione*.



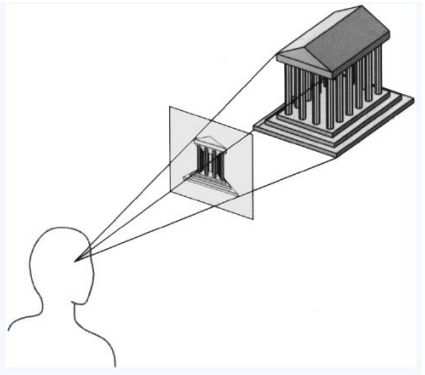


Proiezioni prospettiche



Proiezioni parallele

- La proiezione prospettica è la più **realistica**, in quanto riesce a riprodurre il modo con cui nella realtà vediamo gli oggetti: **oggetti più grandi sono più vicini all'osservatore, oggetti più piccoli sono più lontani.**
- Inoltre le distanze uguali lungo una linea non vengono proiettate su distanze uguali sul piano dell'immagine
- (non è una trasformazione affine)
- • Gli angoli sono conservati solo su piani paralleli al piano di proiezione.



- Le proiezioni parallele fanno sì **che linee parallele nel modello tridimensionale rimangono tali nella proiezione**. Sono utilizzate nel disegno tecnico perché si ha la necessità di compiere misurazioni sul risultato della proiezione.

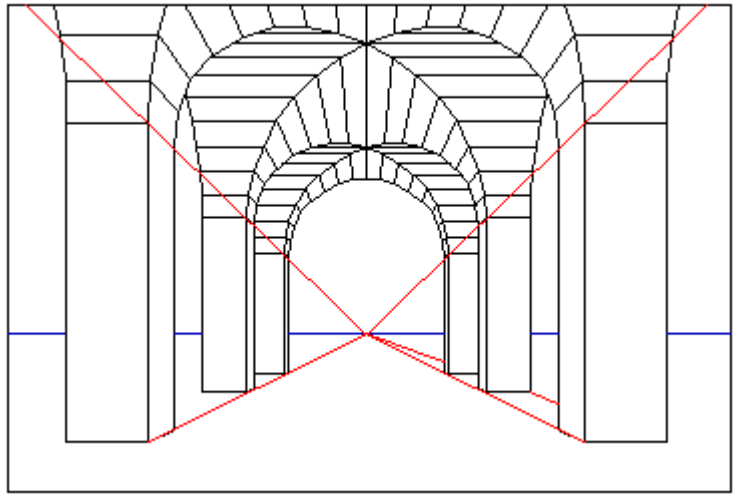
Proiezioni Prospettiche

Sono caratterizzate dai vanishing point.

- La proiezione di ogni insieme di linee parallele non parallele al piano di proiezione converge in un punto detto *vanishing point* (punto di convergenza).
- Il numero di questi punti è infinito, come il numero delle possibili direzioni di fasci di rette parallele.

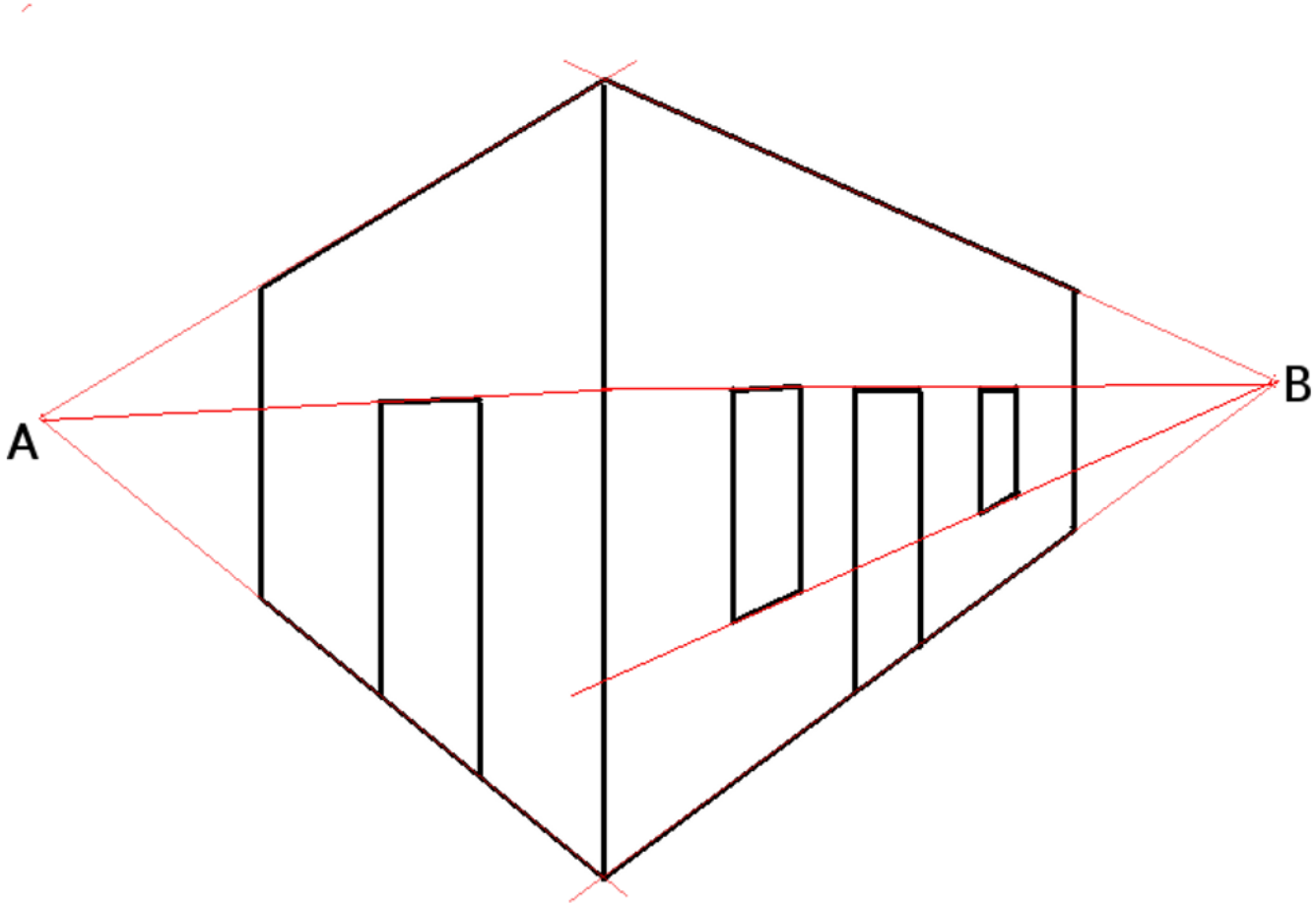
- Se **l'insieme di linee parallele è a sua volta parallelo ad uno degli assi coordinati il punto di convergenza si chiama *axis vanishing point***.
- Di questi punti ce ne possono essere al più tre.
- Le proiezioni si possono classificare in base al numero di vanishing point principali (che è il numero di assi del sistema di coordinate che intersecano il piano di proiezione).

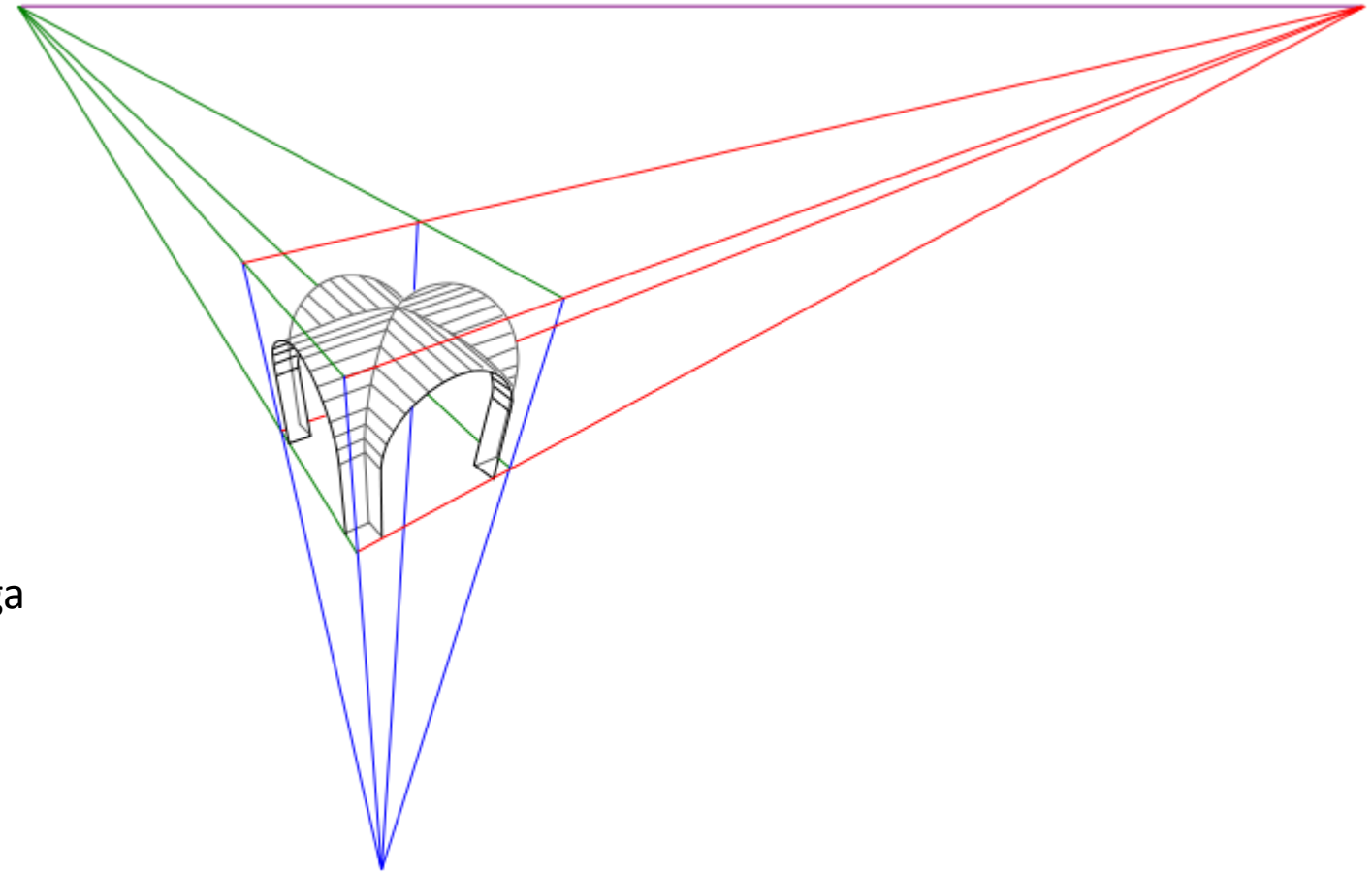
Se il piano di vista è parallelo ad **un piano coordinato** la prospettiva si dice ad 1 punto di fuga.



1 Punti di fuga

Se il piano di vista è parallelo ad **un asse coordinato** la prospettiva si dice a 2 punti di fuga

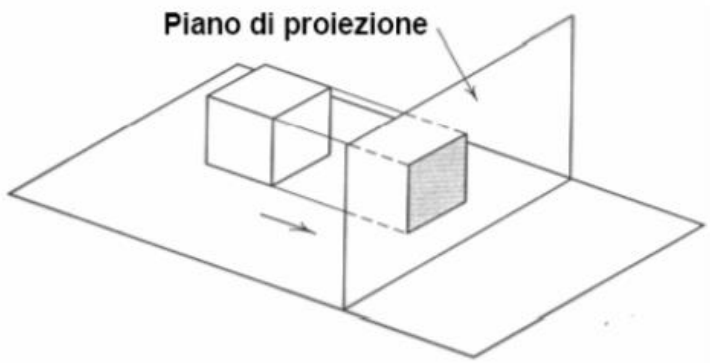




3 punti di fuga

Proiezioni Parallele

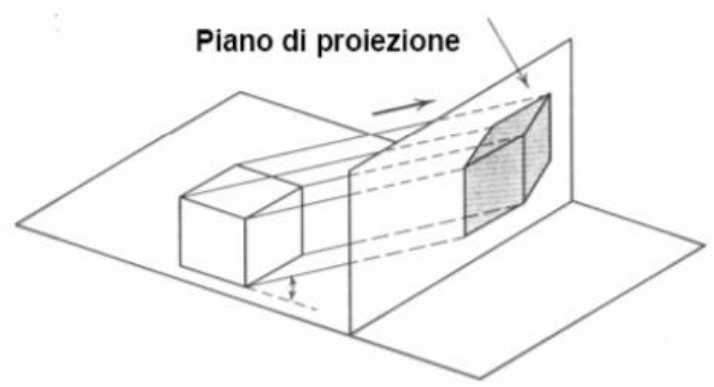
- Si classificano in base alla relazione che c’è tra la direzione di proiezione e la normale al piano di proiezione.



Ortografica

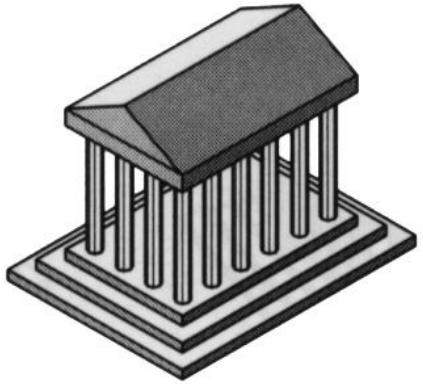
direzione di proiezione coincide con la normale al piano di proiezione si parla di *proiezione ortografiche*.

Obliqua



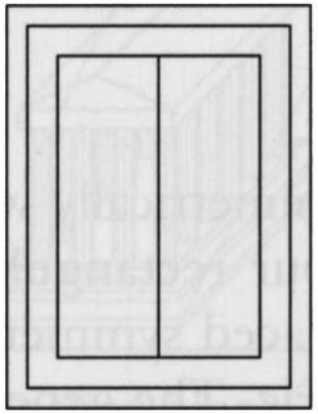
Nel caso contrario si parla di *proiezione obliqua*.

