

# Trasformazione di Vista (Viewing 3D)





# Cambio di Sistema di Riferimento

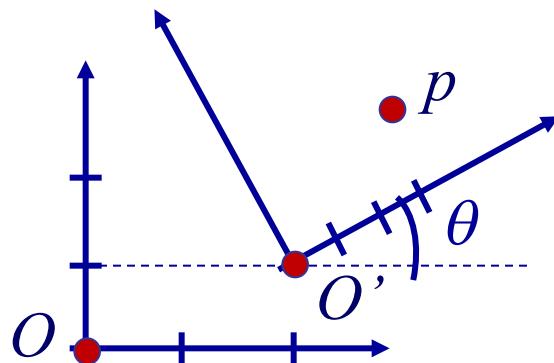
**Problema:** sia dato un Sistema di Riferimento (frame) cartesiano  $(\underline{e}_1, \underline{e}_2, \underline{e}_3, O)$  e un punto  $p$  in questo sistema. Sia poi dato un nuovo frame  $(\underline{u}, \underline{v}, \underline{w}, O')$ . Si determinino le coordinate di  $p$  rispetto al nuovo frame.

Procediamo mediante trasformazioni geometriche elementari per portare il primo frame a coincidere con il secondo; la trasformazione  $M$  cercata, cioè tale che applicata a  $p$  (ossia il prodotto  $M p$ ) fornisca le sue coordinate rispetto al nuovo sistema, si può poi ottenere ...

**componendo nell'ordine le matrici inverse delle trasformazioni elementari per portare il primo sistema sul secondo**

# Cambio di Sistema di Riferimento

**Esempio 2D:** Sia dato il frame cartesiano  $([1,0,0]^T, [0,1,0]^T, [0,0,1]^T)$  e il nuovo frame  $([1/2\cos(\theta), 1/2\sin(\theta), 0]^T, [-1/2\sin(\theta), 1/2\cos(\theta), 0]^T, [2,1,1]^T)$ , dove i vettori e punti sono espressi rispetto al primo sistema.



Le trasformazioni per portare il primo frame sul secondo sono

$$T = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad S = \begin{pmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R(\theta) = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

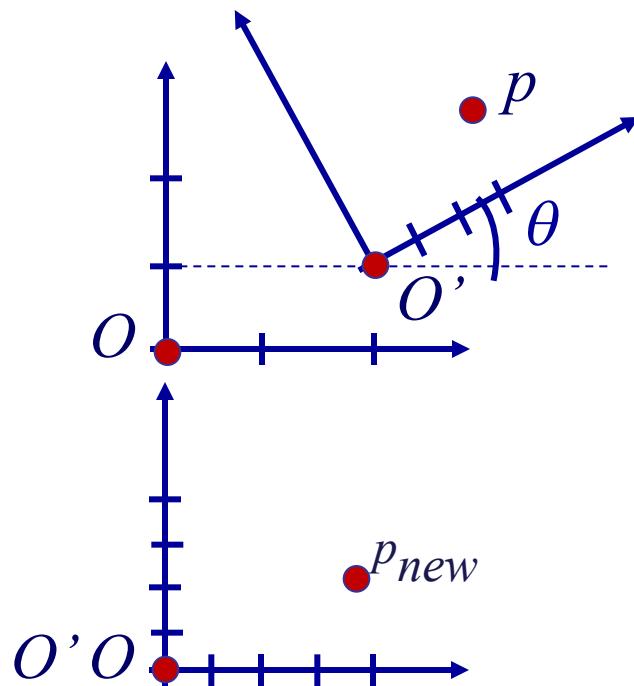
Da cui

$$M = (T S R(\theta))^{-1} = R^{-1}(\theta) S^{-1} T^{-1}$$

dove  $c = \cos(\theta)$ ,  $s = \sin(\theta)$

$$= \begin{pmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

# Cambio di Sistema di Riferimento



E moltiplicando le tre matrici:

$$\mathbf{M} = \begin{pmatrix} 2c & 2s & -4c - 2s \\ -2s & 2c & 4s - 2c \\ 0 & 0 & 1 \end{pmatrix}$$

Concludendo, le coordinate di  $p$  nel nuovo sistema di riferimento saranno:

$$p_{new} = \mathbf{M} p$$



# Cambio di Sistema di Riferimento

Vediamo un modo differente di risolvere il Problema.

Dobbiamo determinare le coordinate  $p_{new} = [u, v, w, 1]$  così che

$$[\underline{u}, \underline{v}, \underline{w}, O'] \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} = [\underline{e1}, \underline{e2}, \underline{e3}, O] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (1)$$

Sappiamo che  $\underline{u}$ ,  $\underline{v}$ ,  $\underline{w}$  sono vettori linearmente indipendenti e formano una base per  $R^3$ . Scriviamo i vettori  $\underline{e1}$ ,  $\underline{e2}$  ed  $\underline{e3}$  in questa base:

$$\underline{e1} = [1, 0, 0, 0]^T = a_{11}\underline{u} + a_{21}\underline{v} + a_{31}\underline{w}$$

$$\underline{e2} = [0, 1, 0, 0]^T = a_{12}\underline{u} + a_{22}\underline{v} + a_{32}\underline{w}$$

$$\underline{e3} = [0, 0, 1, 0]^T = a_{13}\underline{u} + a_{23}\underline{v} + a_{33}\underline{w}$$



# Cambio di Sistema di Riferimento

Poiché  $O-O'$  è un vettore, possiamo trovare la sua rappresentazione nella base  $\underline{u}, \underline{v}, \underline{w}$

$$O - O' = [0, 0, 0, 1] - O' = a_{14}\underline{u} + a_{24}\underline{v} + a_{34}\underline{w}$$

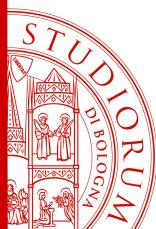
e quindi:

$$O = [0, 0, 0, 1]^T = a_{14}\underline{u} + a_{24}\underline{v} + a_{34}\underline{w} + O'$$

mettendo quest'ultima insieme alle precedenti si ha:

$$[\underline{e}_1, \underline{e}_2, \underline{e}_3, O] = [\underline{u}, \underline{v}, \underline{w}, O'] \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

riprendendo la relazione (1) e sostituendo:



# Cambio di Sistema di Riferimento

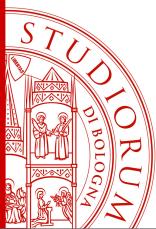
$$[\underline{u}, \underline{v}, \underline{w}, O'] \begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} = [\underline{e}_1, \underline{e}_2, \underline{e}_3, O] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = [\underline{u}, \underline{v}, \underline{w}, O'] \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

da cui risulta:

$$\begin{pmatrix} u \\ v \\ w \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

A

Quanto ottenuto dice che il cambio di sistema di riferimento può essere rappresentato da una matrice  $4 \times 4$



# Cambio di Sistema di Riferimento

Ovviamente si devono determinare i coefficienti  $a_{ij}$  per poter dire di aver risolto il problema.

Le prime tre colonne sono le coord. dei vettori  $\underline{e1}$ ,  $\underline{e2}$  ed  $\underline{e3}$ , nel nuovo sistema, mentre la quarta sono le coord. di  $O$  nel nuovo sistema.

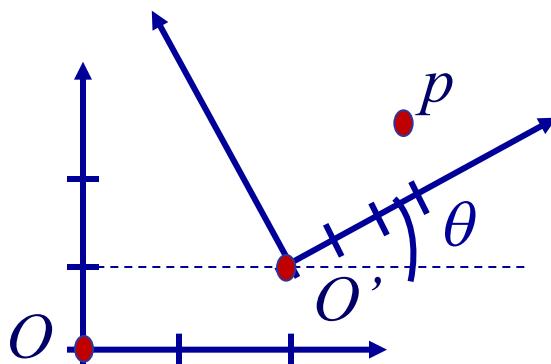
Solitamente, nella pratica, sono note le coordinate di  $\underline{u}$ ,  $\underline{v}$ ,  $\underline{w}$  e  $O'$  rispetto al primo sistema, cioè:

$$[\underline{u}, \underline{v}, \underline{w}, O'] = [\underline{e1}, \underline{e2}, \underline{e3}, O] \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \xleftarrow{\text{B}}$$

Allora la cercata matrice A sarà data da  $B^{-1}$ .

# Cambio di Sistema di Riferimento

Riprendiamo il seguente esempio 2D già visto: sia dato il frame cartesiano  $([1,0,0]^T, [0,1,0]^T, [0,0,1]^T)$  e il nuovo frame  $([1/2\cos(\theta), 1/2\sin(\theta), 0]^T, [-1/2\sin(\theta), 1/2\cos(\theta), 0]^T, [2,1,1]^T)$ , dove i vettori e punti sono espressi rispetto al primo sistema.



$$[\underline{u}, \underline{v}, O'] = [\underline{e}_1, \underline{e}_2, O] \begin{pmatrix} c/2 & -s/2 & 2 \\ s/2 & c/2 & 1 \\ 0 & 0 & 1 \end{pmatrix} \quad B$$

L'inversa della matrice B sarà:  
 (confrontare con quanto trovato  
 precedentemente).

$$A = B^{-1} = \begin{pmatrix} 2c & 2s & -4c - 2s \\ -2s & 2c & 4s - 2c \\ 0 & 0 & 1 \end{pmatrix}$$

con  $c = \cos(\theta)$ ,  $s = \sin(\theta)$

# Definizione oggetto mesh 3D

In un sistema di riferimento Cartesiano xyzO destrorso, consideriamo un oggetto mesh 3D dando la lista dei suoi Vertici (coord. floating point) e la lista delle sue Facce (piane).

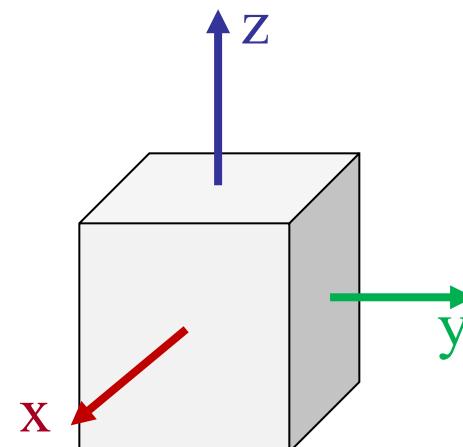
Vogliamo analizzare cosa si deve fare per ottenere una sua rappresentazione grafica 2D (immagine 2D).

## Coordinate (X,Y,Z) dei Vertices

```
1> 1.000000 -1.000000 1.000000
2> -1.000000 -1.000000 1.000000
3> -1.000000 1.000000 1.000000
4> 1.000000 1.000000 1.000000
5> 1.000000 -1.000000 -1.000000
6> -1.000000 -1.000000 -1.000000
7> -1.000000 1.000000 -1.000000
8> 1.000000 1.000000 -1.000000
```

## Indici dei Vertices di ogni Face

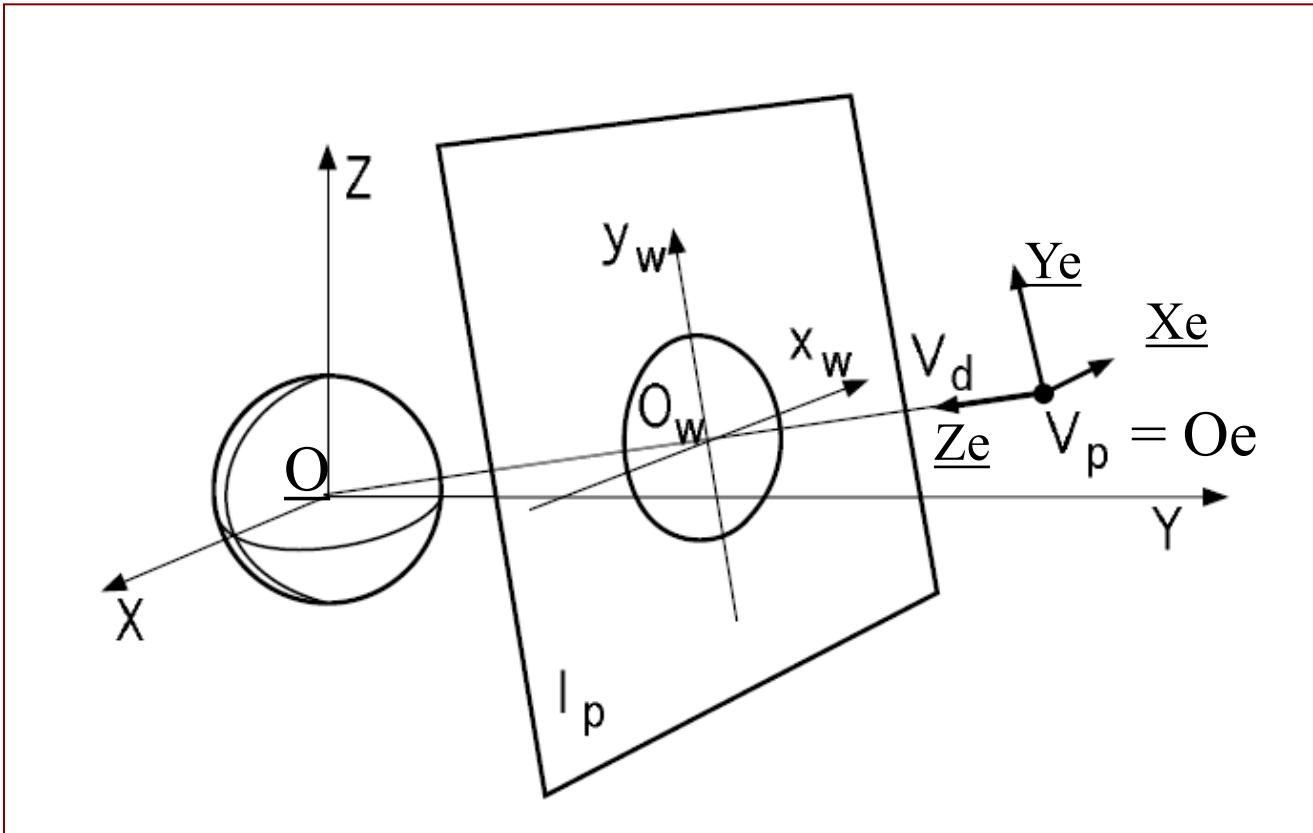
```
1> 4 3 2 1
2> 8 7 3 4
3> 7 8 5 6
4> 5 1 2 6
5> 8 4 1 5
6> ...
```



