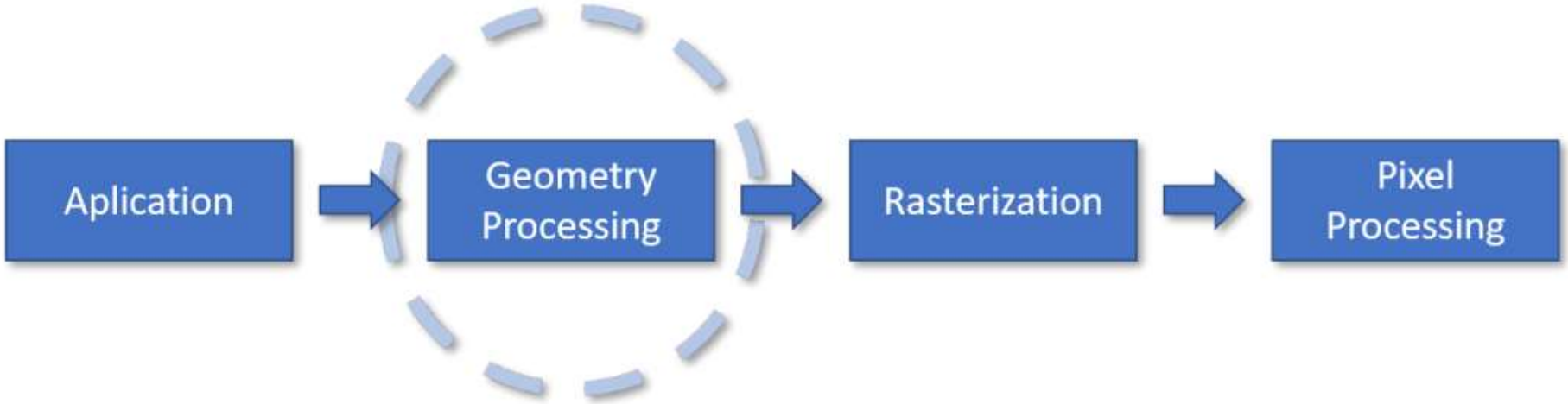
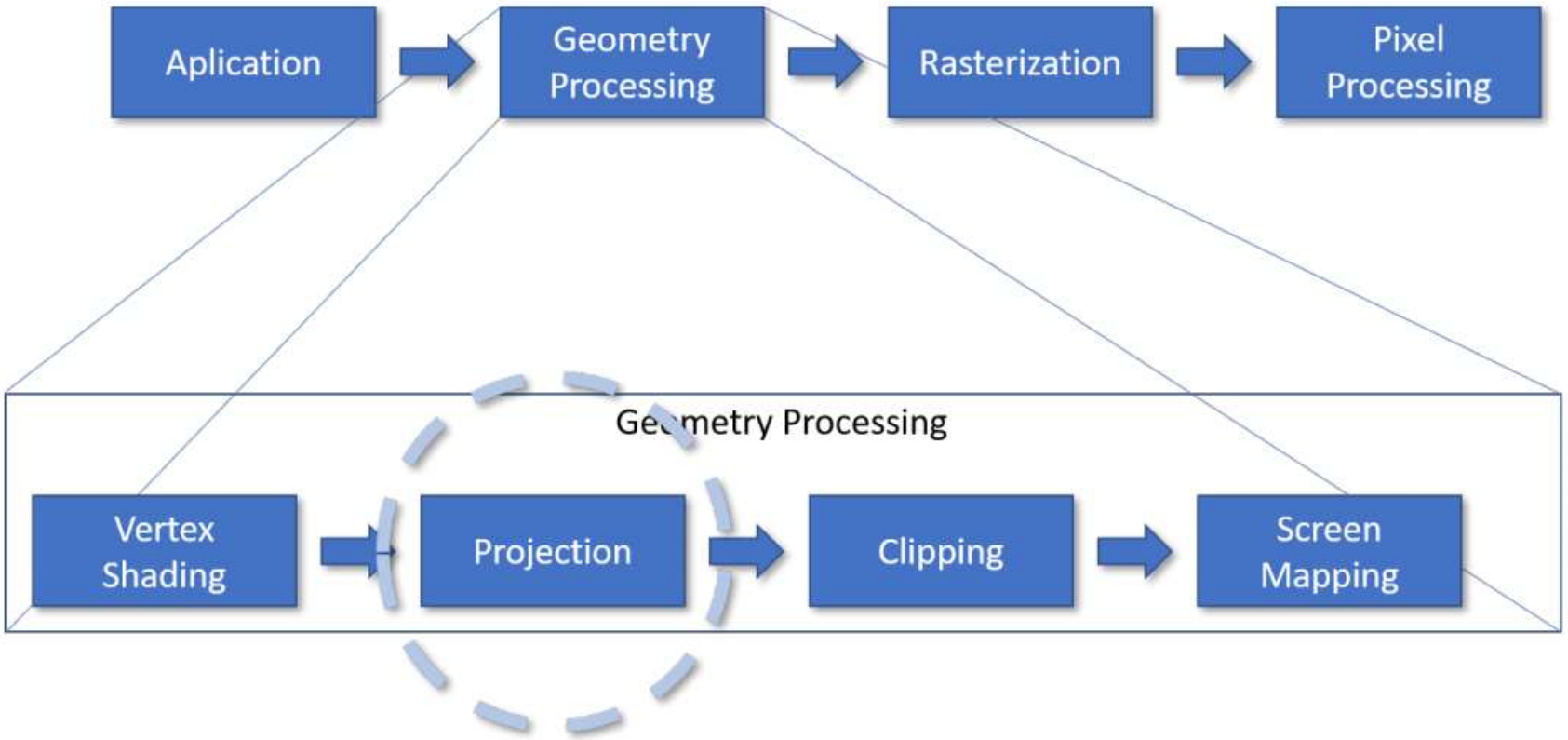
Transformaciones de Vista y Proyecciones

