Tema 4: Viendo en 3D

J. Ribelles

SIE020: Síntesis de Imagen y Animación
Institute of New Imaging Technologies, Universitat Jaume I

Contenido

Introducción

Al igual que en el mundo real se utiliza una cámara para conseguir fotografías, en nuestro mundo virtual también es necesario definir un modelo de cámara que permita obtener vistas 2D de nuestro mundo 3D.

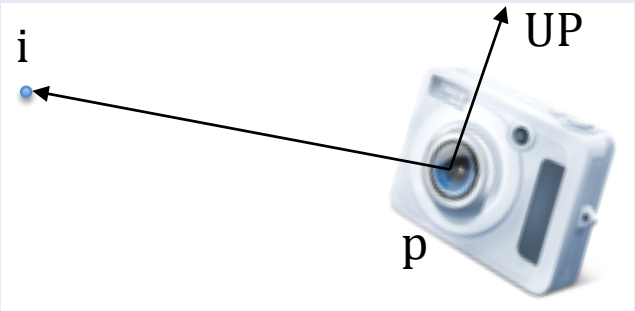
Se implementa como una secuencia de tres transformaciones:

- Transformación de la cámara: ubica la cámara virtual en el origen del sistema de coordenadas orientada de manera conveniente.
- Transformación de proyección: determina cuánto del mundo 3D es visible, a este espacio limitado se le denomina *volumen de la vista*, y proyecta el contenido del volumen en un plano 2D.
- Transformación al área de dibujo: consiste en mover el resultado de la transformación de proyección al espacio de la ventana destinado a mostrar la vista 2D.

Transformación de la cámara

¿En qué consiste?

- Situación inicial de la cámara:
  - La posición es un punto p del espacio 3D.
  - El objetivo queda apuntando a un punto i.
  - La inclinación se establece mediante el vector UP.



- Situación final requerida: cámara situada en el origen de coordenadas apuntando en la dirección del eje Z negativo y coincidiendo el vector de inclinación con el eje Y positivo.

Matriz de transformación de la cámara

Si:

- F es el vector normalizado que desde la posición de la cámara apunta al punto de interés,
- UP' es el vector de inclinación normalizado,
- $S = F \times UP'$ y $U = S \times F$,

Entonces M_C es la matriz de transformación de la cámara.

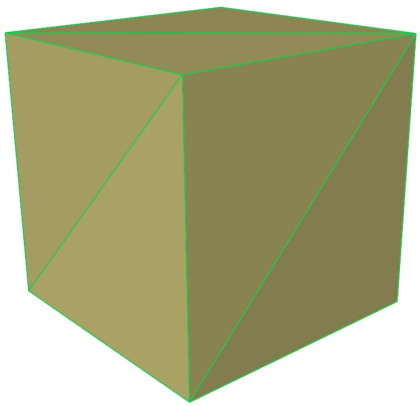
$$M_C = \begin{pmatrix} S_x & S_y & S_z & 0 \\ U_x & U_y & U_z & 0 \\ -F_x & -F_y & -F_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (1)$$

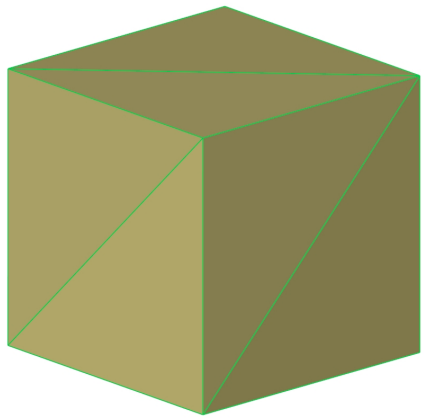
Transformación de proyección

Tipos de vistas

El volumen de la vista determina la parte del mundo 3D que puede ser vista por el observador. Dicho volumen depende del tipo de proyección.

- Vista perspectiva: se utiliza para generar imágenes más fieles a la realidad en aplicaciones como videojuegos, simulaciones o, etc.
- Vista paralela: utilizada principalmente en ingeniería o arquitectura, y se caracteriza por preservar longitudes y ángulos.