## Geometria e Topologia

- **definizione geometrica** (dove sono posizionati nello spazio 3D i vertici)
- **definizione topologica** (come sono connessi i vertici da lati e facce)

# Sistemi di Riferimento



X Y Z O : Sistema di Riferimento del Mondo (destrorso)

Xe Ye Ze Oe : Sistema di Riferimento (sinistrorso) dell'  
Osservatore (Oe =Vp=View Point)

Xw Yw Ow : Sistema di Riferimento 2D (Window)

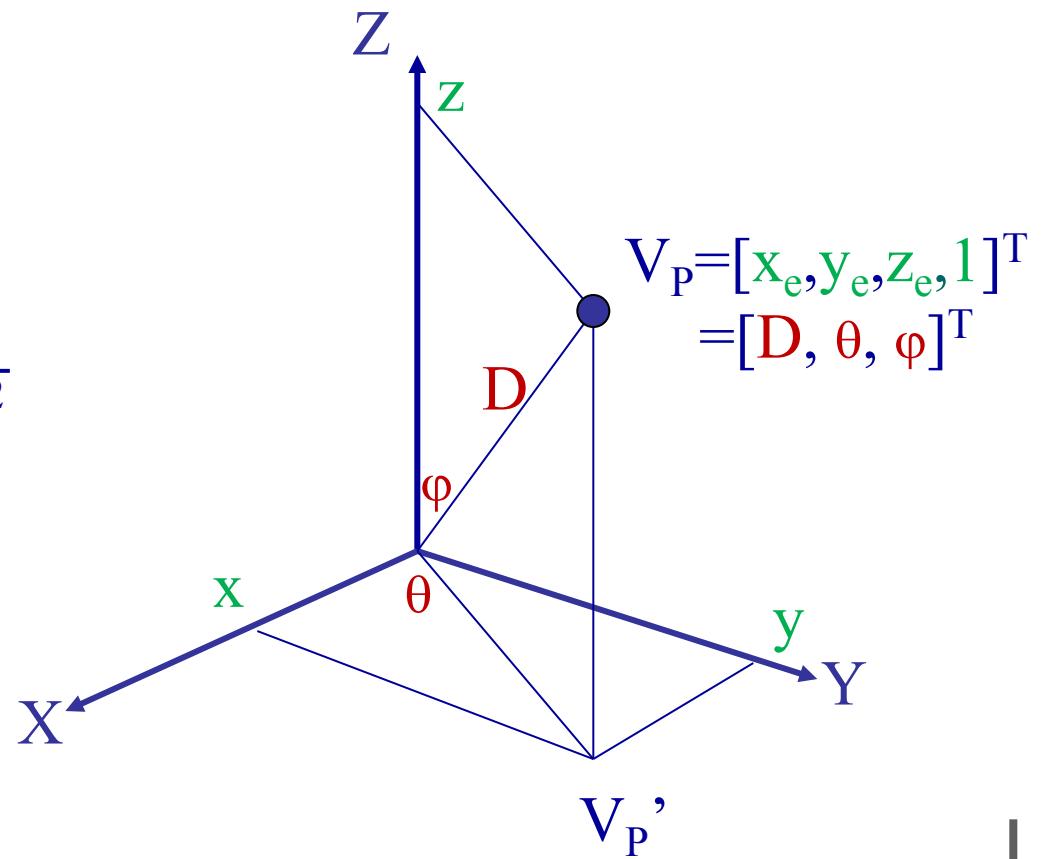


# Coordinate Cartesiane e Sferiche

In certe situazioni è comodo dare il View Point in coordinate sferiche ed in questo caso è bene ricordare come passare da coordinate sferiche a cartesiane e viceversa.

$$\begin{cases} x_e = D \sin(\varphi) \cos(\theta) \\ y_e = D \sin(\varphi) \sin(\theta) \\ z_e = D \cos(\varphi) \end{cases}$$

$$\begin{cases} D = \sqrt{x_e^2 + y_e^2 + z_e^2} \\ \theta = \arctan(y_e / x_e) \\ \varphi = \arccos(z_e / D) \end{cases}$$





# Trasformazione di Vista

Scomponiamo la Trasformazione di Vista in due passi:

1. Trasformazione Sistema di Riferimento

da    X Y Z O        a    Xe Ye Ze Oe

2. Proiezione Geometrica

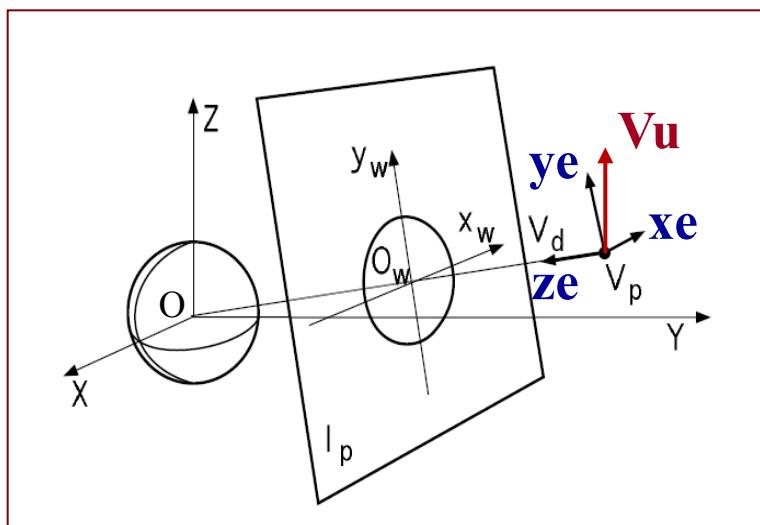
da    Xe Ye Ze Oe    a    Xw Yw Ow

# Trasformazione Sistema di Riferimento e View-Up vector **Vu**

$$\mathbf{Vp} = [D, \theta, \varphi, 1]^T = [D \sin\varphi \cos\theta, D \sin\varphi \sin\theta, D \cos\varphi, 1]^T$$

$$[xe, ye, ze, 1]^T = \mathbf{VM} [x, y, z, 1]^T$$

**VM** è la ViewMatrix

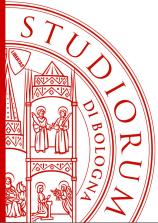


$$\mathbf{ze} = \mathbf{Vd} / \|\mathbf{Vd}\| = -(\mathbf{Vp} - \mathbf{O}) / \|\mathbf{Vp} - \mathbf{O}\|$$

$$\mathbf{xe} = (\mathbf{ze} \times \mathbf{Vu}) / \|\mathbf{ze} \times \mathbf{Vu}\|$$

$$\mathbf{ye} = -(\mathbf{ze} \times \mathbf{xe}) / \|\mathbf{ze} \times \mathbf{xe}\|$$

**Vu** viene definito nel Sistema di Riferimento del Mondo



# Trasformazione Sistema di Riferimento e View-Up vector Vu

$$[x_e, y_e, z_e, 1]^T = VM [x, y, z, 1]^T$$

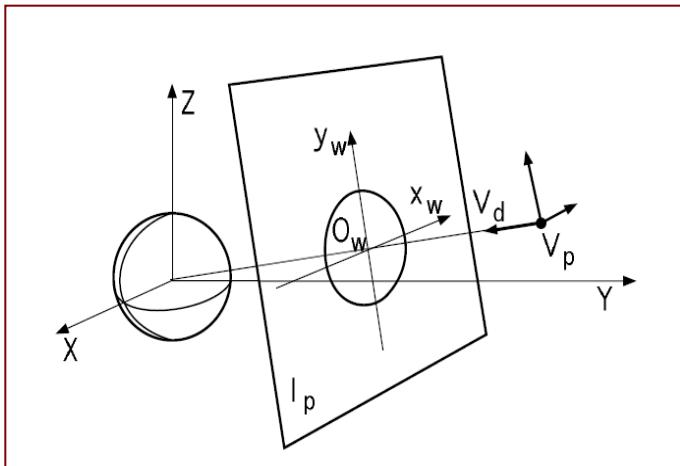
$$B = \begin{pmatrix} \text{xe} & \text{ye} & \text{ze} & D \sin\varphi \cos\theta \\ 0 & 0 & 0 & D \sin\varphi \sin\theta \\ & & & D \cos\varphi \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$VM = B^{-1} = \begin{pmatrix} \text{xe} & 0 \\ \text{ye} & 0 \\ \text{ze} & D \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

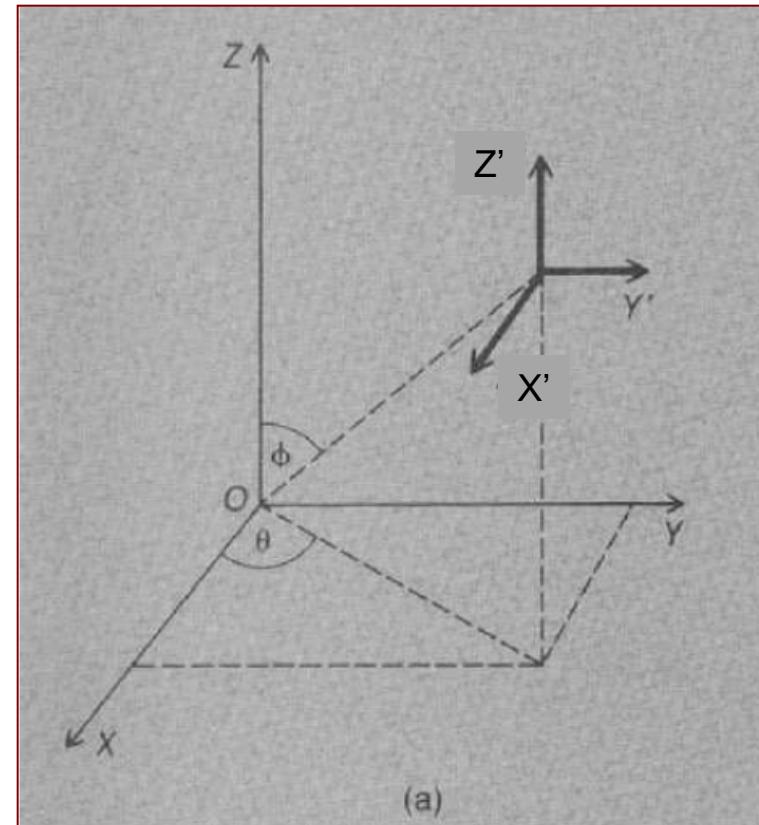
Nelle prossime slide vedremo di ottenere lo stesso risultato portando il primo frame a coincidere con il secondo mediante trasformazioni geometriche elementari.

# Trasformazione Sistema di Riferimento e View-Up vector $\mathbf{Vu}=[0, 0, 1]^T$

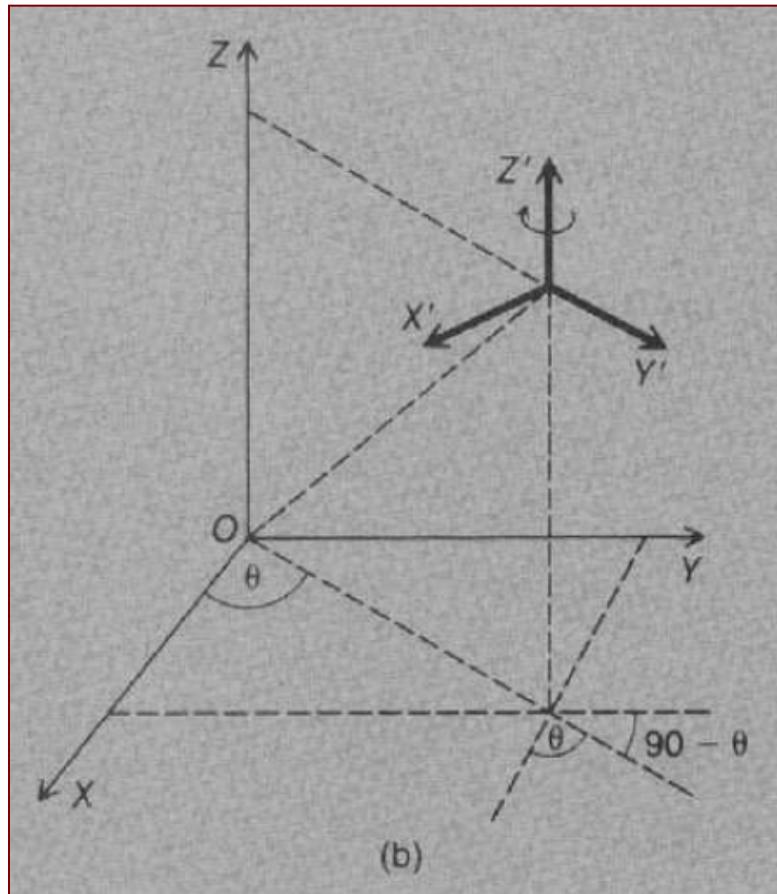
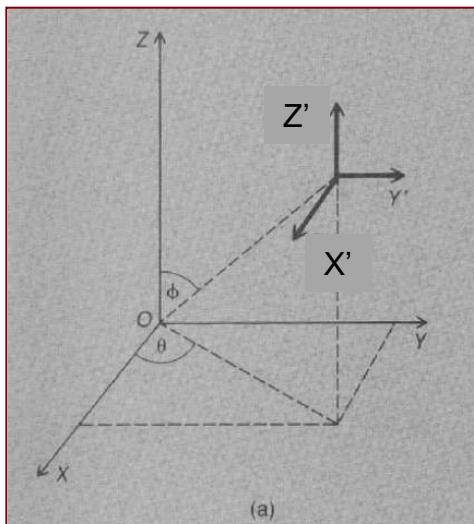
$$\mathbf{Vp} = [D, \theta, \varphi, 1]^T = [D \sin\varphi \cos\theta, D \sin\varphi \sin\theta, D \cos\varphi, 1]^T$$



$$M1 = \begin{pmatrix} 1 & 0 & 0 & D \sin\varphi \cos\theta \\ 0 & 1 & 0 & D \sin\varphi \sin\theta \\ 0 & 0 & 1 & D \cos\varphi \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