Dr. Ivan Sipiran

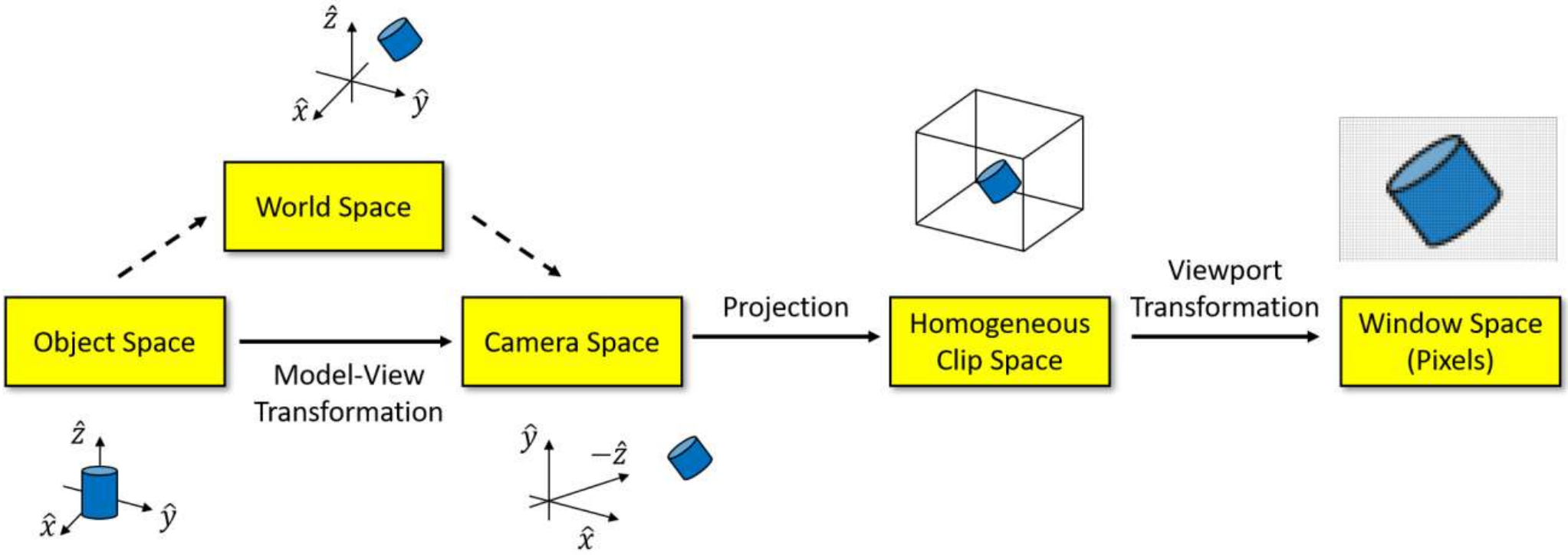
Graphics Rendering Pipeline



Graphics Rendering Pipeline

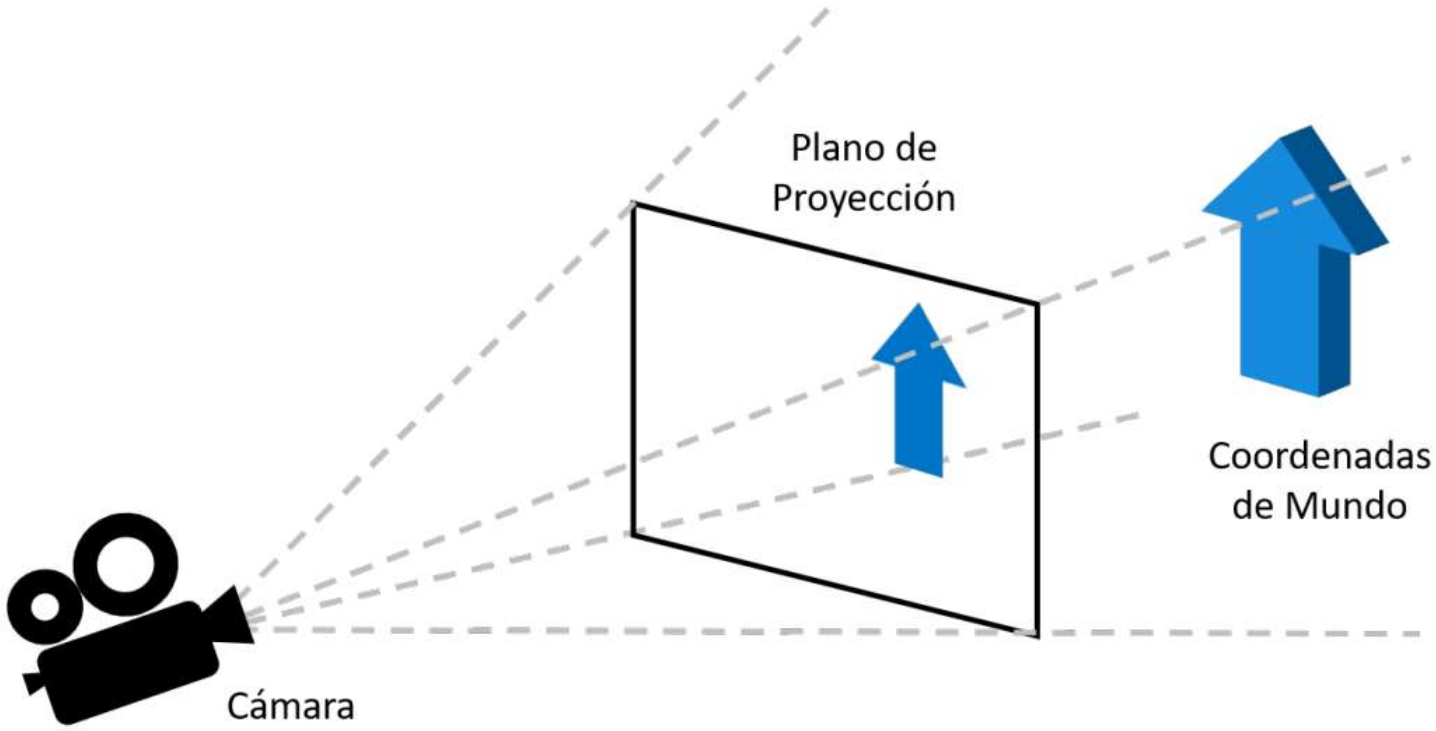


Sistemas de Coordenadas

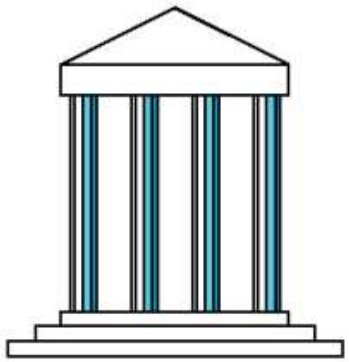


Proyecciones

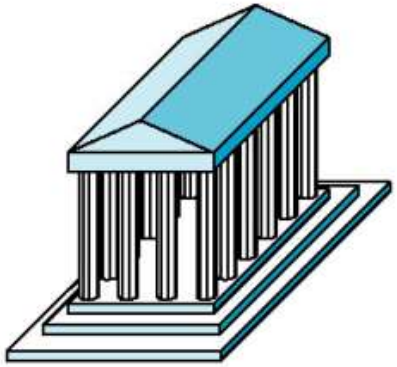
- Proyectar el espacio 3D en coordenadas del mundo sobre un plano de proyección



