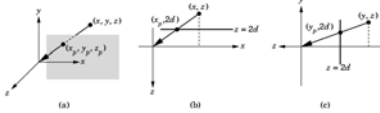


Matrici di Proiezione Prospettica

- ❖ Assunto che siamo nel sistema di riferimento della camera con il centro di proiezione nell'origine, e il piano di proiezione a distanza d lungo l'asse $-z$
- ❖ Vogliamo trovare la proiezione (x_p, y_p, z_p) sul piano di proiezione di un punto (x, y, z)



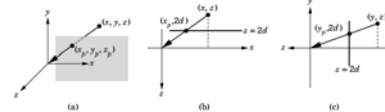
Proiezione Prospettica

- ❖ Si ottiene che:

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

$$x_p = \frac{x}{z/d} \quad y_p = \frac{y}{z/d}$$

- ❖ Nota che questa trasformazione non è lineare, né affine, né reversibile.



Coordinate Omogenee

- ❖ Estendiamo la nostra def di coordinate omogenee dicendo che un punto p può essere rappresentato come

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

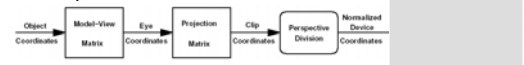
- ❖ Un punto in 3D corrisponde ad una linea in 4d.
- ❖ Posso sempre recuperare la forma con 1 come quarto elemento
- ❖ Posso fare matrici che modificano il quarto elemento.

Coordinate Omogenee

- ❖ In particolare possiamo definire la matrice

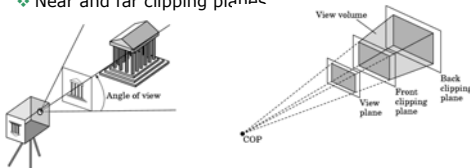
$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \quad \mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \mathbf{q} = \mathbf{M}\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} \approx \begin{bmatrix} x \\ z/d \\ y \\ z/d \\ d \\ 1 \end{bmatrix}$$

- ❖ Che effettua la trasformazione prospettica, purché si normalizzi dividendo per la quarta componente



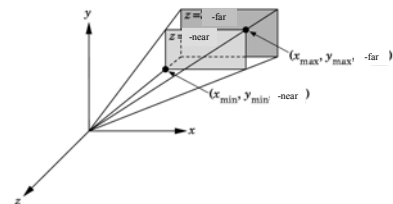
Proiezione prospettica in opengl

- ❖ Finora abbiamo definito solo l'operazione di proiezione.
- ❖ Per definire una camera dobbiamo anche definire il view volume
 - ❖ Angle of view
 - ❖ Near and far clipping planes



Proiezione prospettica in opengl

- ❖ `glFrustum(xmin, xmax, ymin, ymax, near, far);`
- ❖ `gluPerspective(fov, aspect, near, far)`



Proiezioni Ortogonali

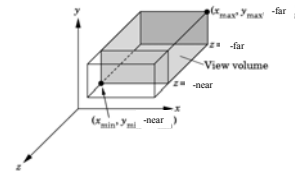
- ❖ Caso particolare di proiezione parallela in cui le linee di proiezione sono perpendicolari al view plane
- ❖ La proiezione è semplicemente

$$x_p = x \quad y_p = y \quad z_p = 0$$

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

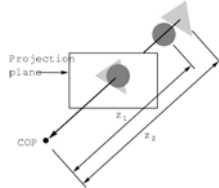
Proiezioni Ortogonali in OpenGL

- ❖ `glOrtho(xmin,xmax,ymin,ymax,near,far);`
- ❖ I clipping planes sono a $z = -near$ e $z = -far$



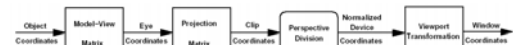
Hidden Surface Removal

- ❖ Si deve disegnare sul frame buffer solo quelle porzioni di primitive che sono davanti a tutte le altre; esistono numerosi algoritmi
- ❖ Tecnica Zbuffer, per ogni pixel dello schermo memorizzo la minima distanza dal centro di proiezione disegnata in quel pixel.
- ❖ Quando rasterizzo, disegno (e aggiorno lo zbuffer, solo se davanti.



Window e Device coords

- ❖ In OpenGL si distingue tra
- ❖ Normalized Device (screen) Coords
 - ❖ Sono 3d, quel che si vede e' nel canonical view volume.
- ❖ Window Coord
 - ❖ Sono sempre 3d, ma con x,y espresse in pixel, e la z in [0,1];



Normalized Device Coord

- ❖ In OpenGL Perspective Transf + division Convertono a Normalized Device Coord
- ❖ La geometria è quindi clippata sul Canonical View Volume

$$\begin{aligned} x &= \pm 1 \\ y &= \pm 1 \\ z &= \pm 1 \end{aligned}$$

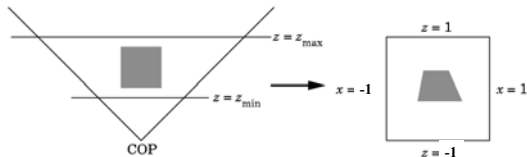
Mapping the view volume

- ❖ Sul Canonical view Volume,
- ❖ In questo caso la proiezione ortografica diventa

$$\begin{bmatrix} \frac{2}{x_{max} - x_{min}} & 0 & 0 & -\frac{x_{max} + x_{min}}{x_{max} - x_{min}} \\ 0 & \frac{2}{y_{max} - y_{min}} & 0 & -\frac{y_{max} + y_{min}}{y_{max} - y_{min}} \\ 0 & 0 & -\frac{2}{far - near} & \frac{far + near}{far - near} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{min} & y_{min} & -near \\ x_{max} & y_{max} & -far \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

Mapping the View Volume

- ❖ Anche le trasformazioni prospettiche mappano sul canonical view volume
- ❖ In questo caso l'applicazione della matrice di proiezione prospettica *distorce* la scena in maniera tale da introdurre il rimpicciolimento prospettico.



Window Coordinate

- ❖ Il passaggio da normalized device coords a window coord. è gestito dalla viewport transformation

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} (p_x/2)x_d + o_x \\ (p_y/2)y_d + o_y \\ (z_d + 1)/2 \end{bmatrix} \begin{matrix} (p_x, p_y) & \text{dimensioni della finestra} \\ (o_x, o_y) & \text{centro della finestra} \\ [x_d, y_d, z_d]^T & \text{punto in normalized device coord} \end{matrix}$$

Window Coordinate

- ❖ In OpenGL il passaggio da Normalized Device (screen) Coords a window coord è controllato dal comando
- ❖ `glViewport(x,y,w,h)`
 x, y origine del lower left corner della finestra
 w, h dimensioni in pixel della finestra
- ❖ Nota in effetti anche la z può essere mappata in un range diverso (ma sempre incluso) da $[0, 1]$
- ❖ `glDepthRange(znear, zfar)`