# Trasformazione Sistema di Riferimento e View-Up vector $\mathbf{Vu}=[0, 0, 1]^T$

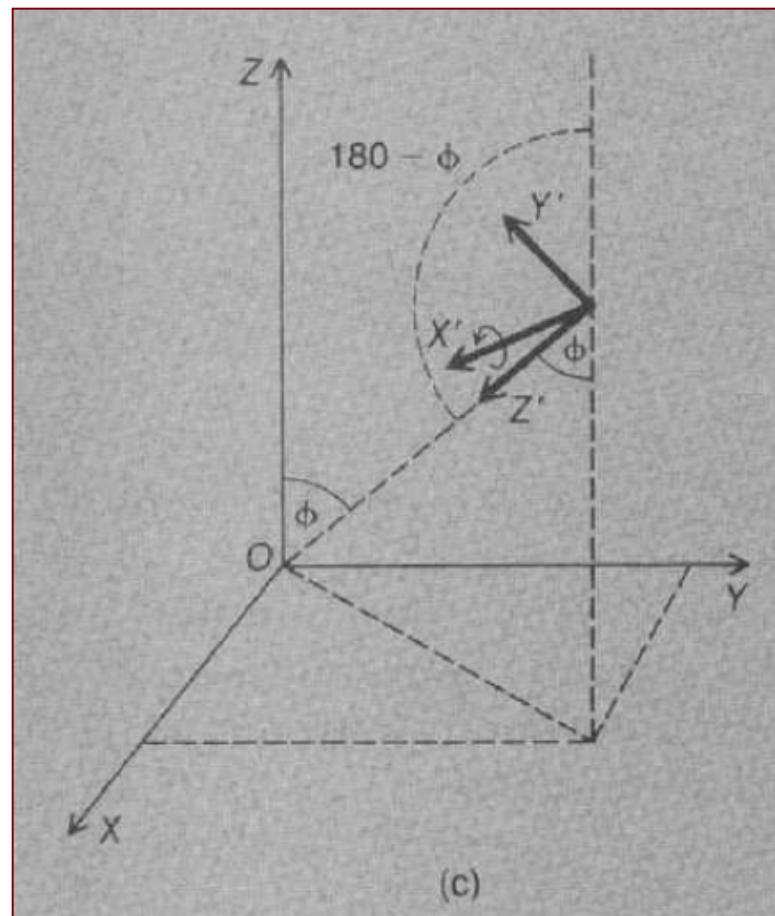
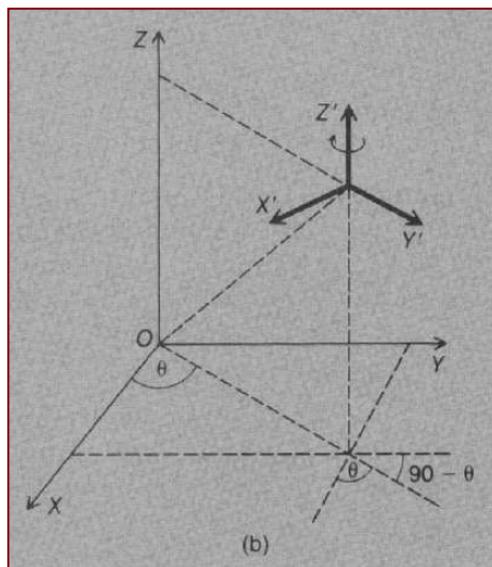


$$M2 = \begin{pmatrix} \sin\theta & \cos\theta & 0 & 0 \\ -\cos\theta & \sin\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\sin -(90-\theta) = -\cos\theta$$

$$\cos -(90-\theta) = \sin\theta$$

# Trasformazione Sistema di Riferimento e View-Up vector $\mathbf{Vu}=[0, 0, 1]^T$

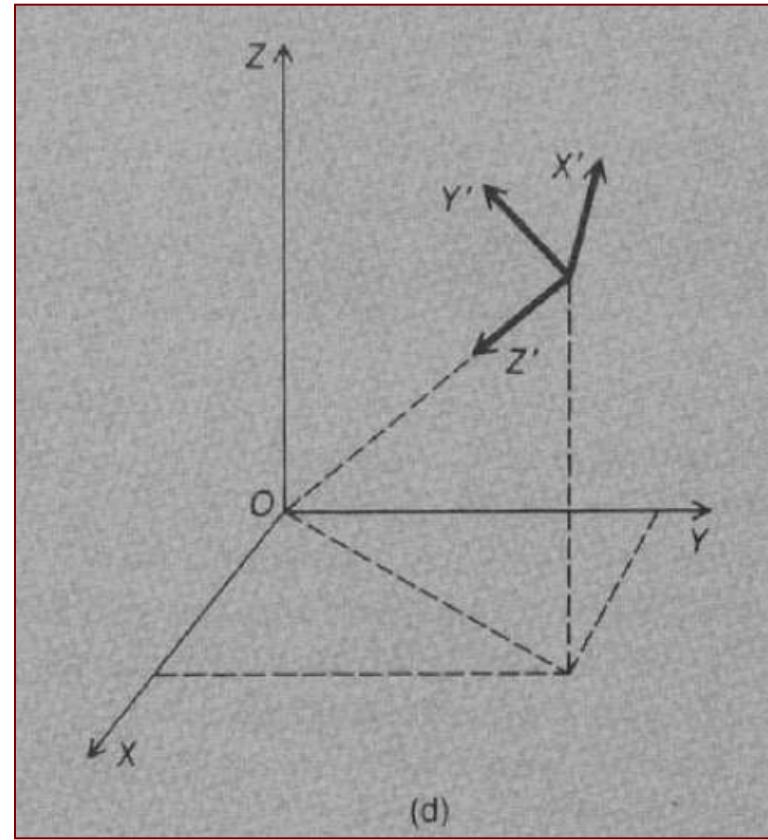
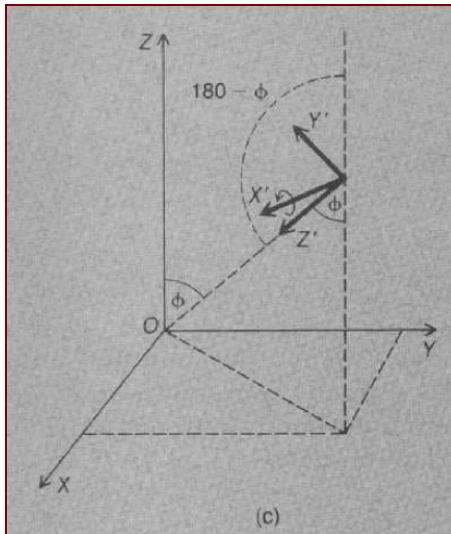


$$M3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & -\cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\sin(180 - \phi) = \sin \phi$$

$$\cos(180 - \phi) = -\cos \phi$$

# Trasformazione Sistema di Riferimento e View-Up vector $\mathbf{Vu}=[0, 0, 1]^\top$



$$M4 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



# Trasformazione Sistema di Riferimento e View-Up vector $\mathbf{Vu}=[0, 0, 1]^T$

$$[x_e, y_e, z_e, 1]^T = VM [x, y, z, 1]^T$$

$$VM = M4^{-1} M3^{-1} M2^{-1} M1^{-1} = \begin{pmatrix} -\sin\theta & \cos\theta & 0 & 0 \\ -\cos\theta \cos\varphi & -\sin\theta \cos\varphi & \sin\varphi & 0 \\ -\cos\theta \sin\varphi & -\sin\theta \sin\varphi & -\cos\varphi & D \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

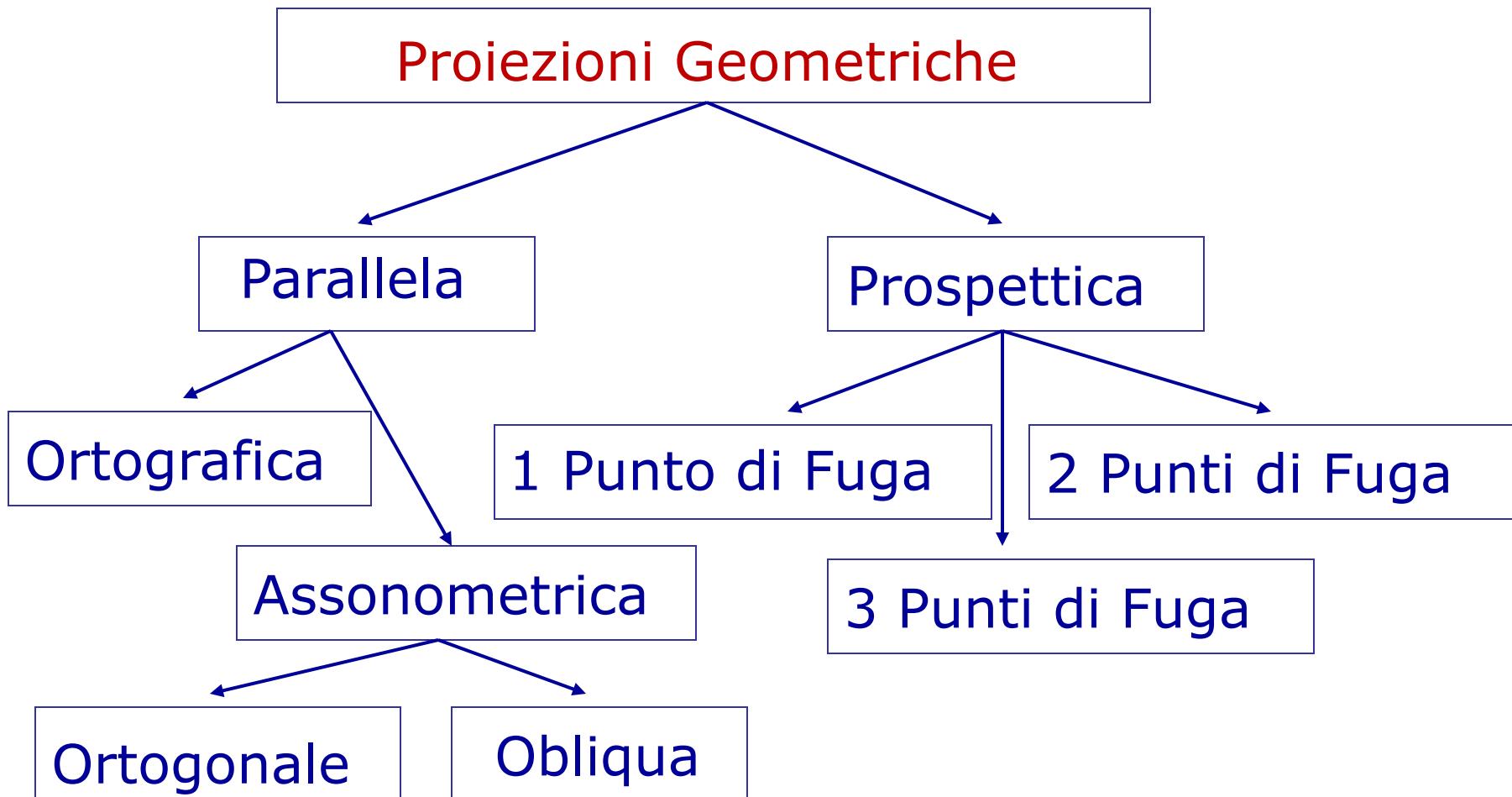
$$x_e = -x \sin\theta + y \cos\theta$$

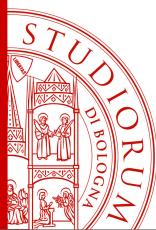
$$y_e = -x \cos\theta \cos\varphi - y \sin\theta \cos\varphi + z \sin\varphi$$

$$z_e = -x \cos\theta \sin\varphi - y \sin\theta \sin\varphi - z \cos\varphi + D$$

