



DeepL

Suscríbete a DeepL Pro para poder traducir archivos de mayor tamaño.

Más información disponible en [www.DeepL.com/pro](http://www.DeepL.com/pro).

VC



# Informática visual

## Objetos gráficos y su programación

Universidad de Darmstadt

Prof. Dr. Elke Hergenröther

Björn Frömmel

Prof. Dr. Benjamin Meyer

---

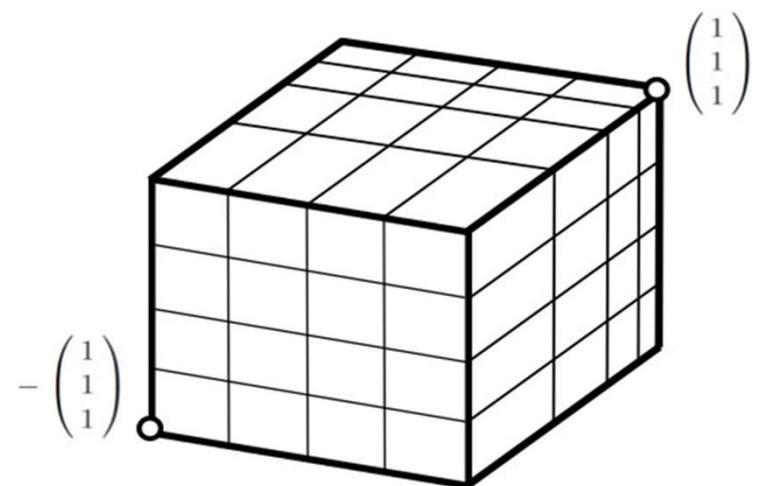
# CAPÍTULO 7

## Cámaras

## 7. cámaras

# El espacio normalizado de dispositivos (NDC)

- Hasta ahora, intuitivamente siempre hemos definido y visualizado nuestra geometría en el intervalo de valores  $[-1, 1]^3$ .
  - Esta zona (normalizada) también se denomina **Espacio de dispositivos normalizado**.
  - Sólo se renderizan los vértices dentro de este cubo.
- Pero la cuestión de qué geometría se renderiza debería depender de la posición del espectador/cámara.



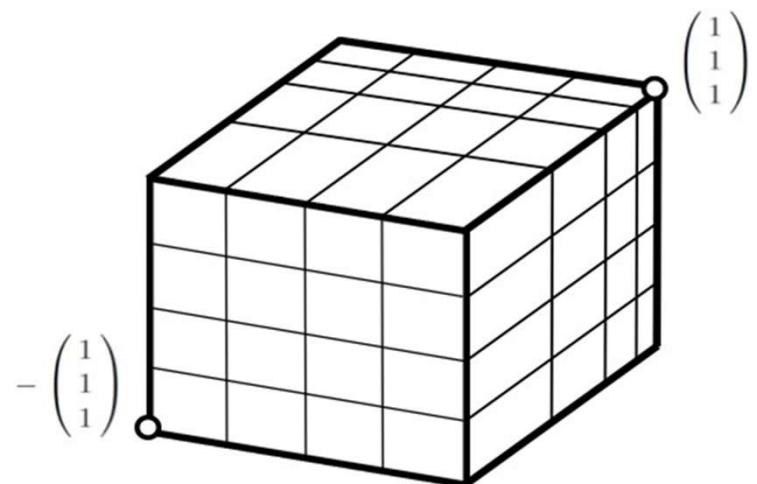
**Espacio de dispositivos  
normalizado: cubo de  $(-1, -1, -1)$  a  
 $(1, 1, 1)$**

## 7. cámaras

# Orientación de la cámara

### Pregunta:

- ¿Cómo realizar una cámara situada en cualquier posición y con cualquier orientación, utilizando únicamente  
¿Utilizar transformaciones?

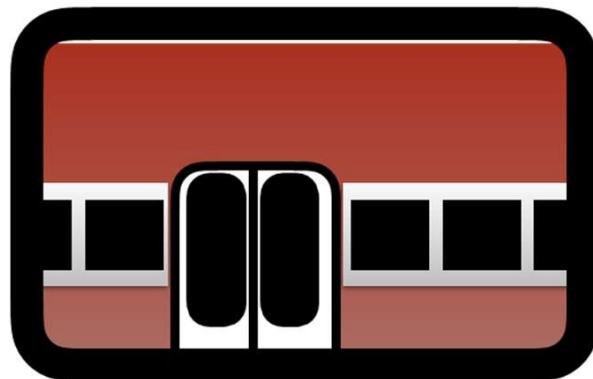


**Espacio de dispositivos  
normalizado: cubo de  $(-1, -1, -1)$  a  
 $(1, 1, 1)$**

## 7. cámaras

# Problema de

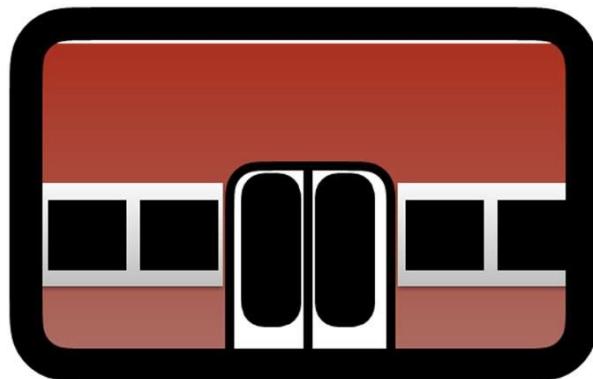
**Situación:** está sentado en un tren y mira por la ventanilla.



## 7. cámaras

# Problema de

**Situación:** está sentado en un tren y mira por la ventanilla.



- ¿Está en marcha tu tren o el otro?
- Imposible decidir con una visión limitada de la escena

## 7. cámaras

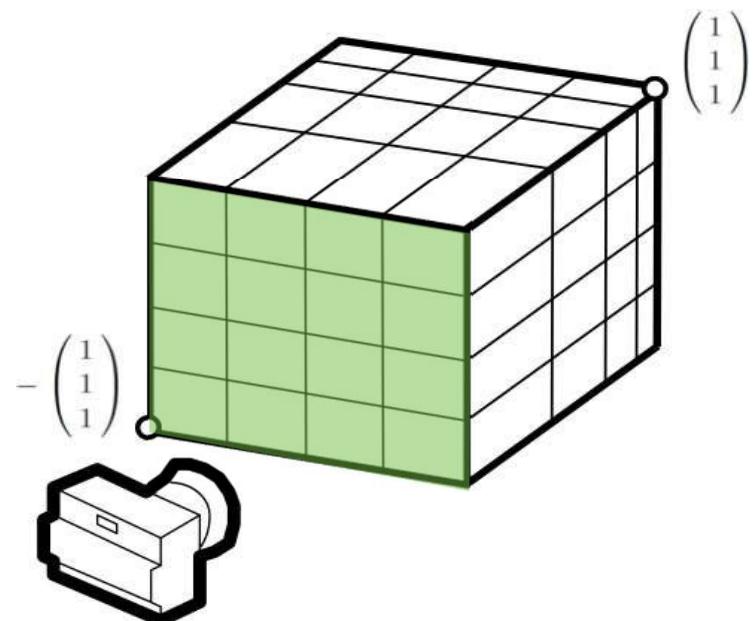
# Orientación de la cámara

### Pregunta:

- ¿Cómo realizar una cámara situada en cualquier posición y con cualquier orientación, utilizando únicamente  
¿Utilizar transformaciones?

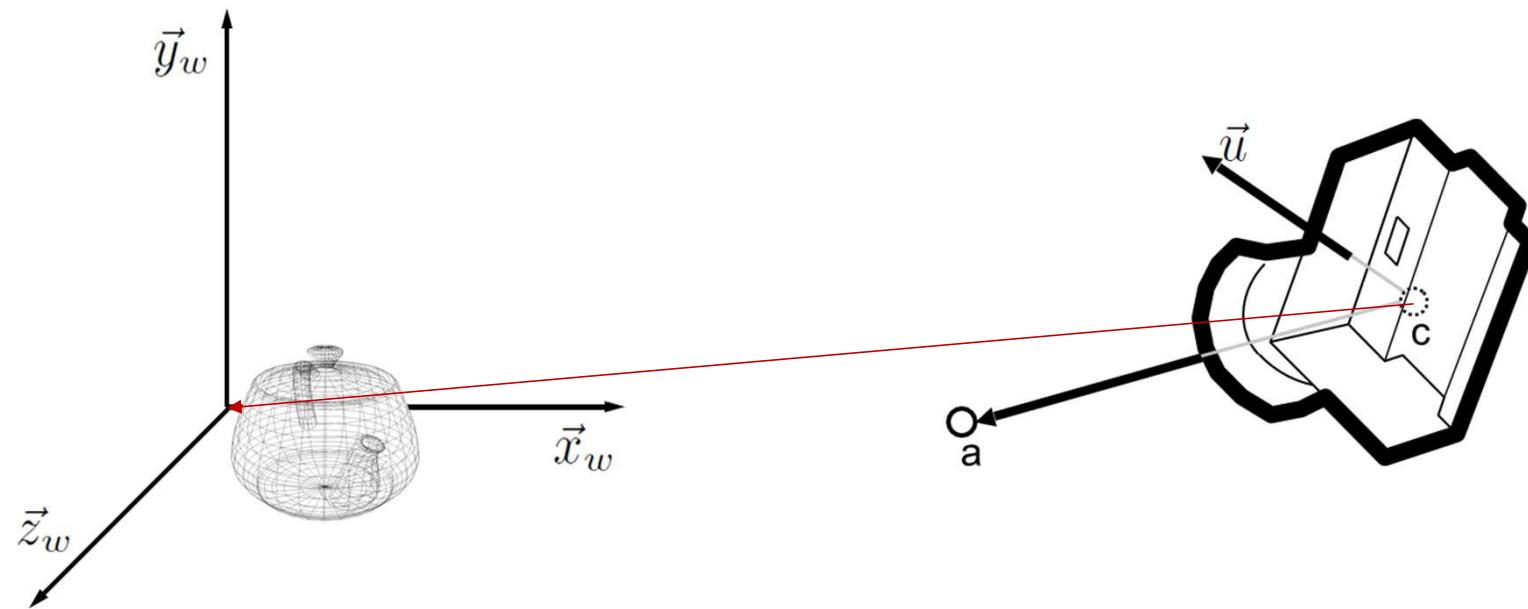
### Respuesta:

- En lugar de mover/rotar la cámara,  
¡mueve/rota la escena!
  - Mismo resultado, más eficaz en la aplicación.
  - Además, ¡ya sabemos cómo hacerlo!
- Objetivo: La cámara está alrededor del origen y mira a lo largo del eje Z negativo



## 7. cámaras

# Parámetros externos (extrínsecos) de la cámara

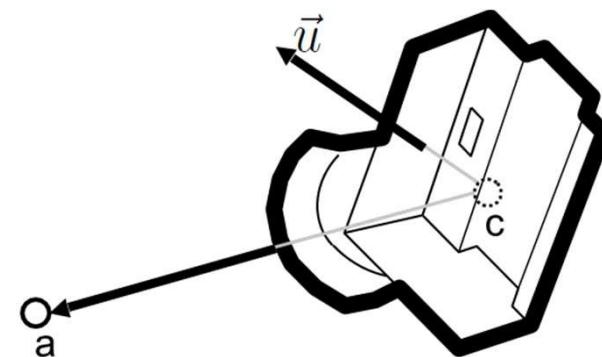


- Sistema mundial de coordenadas
- Colocación de objetos
- Centro de cámara  $\vec{a}$
- Punto de vista  $\vec{a}$
- vector ascendente  $\vec{u}$

## 7. cámaras

# Ver Transformación

- La transformación de la escena puede expresarse como una multiplicación de matrices.
  - Recordatorio: Sistema de coordenadas a la derecha
  - Por defecto, la cámara mira desde cero a lo largo del eje Z negativo.
- Esta denominada **matriz de vistas** puede calcularse automáticamente:



```
lookAt( cx, cy, cz, // Centro de cámara c  
        ax, ay, az, // punto de vista  
        a ux, uy, uz ); // vector ascendente u
```

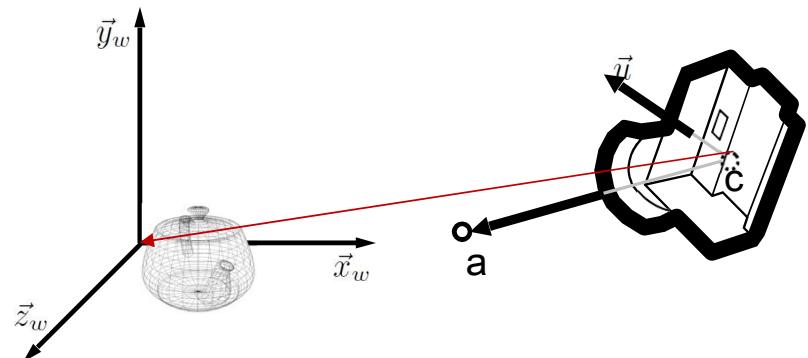
- El sistema de coordenadas resultante se denomina **Espacio de Vista**.

## 7. cámaras

# Ver la matriz de

### Punto de partida:

- La cámara y la geometría están ambas en el sistema de coordenadas mundo
- Sin embargo, la cámara no está en el origen (el vector rojo apunta en la dirección del origen).



**Objetivo:** la cámara se sitúa en el origen del sistema de coordenadas mundo y mira a lo largo del eje Z negativo.

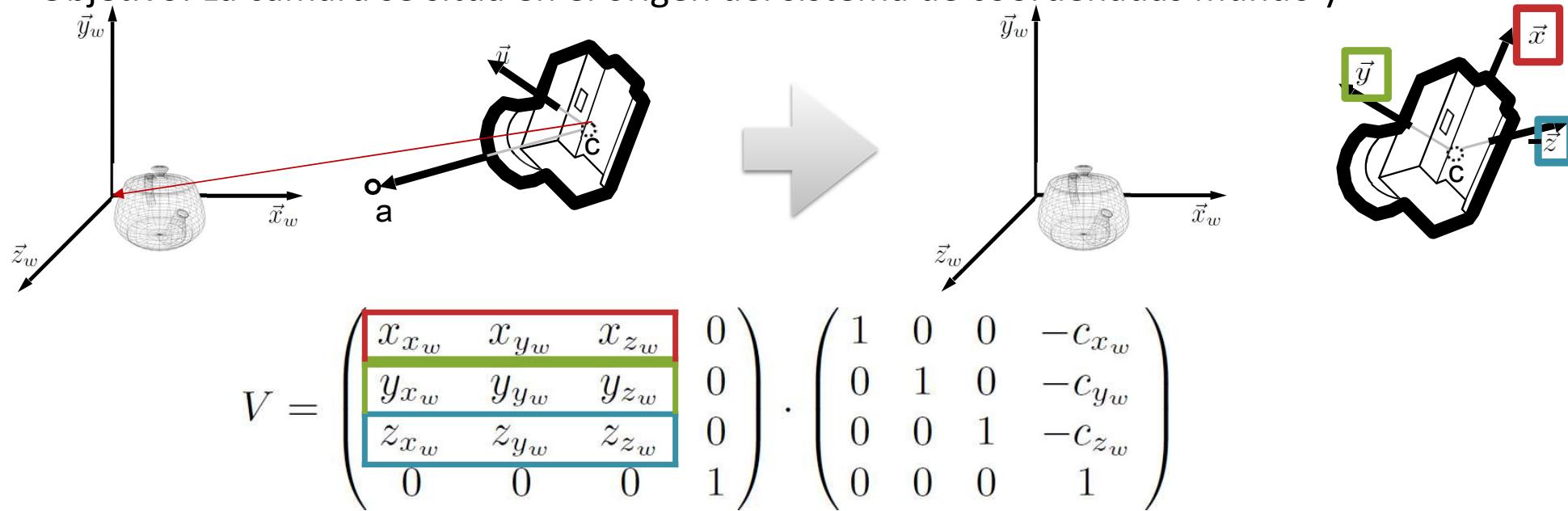
### Implantación:

- Para ello, hay que transformar la cámara.
- Las transformaciones de la cámara también se aplican a las geometrías en el sistema de coordenadas mundo.  
aplicada para que la disposición de las geometrías siga siendo relativa a la cámara.
- Esta transformación se denomina transformación de vista.

## 7. cámaras

### Ver la matriz de

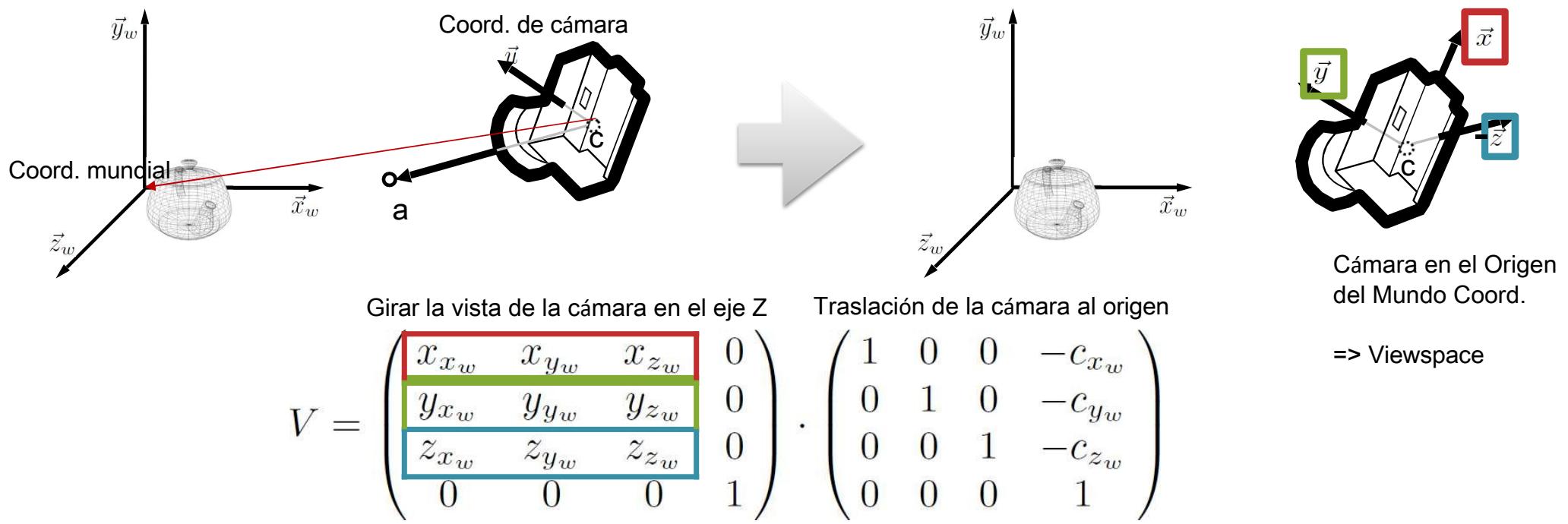
- La cámara y la geometría están ambas en el sistema de coordenadas mundo
- Sin embargo, la cámara no está situada en el origen (el vector rojo apunta en la dirección del Origen).
- Objetivo: La cámara se sitúa en el origen del sistema de coordenadas mundo y



## 7. cámaras

### Ver la matriz de

- El primer paso es mover la cámara al origen (Traslación -C)
- El segundo paso es la cámara para girar un vector de tal manera que se mueve a lo largo de la eje Z negativo.