Proiezioni Ortografiche: mostrano solo una faccia

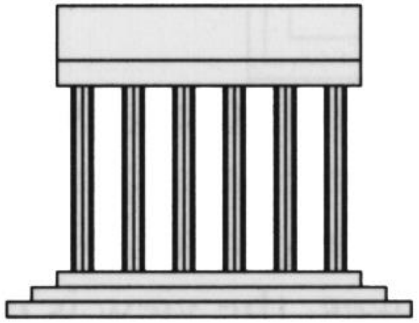


Vista prospettica

Vista frontale



Vista dall'alto

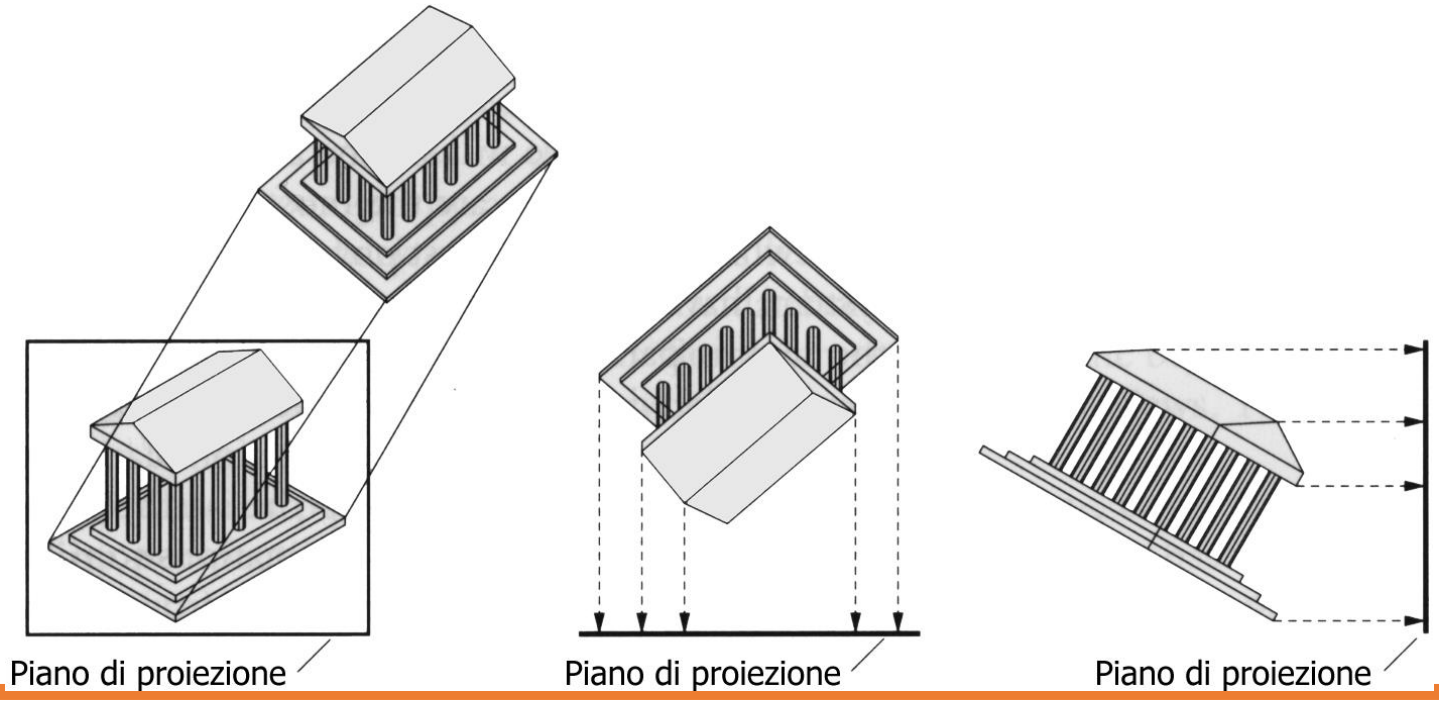


Vista laterale

Direzione di proiezione allineata ad un asse principale

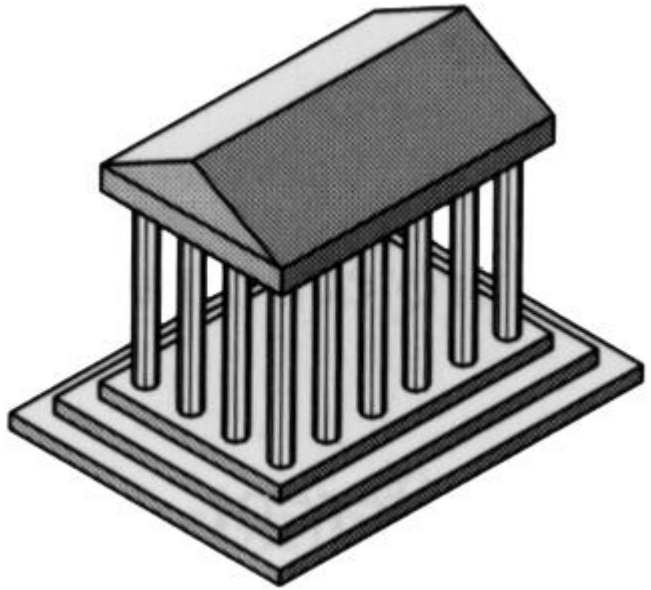
Proiezioni ortografiche assonometriche

- Sono ancora proiezioni **ortografiche**, ma **non essendo la direzione di proiezione allineata con un asse principale, mostrano facce diverse di un oggetto**, assomigliando in questo alle proiezioni prospettiche.



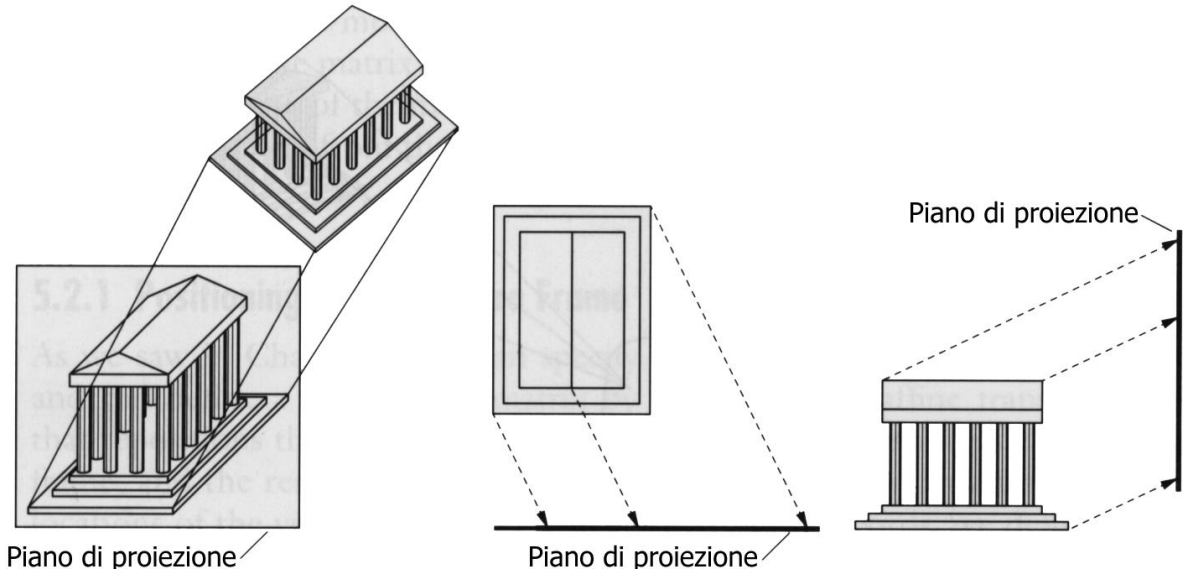
Proiezione Isometrica

- Assonometria isometrica: la direzione di proiezione è identificata da una delle bisettrici degli ottanti dello spazio cartesiano.



Proiezioni Oblique

- Sono il tipo più generale di proiezioni parallele, caratterizzate dal fatto che i **proiettori possono formare un angolo qualsiasi con il piano di proiezione**.

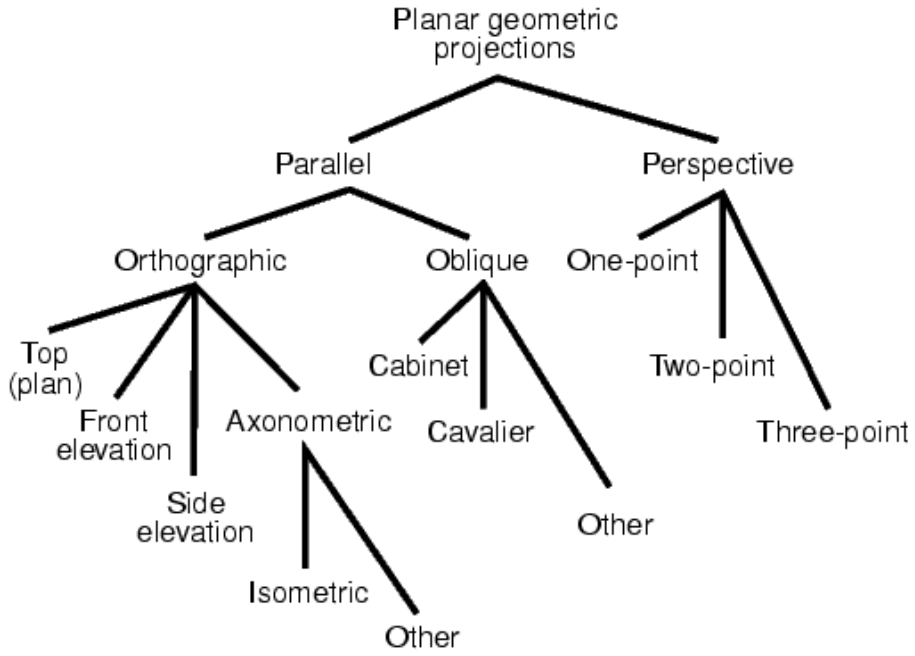


- I due più frequenti tipi di proiezione obliqua sono la
- Cavaliera, in cui la direzione di proiezione forma un angolo di **45°** con il piano di proiezione e quindi linee ortogonali al piano conservano la loro lunghezza.
- Cabinet: la direzione di proiezione forma un angolo di artg(2) = **63.4°** e quindi linee perpendicolari al piano hanno lunghezza pari alla metà di quella reale.

Riassunto Proiezioni Planari Geometriche

Tutte le proiezioni richiedono:

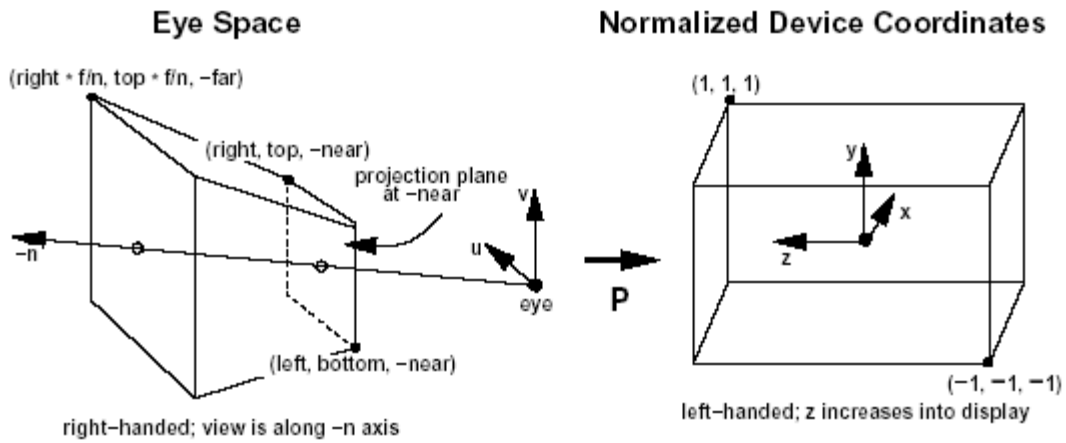
- piano di proiezione.
- posizione del centro di proiezione.
- distanza centro di proiezione-piano:
  - Finita → prospettiche.
  - Infinita → parallele



Prima di una proiezione geometrica piana da 3D -> 2D:

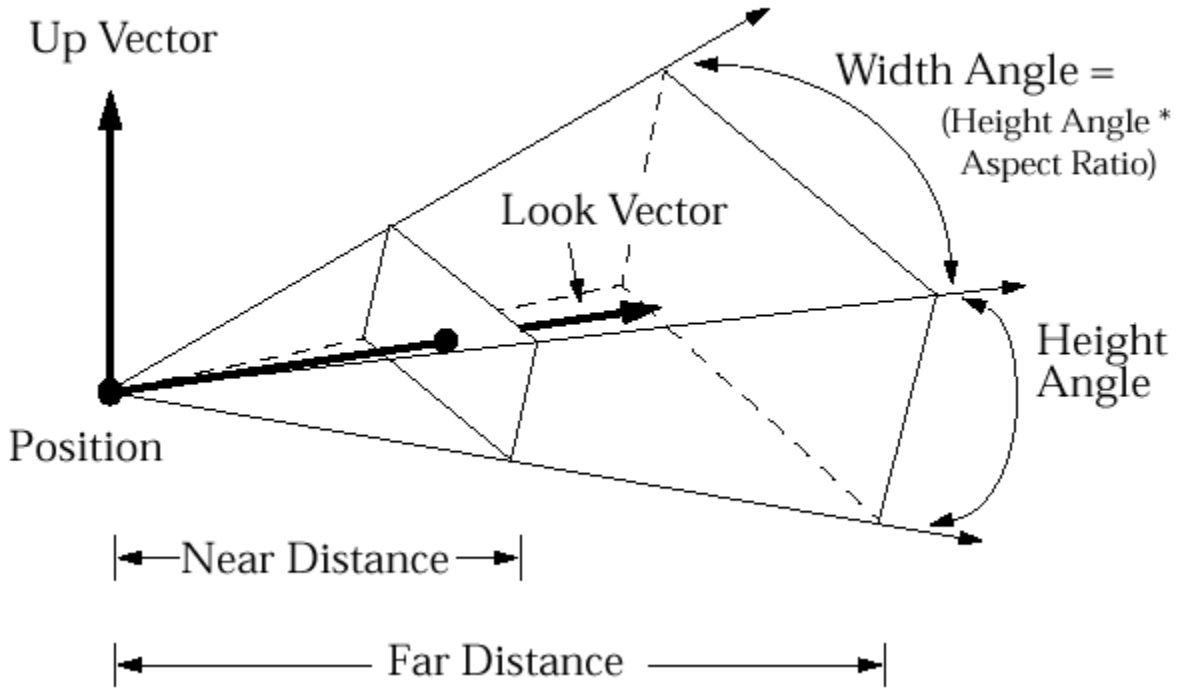
1. **Determinare il volume di spazio tra i piani di clipping anteriore e posteriore** che definisce lo spazio limitato che la telecamera può "vedere" (volume di vista)
Il tipo di proiezione definisce la forma del volume della vista

2. **Proiettare il volume di vista nel sistema di coordinate normalizzate**, cuboide con il centro nell'origine definito in [-1,1]x[-1,1]x[-1,1];



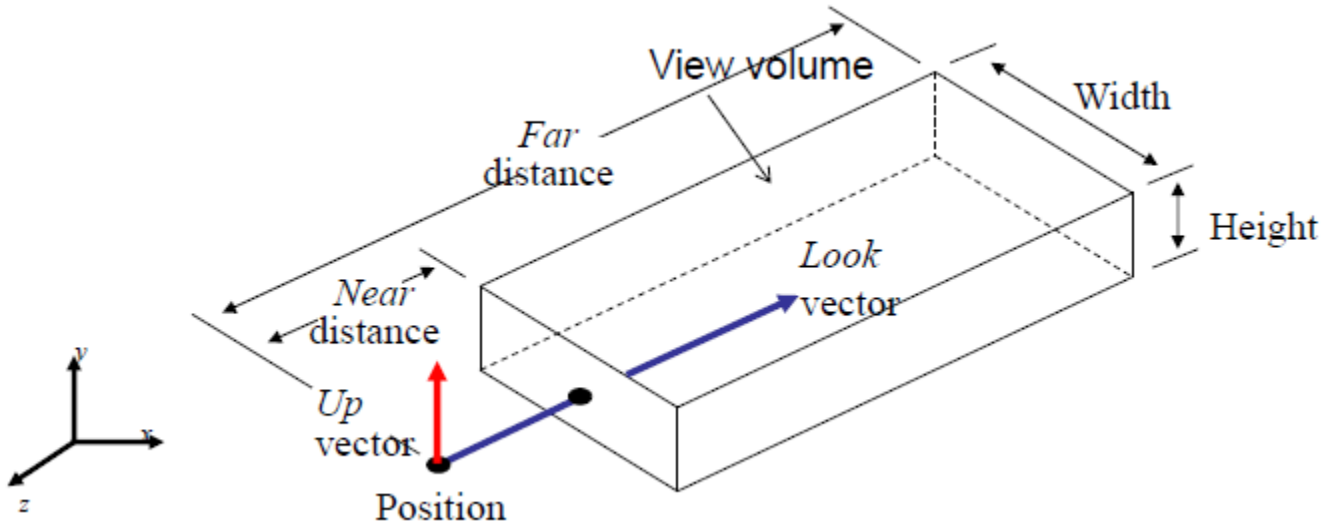
Il clipping rispetto ad un cubo con assi paralleli agli assi coordinate è più semplice da realizzare.