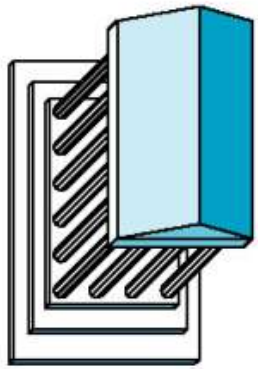
Proyecciones clásicas



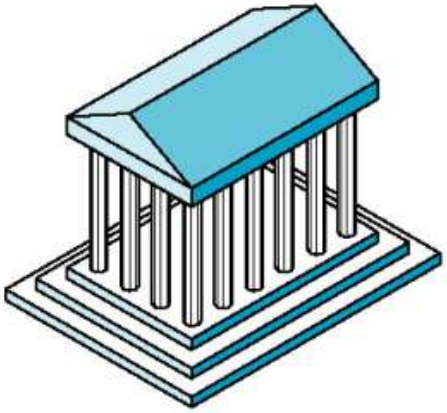
Front elevation



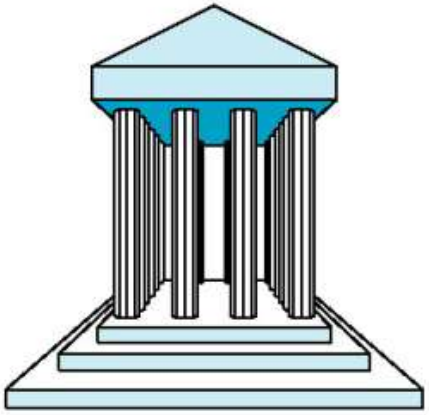
Elevation oblique



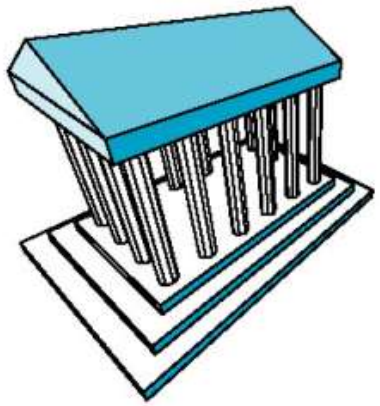
Plan oblique



Isometric

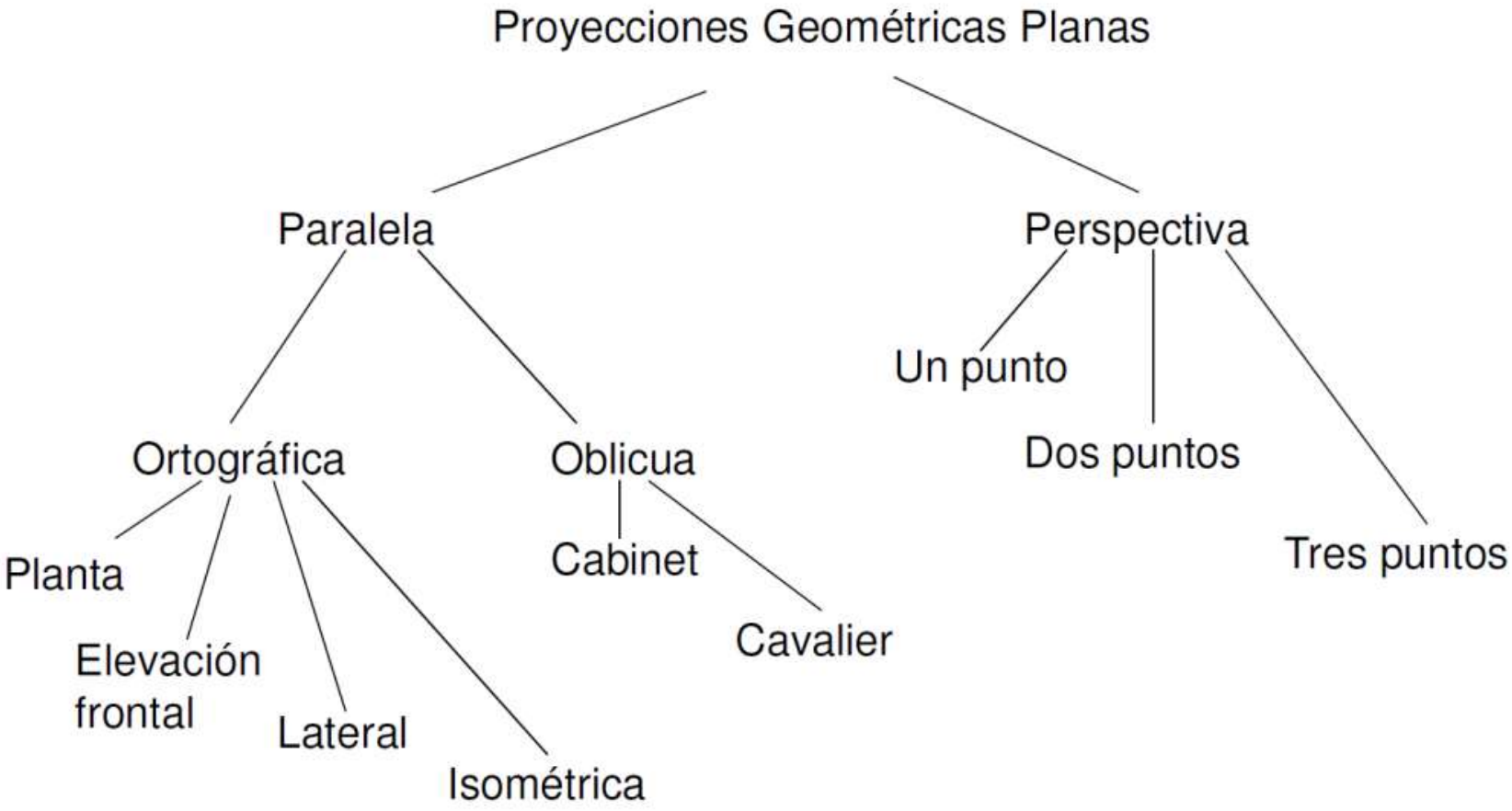


One-point perspective



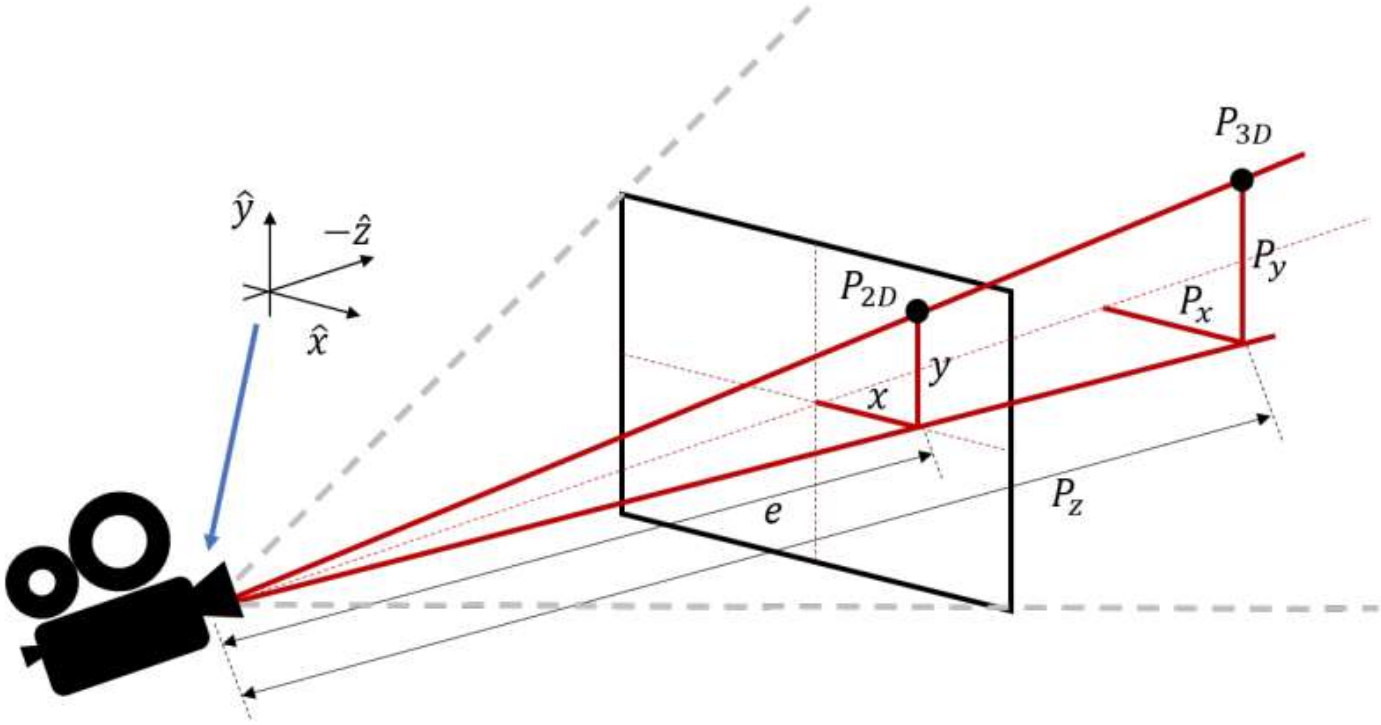
Three-point perspective

Tipos de proyecciones



Proyección sobre un plano

- El rayo que viaja desde el origen de la cámara hasta el punto P_{3D} en coordenadas del mundo intersecta el plano de proyección.
- Dicha intersección nos entrega el punto de interés proyectado.



Proyección sobre un plano

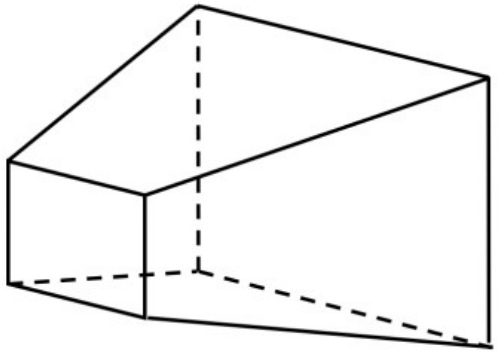
- Por semejanza de triángulos, tenemos