## Transformaciones de vértices hasta ahora

**Transformación del modelo**  $M_{Model}$ : Transformaciones acumuladas, coloca objetos en el Escena (Mundo)

**Transformación de vista V**: Rota y traslada toda la escena para que la cámara esté en el origen y mirando a lo largo del eje Z negativo.

$$\mathbf{p}' = V \cdot \underbrace{T \cdot \dots \cdot T \cdot S \cdot R}_{M_{Model}} \cdot \mathbf{p}$$

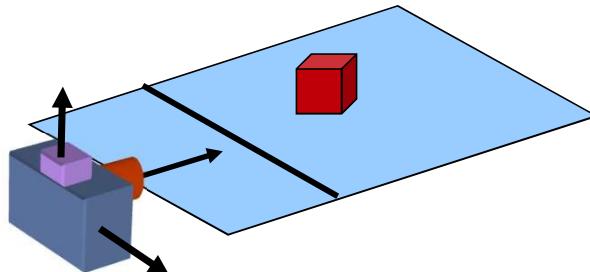


Orden lógico de ejecución

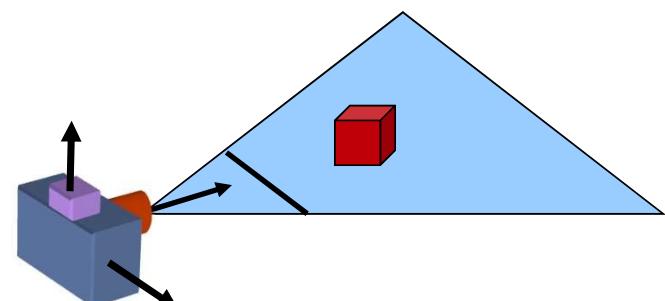
## 7. cámaras

# Parámetros intrínsecos de la cámara

- Los parámetros extrínsecos de la cámara (posición y orientación) ya están integrados en la matriz de vista, pero faltan los parámetros intrínsecos:
  - Simplificado: qué zona de la escena debe renderizarse
  - En una cámara real, esto correspondería al ángulo de apertura y a la distancia focal
- Opciones de proyección:



Proyección paralela u ortográfica

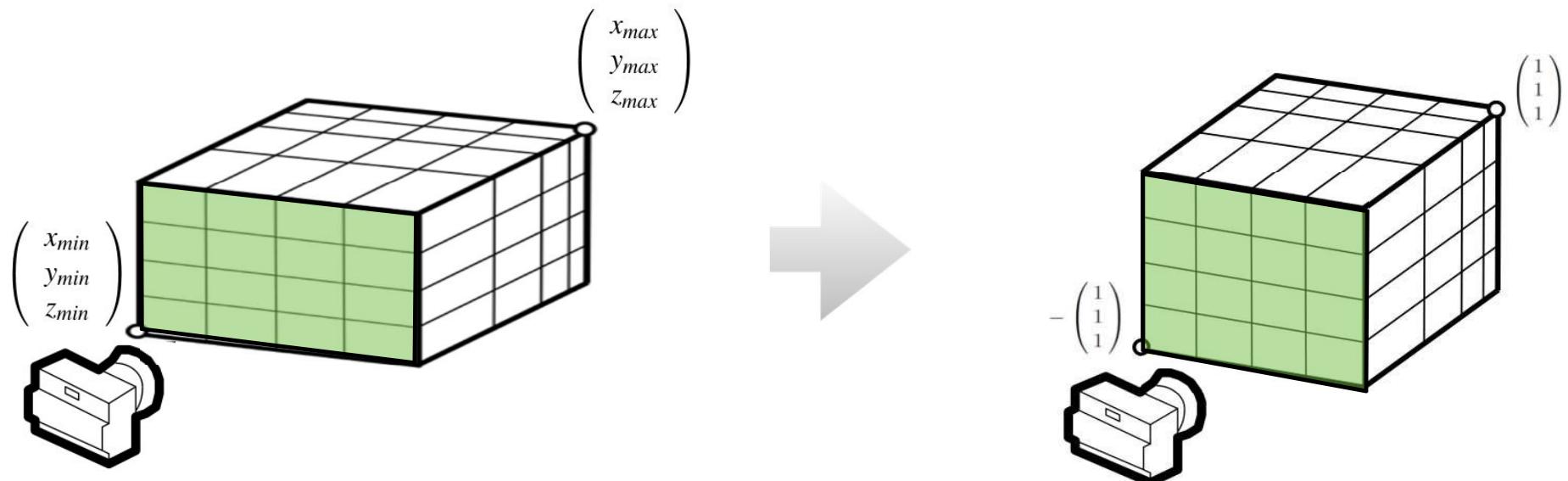


Proyección en perspectiva

## 7. cámaras

# Proyección ortográfica

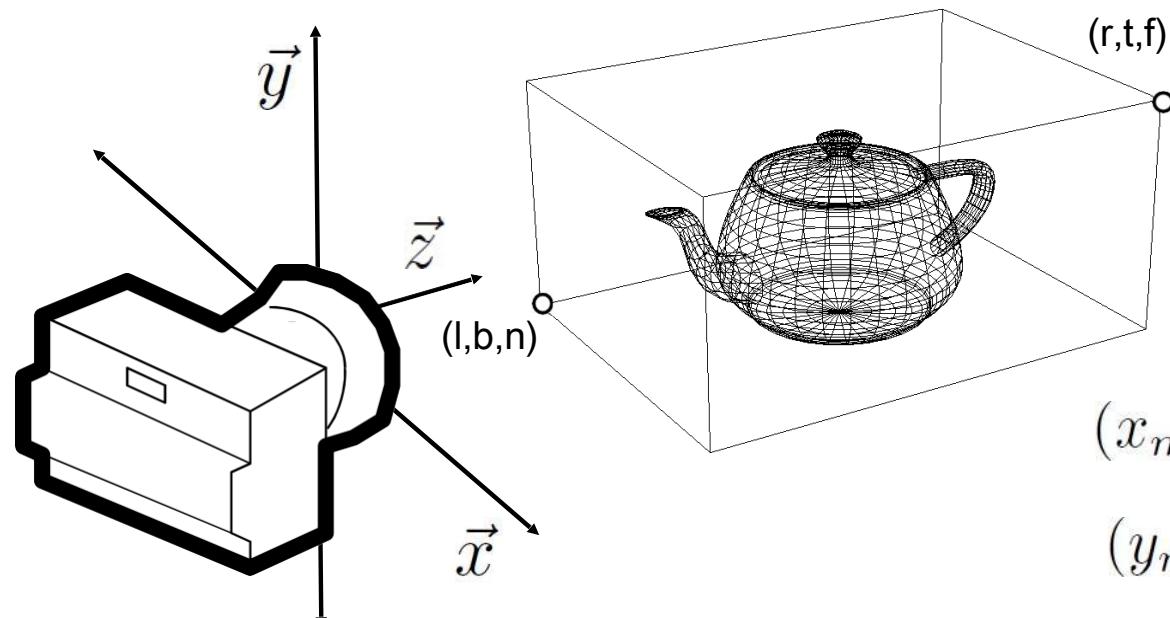
- Se pueden simular **diferentes relaciones de aspecto** transformando toda la escena



## 7. cámaras

# Proyección ortográfica II

- Definición del volumen a renderizar y transformación en un volumen canónico (Normalized Device Space)