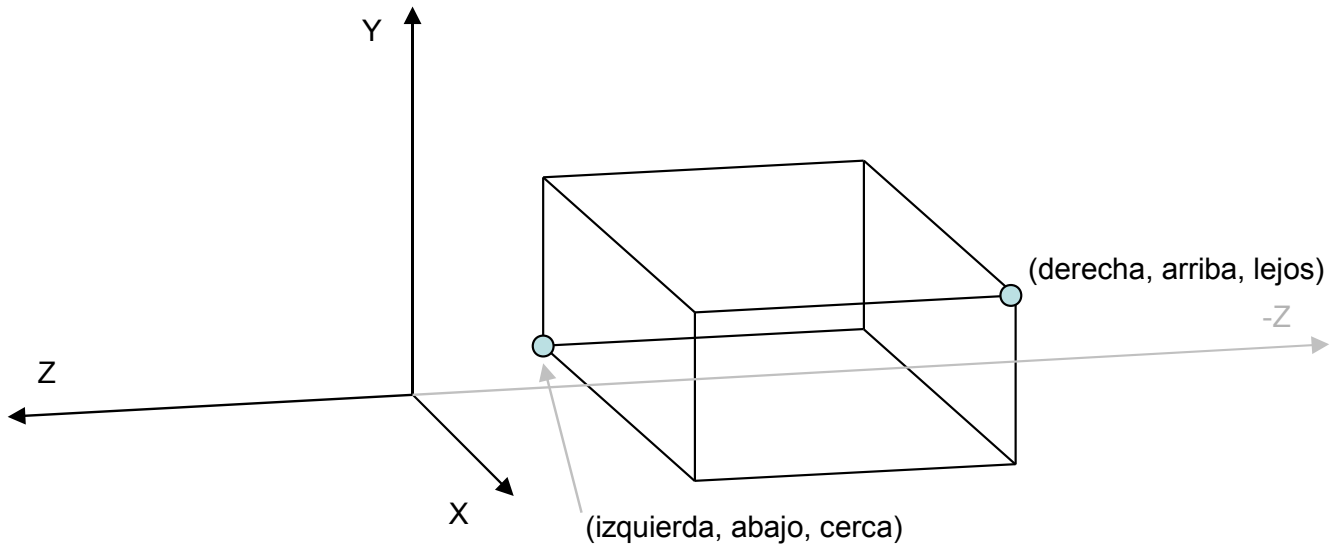




Proyección Paralela

Propiedades

- Los rayos de proyección son paralelos entre sí e intersectan de forma perpendicular con el plano de proyección.
- El volumen de la vista tiene forma de caja que se alinea con los ejes de coordenadas.



Matriz de transformación paralela

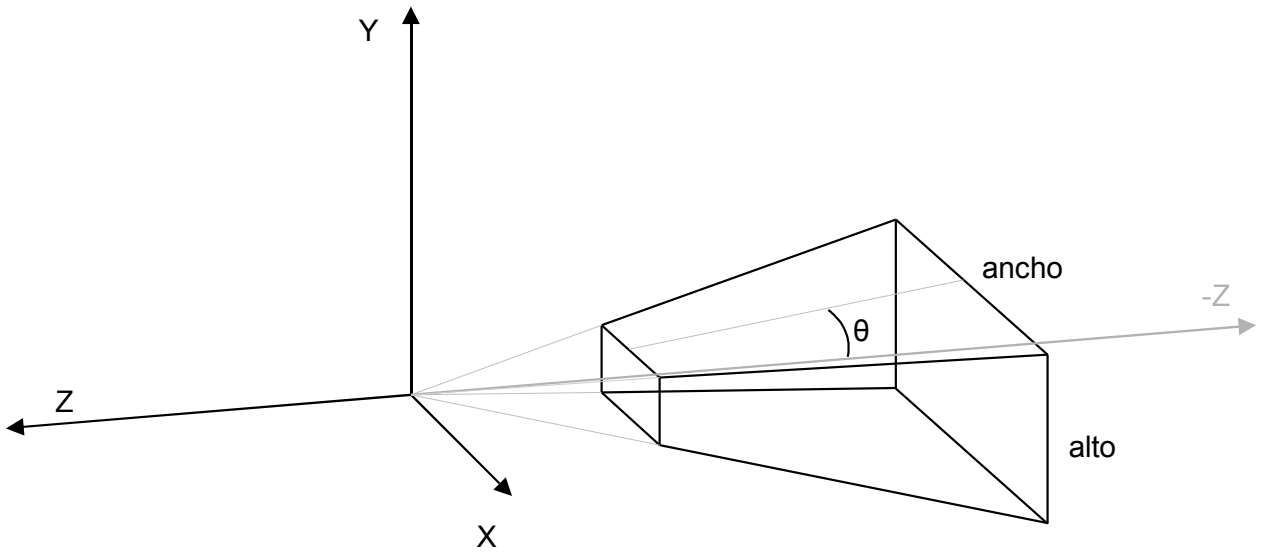
- Los sistemas gráficos trasladan y escalan esa caja de manera que la convierten en un cubo centrado en el origen de coordenadas.
- A este cubo se le denomina *volumen canónico de la vista*.
- Y a las coordenadas en este volumen *coordenadas normalizadas del dispositivo*.
- Para un cubo de lado dos:

$$M_{par} = \begin{pmatrix} \frac{2}{derecha-izquierda} & 0 & 0 & -\frac{derecha+izquierda}{derecha-izquierda} \\ 0 & \frac{2}{arriba-abajo} & 0 & -\frac{arriba+abajo}{arriba-abajo} \\ 0 & 0 & \frac{2}{lejos-cerca} & -\frac{lejos+cerca}{lejos-cerca} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$