Volume di vista: caso proiezione prospettica:piramide troncata o frustum



Volume di vista: caso proiezione parallela ortografica

volume di vista troncato, cuboide



Normalizzazione

Proiettare il volume di vista nella sistema di coordinate normalizzato.

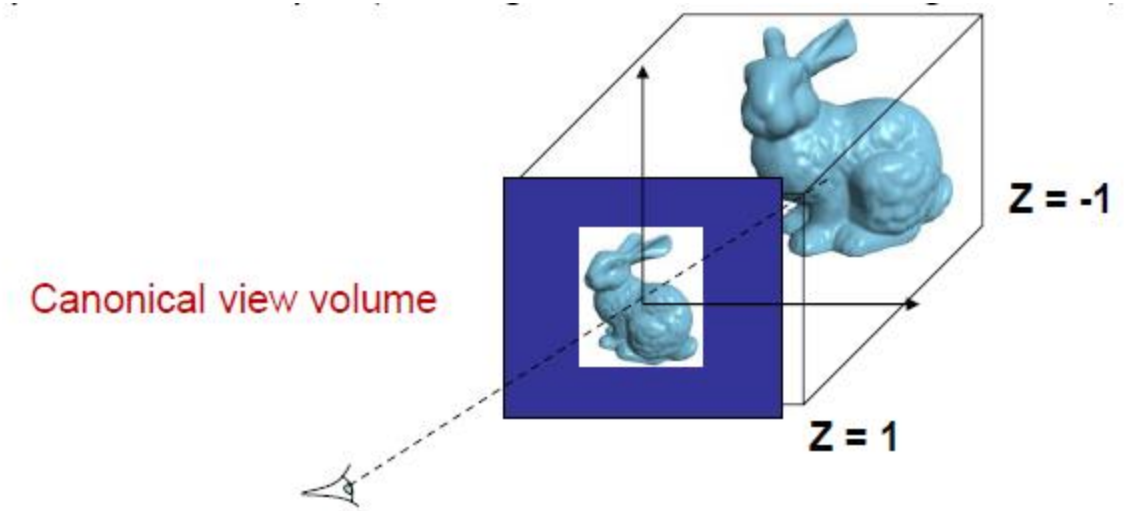
Trasformare il volume di vista in un volume di vista parallelo (cubo) (Spazio immagine)

Perché si fa?

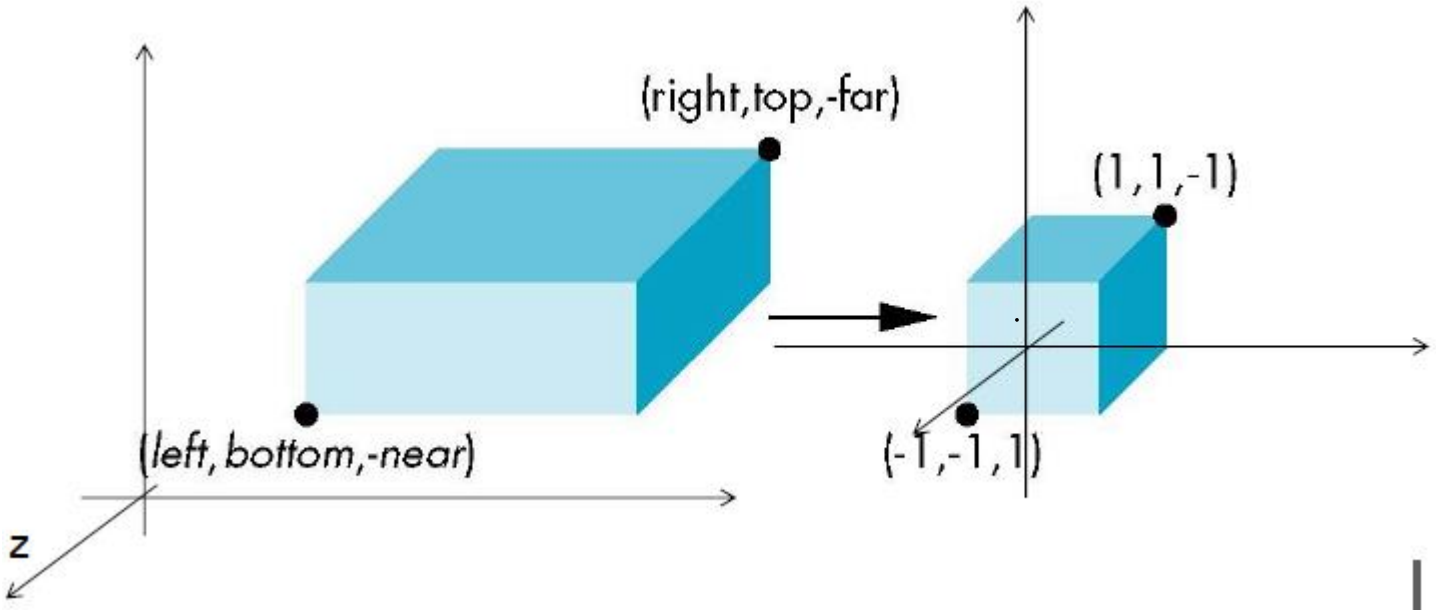
- La normalizzazione consente una **singola pipeline sia per la visualizzazione prospettica che per la visualizzazione ortografiche**
- Rimaniamo in quattro coordinate omogenee dimensionali il più a lungo possibile per conservare le informazioni tridimensionali necessarie per la rimozione delle superfici nascoste
- Semplifichiamo sia il clipping che la proiezione

Proiettare il volume di vista nel sistema di coordinate normalizzato **(Image Space) o** Spazio immagine

La scena è delimitata da un cuboide centrato nell' origine con coord. x, y in [-1, 1] e anche z va da -1 a 1 che rappresenta la profondità (1 più vicino e -1 più lontano)
La visualizzazione 2D finale della scena 3D (l'immagine finale) verrà infine calcolata proiettando la porzione di scena contenuta nel volume della vista canonica in una finestra nel piano dell'immagine

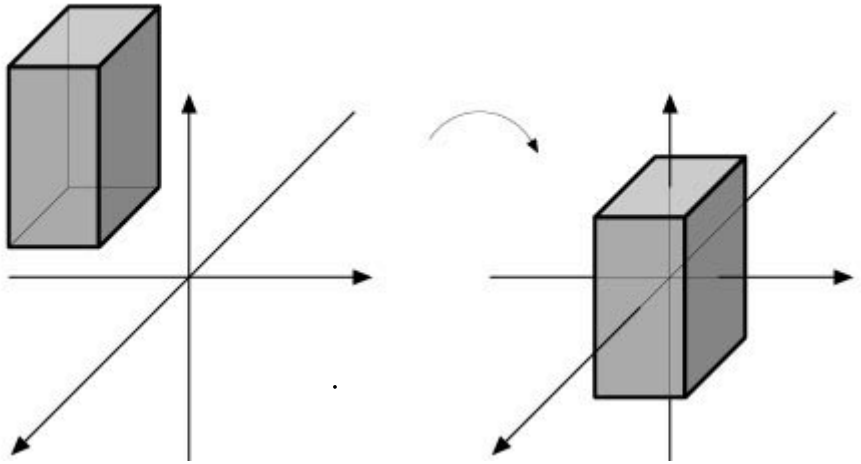


Normalizzazione del volume di vista nel caso di proiezione ortogonale

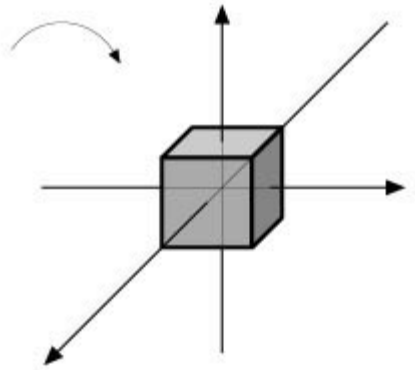


Due passi

- Spostare il centro nelll'origine
T (- (left+ right) / 2, - (bottom+ top) / 2, (near + far) / 2))



- Scala per avere lati di lunghezza 2
S (2 / (right-left), 2 / (top-bottom), 2 / (near-far))

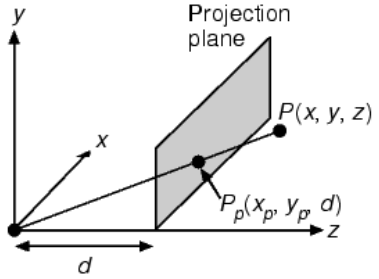


$$\mathbf{v}' = \mathbf{P} \cdot T_v \cdot T_m \cdot \mathbf{v}$$

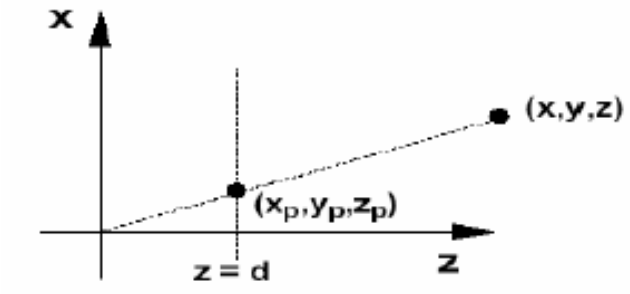
$$\mathbf{P}(left, right, top, bottom, near, far) = \begin{bmatrix} \frac{2}{right - left} & 0 & 0 & -\frac{right + left}{right - left} \\ 0 & \frac{2}{top - bottom} & 0 & -\frac{top + bottom}{top - bottom} \\ 0 & 0 & \frac{2}{near - far} & \frac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matematica delle proiezioni

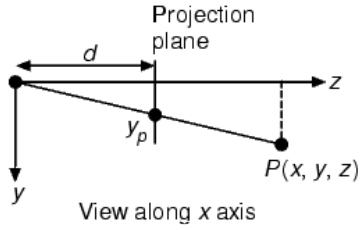
Proiezione sul piano z=d



$x:x_p=z:d$



$$x_p = \frac{d \cdot x}{z} = \frac{x}{z/d}$$



$y:y_p=z:d$

$$y_p = \frac{d \cdot y}{z} = \frac{y}{z/d}$$

$$z_p = d$$

- In coordinate omogenee possiamo esprimere la precedente proiezione sul piano z=d come una matrice di trasformazione 4x4
- Matrice di proiezione prospettica
- $p = (x, y, z, 1)$
- q= $(x_p, y_p, z_p, 1)$

$$P_p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix}$$

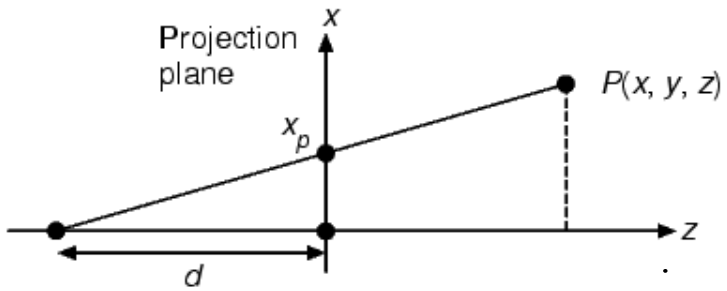
$$q = P_p\, p$$

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

Passando alle coordinate omogenee, (facendo sì che la 4 coordinata sia 1)

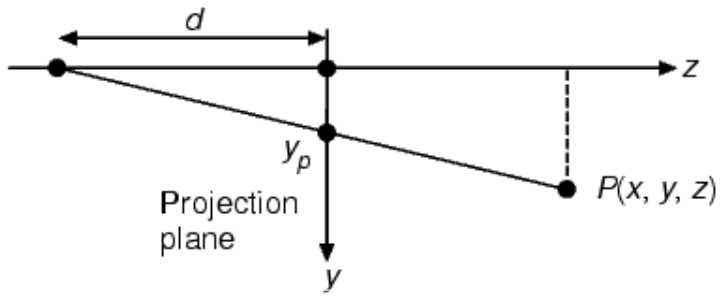
$$\begin{pmatrix} x * d/z \\ y * d/z \\ d \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z/d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Punto di vista (0 0 -d), piano di proiezione z=0:



$x:x_p=d+z:d$

$$x_p = \frac{d \cdot x}{z+d} = \frac{x}{(z/d)+1}$$



$y:y_p=d+z:d$

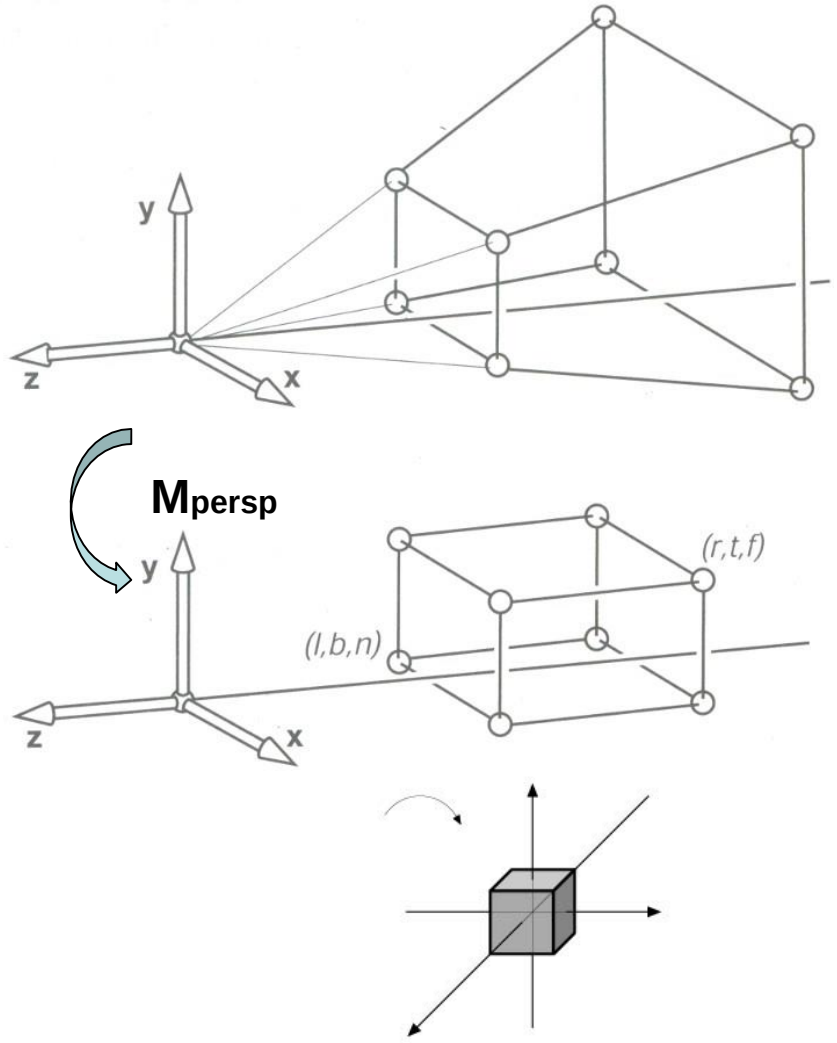
$$y_p = \frac{d \cdot y}{z+d} = \frac{y}{(z/d)+1}$$

$$\begin{pmatrix} x \\ y \\ 0 \\ (z+d)/d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x*d/(z+d) \\ y*d/(z+d) \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \\ (z+d)/d \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