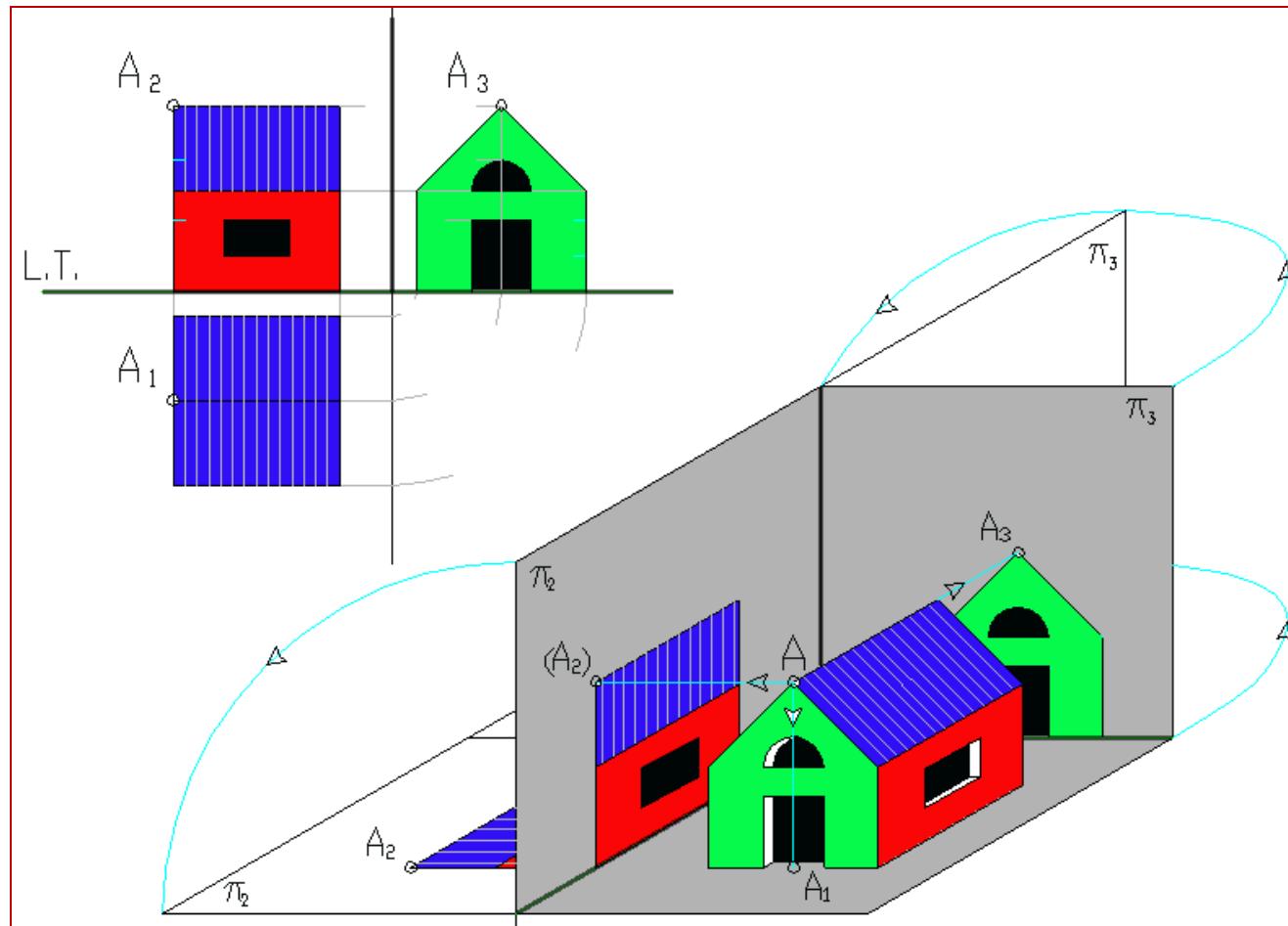


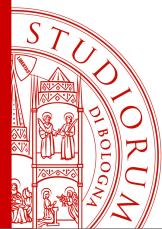
# Proiezioni Geometriche



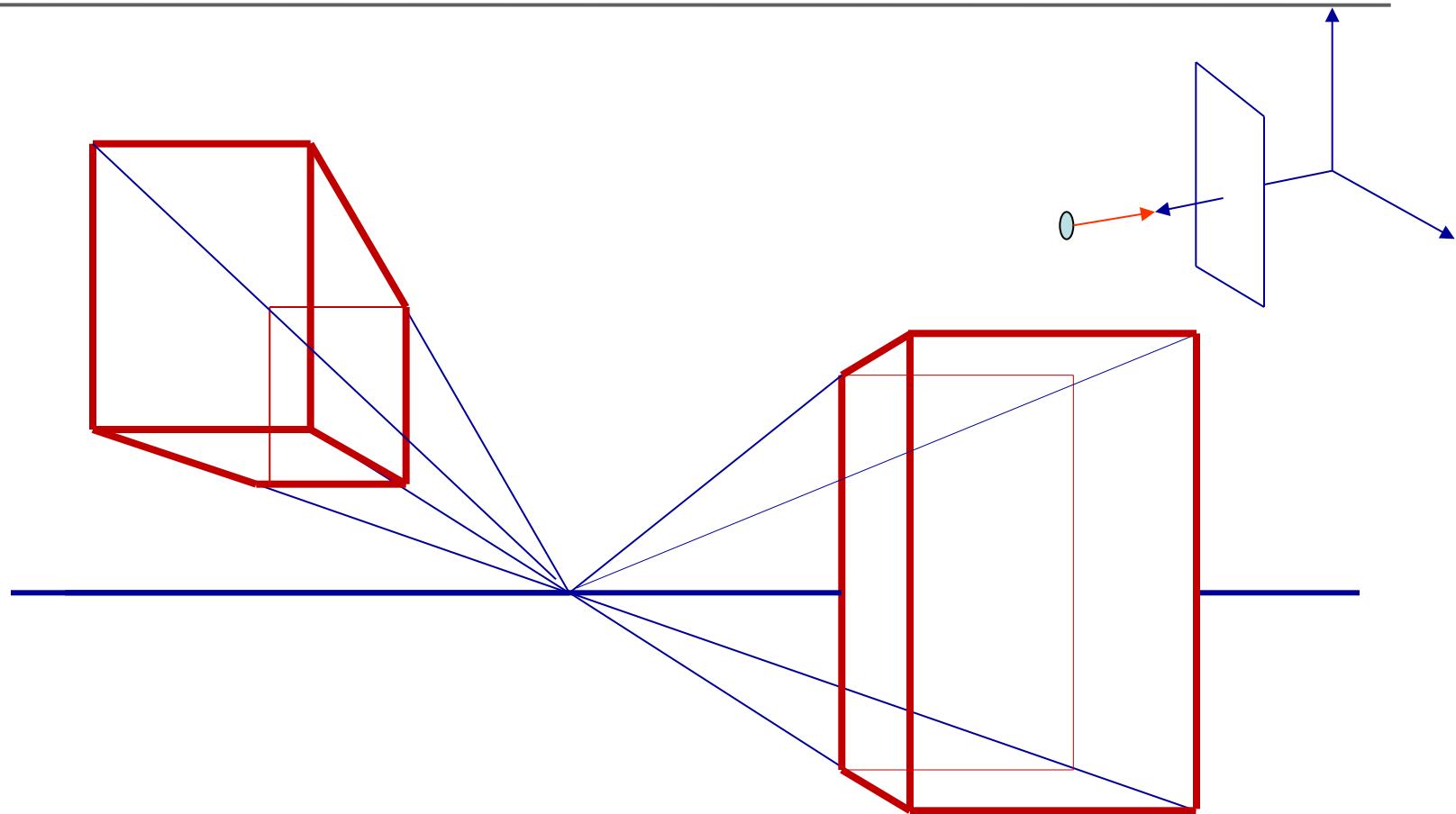


# Esempio: Proiezione Ortografica

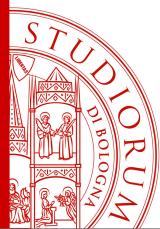




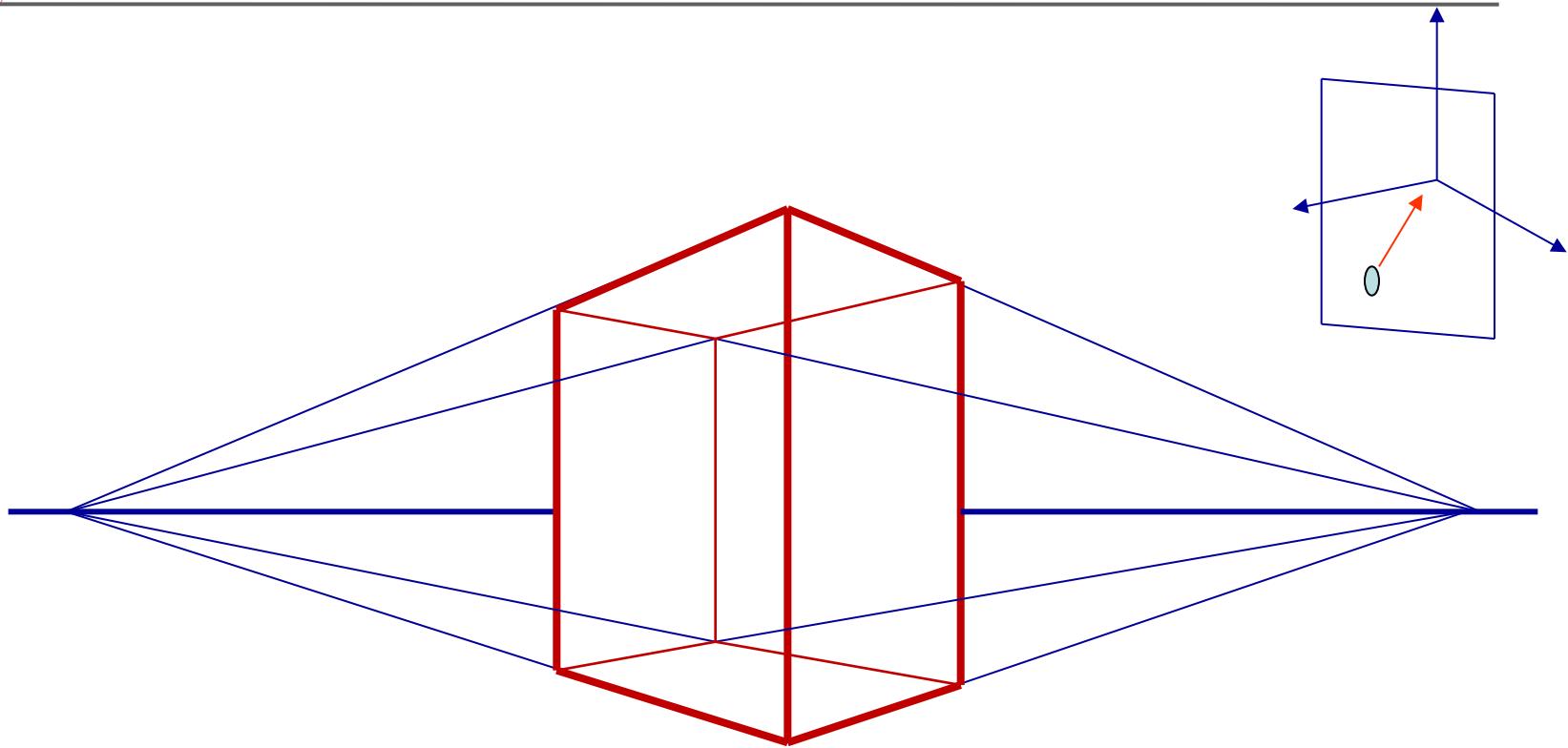
# Proiezione a 1 Punto di Fuga



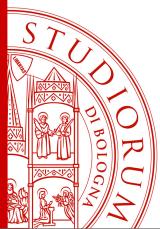
Piano di proiezione ortogonale ad uno degli assi e  
quindi parallelo agli altri due