$$(x_{min}, x_{max}) = (left, right)$$

$$(y_{min}, y_{max}) = (bottom, top)$$

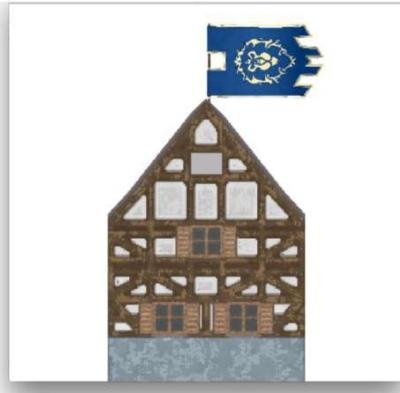
$$(z_{min}, z_{max}) = (near, far)$$

## 7. cámaras

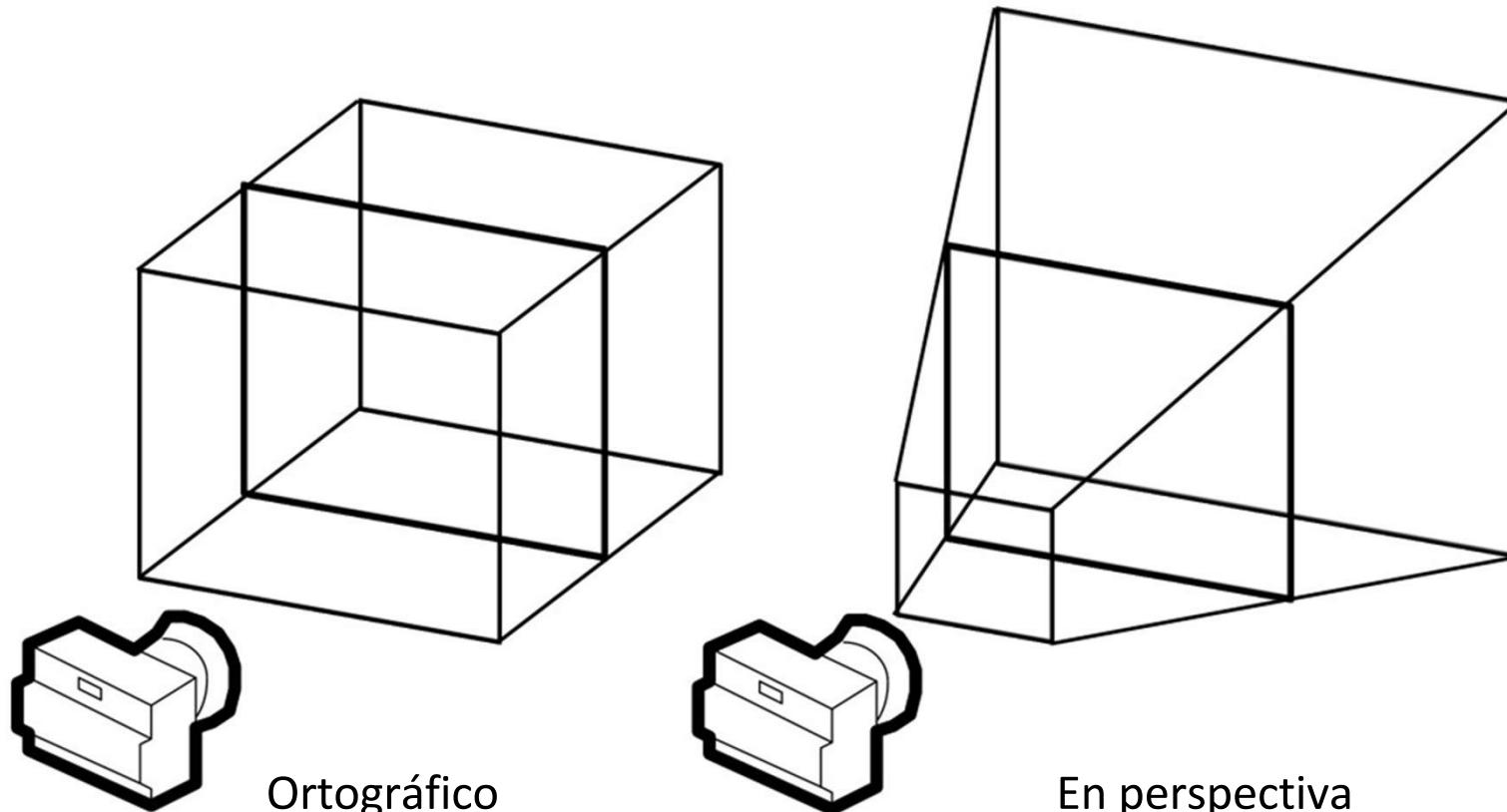
# Proyección ortográfica III

- El establecimiento del espacio de volumen y el cálculo de la matriz de proyección se realiza mediante:  

```
glOrtho( GLdouble izquierda, GLdouble derecha,
          GLdouble abajo, GLdouble arriba,
          GLdouble cerca, GLdouble lejos);
```
- No olvides ajustar también la resolución de tu ventana
  - La relación de aspecto de la cámara y la ventana debe ser la misma para evitar distorsiones



## Diferentes proyecciones



## 7. cámaras

# Percepción de la perspectiva

- Percepción en perspectiva significa:
- Cuanto más atrás esté el objeto, más pequeño nos parecerá.
- Si se incumple esta norma, recibimos, por ejemplo, información incorrecta sobre la talla.



Fuente: Wikipedia - Ilusiones ópticas

## 7. cámaras

# Percepción en perspectiva II

Proyección en perspectiva

- "Representación natural"
- Tipo de proyección más utilizado
- Los rayos de la visión se encuentran en un **Punto de fuga**



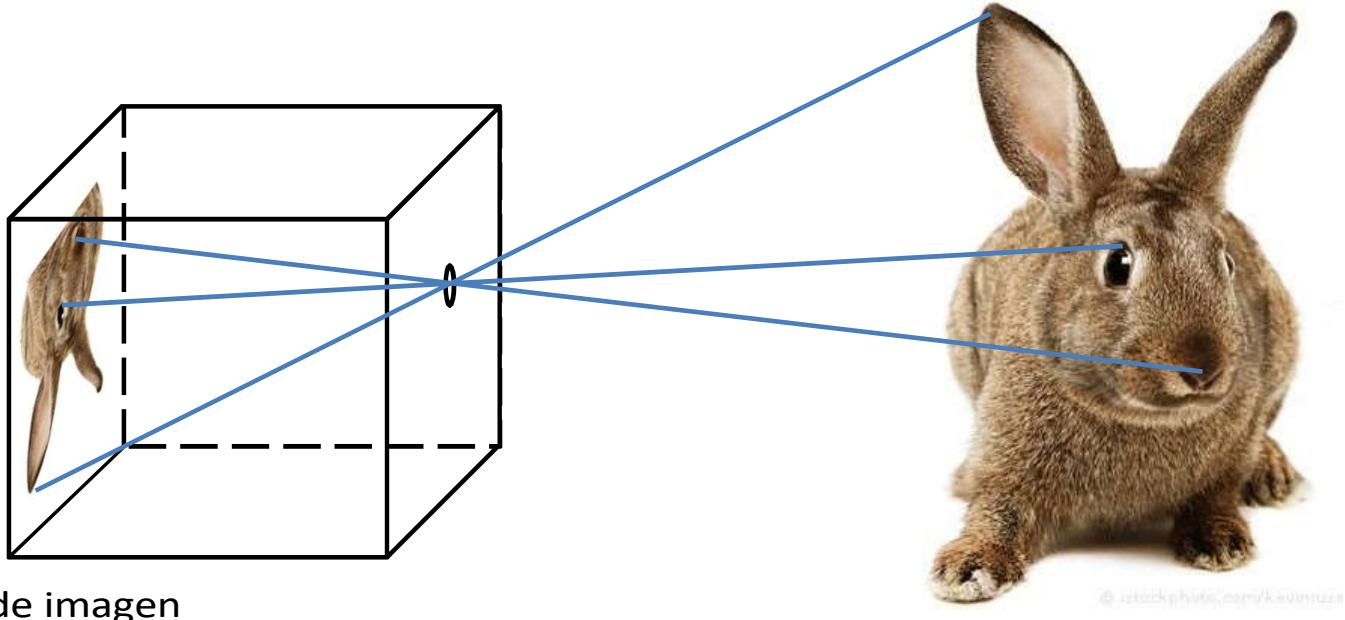
---

Continúe aquí el 22.05.2023 ....

## 7. cámaras

# Proyección en perspectiva

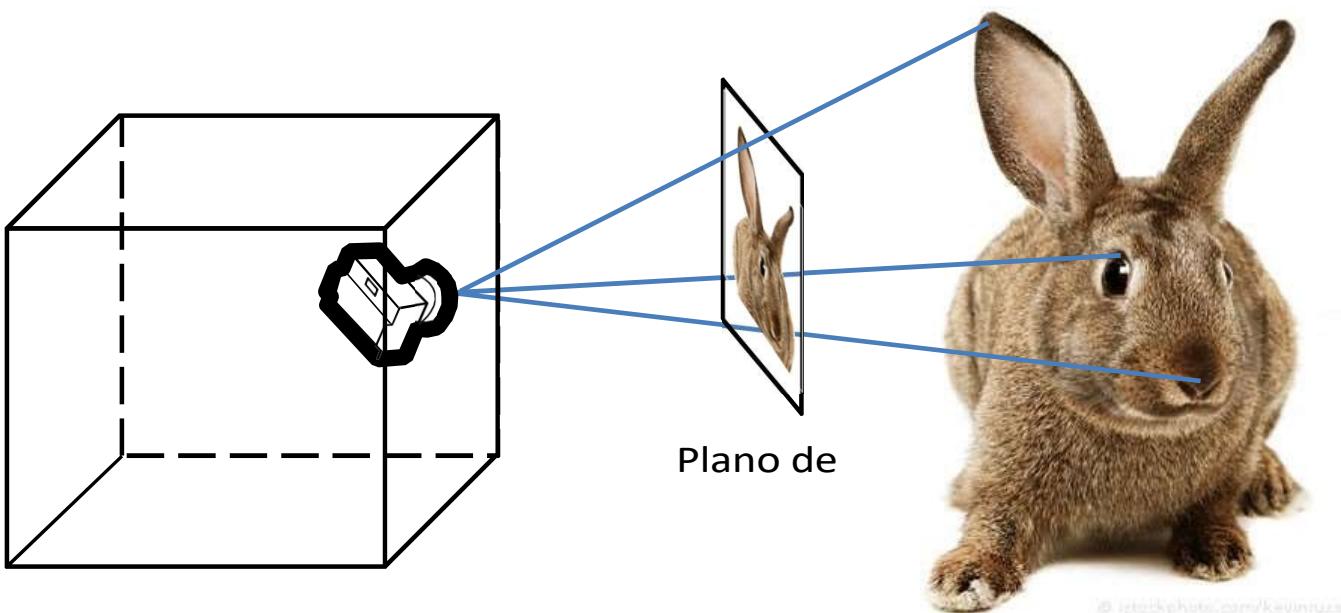
- Cámara estenopeica:
  - Caja con agujero
  - Imagen perfecta para un orificio "del tamaño de un punto"