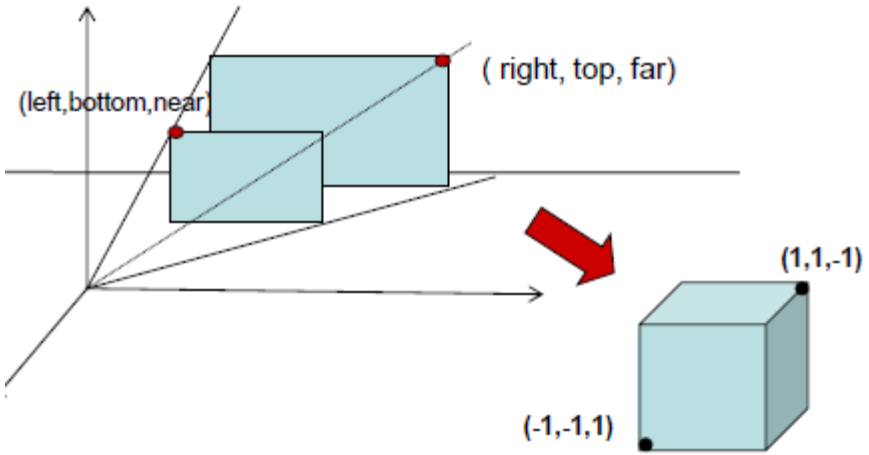
$$M'_{per} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/d & 1 \end{pmatrix}$$

Normalizzazione del volume di vista nel caso di proiezione prospettica



- Nel caso di proiezione prospettica possiamo pensare al volume di vista come ad un cubo distorto, poiché esso ha sei facce, ognuna con 4 spigoli.
- Dobbiamo individuare la matrice di proiezione prospettica M_{persp} che lascia inalterati i punti sul piano z=n e mappa il rettangolo grande z=f posto sul retro del volume prospettico nel rettangolo piccolo sul retro del volume ortografico .
- Applicare la matrice di normalizzazione del caso di proiezione ortogonale per mappare il parallelepipedo ottenuto dalla fase precedente in un cubo con centro l'origine e x,y,z in [-1,1]

$$\mathbf{P = P_{ortho} M_{persp}}$$



$V=P*T_V*T_m$

$$P=\begin{bmatrix} \frac{2*near}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2*near}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{-2far*near}{far-near} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

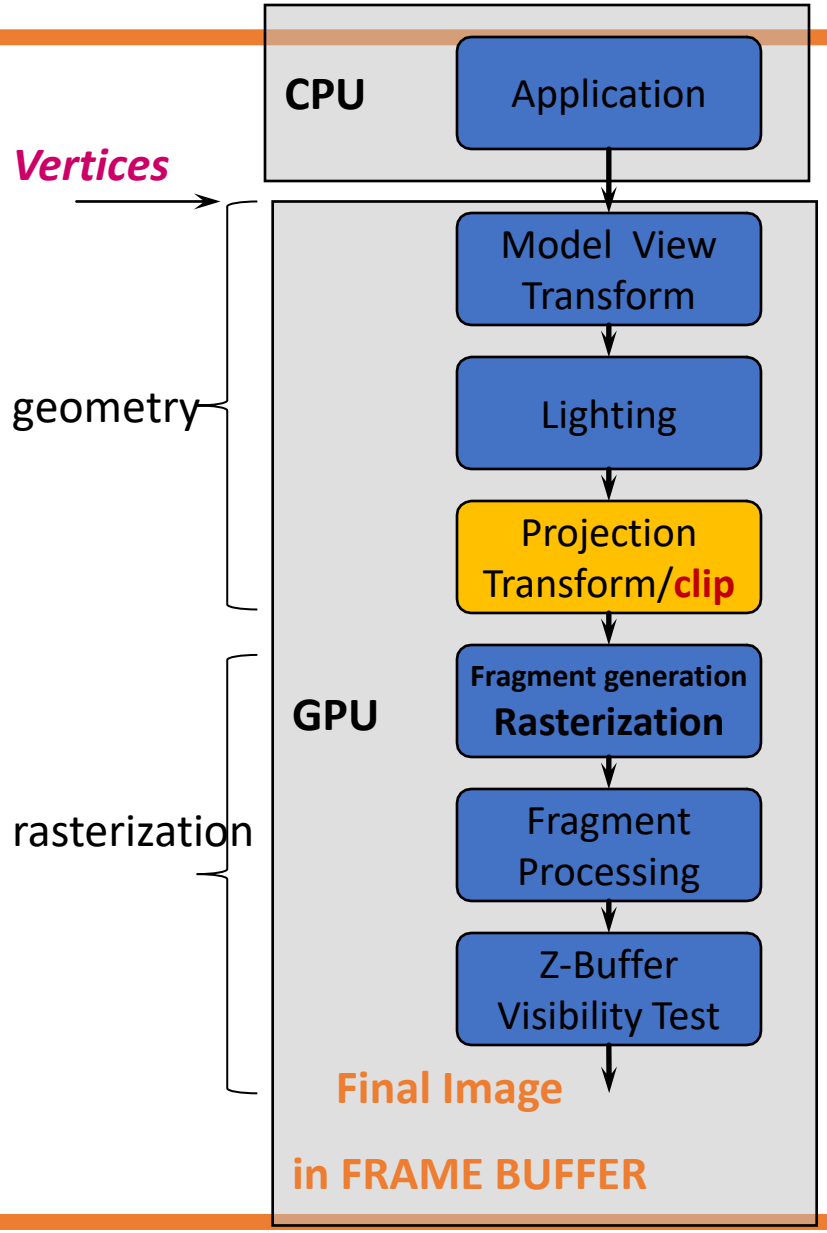
Se osserviamo la matrice prospettica, vediamo che non ha [0 0 0 1] nella riga inferiore

Ciò significa che quando trasformiamo un vettore di posizione 3D [vx vy vz 1], non finiremo necessariamente con un 1 nella quarta componente del vettore risultato

Invece, si ottiene un vero vettore 4D [vx' vy' vz' vw']
Il passaggio finale della proiezione prospettica è mappare questo vettore 4D nel vettore con quarta coordinate pari ad 1:

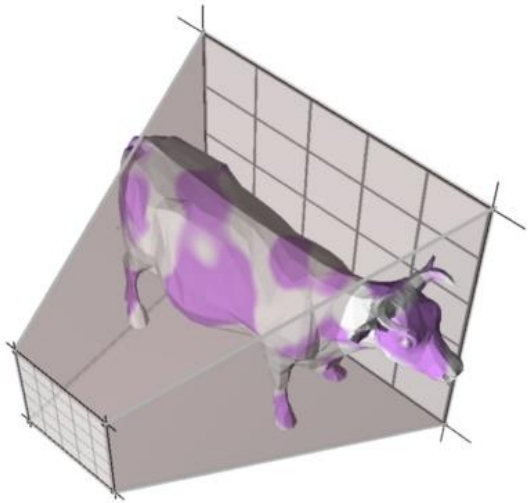
$$\begin{bmatrix} v_x & v_y & v_z & v_w \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{v_x}{v_w} & \frac{v_y}{v_w} & \frac{v_z}{v_w} \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{v_x}{v_w} & \frac{v_y}{v_w} & \frac{v_z}{v_w} & 1 \end{bmatrix}$$

Geometry stage: Clipping

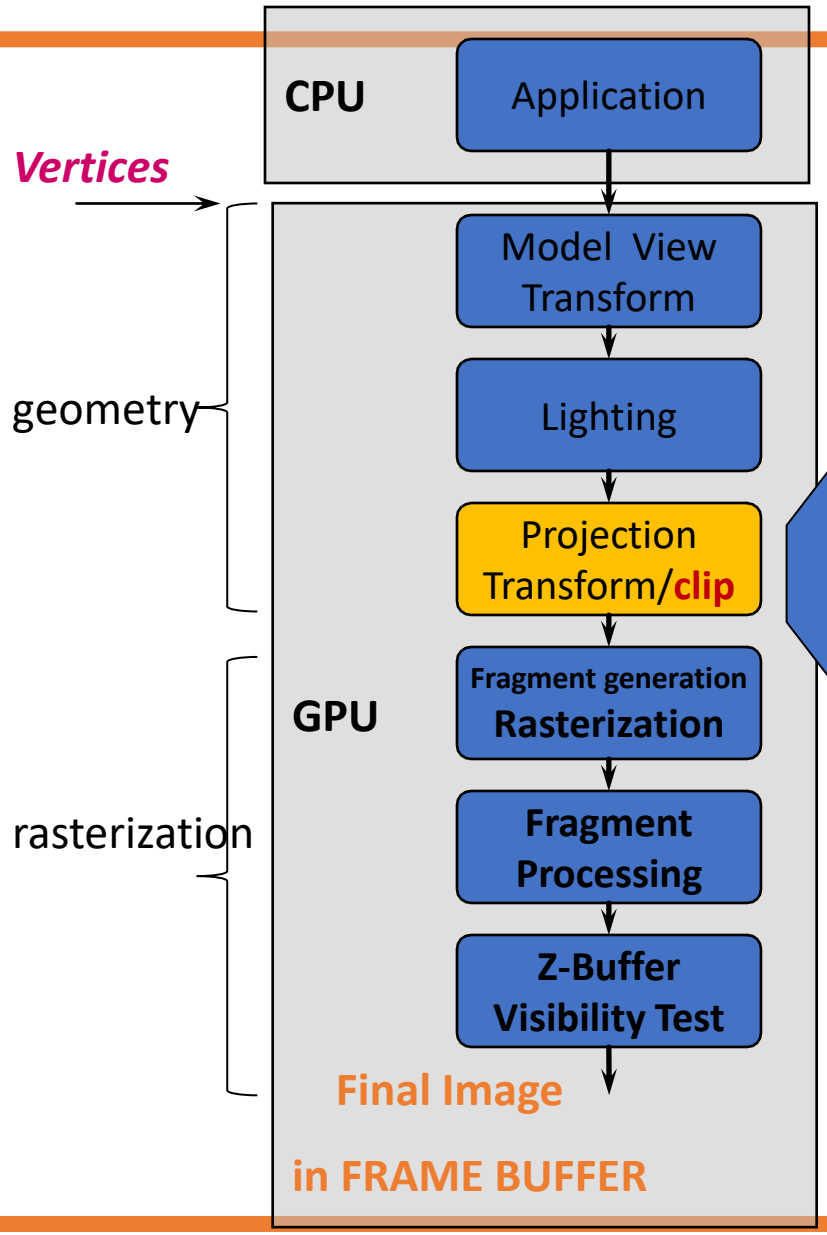


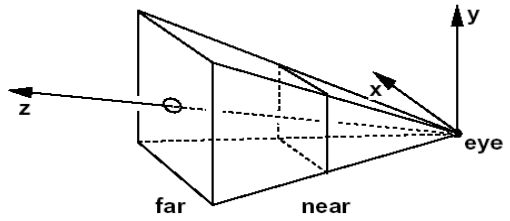
Bisogna tagliare la scena intorno alle facce del volume di vista

Le porzioni degli oggetti fuori dal volume di vista vengono rimosse

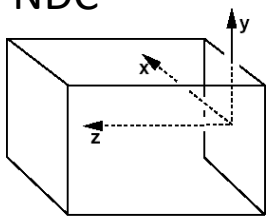


Clipping: quando?





projection

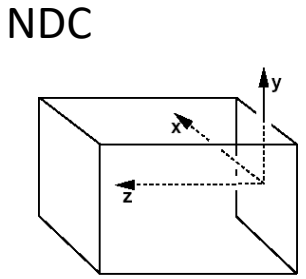


Opengl, per convenzione, durante la proiezione, inverte l'asse z del cuboide generando un left handed coordinate system

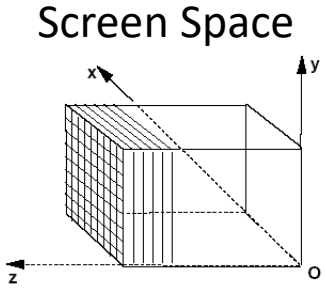
Trasformazione di Proiezione (Prospettica/Ortografica)

Clipping

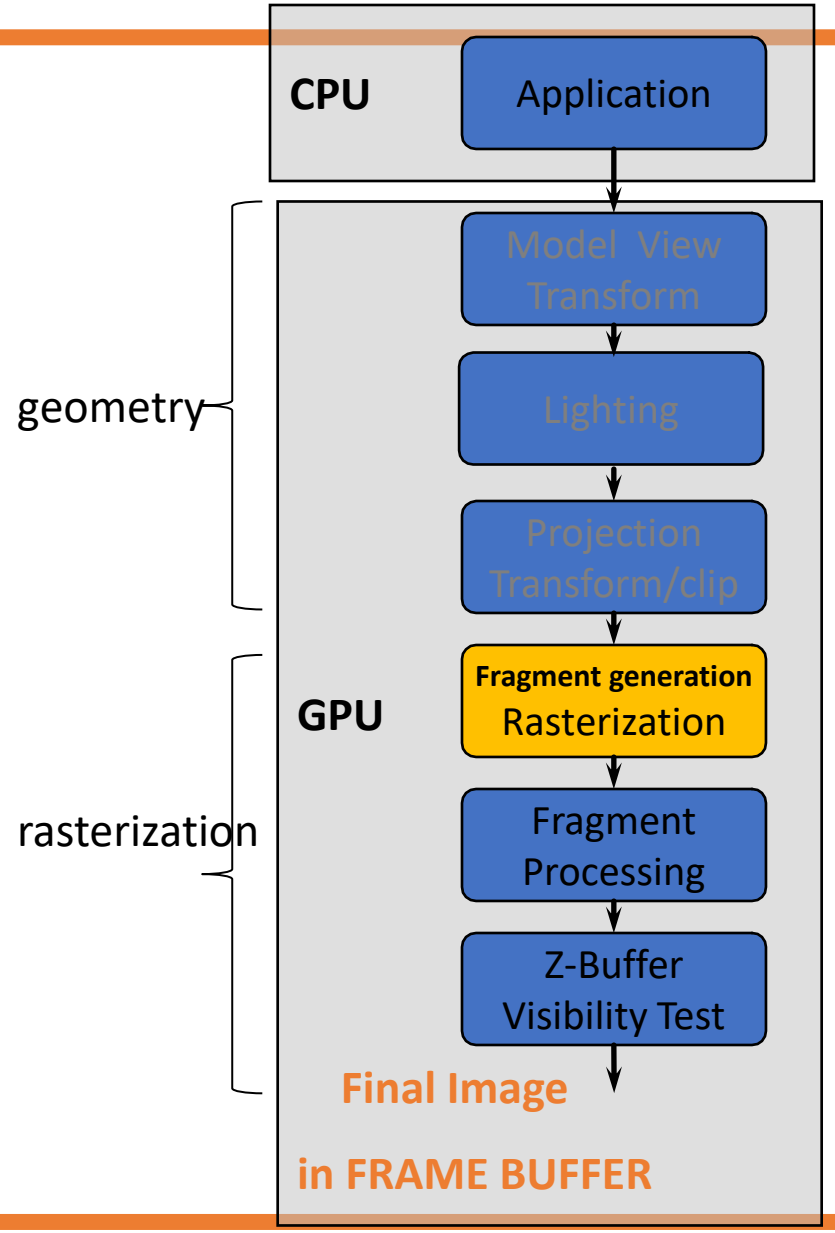
Proiezione (sullo spazio dello schermo)



projection

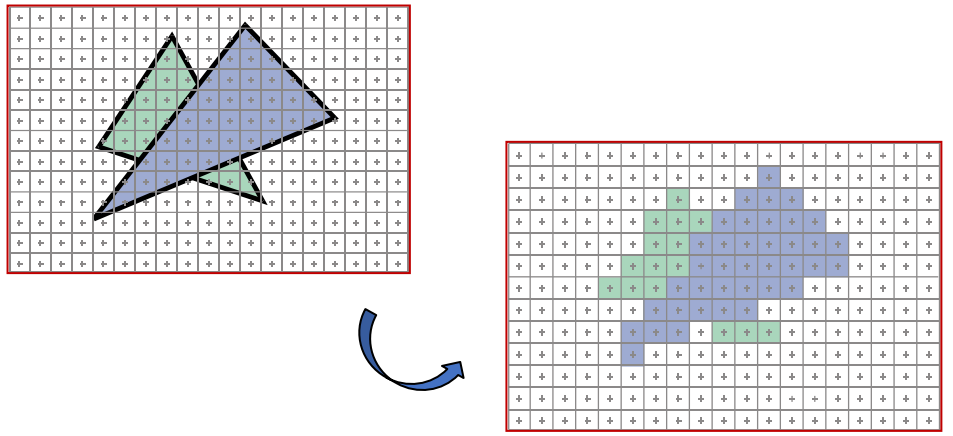


Scan Conversion (Rasterization)