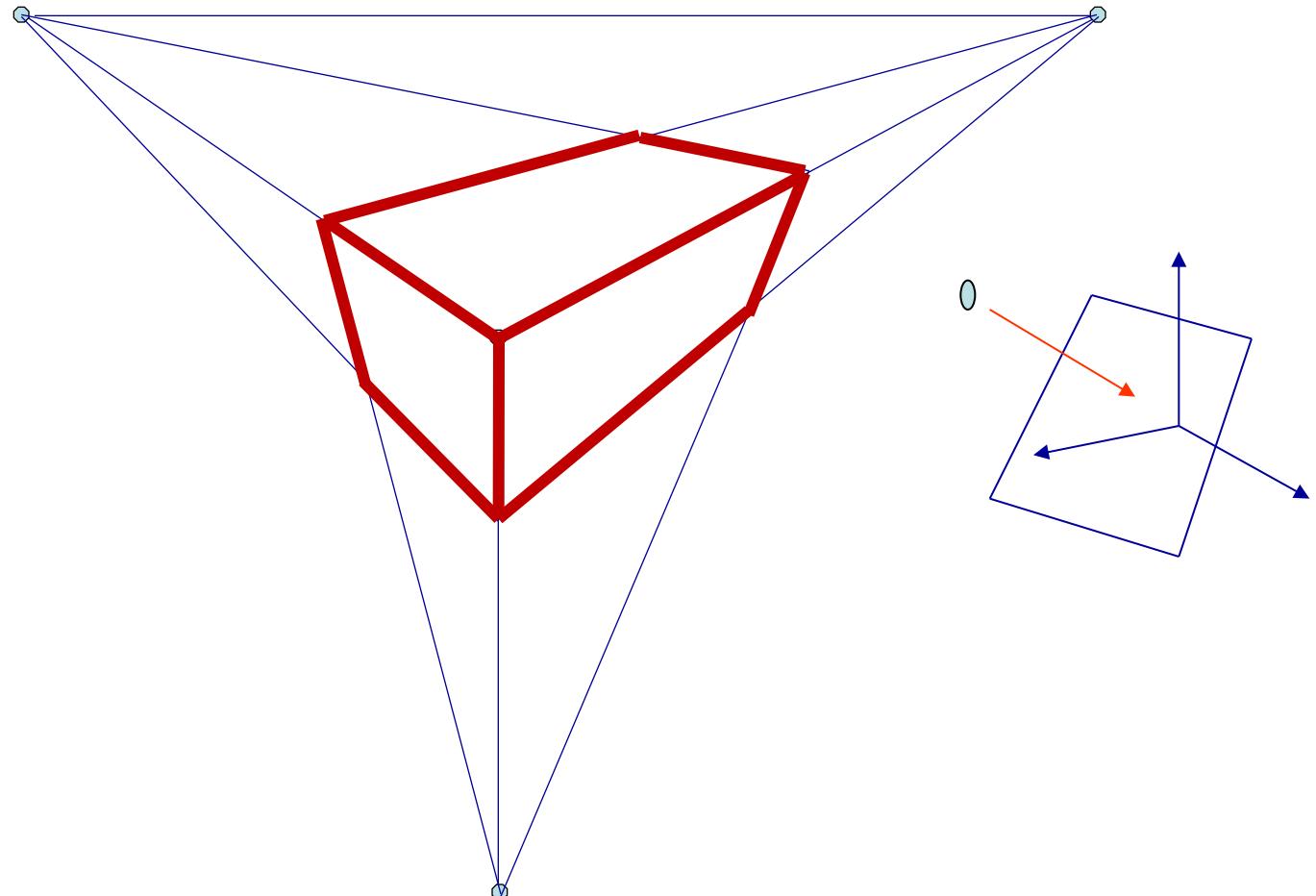
# Proiezione a 2 Punti di Fuga



Piano di proiezione parallelo ad uno degli assi coordinati



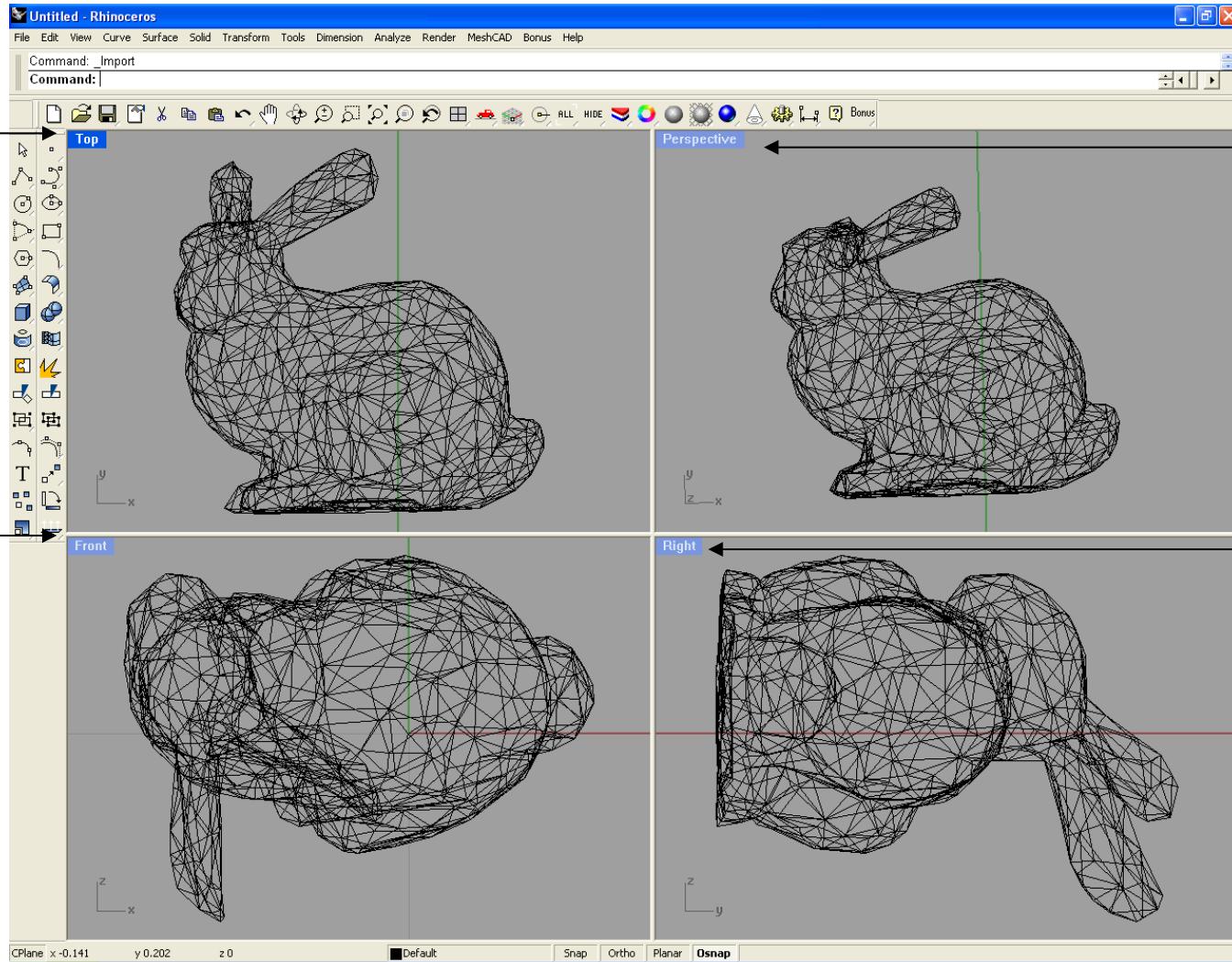
# Proiezione a 3 Punti di Fuga



Piano di proiezione in posizione generica

# Esempio

Top  
1 punto  
di fuga



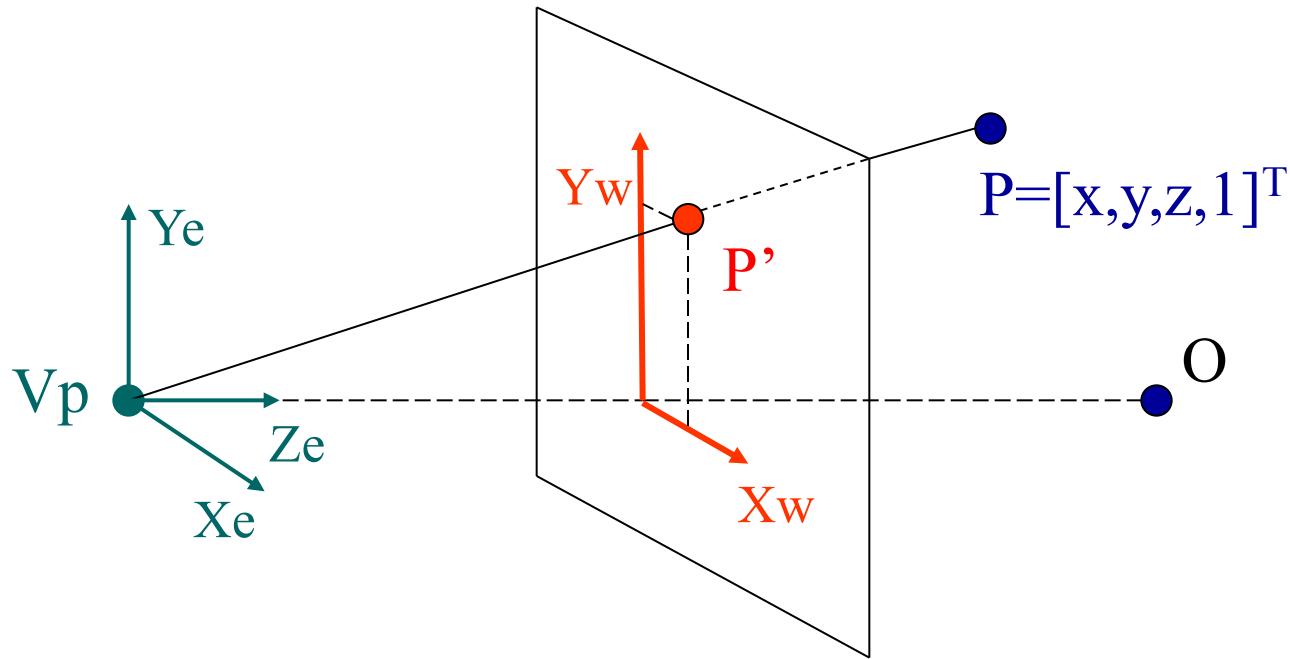
Perspective  
3 punti  
di fuga

Front  
1 punto  
di fuga

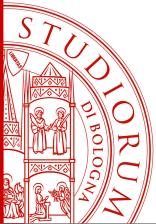
Right  
1 punto  
di fuga

# Proiezione Geometrica

## Proiezione a 3 punti di fuga

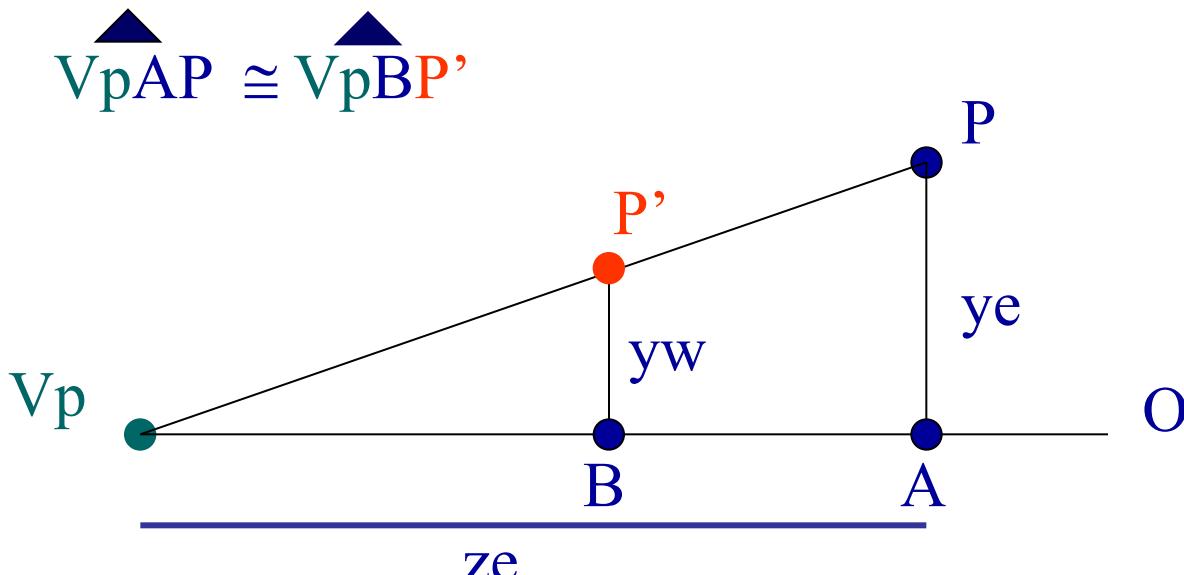


$$[x_e, y_e, z_e, 1]^T = M P$$



# Proiezione Geometrica

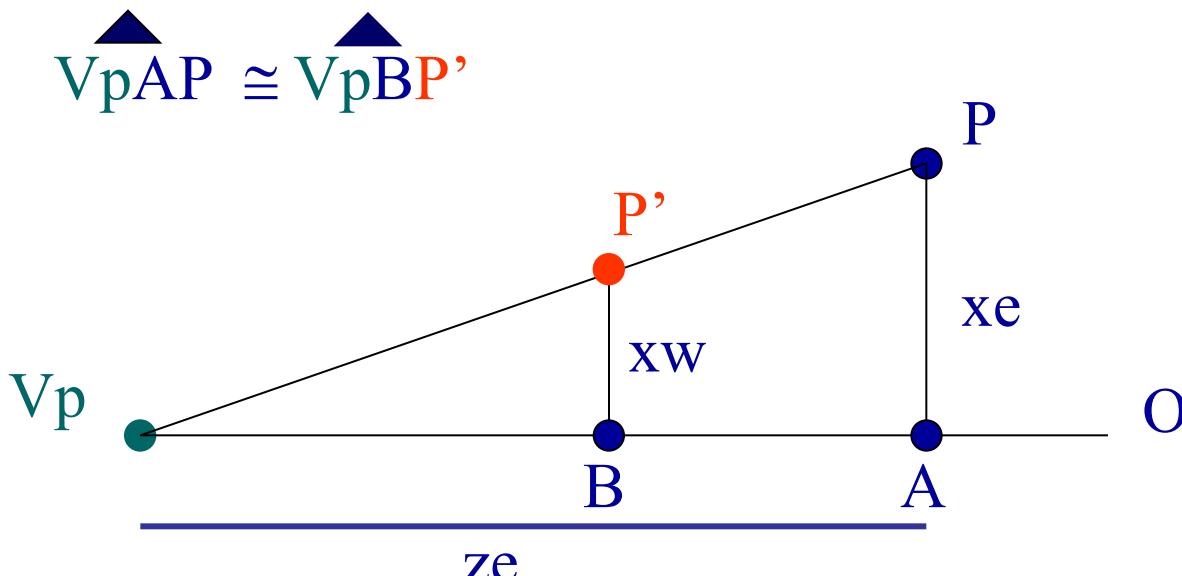
## Proiezione a 3 punti di fuga



$$yw = \frac{d \cdot ye}{ze}$$

# Proiezione Geometrica

## Proiezione a 3 punti di fuga



$$x_w = \frac{d \cdot x_e}{ze}$$



# Proiezione Geometrica

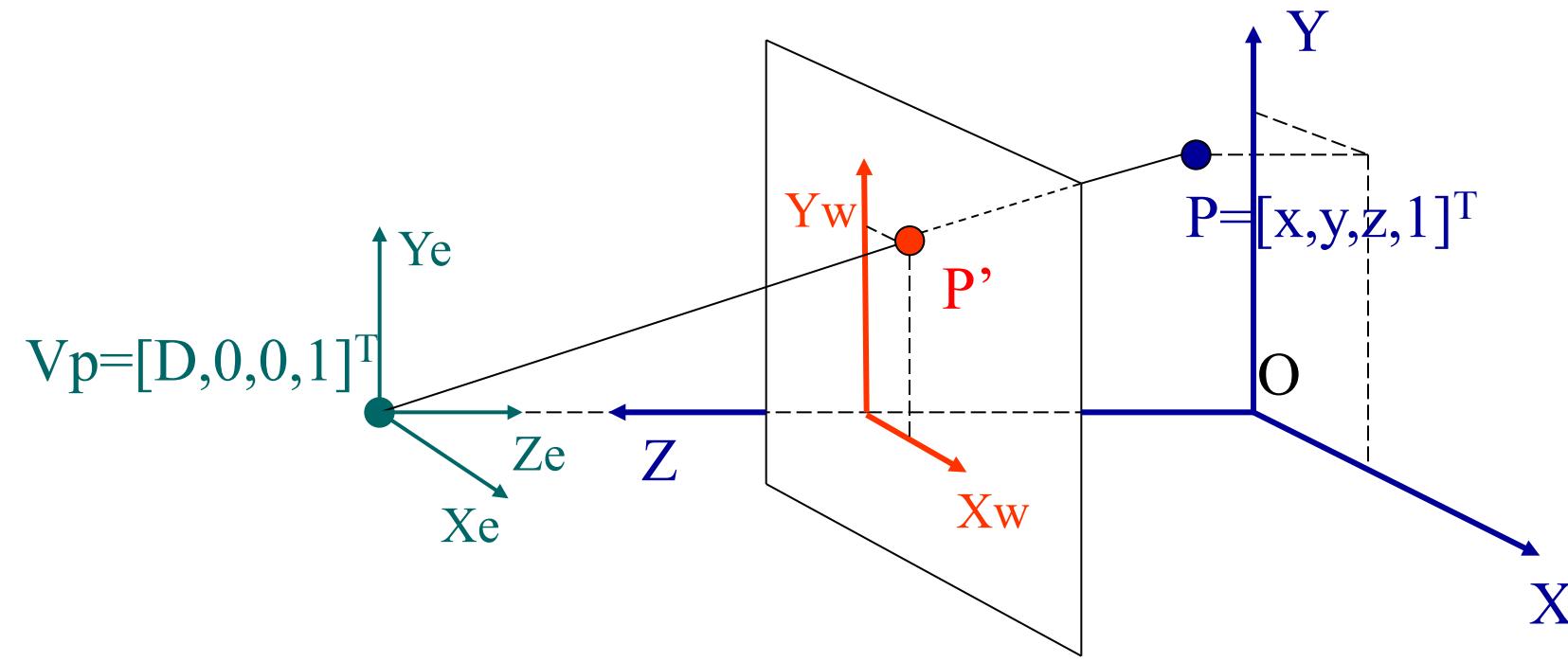