Proyección Perspectiva

Propiedades

- Los rayos de proyección parten todos ellos desde la posición del observador.
- El volumen de la vista tiene forma de pirámide, definida por cuatro parámetros: los planos cerca y lejos, el ángulo θ en la dirección Y y la relación de aspecto de la base de la pirámide *ancho*/*alto*.



Matriz de transformación perspectiva

- El cálculo de la proyección perspectiva
- Los sistemas gráficos convierten ese volumen con forma de pirámide en el volumen canónico de la vista.
- Para un cubo de lado dos:

$$M_{per} = \begin{pmatrix} \frac{aspect}{tan(\theta)} & 0 & 0 & 0 \\ 0 & \frac{1}{tan(\theta)} & 0 & 0 \\ 0 & 0 & \frac{lejos+cerca}{cerca-lejos} & \frac{2 \cdot lejos \cdot cerca}{cerca-lejos} \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad (3)$$

Transformación al Área de Dibujo

Descripción

- El área de dibujo, también conocido por *viewport*, es la parte de la ventana de la aplicación donde se muestra la vista 2D.
- Consiste en mover el resultado de la proyección a dicha área.
- Se asume que la geometría a visualizar reside en el volumen canónico de la vista.
- Si n_x y n_y son el ancho y el alto del *viewport* en píxeles, y o_x y o_y son el píxel de la esquina inferior izquierda del *viewport* en coordenadas de ventana, la matriz de transformación es la siguiente:

$$M_{vp} = \begin{pmatrix} \frac{n_x}{2} & 0 & 0 & \frac{n_x-1}{2} + o_x \\ 0 & \frac{n_y}{2} & 0 & \frac{n_y-1}{2} + o_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

Eliminación de partes ocultas

Descripción

- Consiste en determinar qué primitivas de la escena son tapadas por otras primitivas desde el punto de vista del observador.
- El algoritmo más utilizado es el algoritmo conocido como *Z-Buffer*.
  - Requiere de un buffer denominado *buffer de profundidad*.
  - No importa el orden en que se pinten las primitivas.
  - Sí es muy importante que el buffer de profundidad se inicialice siempre al valor de profundidad antes de ...

Listado 1: Algoritmo del Z-Buffer

```
if (pixel.z < bufferProfundidad(x,y).z)
{
  bufferProfundidad(x,y).z = pixel.z;
  bufferColor(x,y).color   = pixel.color;
}
```

Viendo en OpenGL

Lo que el programador debe hacer

- Construir la matriz de transformación de la cámara y la matriz de proyección.
- La librería GLM proporciona diversas funciones:
  - mat4 lookAt (vec3 p, vec3 i, vec3 UP)
  - mat4 ortho (float izquierda, float derecha, float abajo, float arriba, float cerca, float lejos)
  - mat4 perspective (float θ, float ancho/alto, float cerca, float lejos)
- Suministrar ambas matrices al procesador de vértices donde ...
- Especificar la ubicación del *viewport*:
  - void glViewport (int x, int y, int *ancho*, int *alto*)
  - Debe realizarse cada vez que cambie en el tamaño de la ventana.
  - Es importante que la relación de aspecto del *viewport* sea igual a la relación de aspecto del volumen de la vista.

Lo que la funcionalidad fija de la GPU hace

- Entre el procesador de vértices y el de fragmentos:
  - Los elementos, o partes de ellos, que queden fuera del volumen de la vista son eliminados en la etapa de recortado.
  - Cada vértice es dividido por w, proceso denominado *división perspectiva*.
  - La transformación al área de dibujo.
- Tras el procesador de fragmentos:
  - Test de profundidad: la GPU comprueba la profundidad de cada fragmento.
  - Requiere ayuda del programador:
    1. Solicitar la creación del buffer de profundidad. glutInitDisplayMode (... | GLUT_DEPTH);
    2. Habilitar la operación del test de profunidad: glEnable (GL_DEPTH_TEST);
    3. Inicializar el buffer a la profundidad máxima antes de comenzar a dibujar: glClear (... | GL_DEPTH_BUFFER_BIT);