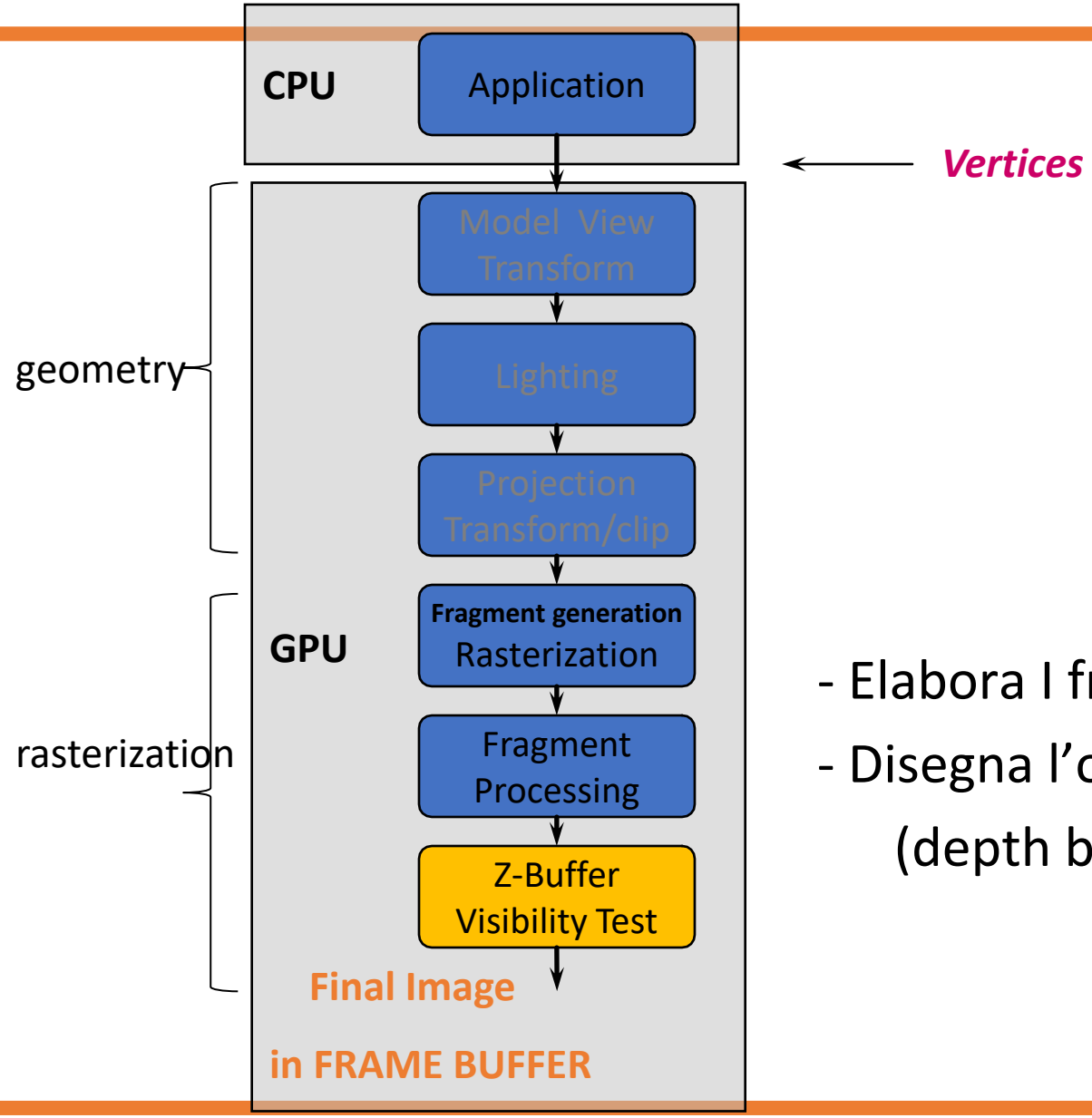


← ***Vertici***

- Determina quali pixels sono interni alla primitiva specificata da un insieme di vertici
- Produce un insieme di frammenti.
- **I Frammenti** hanno una posizione (pixel location) e altri attributi, come colore e coordinate di texture che vengono determinati interpolando i valori sui vertici.



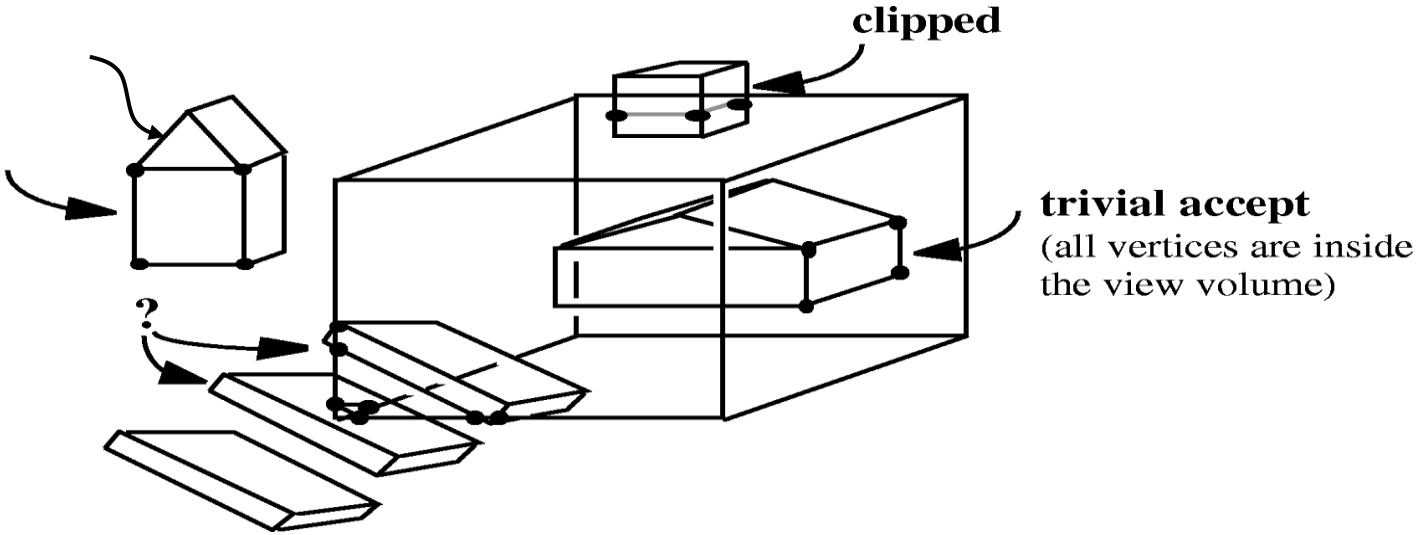
Visibility / Display



- Elabora I frammenti(texturing)
- Disegna l'oggetto più vicino.
  (depth buffer)

Clipping Rispetto al Volume di vista

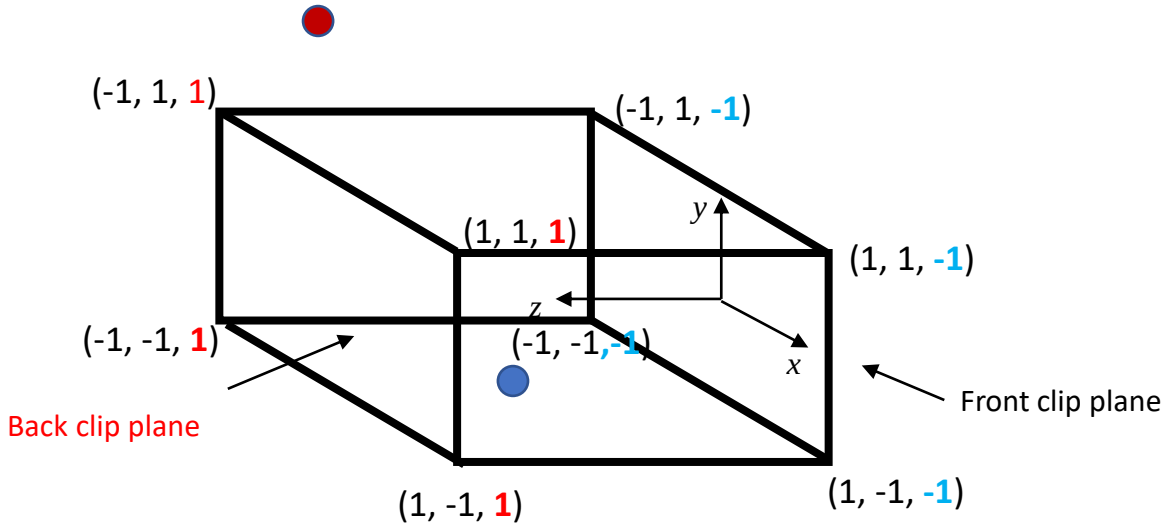
- **I poliedri** trasformati in coordinate normalizzate vengono ritagliati rispetto ai limiti del volume di vista canonico, un poligono alla volta
- **I poligoni vengono tagliati uno spigolo alla volta.**
- I calcoli di intersezione sono banali grazie ai piani normalizzati del volume di vista canonico.
- Quando gli oggetti vengono tagliati vengono creati nuovi vertici.



- L'utilizzo di **bounding volumes** permette di rigettare banalmente gruppi di oggetti alla volta.

Clipping di punti

- Il volume di vista è un cuboide con centro l'origine e x,y e z che variano tra -1 ed 1. (Opengl, per sua convenzione, durante la proiezione, inverte l'asse z del cuboide generando un left handed coordinate system)
- Test sulle coordinate dei vertici -1 <= *x,y,z* <=1

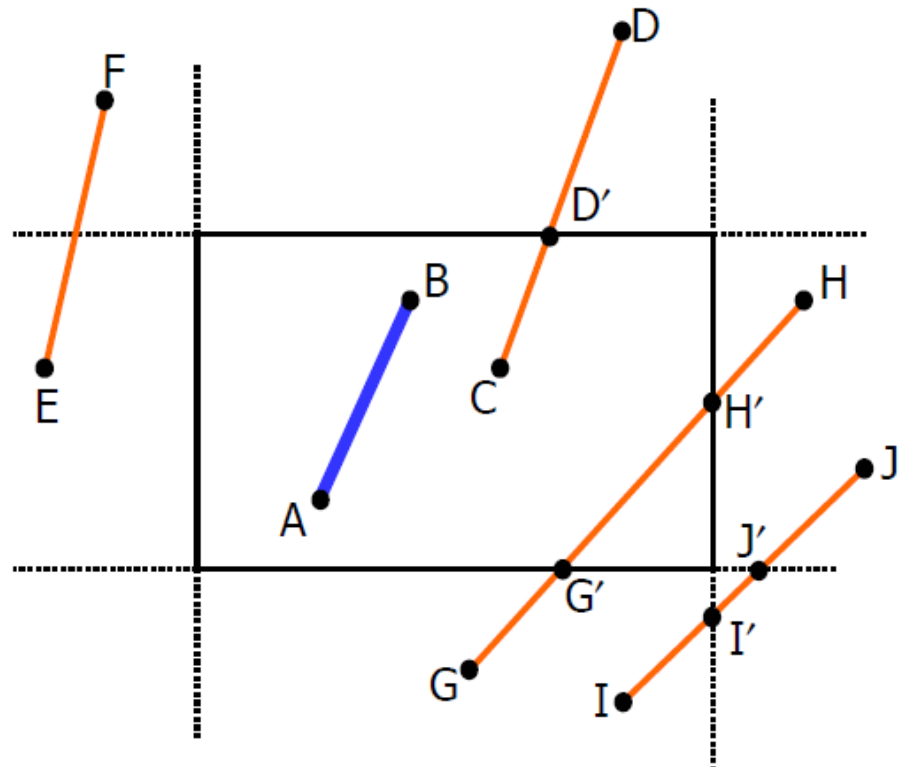


- I vertici che cadono dentro questi valori vengono conservati , e I vertici che cadono fuori vengono tagliati.

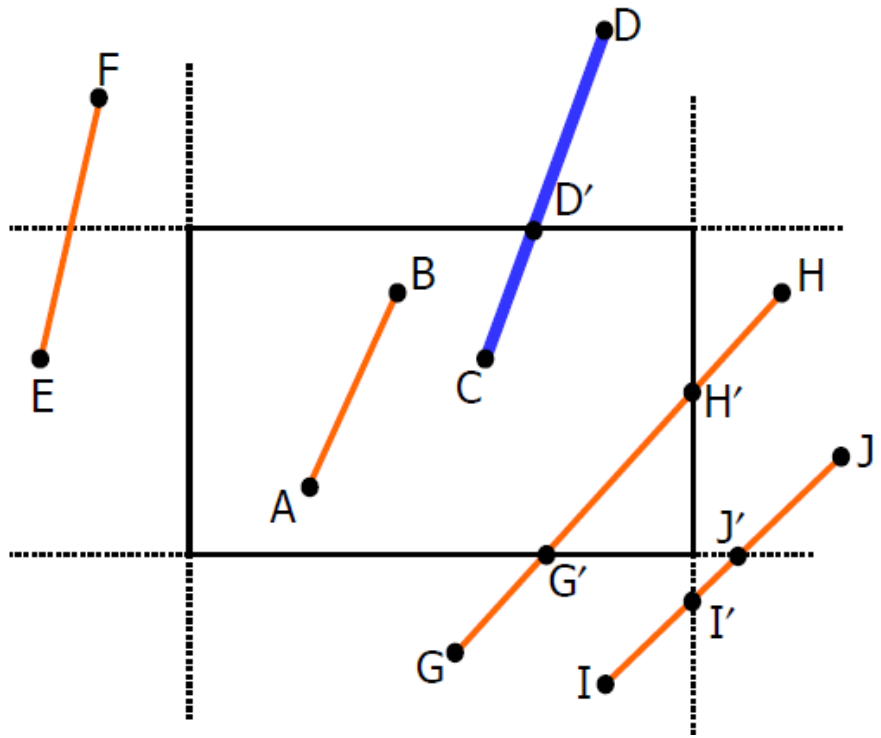
Clipping di un segmento (caso 2D)

- Clipping di un segmento: necessario analizzare le posizioni dei suoi punti estremi.
  - Se gli estremi sono entrambi interni al rettangolo di clipping, il segmento è interno;
  - Se un estremo è interno e l'altro esterno, allora il segmento interseca il rettangolo di clipping ed è necessario determinare l'intersezione;
  - Se entrambi gli estremi sono esterni al rettangolo, il segmento può intersecare o meno il rettangolo di clipping e si rende necessaria una analisi più accurata per individuare le eventuali parti interne del segmento.