## 7. cámaras

# Proyección en perspectiva II

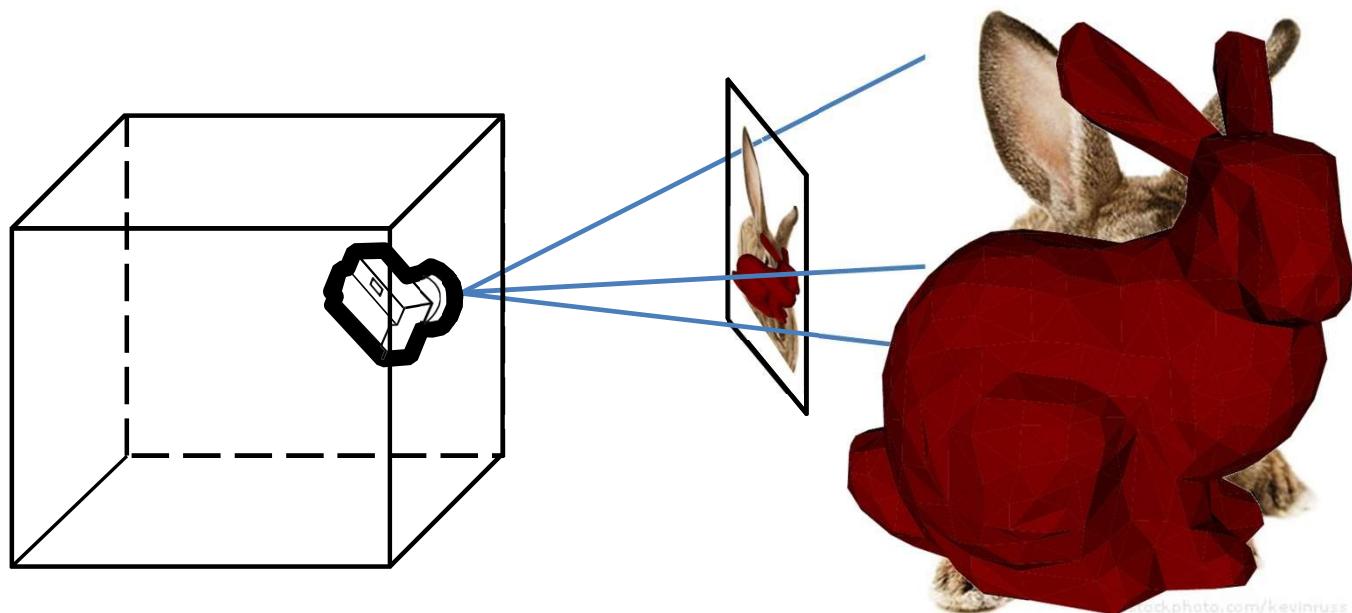
- Cámara estenopeica:
  - Caja con agujero
  - Imagen perfecta para un agujero "del tamaño de un punto"



## 7. cámaras

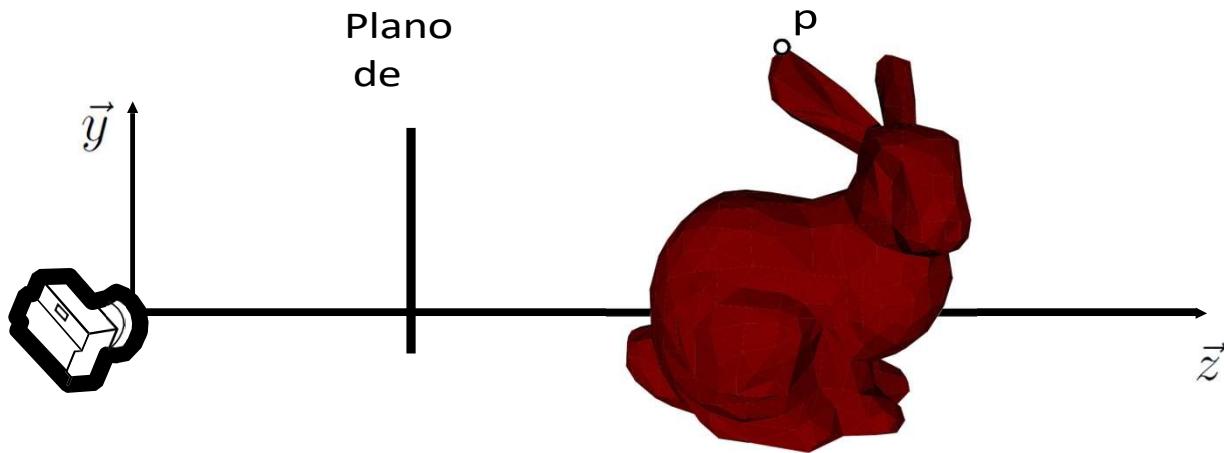
# Proyección en perspectiva III

- Cámara estenopeica:
  - Caja con agujero
  - Imagen perfecta para un agujero "del tamaño de un punto"



## Proyección en perspectiva IV

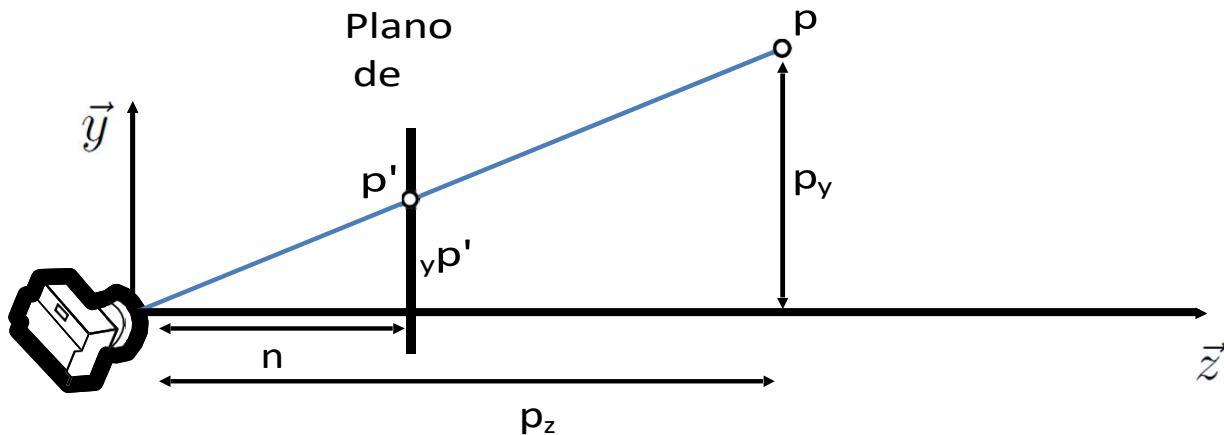
- Proyección sobre el plano de la imagen:
  - (Aquí ejemplo en 2D)



## 7. cámaras

# Proyección en perspectiva V

- Proyección sobre el plano de la imagen:
  - (Aquí ejemplo en 2D)



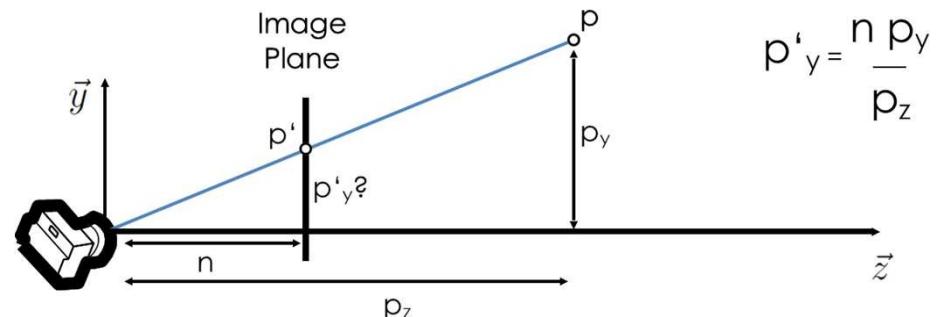
- Juego de vigas:

$$\frac{y}{n} = \frac{p'}{p_z} \quad \Rightarrow \quad p'_y = \frac{n p_y}{p_z}$$