$$x = -\frac{e}{P_z} P_x \qquad\qquad y = -\frac{e}{P_z} P_y$$

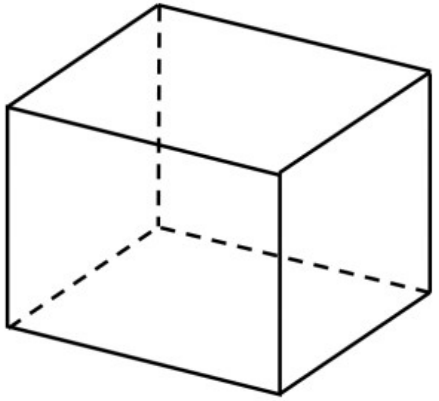
- Si embargo, perdemos la información de profundidad
- En general, necesitamos dicha información para saber qué objetos se encuentran detrás de otros.
- Usaremos coordenadas homogéneas para proyectar los vértices a un espacio 4D.

Volumen de vista

- Volumen de vista o Frustum es la región del espacio que es observada por la cámara.
- Descartaremos elementos fuera de esta región mediante clipping
- Dependiendo del tipo de proyección, tenemos distintas formas para el volumen de vista.



Perspectiva



Ortográfica

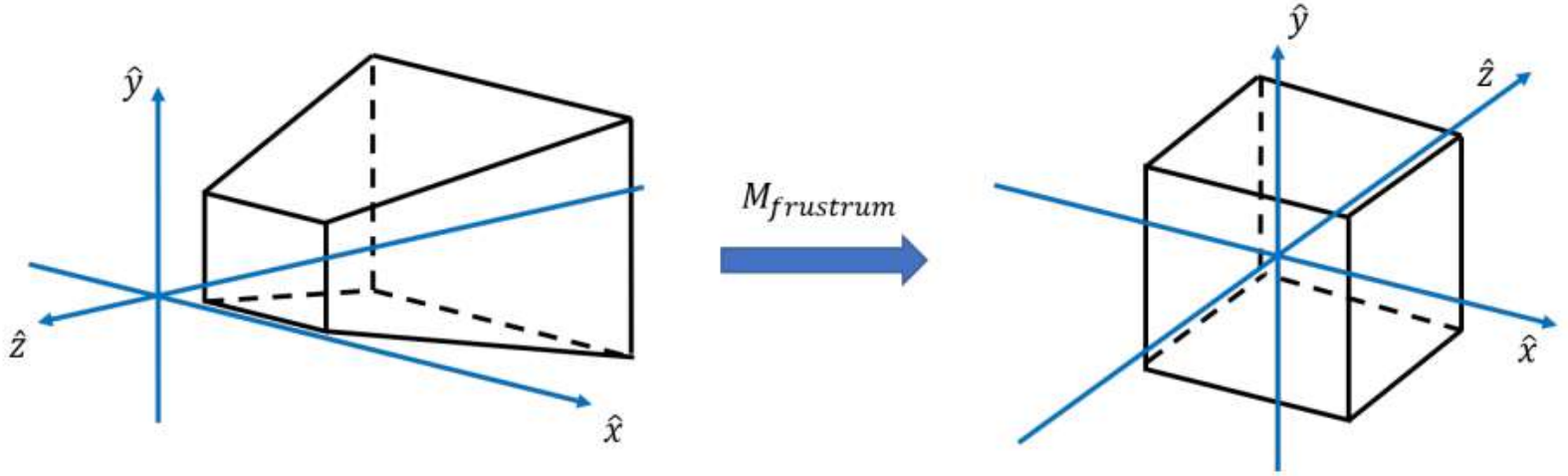
- Antes se proyectan los objetos a este volumen de vista
- La tarea de clipping se simplifica al rabajar sobre el volumen de vista conocido.