## Proiezione a 3 punti di fuga

$$[x_w, y_w, 1]^T = \text{PrGe} ( [x_e, y_e, z_e, 1]^T )$$

$$x_w = \frac{\mathbf{d} \cdot \mathbf{x}_e}{z_e}$$

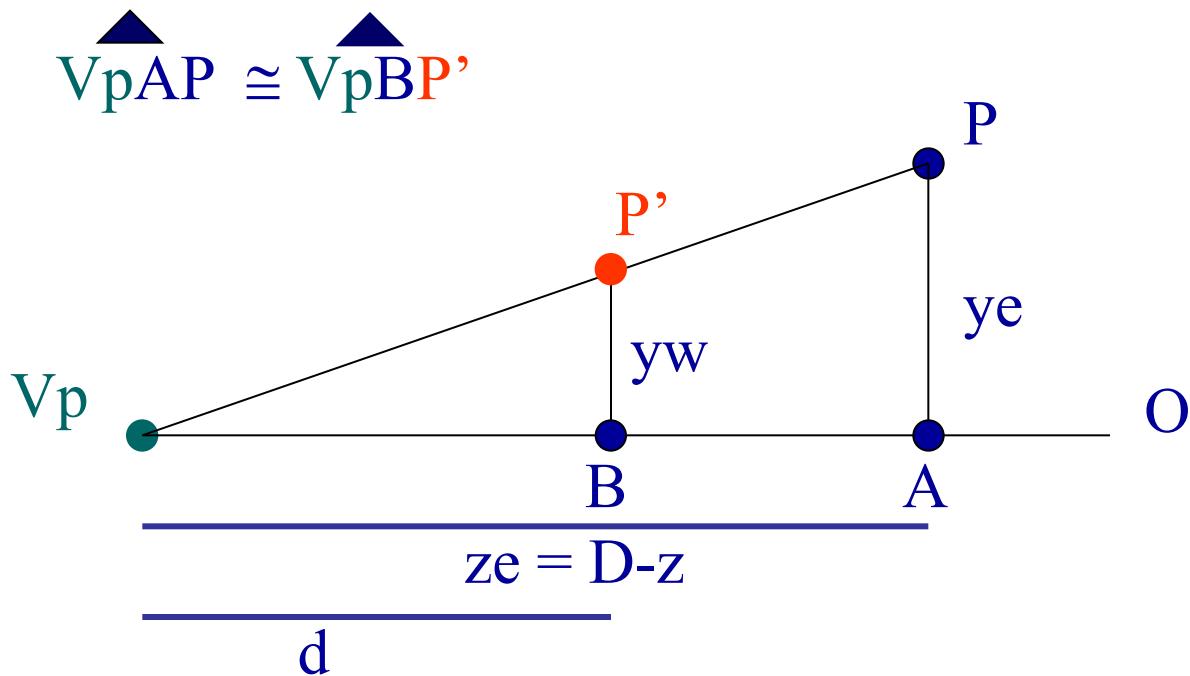
$$y_w = \frac{\mathbf{d} \cdot \mathbf{y}_e}{z_e}$$

# Trasformazione Sistema di Riferimento Proiezione centrale (1 punto di fuga)



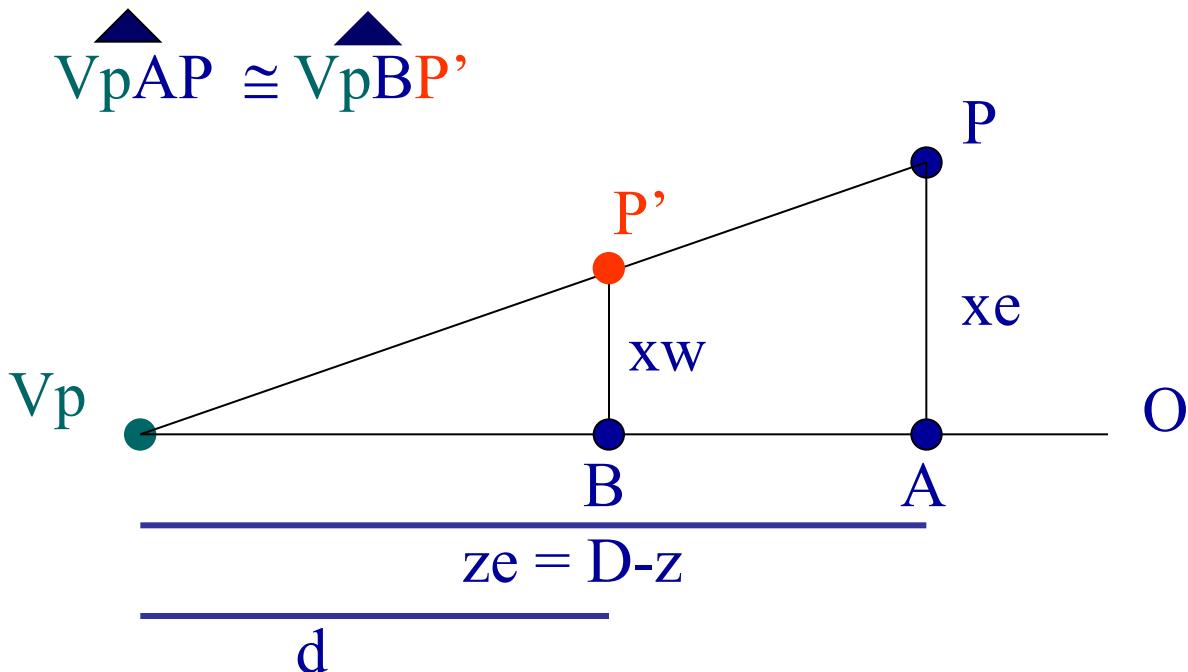
$$P = [x_e, y_e, z_e, 1]^T = [x, y, D-z, 1]^T$$

# Proiezione centrale (1 punto di fuga)



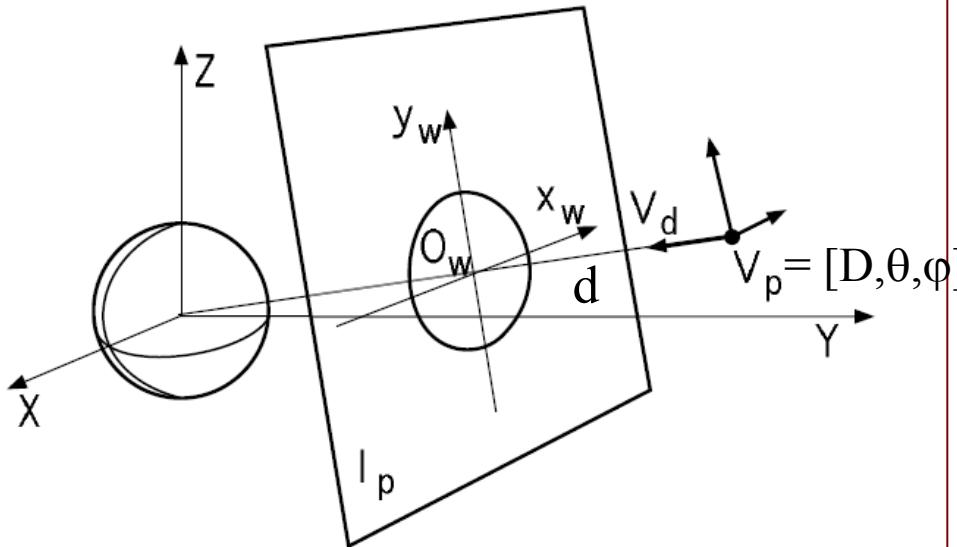
$$yw = \frac{d \cdot ye}{ze} = \frac{d \cdot y}{D - z}$$

# Proiezione centrale (1 punto di fuga)



$$xw = \frac{d \cdot xe}{ze} = \frac{d \cdot x}{D - z}$$

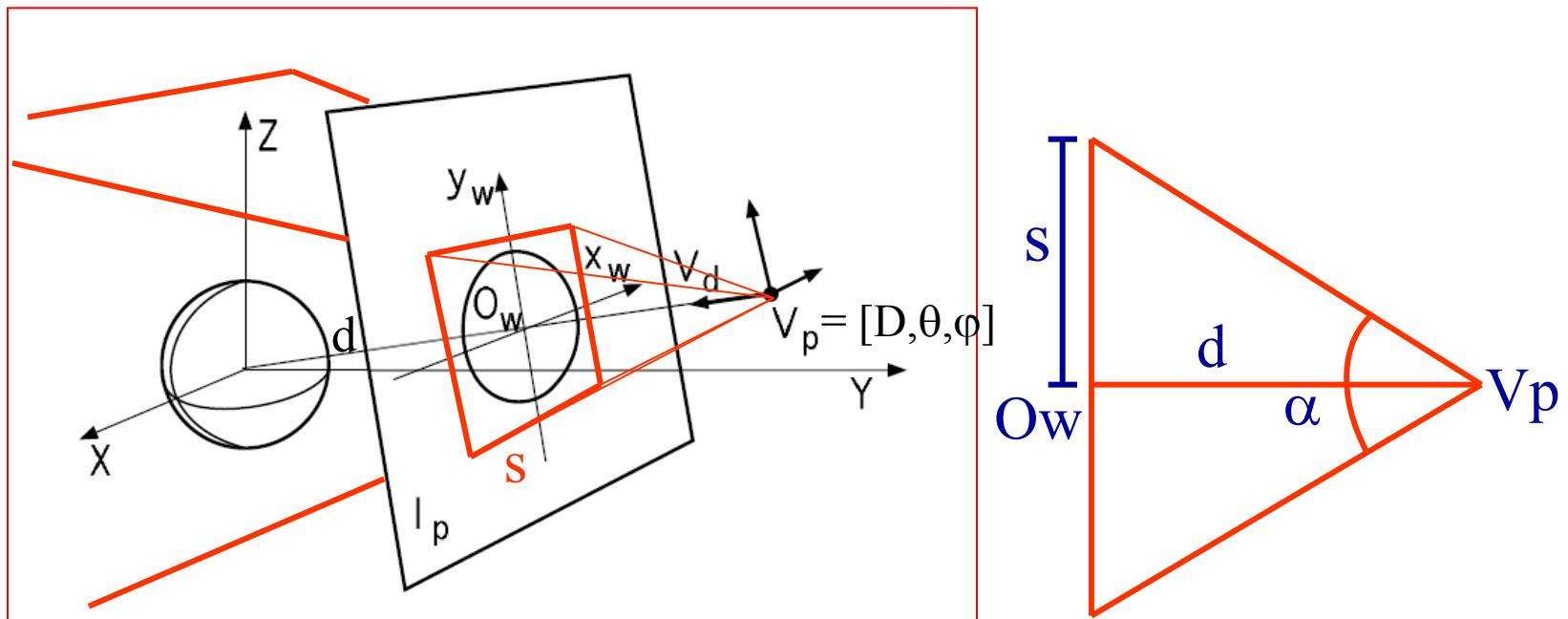
# Parametri di Vista



- Variando  $\theta$  e  $\phi$  è possibile osservare un oggetto da differenti angolazioni;
- Variando  $D$  è possibile spostare l'osservatore vicino o lontano dall'oggetto;