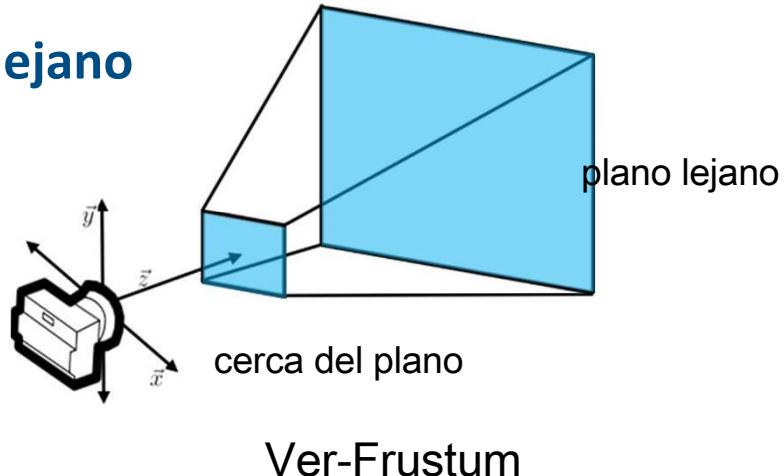
## 7. cámaras

# Proyección en perspectiva VI

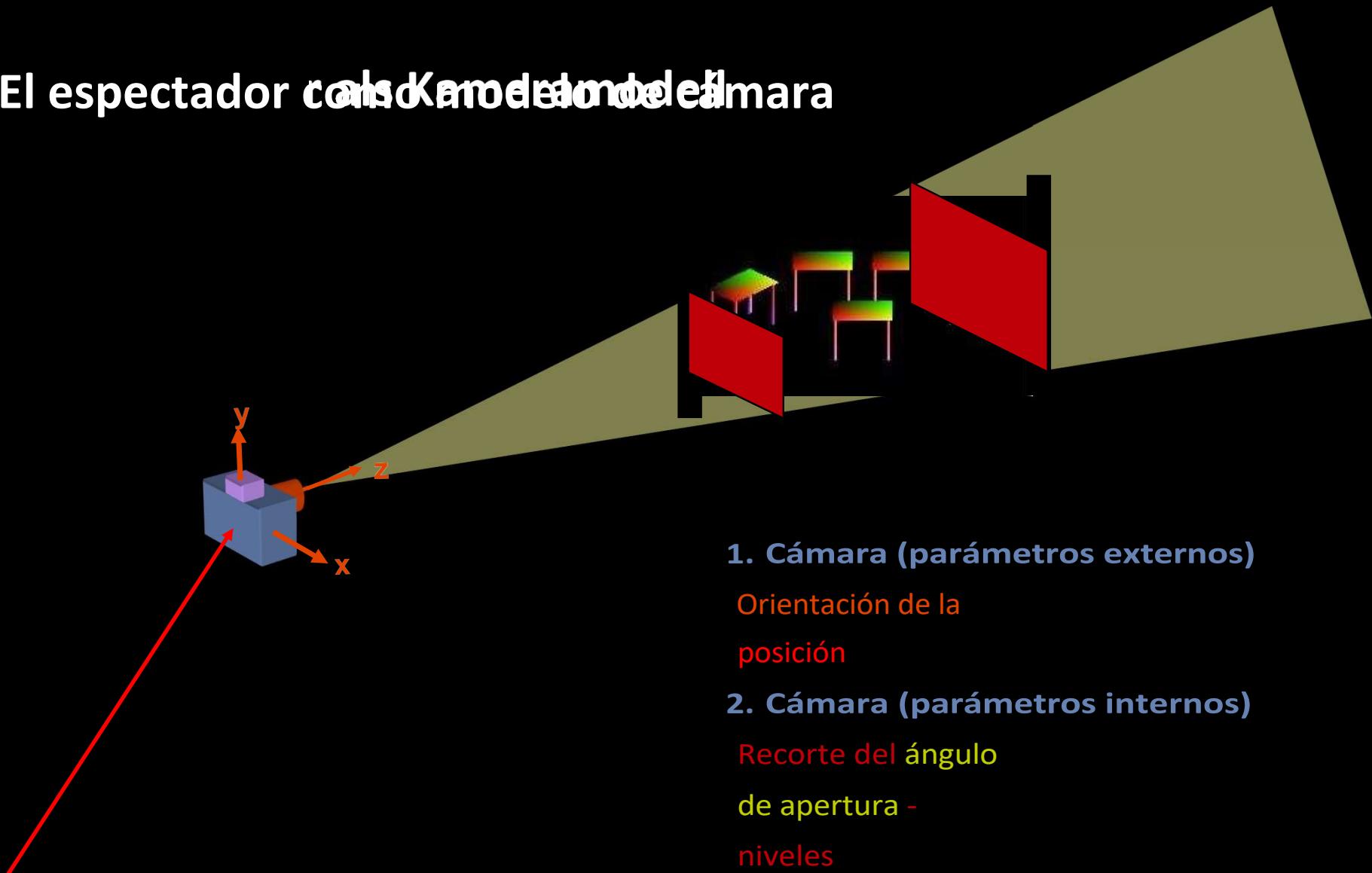
- ¿Cómo puede realizarse esta proyección como una multiplicación de matrices?



- Solución: **niveles de recorte cercano y lejano**

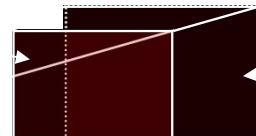
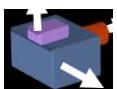
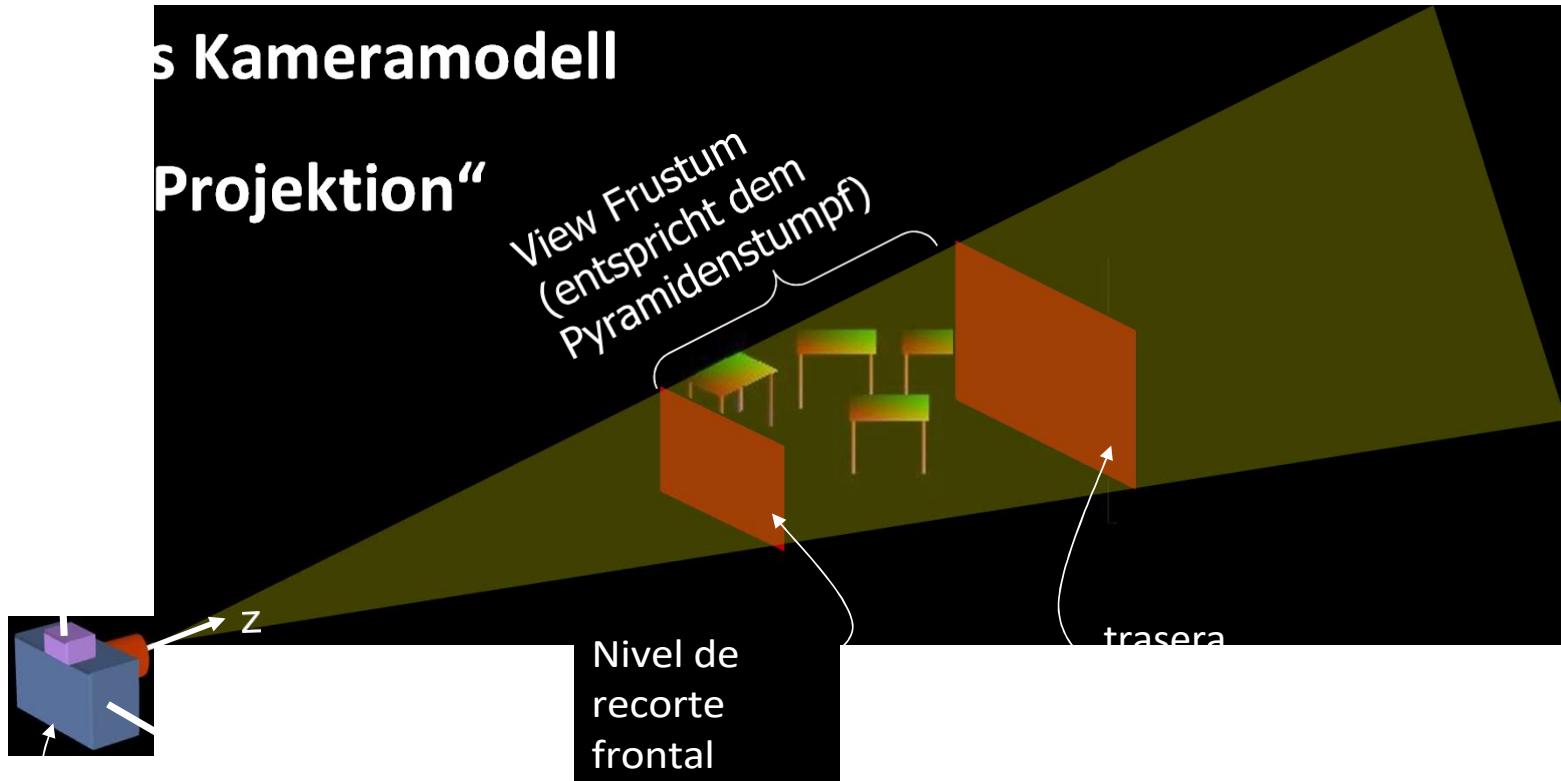


# El espectador ~~consigue~~ ~~que~~ ~~la~~ ~~cámara~~ ~~vea~~ ~~lo~~ ~~que~~ ~~desea~~

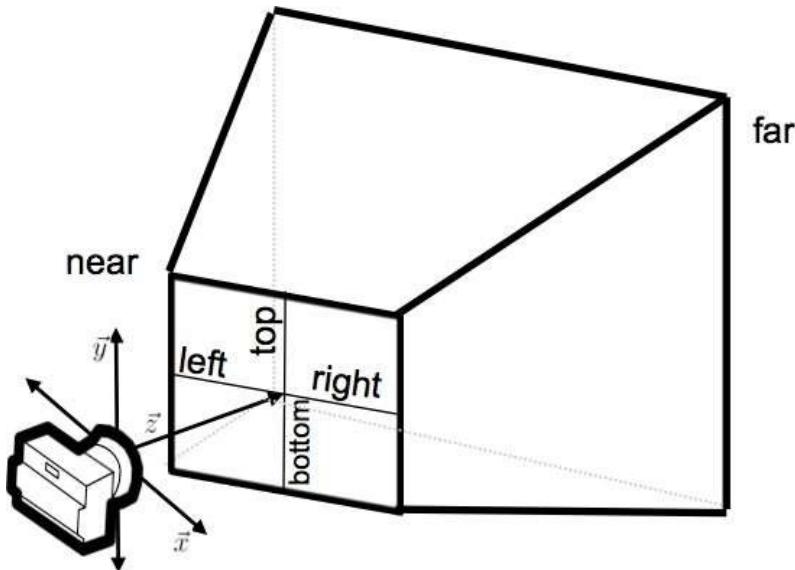


# Das Kameramodell

## Projektion“

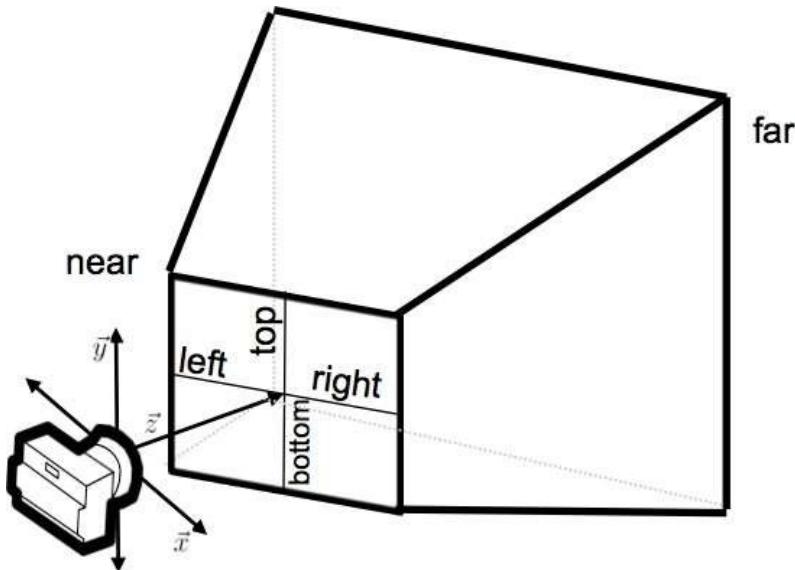


## Definición de Ver Frustum



- Cerca, lejos en relación con la alternativa de la cámara
- Arriba/abajo/izquierda/derecha es la distancia al centro en el plano cercano
- `glm::frustum(float izquierda, float derecha, float arriba, float abajo, float near, float far);`