Preguntas?

Proyección Perspectiva

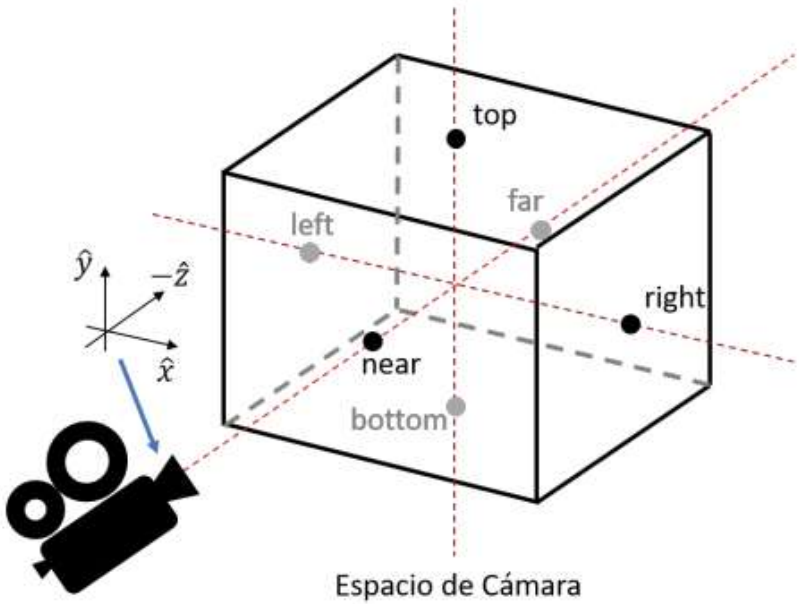
Proyección perspectiva

- Proyectaremos el volumen de vista a un cubo
- Las coordenadas resultantes pertenecen al espacio de clipping homogéneo
- En OpenGL, este espacio está centrado en el origen y se extiende desde -1 hasta +1 en cada eje.
- Por lo tanto, se normaliza a este rango numérico
- Esta transformación dejará un sistema de mano izquierda

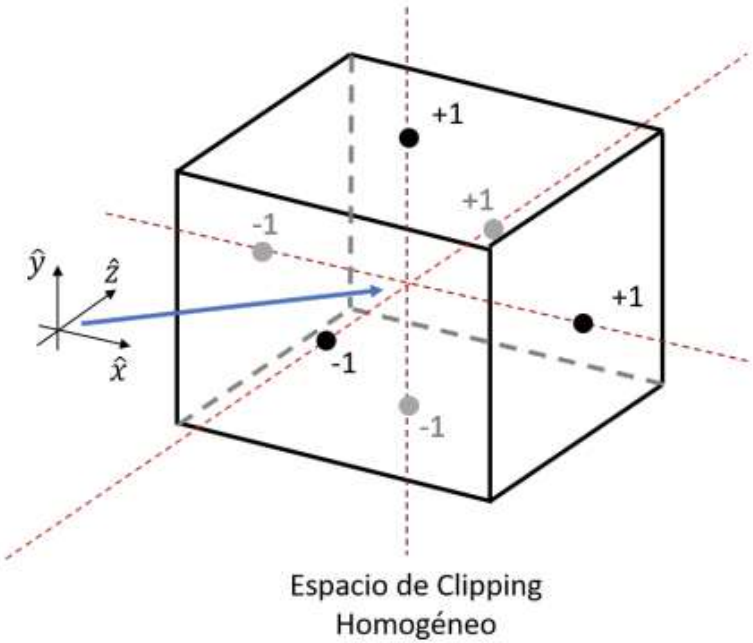


Espacio de clipping homogéneo

- Sea $P = (P_x, P_y, P_z, 1)$ un punto en coordenadas homogéneas en el espacio de cámara.
- Sean n y f las distancias en el eje Z para los planos $near$ y far.
- Sean l y r las distancias en el eje X para los planos $left$ y $right$.
- Sean t y b las distancias en el eje Y para los planos top y $bottom$.



Espacio de Cámara



Espacio de Clipping Homogéneo

Proyectando x e y

- Proyectando las coordenadas x e y con el plano $near$

$$x = -\frac{n}{P_z}P_x \qquad\qquad y = -\frac{n}{P_z}P_y$$

- Si P pertenece al volumen de vista, se cumple

$$l \leq x \leq r \qquad\qquad b \leq y \leq t$$

- Para normalizar el rango a $[-1,1]$, hacemos7

$$x' = (x\ - l)\frac{2}{(r-l)} - 1 \qquad\qquad y' = (y\ - b)\frac{2}{(t-b)} - 1$$