- Tenendo fissato  $V_p$ , variando  $d$  si modifica la dimensione dell' immagine proiettata, ma non della prospettiva;
- Tenendo fissato  $d$ , variando  $D$  si modifica la dimensione dell' immagine proiettata e la prospettiva,

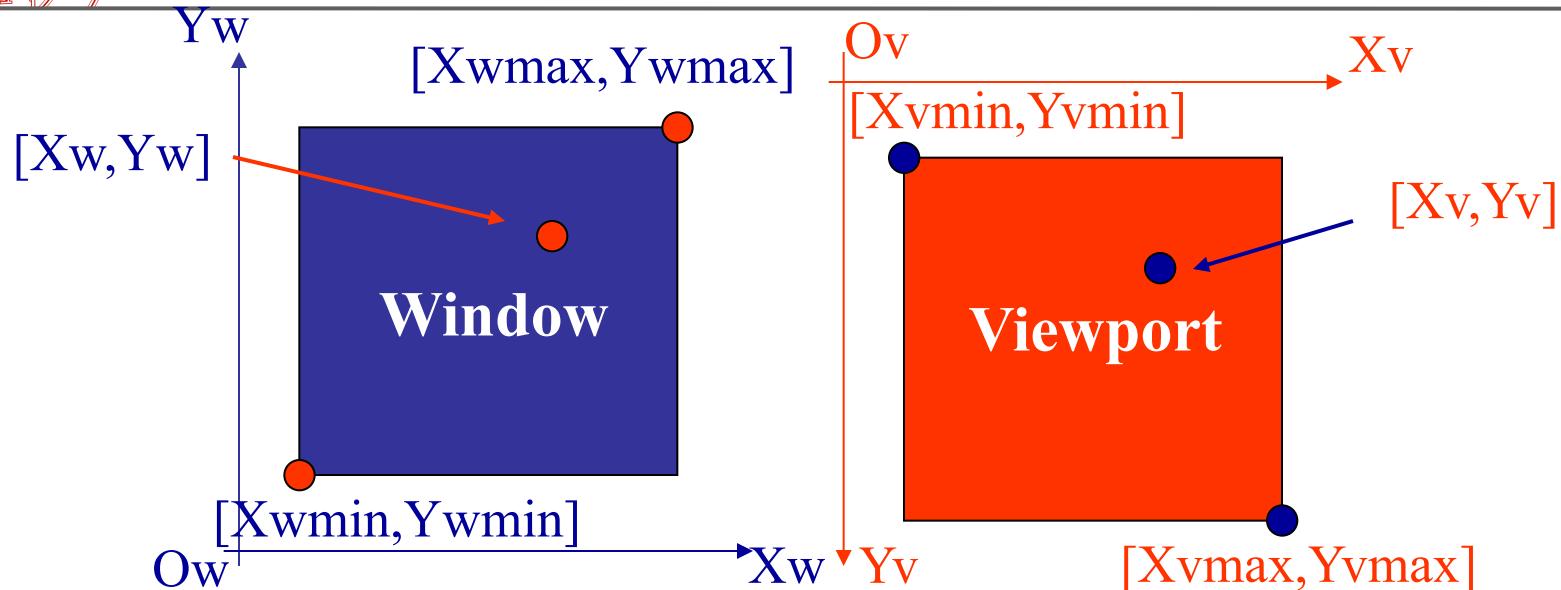
# Determinazione Window



Per determinare l'ampiezza della Window si simula l'ampiezza del cono (o piramide) di vista dell'osservatore mediante la semiampiezza angolare  $\alpha$  e la distanza  $d$  dal piano di proiezione, da cui

$$s = d \cdot \tan(\alpha / 2)$$

# Trasformazione Window-Viewport



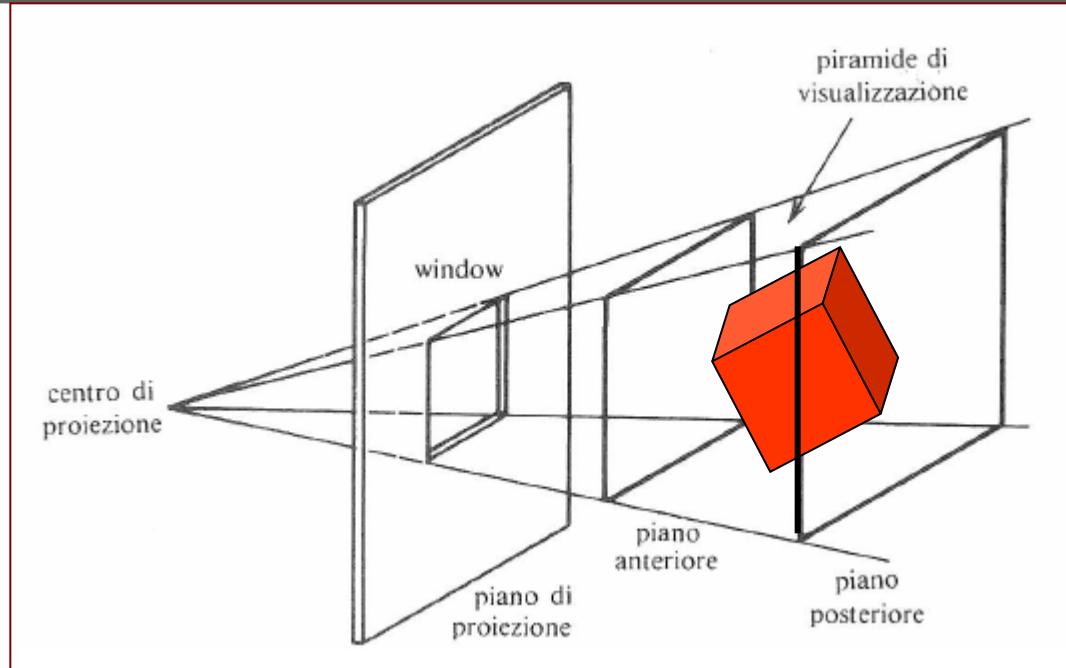
$$X_v = (\text{int})(S_x (X_w - X_{wmin}) + X_{vmin} + 0.5)$$

$$Y_v = (\text{int})(S_y (Y_{wmin} - Y_w) + Y_{vmax} + 0.5)$$

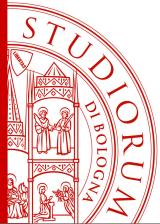
$$S_x = (X_{vmax} - X_{vmin}) / (X_{wmax} - X_{wmin})$$

$$S_y = (Y_{vmax} - Y_{vmin}) / (Y_{wmax} - Y_{wmin})$$

# Tronco di Piramide (Frustum)

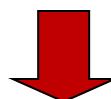


La determinazione di una piramide di vista con l'aggiunta di un front plane ed un back plane dà luogo ad un tronco di piramide detto Volume di Vista o Frustum; solo le parti dell'oggetto interne a tale Volume di Vista saranno visibili e quindi da processare (clipping 3D di punti, di linee e di poligoni).



# Pipeline di Rendering o di Vista

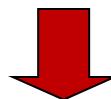
Trasformazione Sistema di Riferimento  
 $(x,y,z,O) \rightarrow (xe,ye,ze,Vp)$



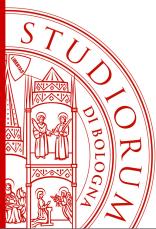
Clipping 3D rispetto al Volume di Vista



Proiezione Geometrica  
 $(xe,ye,ze,Vp) \rightarrow (Xw,Yw,Ow)$



Trasformazione Window-Viewport  
 $(Xw,Yw,Ow) \rightarrow (Xv,Yv,Ov)$



# Esempio

Nella cartella `HTML5_2d_2` vediamo insieme il codice `persp_cube` che usa HTML5 + canvas (contesto 2d) e JavaScript e implementa la trasformazione di vista analizzata:

codice: `persp_cube.html (.js)`

Il codice `persp_cube.js` è costituito dalle seguenti function:

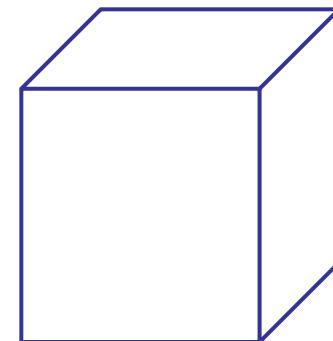
`init()`, `define_object()`, `define_view()`, `render()`,  
`trasf_prosp()` e `win_view()`

vediamole una per una e vediamo in che ordine vengono richiamate.

# function init( )

Questa function viene eseguita una sola volta al caricamento del codice e nell'ordine:

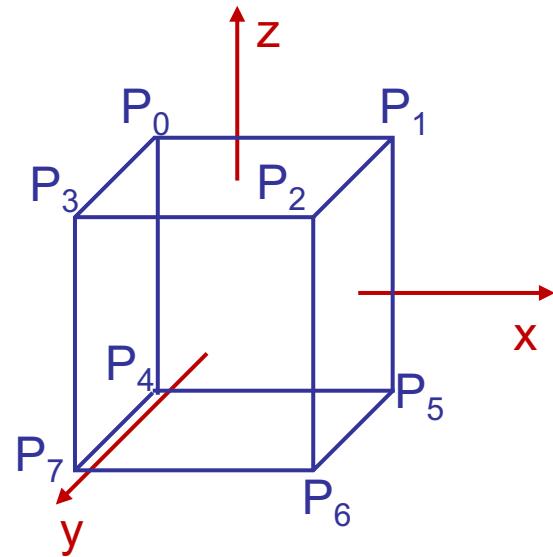
- definisce la viewport
- chiama la function `define_object()` //un cubo
- chiama la function `define_view()`
- chiama la function `render()`



# function define\_object( )

In questa function viene definito un cubo mediante i vertici e le facce quadrate:

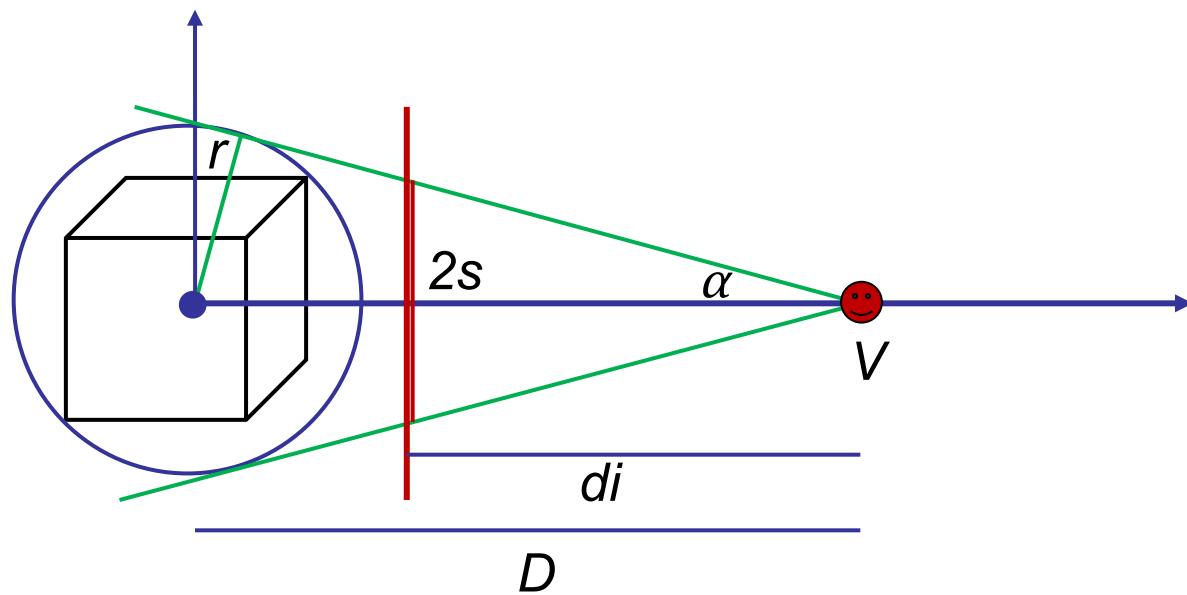
```
nvert=8;  
x=[-1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0];  
y=[-1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, -1.0];  
z=[1.0, 1.0, 1.0, 1.0, -1.0, -1.0, -1.0, -1.0];
```



```
nface=6;  
face=[[3, 2, 1, 0],[0, 1, 5, 4],[6, 5, 1, 2],[3, 7, 6, 2],[4, 7, 3, 0],[4, 5, 6, 7]];
```