## Definición de Ver Frustum



Alternativa:

**theta:** Ángulo de apertura vertical  
**aspect:** Relación de aspecto altura : anchura

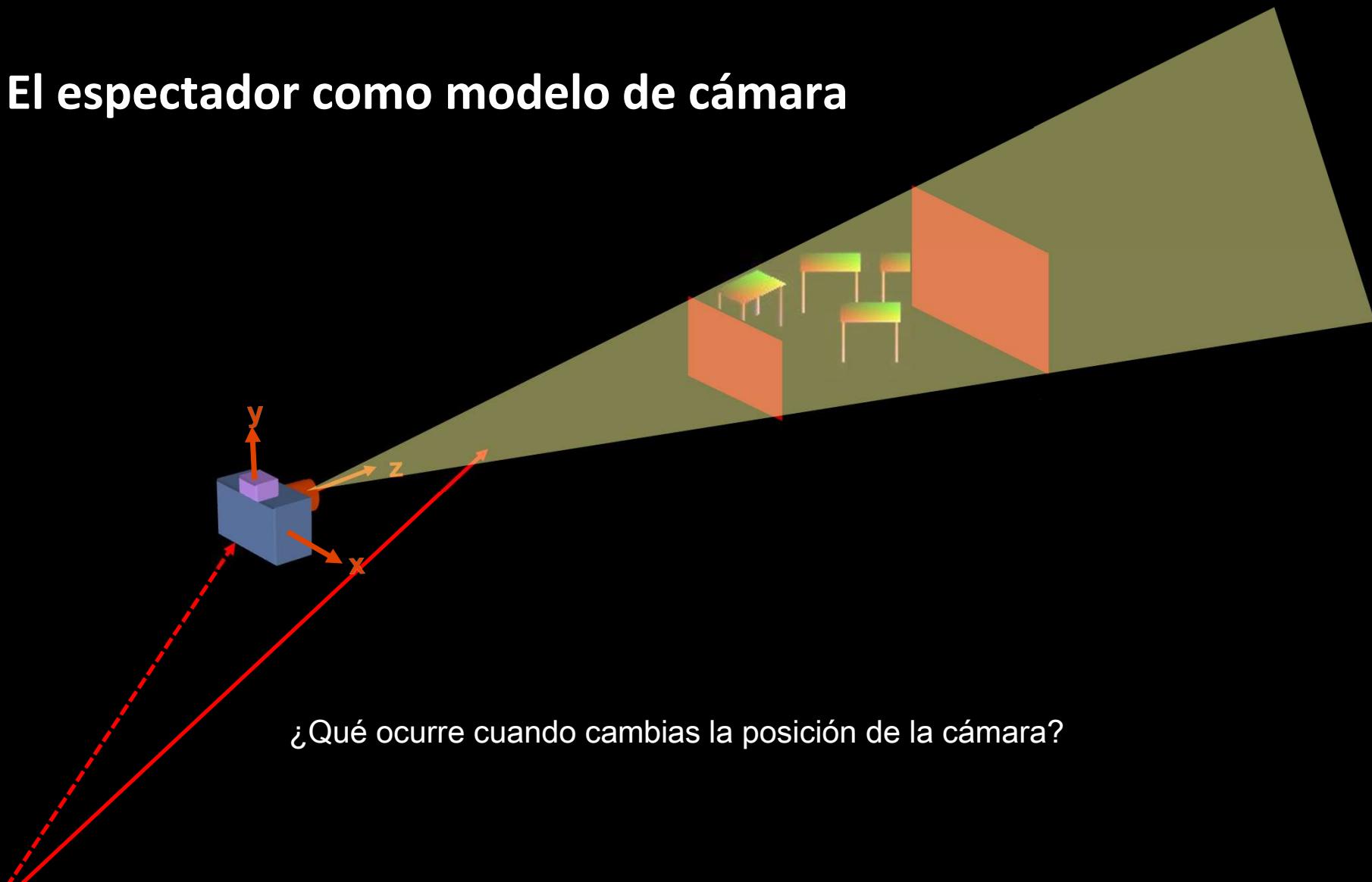
- `glm::perspective(float theta, float aspect, float near, float far);`
- Limitación: sólo frustum simétrico

## Perspektivische Projektionsmatrix

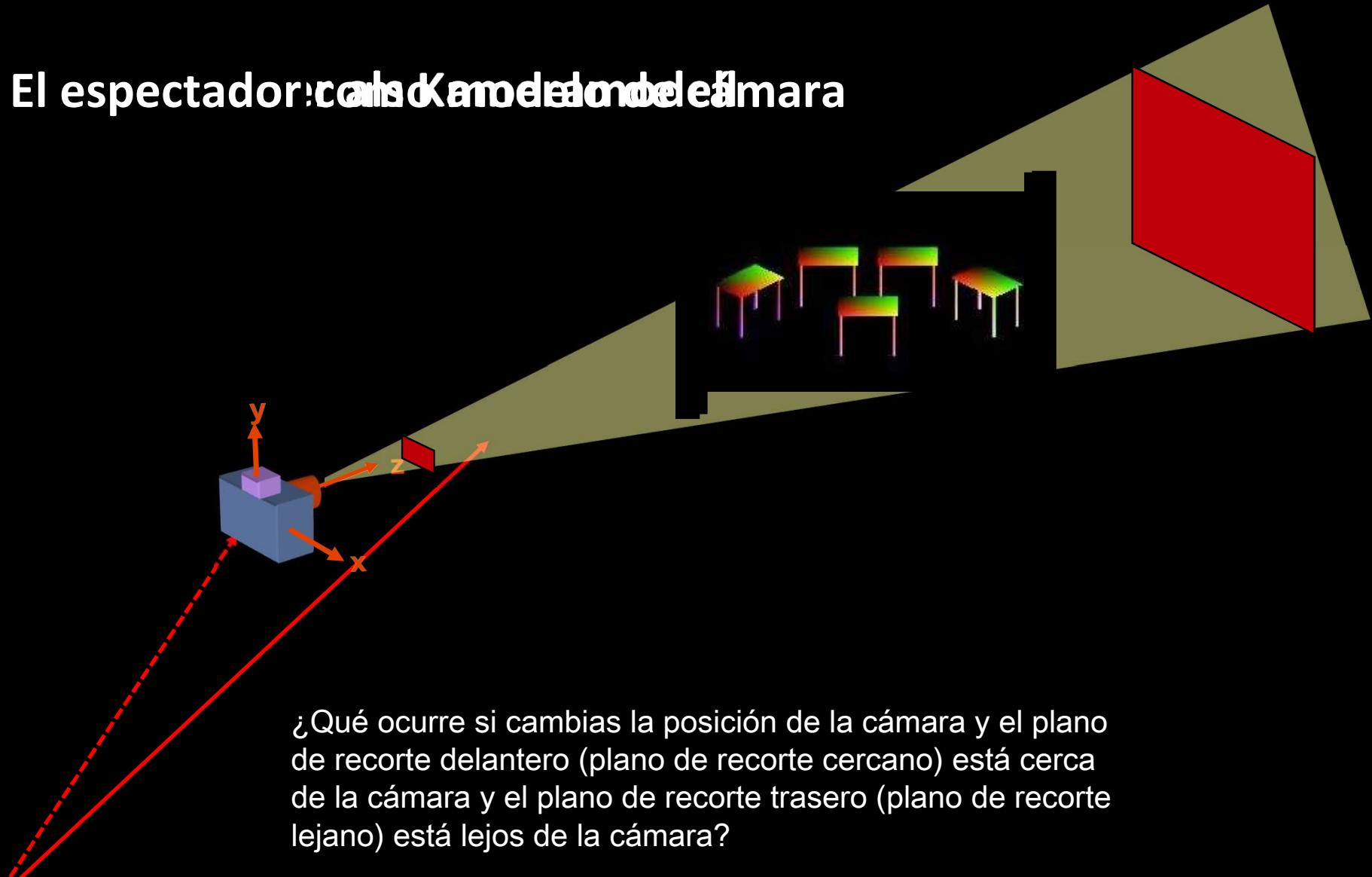
$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ w \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{n+f}{n} & -f \\ 0 & 0 & 1/n & 0 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ w \end{pmatrix} = \begin{pmatrix} p_x \\ p_y \\ p_z \cdot \frac{n+f}{n} - f \\ p_z/n \end{pmatrix} = \begin{pmatrix} np_x/p_z \\ np_y/p_z \\ n + f - nf/p_z \\ 1 \end{pmatrix}$$