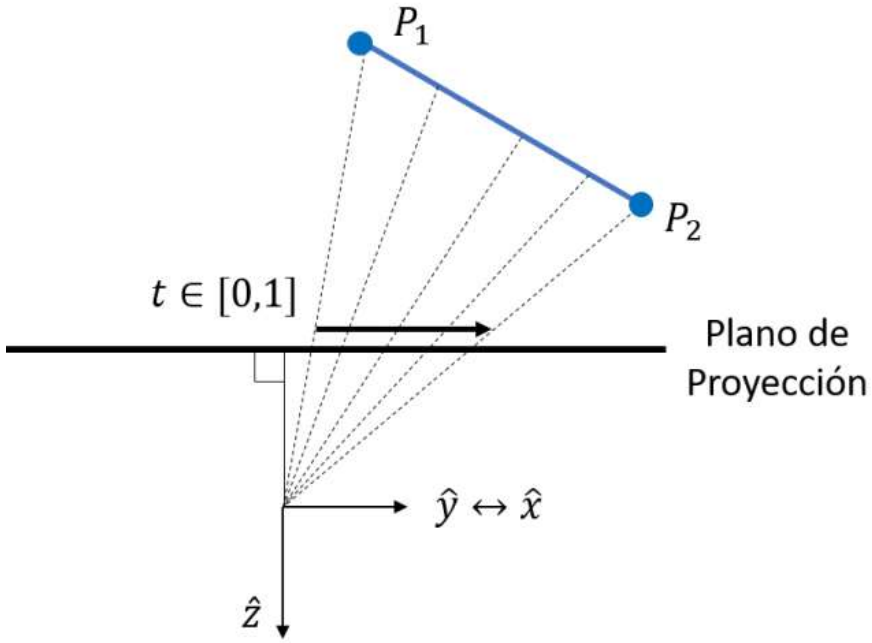
- Reemplazando x e y:

$$x' = \frac{2n}{r-l}\left(-\frac{P_x}{P_z}\right) - \frac{r+l}{r-l} \qquad\qquad y' = \frac{2n}{t-b}\left(-\frac{P_y}{P_z}\right) - \frac{t+b}{t-b}$$

Proyectando z

- Proyectar la componente z es más complejo
- z representa la profundidad, y su recíproco es interpolado linealmente en tiempo de rasterización.
- Luego, formulamos

$$z' = \frac{A}{z} + B$$



Proyectando z

- Si P pertenece al volumen de vista, entonces $-f \leq z \leq -n$
- Mapeando $-f \rightarrow 1$ y $-n \rightarrow -1$

$$-1 = \frac{A}{-n} + B \qquad\qquad 1 = \frac{A}{-f} + B$$

$$A = \frac{2nf}{f-n} \qquad\qquad B = \frac{f+n}{f-n}$$

- Luego, la expresión para z' queda

$$z' = -\frac{2nf}{f-n}\left(-\frac{1}{P_z}\right) + \frac{f+n}{f-n}$$

Coordenadas Homogéneas

- Las proyecciones para cada componente son:

$$x' = \frac{2n}{r-l}\left(-\frac{P_x}{P_z}\right) - \frac{r+l}{r-l}$$

$$y' = \frac{2n}{t-b}\left(-\frac{P_y}{P_z}\right) - \frac{t+b}{t-b}$$

$$z' = -\frac{2nf}{f-n}\left(-\frac{1}{P_z}\right) + \frac{f+n}{f-n}$$

- En coordenadas homogéneas

$$P' = (-x'P_z, -y'P_y, -z\,P_z, -P_z)$$

Coordenadas homogéneas

- Cada término de P' quedaría

$$-x'P_z = \frac{2n}{r-l}P_x + \frac{r+l}{r-l}P_z$$

$$-y'P_z = \frac{2n}{t-b}P_y + \frac{t+b}{t-b}P_z$$

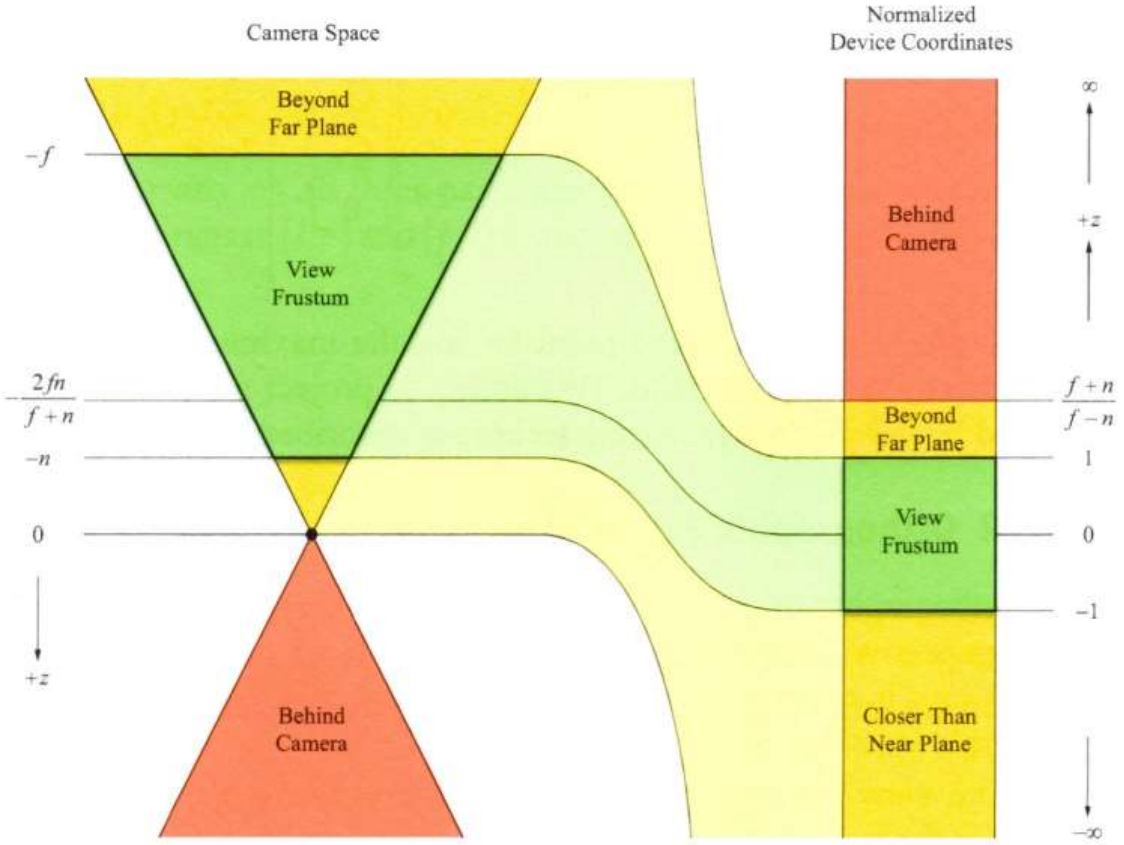
$$-z'P_z = \frac{2nf}{f-n} - \frac{f+n}{f-n}P_z$$

Matriz de transformación

- La transformación de proyección en perspectiva queda dada entonces por la siguiente matriz de transformación de 4×4

$$P' = M_{frustum} P = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & \frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Analizando la componente z