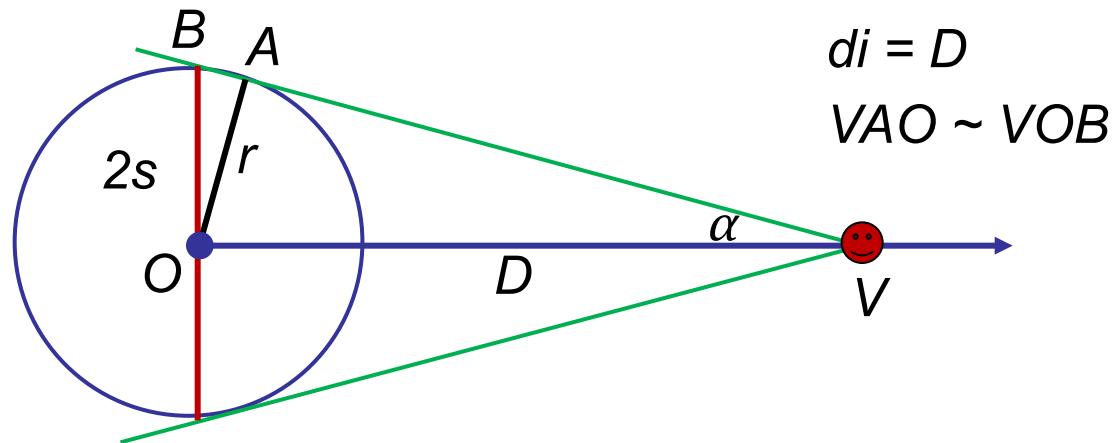
# function define\_view( )

Questa function definisce i parametri di vista in modo tale da garantire che il cubo venga interamente visualizzato all'interno del canvas



# function define\_view( )

In realtà assume che  $di=D$  così da semplificare qualche conto, ma senza perdere in generalità (considera il piano di proiezione passante per l'origine del sistema dell'oggetto)



$$di = D$$
$$VAO \sim VOB$$

$$AV = \sqrt{D^2 - r^2}$$

$$OB : OA = OV : AV$$

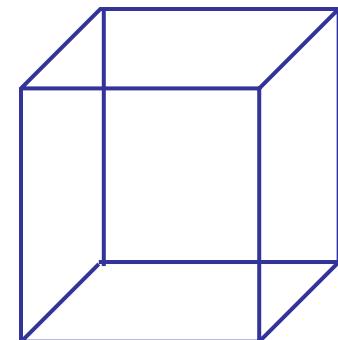
$$OB = s = r * D / \sqrt{D^2 - r^2}$$

$$\alpha = \arctan(s/D)$$

# function render( )

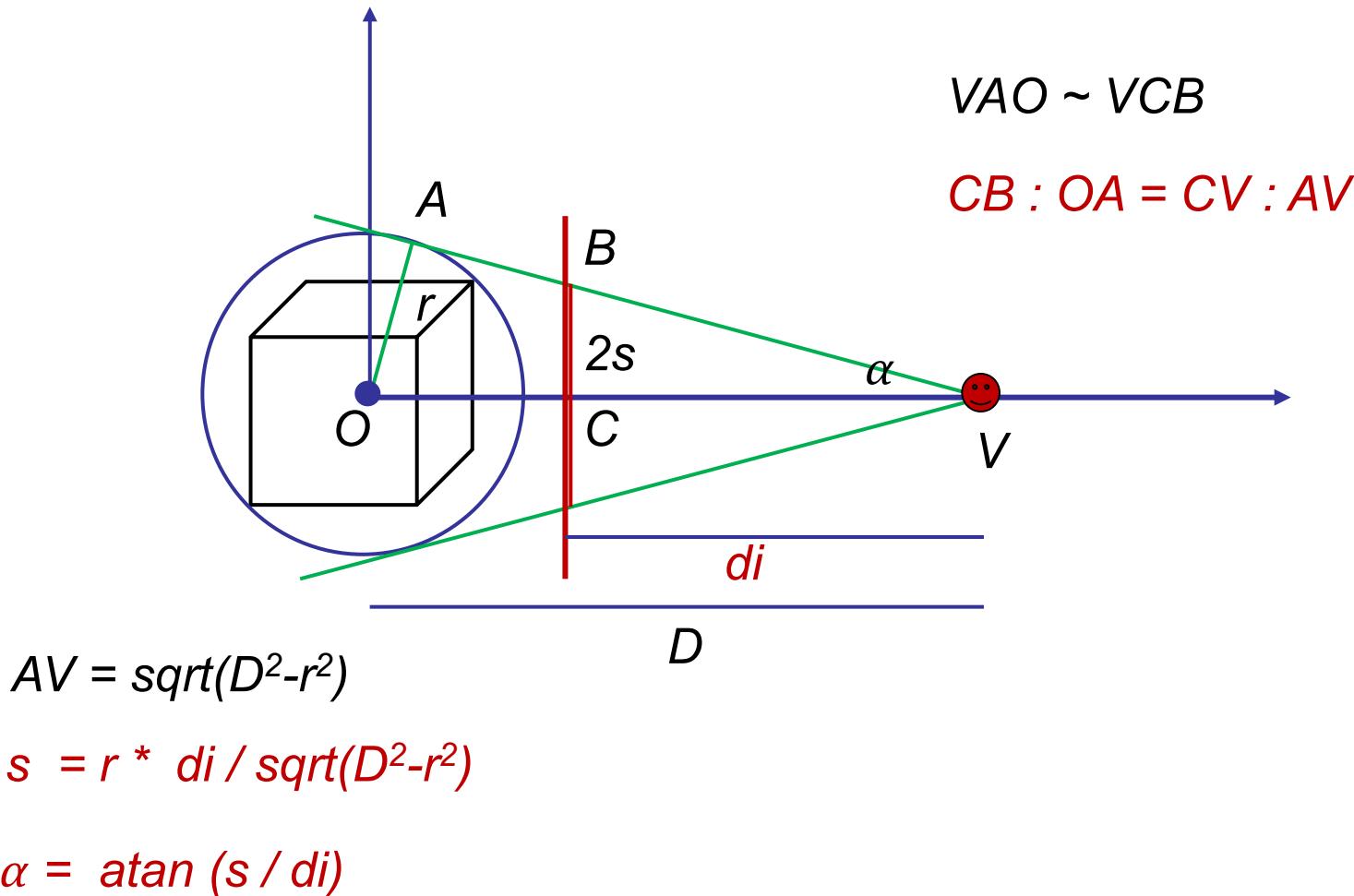
Questa function viene richiamata inizialmente, ma poi richiamata ed eseguita ogni volta che viene cambiato un parametro di vista e nell'ordine:

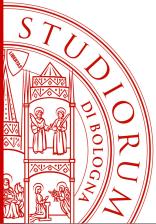
- Pulisce il canvas/viewport
- Definisce la window
- Trasla l'oggetto nell'origine
- Cambia il sistema di riferimento
- Applica la trasformazione prospettica  
function **trasf\_prosp\_gen()**
- Applica la trasformazione window-viewport  
function **win\_view()**
- Disegna tutti lati del cubo  
function **moveTo()** e **lineTo()**



# function define\_view()

Se volessimo tenere distinti  $di$  e  $D$  dovremmo fare:





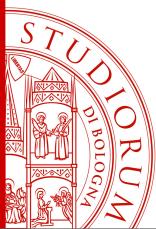
# Esercizio 2

Modificare il codice `persp_cube.js` per sperimentare:

- 1.gestione interattiva dei parametri di vista (angoli  $\theta$  e  $\phi$ ) col mouse (rotella per la distanza);
- 2.inserire la trasformazione di vista con View-Up vector;
- 3.utilizzando la function `copy_trasl` (trasformazione geometrica di traslazione) realizzare il logo del corso;
- 4.disegnare una triangolazione delle facce quadrilatero;
- 5.determinare una lista non ripetuta dei lati del cubo per disegnarli una sola volta;

Io si chiama `persp_mesh.html` , `.js`

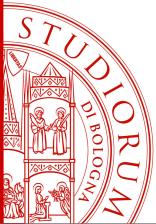
**Nota:** nella pipeline di vista, per ora e per semplicità, non si gestisce il clipping 3D rispetto al frustum.



# Proposte di Esercizi (per esperti)

Modificare il codice `persp_mesh.js` per:

1. animazione di un oggetto sfera che rimbalza sulle pareti di una stanza a forma di cubo (generalizzare il codice `ball3.js`);
2. modificare interattivamente una mesh 3D con il mouse agendo sui vertici.
3. caricare un oggetto mesh 3D dato in formato Wavefront `file.obj` (si utilizzino i file `.obj` nella cartella `data`);



# Formato OBJ

[https://en.wikipedia.org/wiki/Wavefront\\_.obj\\_file](https://en.wikipedia.org/wiki/Wavefront_.obj_file)

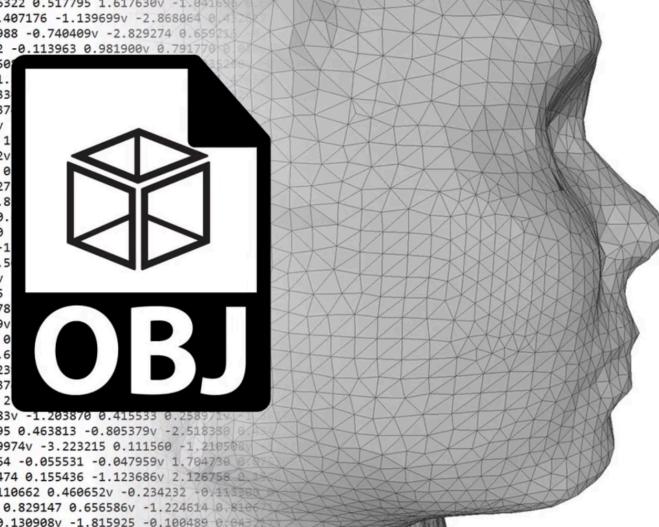
E' un formato commerciale della Alias-Wavefront molto diffuso. Puo' essere in binario o in ASCII (testo). Permette la memorizzazione di mesh 3D e varie info utili per il loro disegno:

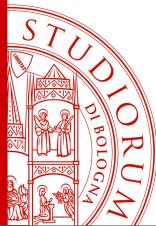
```
# Blender v3.0 (sub 0) OBJ File: ''
# www.blender.org

# 8 vertices
v 1.000000 -1.000000 1.000000
v 1.000000 1.000000 1.000000
v -1.000000 1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 1.000000 -1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -1.000000
v 1.000000 -1.000000 -1.000000
...

```

```
/ @.104620v 1.204145 @.081610v 1.692585 @.081610v
255595 0.739515v -1.056322 0.517795 1.617630v -1.04185
-1.230242v -2.946309 0.407176 -1.139699v -2.868064 0.1
53995v -2.454060 0.589888 -0.748409v -2.829274 0.6591
4338 0.891223v 1.118822 -0.113963 0.981908v 0.791770
-0.445145 0.067123 1.650
-0.112094 -0.642649v 1.
.482656 0.030199 -0.2533
343066 -1.268350v -2.637
368 0.091403 -0.582442v
87v -0.218858 0.351707 1
1707 0.144306 -1.252632v
568v 1.099426 0.566102 0
0.127559 -0.885735v 2.27
9v 0.7766052 0.633870 0.8
93 0.384322v 0.350730 0.
5333 0.056744v 1.273430
5v 2.060726 -0.097557 -1
-0.144497 -1.488848v 2.5
326 0.195691 -1.106098v
3172v 2.569728 0.032102
304097 -1.215309v 1.4478
8552 -0.250406 0.694189v
31v -0.620278 0.810392 0
0.729658 0.097510v -2.6
0.260683 0.763564v -0.23
.081634v 1.601403 -0.037
712 0.409308 0.357014v 2
357714 0.597624 0.218983v -1.203870 0.415533 0.258970v
890 -0.521673v -2.431395 0.463813 -0.805379v -2.518380v
.189487 0.053463 -1.319974 -3.223215 0.111566 -1.210584
56499 0.012985v 1.510364 -0.055531 -0.047959v 1.704778v
55564 -0.850687v 2.100471 0.155436 -1.123686v 2.126758v
.415715v -0.025497 -0.110662 0.466652v -0.234232 0.111566
83 0.746149v -1.458419 0.829147 0.656558v -1.224614 -0.1407
v -1.679804 -0.106087 0.130908v -1.815925 -0.100489 0.0408
```





# Formato OBJ continua

```
...
# 8 texture coordinates
vt 0.5 0.5
...
# 8 normal vertices
vn 1.0 0.0 0.0
...
# 6 faces

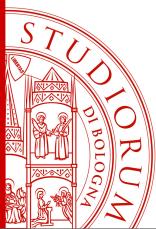
f 1/1/4 2/2/1 3/3/2 4/4/3
f 4/4/4 3/3/1 5/5/2 6/4/3
f 7/7/4 5/5/1 3/5/2 2/2/3
f 1/1/4 8/8/1 7/7/2 2/2/3
f 6/6/4 8/8/1 1/1/2 4/4/3
f 6/6/4 5/5/1 7/7/2 8/8/3
```

Indice coord. vertice

f v/vt/vn v/vt/vn v/vt/vn

Indice coord. texture

Indice coord. normale



# glm\_utils.js e mesh\_utils.js

[HTML5\\_2d\\_2/resources/glm\\_utils.js](#)

in particolare:

**glm\_readOBJ**: carica un file in formato OBJ Wavefront e ritorna in una struttura **subd\_mesh** tutte le informazioni utili per il disegno di una mesh (vertici, facce, normali, coordinate texture, ecc.);

**Unitize**: scala e trasla una mesh per portarla in una sfera unitaria centrata nell'origine;

[HTML5\\_2d\\_2/resources/mesh\\_utils.js](#)

in particolare:

**LoadSubdMesh**: partendo da una definizione di mesh in termini di vertici V e facce FV (Vertici di ogni Faccia), determina tutte le informazioni utili per la gestione di **unstructured mesh** (progetto **subdivision surfaces**) (la rivedremo più avanti)



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA

**Giulio Casciola**  
Dip. di Matematica  
[giulio.casciola at unibo.it](mailto:giulio.casciola@unibo.it)