## El espectador como modelo de cámara

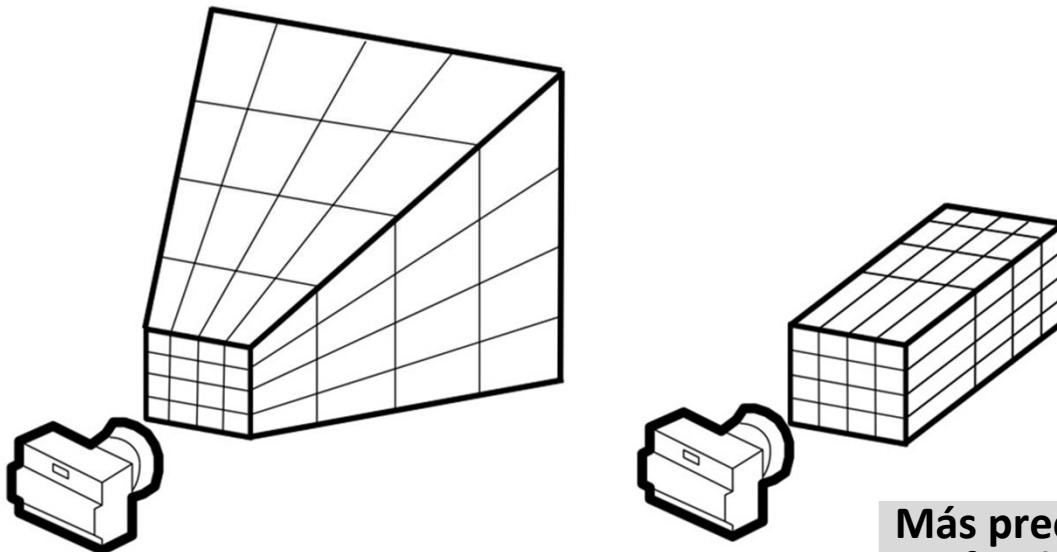


## El espectador: ~~cámaras~~ Kam de la cámara



## 7. cámaras

# Resultado de la matriz de



Más precisión de profundidad para

Transformación 3D:

- Los puntos del primer plano no se modifican
- las coordenadas x,y se escalan en perspectiva
- Escalado no lineal de las coordenadas z

## Transformaciones de vértices hasta ahora

**Transformación del modelo**  $M_{Model}$ : Transformaciones acumuladas, coloca objetos en el Escena (Mundo)

**Transformación de vista V**: Rota y traslada toda la escena para que la cámara esté en el origen y mirando a lo largo del eje Z negativo.

**Transformación de la proyección**  $M_{Proj}$ : Calcula la proyección, forma el frustum de la cámara en el Cubo  $[-w, w]^3$  abajo; dirección de visión a lo largo del eje z

$$\mathbf{p}' = \underbrace{M_{Projection} \cdot V \cdot T \cdot \dots \cdot T \cdot S \cdot R \cdot \mathbf{p}}_{M_{View} \cdot M_{Model}}$$

## 7. cámaras

# Negación de la coordenada z

Todavía falta una cosita:

- A los diseñadores no les gusta trabajar en un sistema de **coordenadas zurdo**, pero el **Espacio Normalizado de Dispositivos utiliza un sistema de coordenadas zurdo** (para tener valores z positivos para la distancia)
  - Recordatorio: En el Espacio-Cámara nuestra cámara debe mirar a lo largo del eje Z negativo
- ¿Cómo se pasa de un sistema de coordenadas izquierdo a un sistema de coordenadas derecho? ¿Y al revés?

$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{M_{R \rightarrow L}} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix}$$

## 7. cámaras

# Transformaciones de vértices hasta ahora

**Transformación del modelo**  $M_{Model}$ : Transformaciones acumuladas, coloca objetos en el Escena (Mundo)

**Transformación de vista V**: Rota y traslada toda la escena para que la cámara esté en el origen y mirando a lo largo del eje Z negativo.

**Transformación de la proyección**  $M_{Proj}$ : Calcula la proyección, forma el frustum de la cámara en el Cubo  $[-w, w]^3$  abajo; dirección de visión a lo largo del eje z

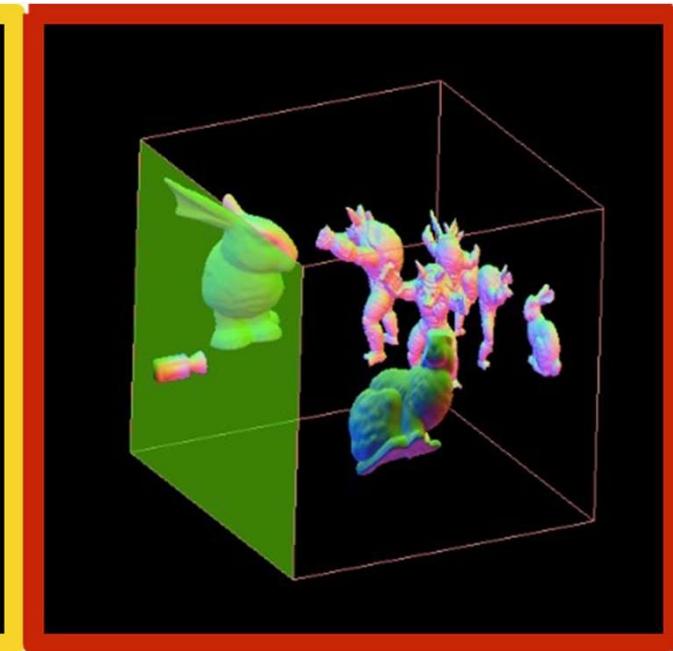
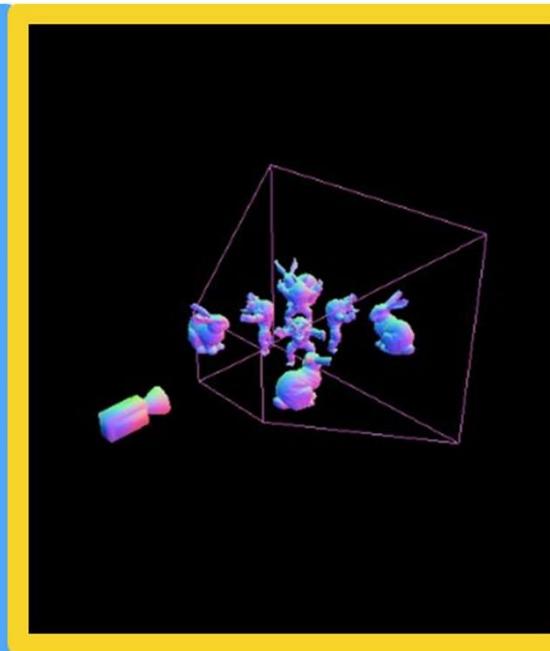
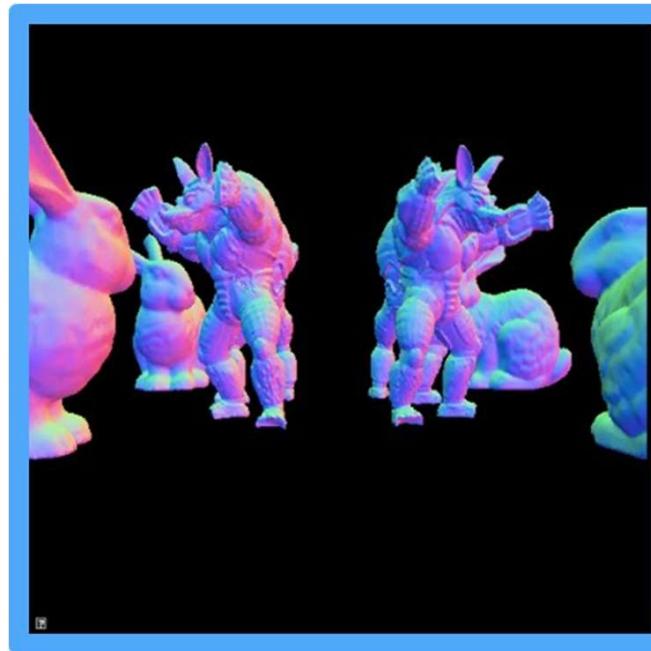
**Negación del eje Z**: multiplicación de los valores Z por -1

Inversión del eje Z ya integrada en los métodos gl

$$\mathbf{p}' = \underbrace{M_{Ortho} \cdot M_{Persp} \cdot M_{R \rightarrow L}}_{M_{ortho} \text{ y } M_{persp}} \cdot V \cdot \underbrace{T \cdot \dots \cdot T \cdot S \cdot R}_{M_{Model}} \cdot \mathbf{p}$$

## 7. cámaras

# Ejemplo de proyección



Vista de la cámara  
Espacio de la cámara  
dispositivo

Espacio normalizado del

## 7. cámaras

# Implementación de las transformaciones de vértices en el shader

### Sombreador de vértices

```
#Versión 330
layout(location = 0) en vec3 vertex;

uniform mat4 modelo;
e
uniform mat4 vista;
e
uniform mat4 proyección;
e

void main()
{
    gl_Position = proyección * vista * modelo * vec4(vértice, 1.0);
}
```

- Requiere sólo 4 líneas de código en el Vertex Shadershader
- La salida es la posición del vértice en coordenadas homogéneas, antes de la **perspectiva División** (homogeneización)

# Película de recambio

---