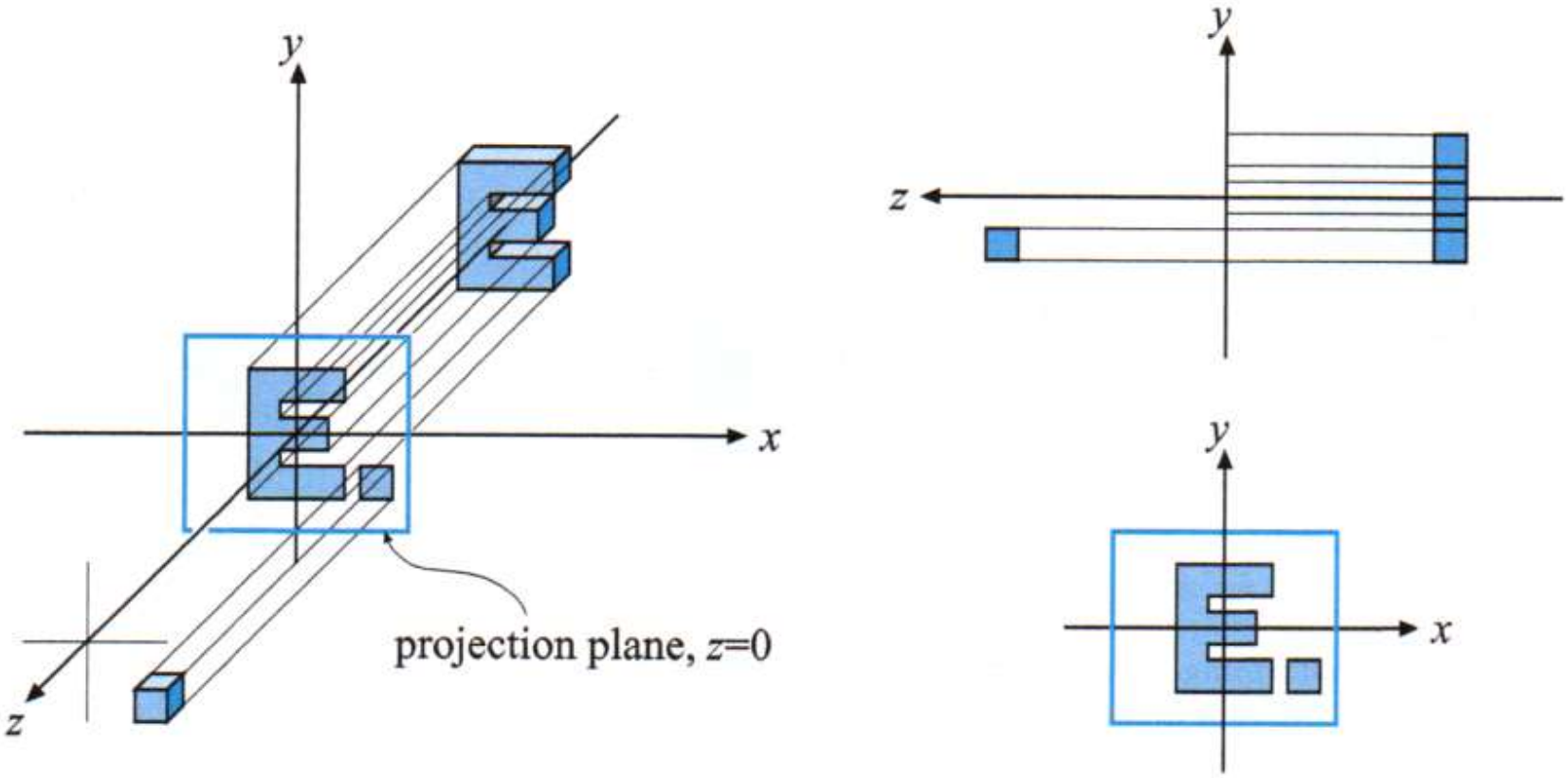


- Puntos dentro el volumen están en rango [-1, 1]
- El espacio infinito detrás del plano lejano queda restringido al intervalo $\left[1, \frac{f+n}{f-n}\right]$
- El espacio finito antes del plano cercano es expandido al rango infinito $[-1, -\infty]$

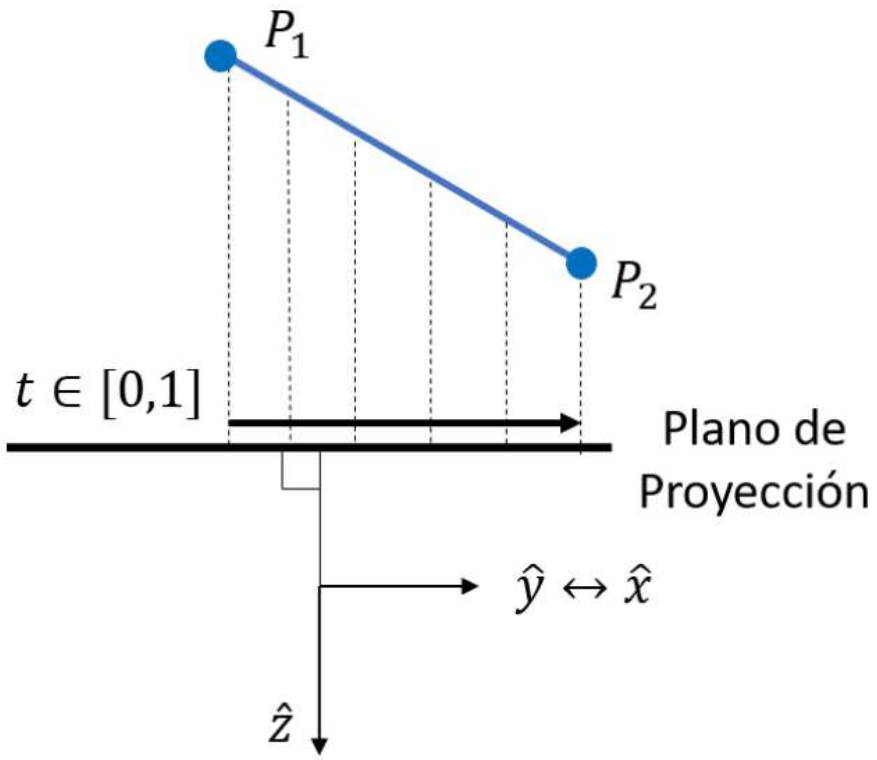
Proyección Ortográfica

Proyección ortográfica

- Líneas paralelas se mantienen paralelas después de la proyección



Proyección ortográfica



- No hay distorsión de perspectiva
- Los valores de profundidad pueden ser interpolados linealmente
- El mapeo de coordenadas homogéneas de clipping es lineal en los 3 ejes