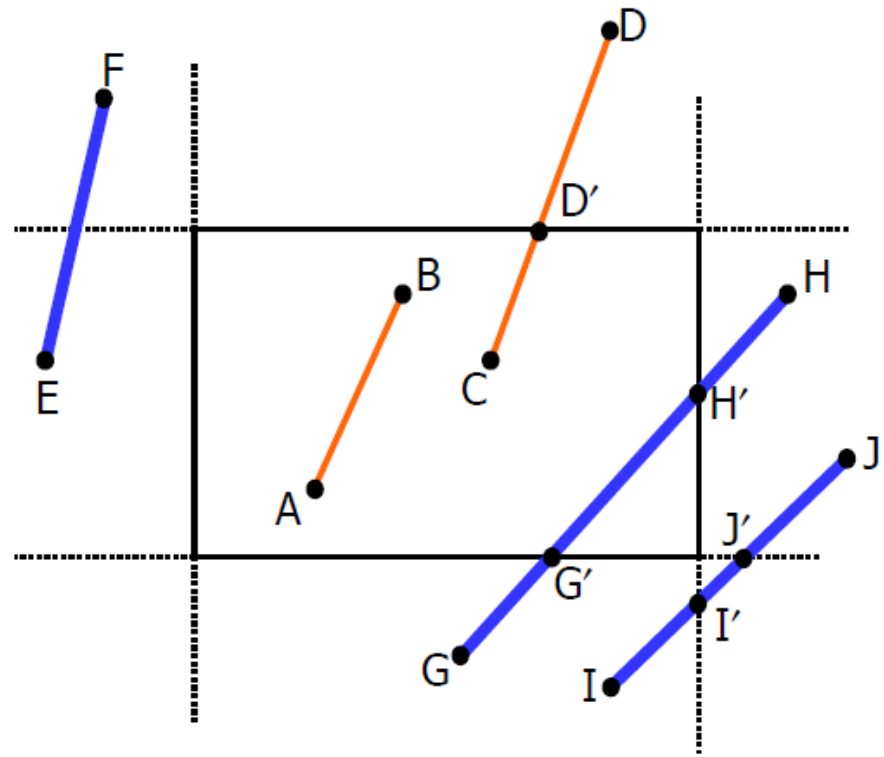
- Se gli estremi sono entrambi interni al rettangolo di clipping, il segmento (AB) è interno.



- Se un estremo è interno e l'altro esterno, allora il segmento interseca il rettangolo di clipping ed è necessario determinare l'intersezione (CD)



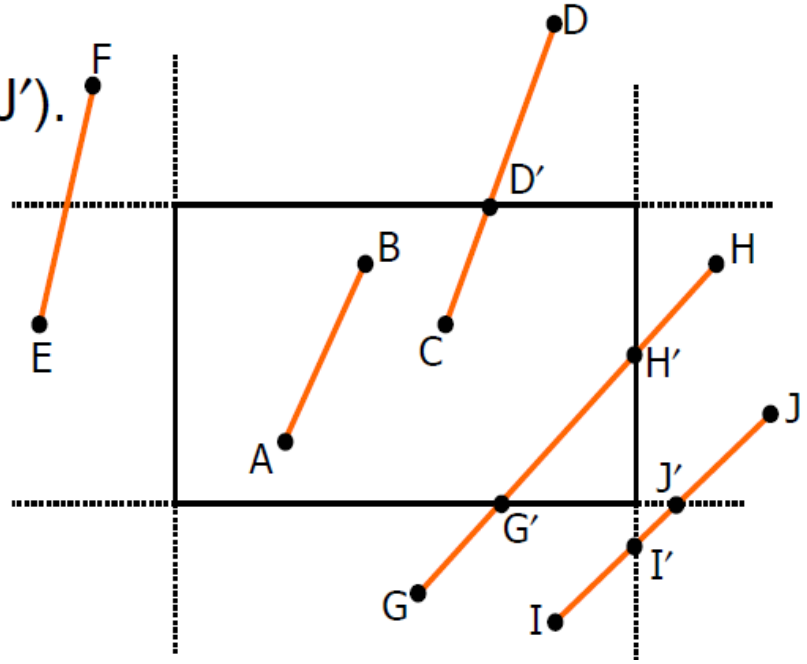
- Se entrambi gli estremi sono esterni al rettangolo, il segmento può intersecare o meno il rettangolo di clipping e si rende necessaria una analisi più accurata per individuare le eventuali parti interne del segmento (EF, GH, IJ).



Algoritmo diretto

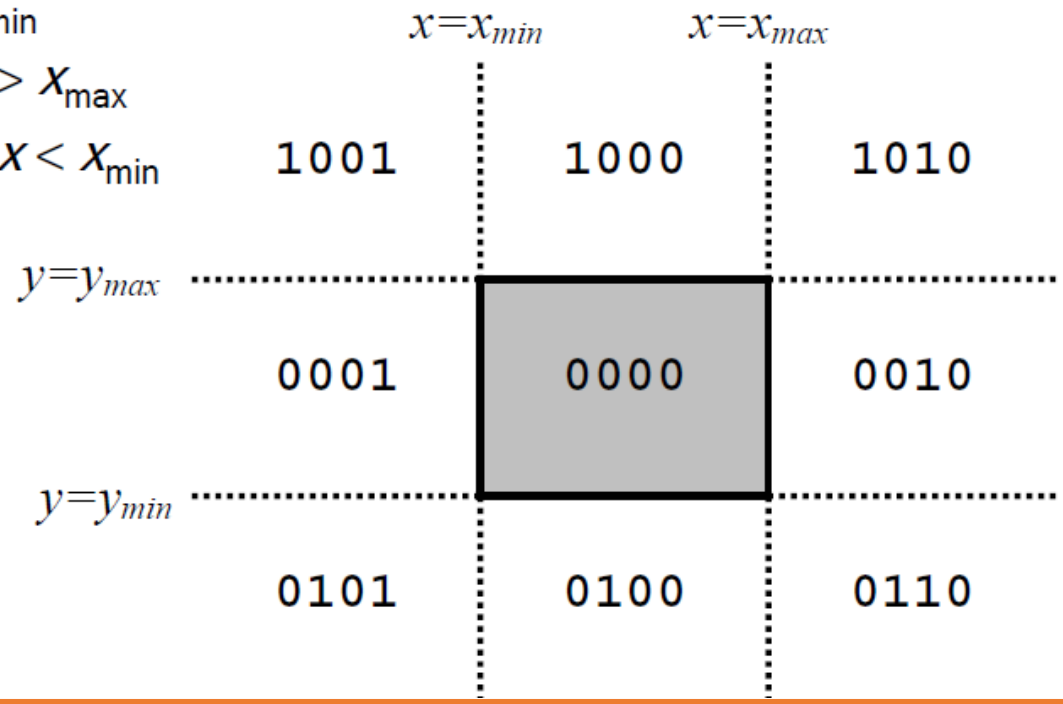
- L'approccio diretto alla soluzione del problema è quello di determinare le intersezioni tra la retta su cui giace il segmento e le 4 rette su cui giacciono i lati del rettangolo di clipping;
- Individuati i punti di intersezione occorre verificare l'effettiva appartenenza al rettangolo di clipping (G′ e H′) o meno (I′ e J′).
- Le intersezioni si determinano mediante l'eq. parametrica dei segmenti relativi.

$$x = x_a + t(x_b - x_a)$$
$$y = y_a + t(y_b - y_a)$$
$$t \in [0,1]$$

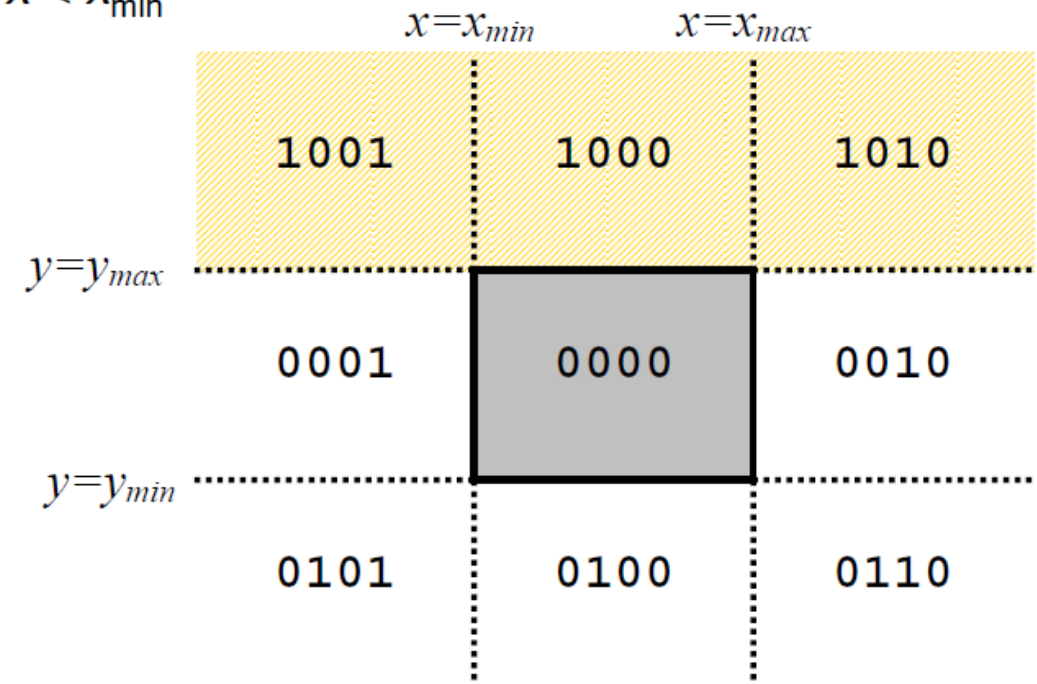


Algoritmo di Cohen-Sutherland

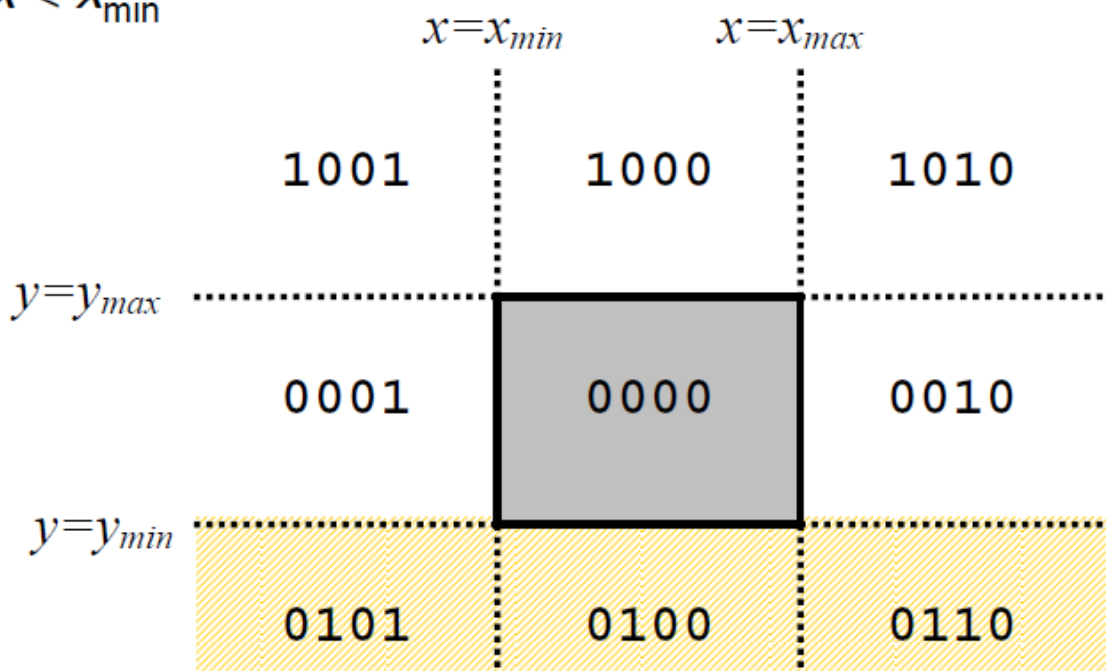
- Idea di base: le rette che delimitano il rettangolo di clipping suddividono il piano in nove regioni;
- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



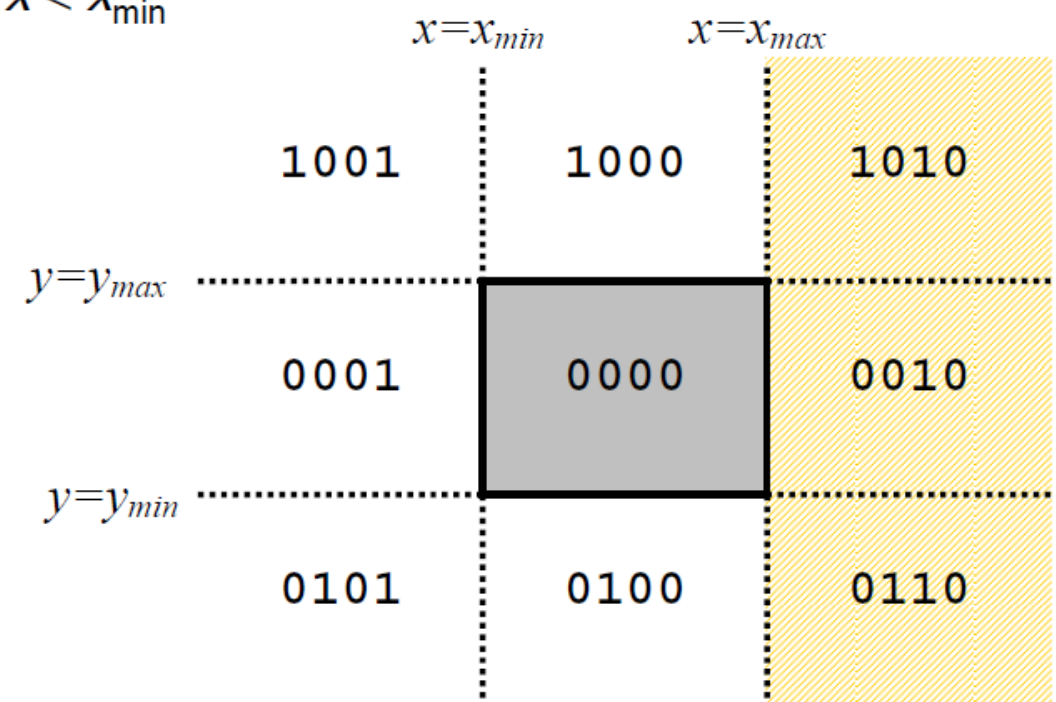
- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



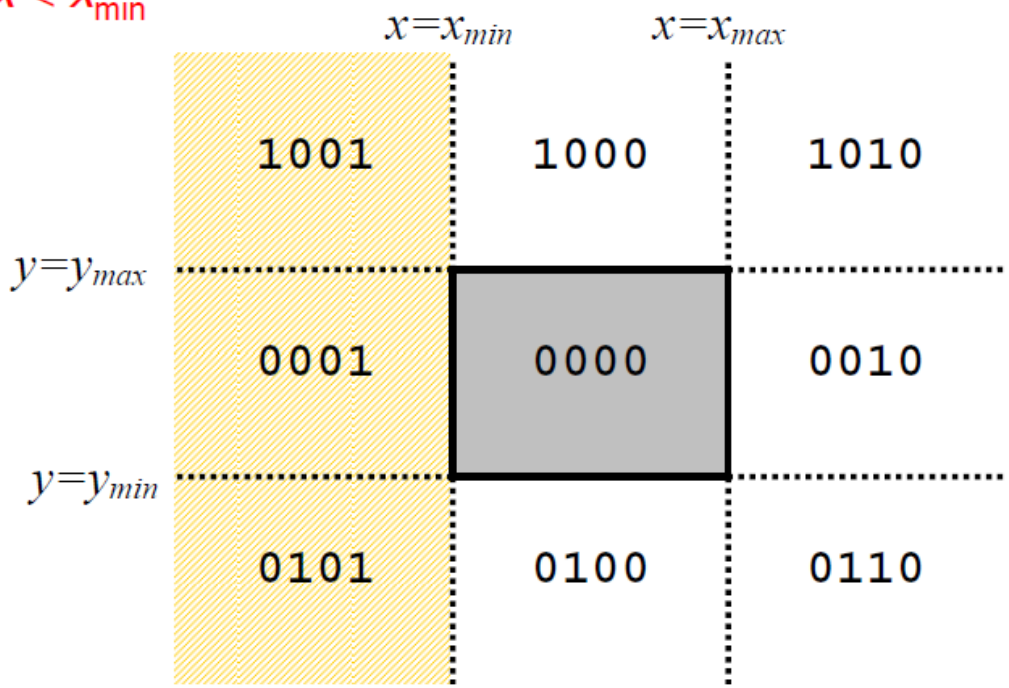
- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$

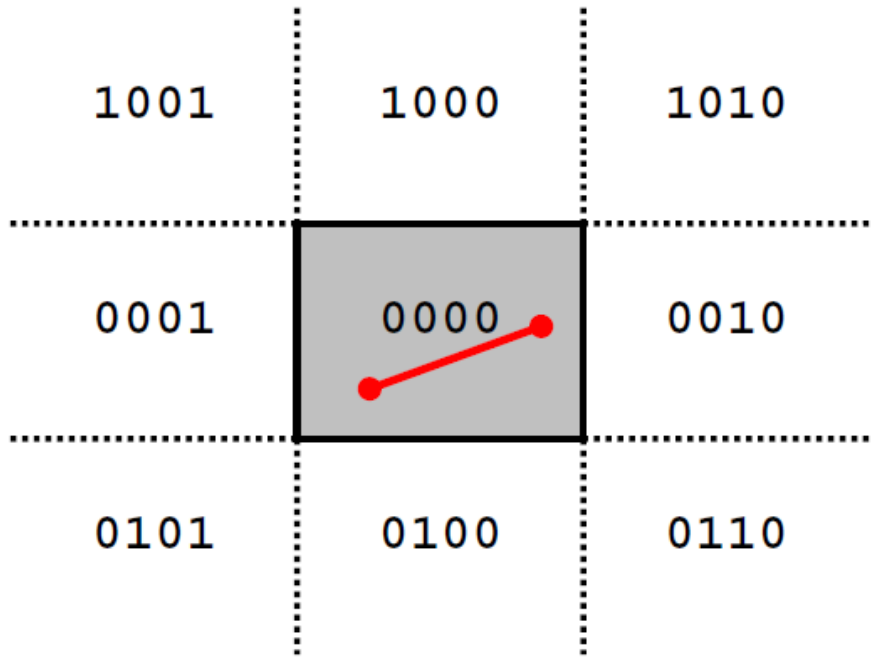


- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



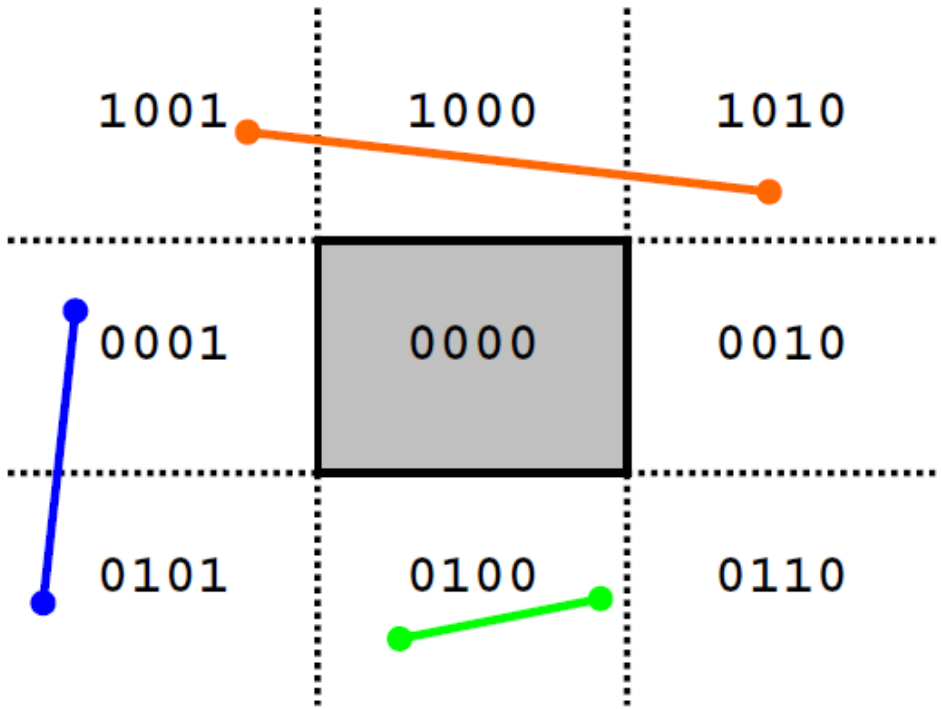
- Il clipping di un segmento prevede la codifica (e confronto) dei suoi estremi sulla base delle regioni di appartenenza;

- Se il codice di entrambi gli estremi è 0000 (OR logico tra i codici ritorna un risultato nullo), allora si può banalmente decidere che il segmento è interamente interno al rettangolo di clipping.



$0000 \vee 0000 = 0000$

- Se l'operazione di AND logico tra i codici degli estremi restituisce un risultato non nullo allora il segmento è esterno al rettangolo di clipping.
- In questo caso, infatti, gli estremi giacciono in uno stesso semipiano (quello identificato dal bit a 1 del risultato) e quindi il segmento non interseca il rettangolo di clipping.

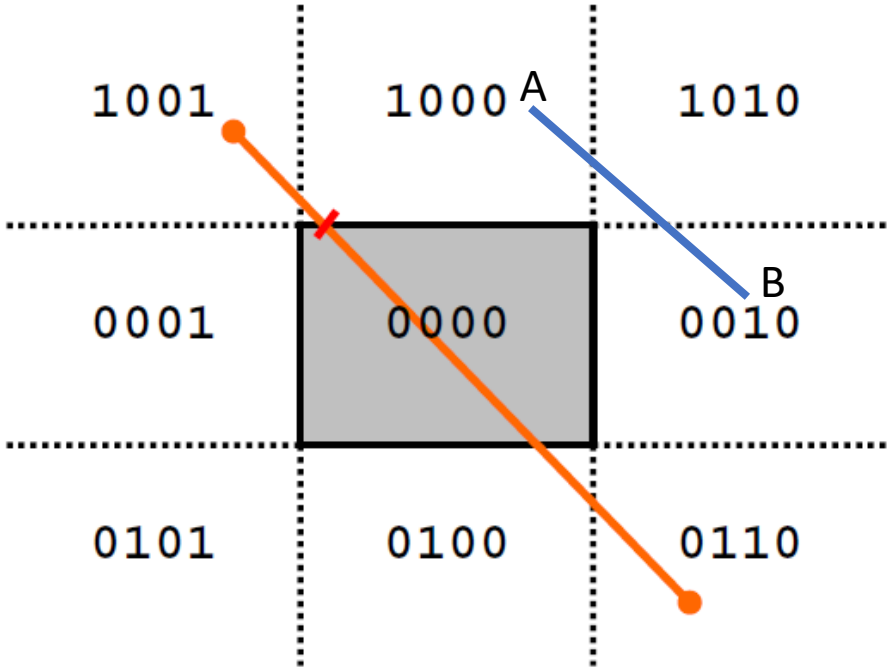


1001 ∧ 1010 = 1000

0100 ∧ 0100 = 0100

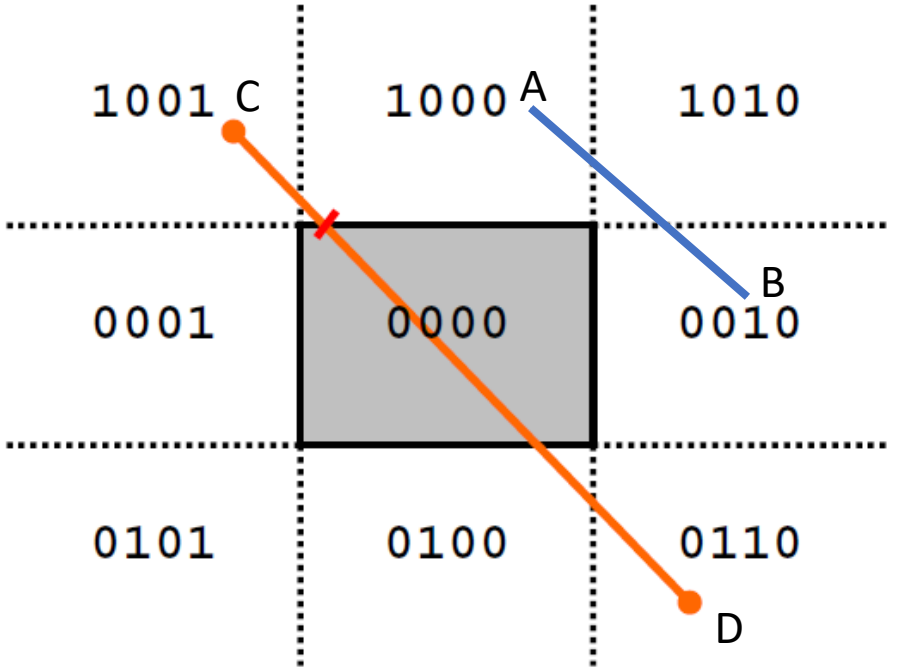
0001 ∧ 0101 = 0001

- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);



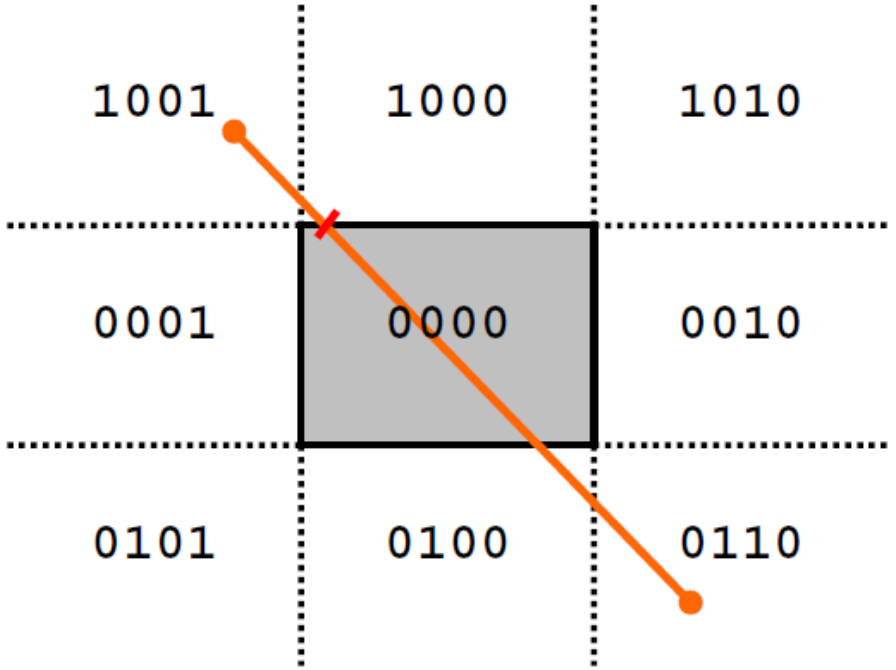
- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);

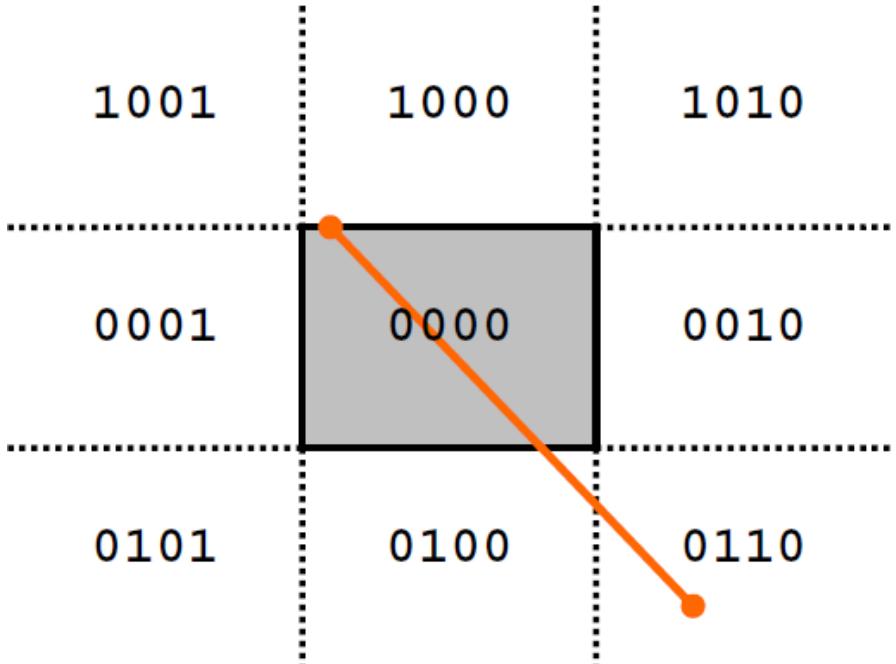
Se l'intersezione non appartiene al rettangolo di clipping e quindi non ha codifica (0000) allora il segmento viene rigettato. (Caso AB)



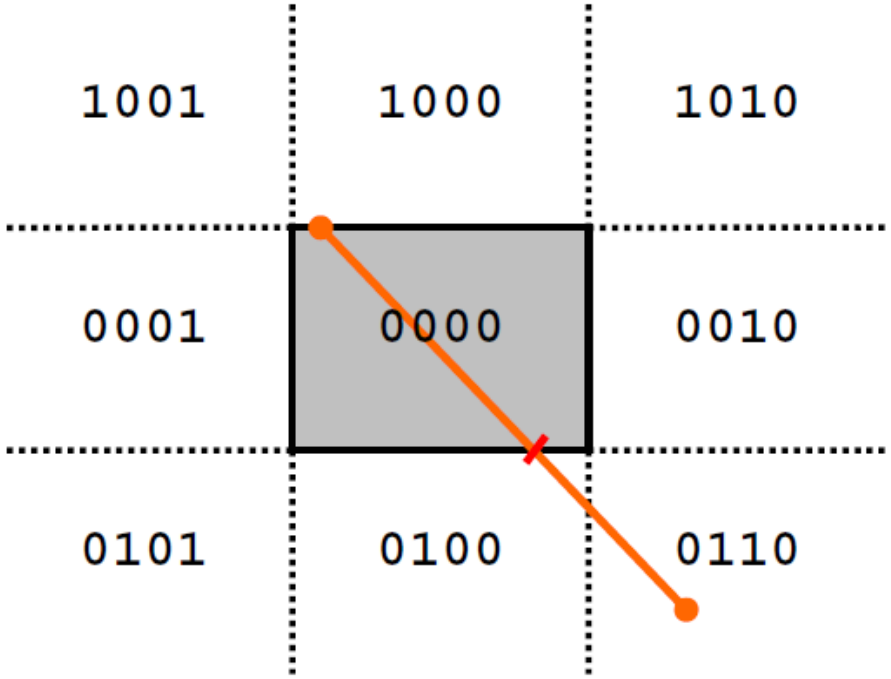
- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);