$$x' = \frac{2}{r-l}x - \frac{r+l}{r-l}$$

$$x' = \frac{2}{t-b}y - \frac{t+b}{t-b}$$

$$x' = \frac{-2}{f-n}x - \frac{f+n}{f-n}$$

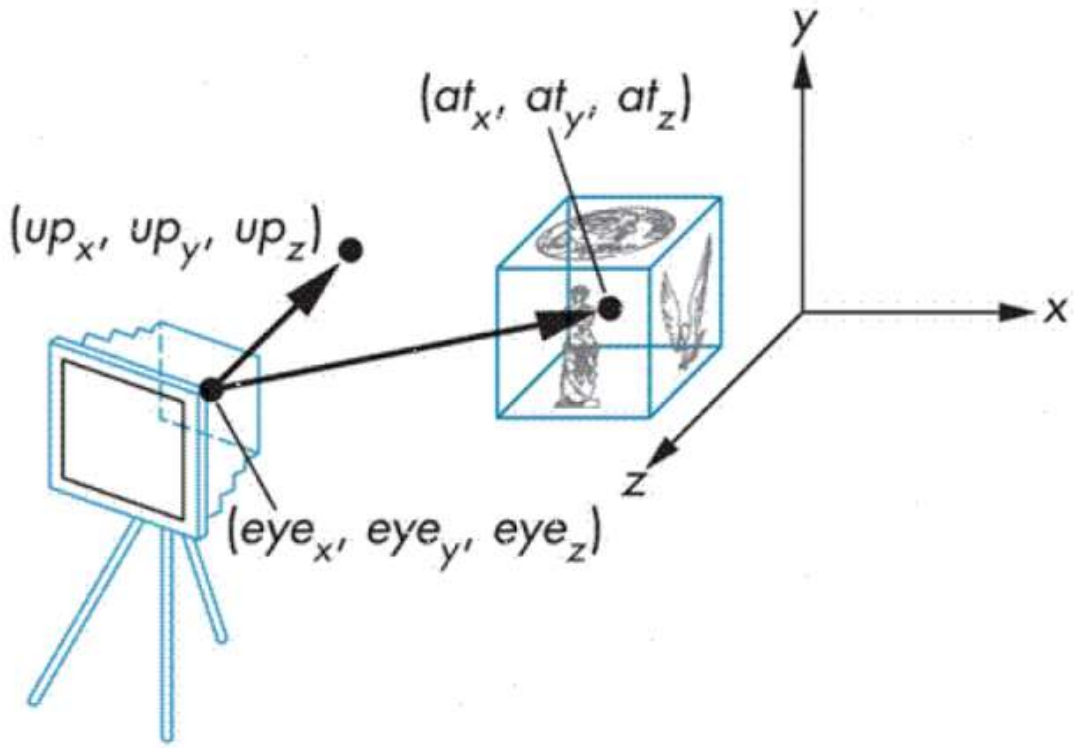
Matriz de proyección ortográfica

- Utilizando las ecuaciones anteriores, construimos el siguiente sistema matricial

$$P' = M_{Ortografica} P = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix}$$

Transformación de Vista

Transformación de vista



- Tenemos pendiente convertir cada punto en coordenadas del mundo a coordenadas de la cámara.

Configuración de cámara

- Disponemos de:
  - e: posición del ojo que observa (eye)
  - u: vector hacia arriba (up)
  - a: vector hacia donde miramos (at)

- Definamos el vector hacia adelante $f = \frac{a-e}{\|a-e\|}$ (forward vector)
- Un vector s hacia el lado puede ser obtenido como $s = f \times u$ (sideways vector)
- Note que s y u no son necesariamente ortogonales.
- Nuevo vector hacia arriba u' es obtenido como $u' = s \times f$, asegurando ortogonalidad.
- Con estos tres vectores ortogonales, creamos una matriz de rotación.

Configuración de cámara

$$R_{\text{cámara a mundo}} = \begin{bmatrix} s_x & u'_x & f_x & 0 \\ s_y & u'_y & f_y & 0 \\ s_z & u'_z & f_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Su inversa nos convierte coordenadas del mundo a coordenadas de cámara

$$R_{\text{mundo a cámara}} = R^{-1}_{\text{cámara a mundo}} = R^{T}_{\text{cámara a mundo}}$$

$$R_{\text{mundo a cámara}} = \begin{bmatrix} s_x & s_y & s_z & 0 \\ u'_x & u'_y & u'_z & 0 \\ f_x & f_y & f_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Configuración de la cámara

- Si la cámara se encuentra en la posición e, hay que hacer una traslación:

$$T_{\text{mundo a cámara}} = \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- La transformación de vista total es:

$$\text{Vista'} = R_{\text{mundo a cámara}} \cdot T_{\text{mundo a cámara}}$$

$$\text{Vista'} = \begin{bmatrix} s_x & s_y & s_z & 0 \\ u'_x & u'_y & u'_z & 0 \\ f_x & f_y & f_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_x \\ 0 & 1 & 0 & -e_y \\ 0 & 0 & 1 & -e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & s_y & s_z & -e \cdot s \\ u'_x & u'_y & u'_z & -e \cdot u' \\ f_x & f_y & f_z & -e \cdot f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Configuración de la cámara

- Como las matrices de proyección son para un sistema de mano izquierda, es necesario invertir la coordenada z de la transformación de vista:

$$\mathrm{V} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_x & s_y & s_z & -e \cdot s \\ u'_x & u'_y & u'_z & -e \cdot u' \\ f_x & f_y & f_z & -e \cdot f \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & s_y & s_z & -e \cdot s \\ u'_x & u'_y & u'_z & -e \cdot u' \\ -f_x & -f_y & -f_z & e \cdot f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Implementación en OpenGL

- Una sola matriz de transformación enviada al shader.

GLSL

```
#version 130
uniform mat4 transform;
in vec3 position;

void main()
{
    gl_Position = transform * vec4(position, 1.0f);
}
```

Python

```
transform = tr.matmul([projection, view, model_traslation, model_rotation])
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "transform"), 1, GL_TRUE, transform)
```

Implementación en OpenGL

- Cada transformación por separado

GLSL

```
#version 130
uniform mat4 projection;
uniform mat4 view;
uniform mat4 model;
in vec3 position;

void main()
{
    gl_Position = projection * view * model * vec4(position, 1.0f);
}
```

Python

```
model = numpy.matmul(model_traslation, model_rotation)
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "projection"), 1, GL_TRUE, projection)
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "view"), 1, GL_TRUE, view)
glUniformMatrix4fv(glGetUniformLocation(shaderProgram, "model"), 1, GL_TRUE, model)
```