Se l'intersezione appartiene al rettangolo di clipping ed ha codifica (0000), allora l'estremo con bit a 1 viene rimpiazzato con l'intersezione calcolata



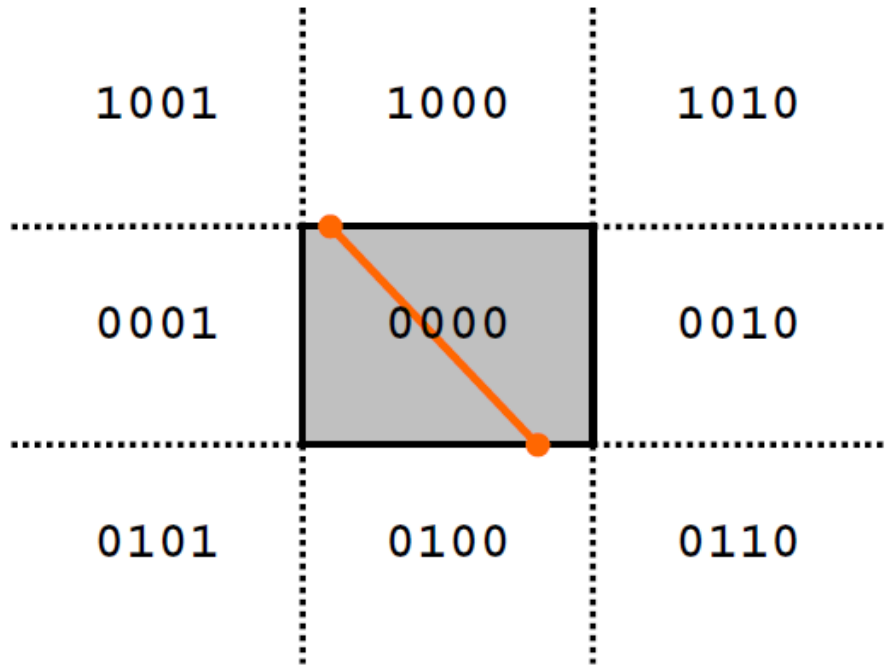

1001
1000
1010
0001
0000
0010
0101
0100
0110

Si itera il procedimento, si interseca il segmento con il lato del poligono di clipping corrispondente al secondo bit discordante tra i codici (in fig. y=ymin). L'estremo con bit ad 1, viene rimpiazzato dal punto di intersezione.



- Ad ogni iterazione si controlla l'eventuale terminazione del processo (OR logico nullo);

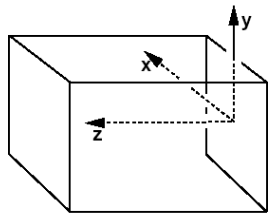
- L'algoritmo rimuove progressivamente le parti esterne; risulta efficiente quando molti dei segmenti da clippare sono completamente esterni al rettangolo di clipping.
- Nota: per l'inclusione del punto in 0000 il rettangolo di clipping "include" il bordo



Clipping di rette

```
Ripeti
        Valuta i codici dei due estremi
        Se entrambi i codici sono nulli
                La linea è totalmente visibile // caso AB
        Altrimenti se l'AND bit a bit degli estremi è diverso da 0
                La linea è totalmente invisibile // caso CD: si tratta di
                                                 // linee che sono tutte in una
                                                 // delle 4 regioni definite dai
                                                 // bit del codice perché gli
                                                 // estremi hanno un bit 1 nella
                                                 // stessa posizione)
        Altrimenti
                Si considera il primo estremo con codice diverso da 0
                        // caso EF e IL: se un estremo ha codice pari a 0
                        // allora è visibile
                Si effettua l'intersezione con la retta corrispondente al
                primo bit 1 del codice // punto H in EF e punto M in IL
                Si considera il nuovo segmento che va dal punto
                intersezione al secondo estremo e si calcola il nuovo
                codice per l'intersezione
Finché non ricadi in uno dei due casi banali
```

Cohen-Sutherland - Line Clipping in 3D



- Molto simile al 2D
- Divide il volume in 27 regioni
- **6 bit outcode** :

Primo bit: dietro al piano posteriore

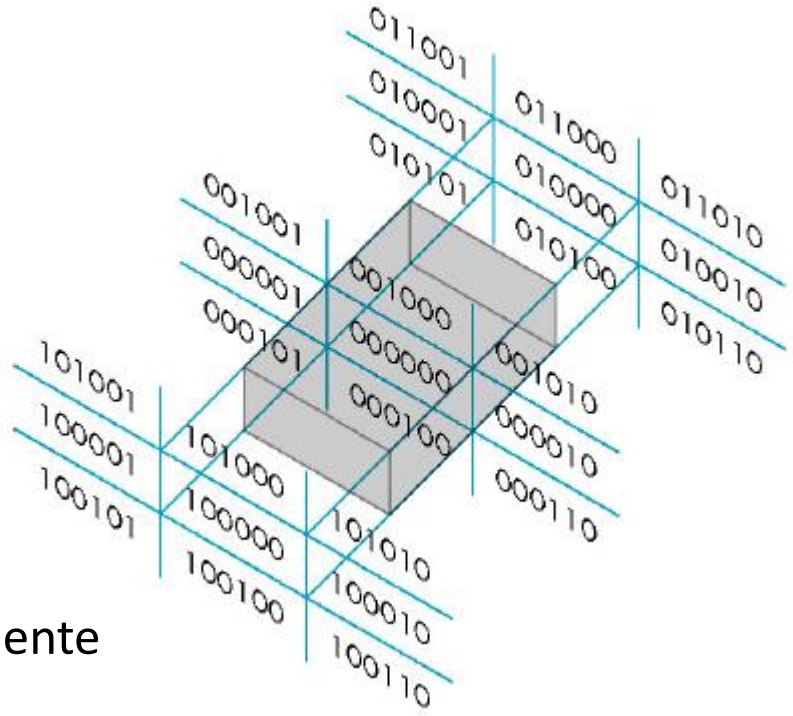
Secondo bit: davanti al piano frontale

Terzo bit: sopra il piano top

Quarto bit: sotto il piano bottom

Quinto bit: a destra del piano di destra

Sesto bit: a sinistra del piano di sinistra



Linee con OC_0 = 000000 and OC_1 = 000000 possono essere banalmente accettate

- Linee che stanno interamente in un volume all'esterno di un piano possono essere banalmente rigettate.
- Altrimenti CLIP

Window Transformation

Una volta che il volume di vista è stato mappato in un cuboide con centro l'origine e lato 2, **ed I poliedri** trasformati in coordinate normalizzate vengono ritagliati rispetto ai limiti del volume di vista canonico, un poligono alla volta, l'immagine 2D finale è ottenuta semplicemente annullando la coordinata dopo la proiezione ortogonale nel piano z = 0.

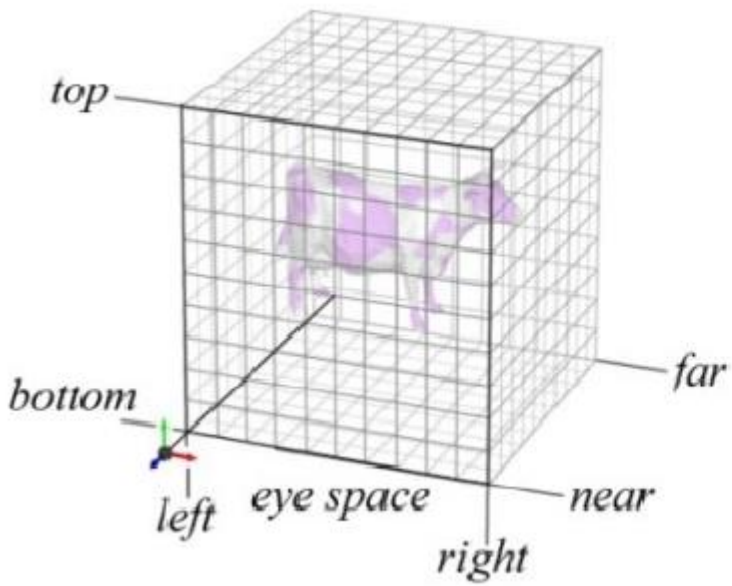
- La Window Transformation è una proiezione ortogonale che mappa punti in NCS (x, y, z) in [-1,1] x[-1,1]x[-1,1] (spazio immagine 3D) in una regione rettangolare (x, y) in [-1,1] x[-1,1] (finestra 2D)

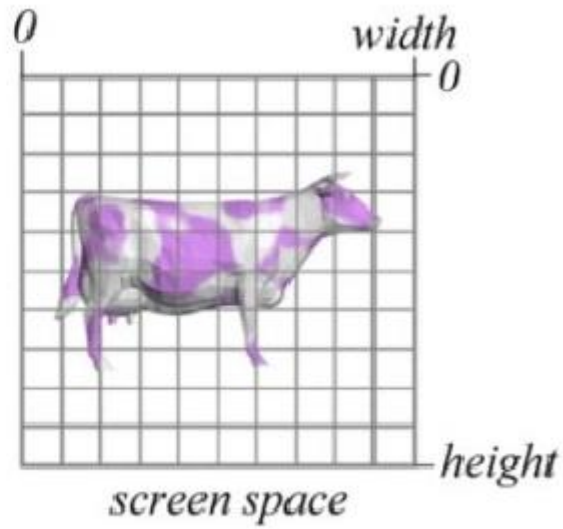
- I punti proiettati nel piano della vista (z = 0) mantengono le coordinate x, y, ma z = 0.

$$P_{ortho} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \Rightarrow \quad P_{ortho} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ 0 \\ 1 \end{pmatrix}$$

$$v' = P_{ortho} \cdot P \cdot T_v \cdot T_M \cdot v$$

Adesso bisogna mappare la window sul viewport





Trasformazione window-viewport

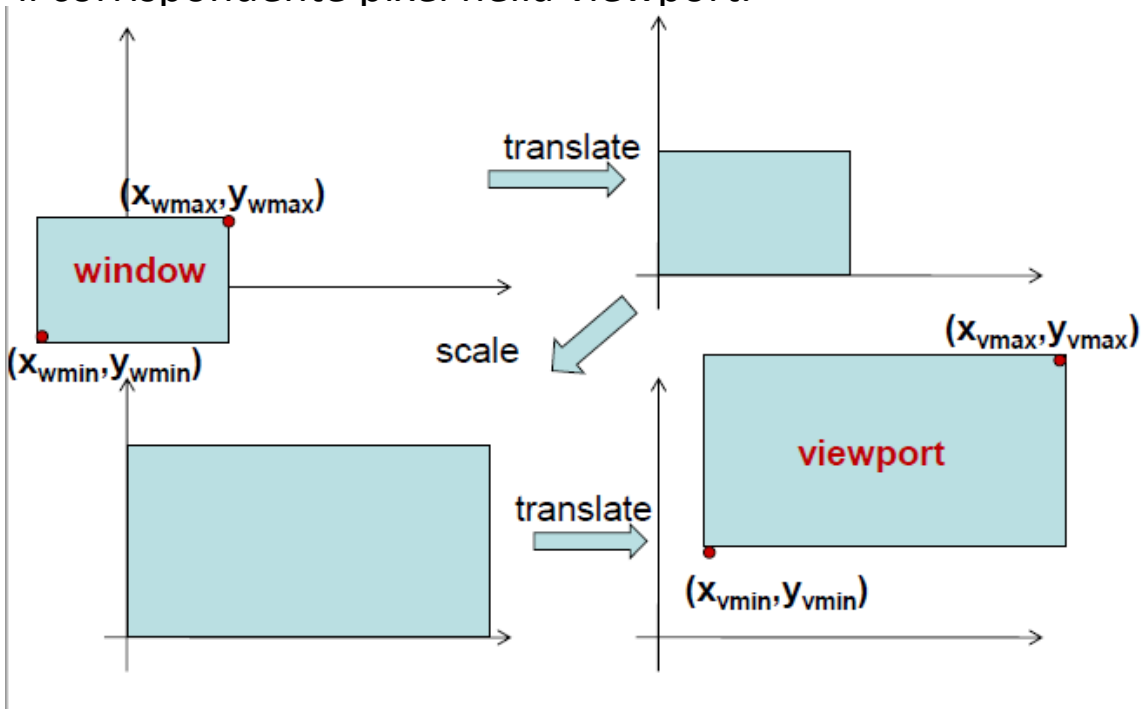
Una window ed una viewport sono aree rettangolari.

La window è la finestra 2D attraverso la quale si guarda la scena 3D e che verrà ***visualizzata sullo schermo in un'area rettangolare detta viewport (sistema di coordinate schermo).***

La viewport quindi è un array di (*nx* × *ny*) pixel.

Alle coordinate reali di ogni vertice nella window dovranno corrispondere coppie di indici interi che individuano il corrispondente pixel nella viewport.

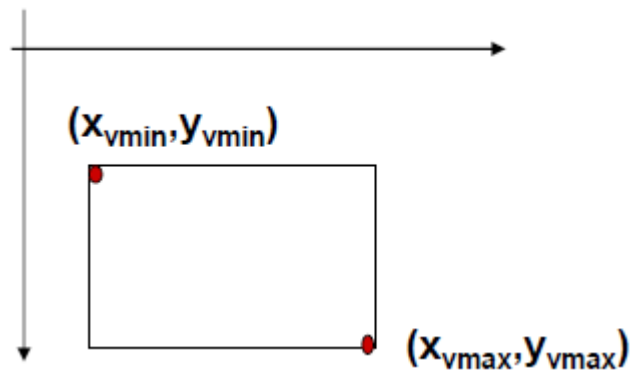
Window:
Sistema di coordinate normalizzate di device (NDC) in [-1,1]x[-1,1]



$$x_v = ceil(S_x(x_w - x_{w\min}) + x_{v\min})$$

$$y_v = ceil(S_y(y_{w\min} - y_w) + y_{v\max})$$

$$S_x = \frac{x_{v\max} - x_{v\min}}{x_{w\max} - x_{w\min}} \qquad S_y = \frac{y_{v\max} - y_{v\min}}{y_{w\max} - y_{w\min}}$$



In forma matriciale

$$T_{wv} = \begin{bmatrix} 1 & 0 & x_{v\min} \\ 0 & 1 & y_{v\min} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_{w\min} \\ 0 & 1 & -y_{w\min} \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} S_x & 0 & x_{v\min} - S_x x_{w\min} \\ 0 & S_y & y_{v\min} - S_y y_{w\min} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & \dfrac{x_{v\min} x_{w\max} - x_{v\max} x_{w\min}}{x_{w\max} - x_{w\min}} \\ 0 & S_y & \dfrac{y_{v\min} y_{w\max} - y_{v\max} y_{w\min}}{y_{w\max} - y_{w\min}} \\ 0 & 0 & 1 \end{bmatrix}$$