Viendo en 3D

Transformacion de la cámara

La posición de una cámara (el lugar desde el que se va a tomar la fotografía), se establece especificando un punto p del espacio 3D. Una vez posicionada, la cámara se orienta de manera que su objetivo quede apuntando a un punto específico de la escena. A este punto i se le conoce con el nombre de punto de interés.

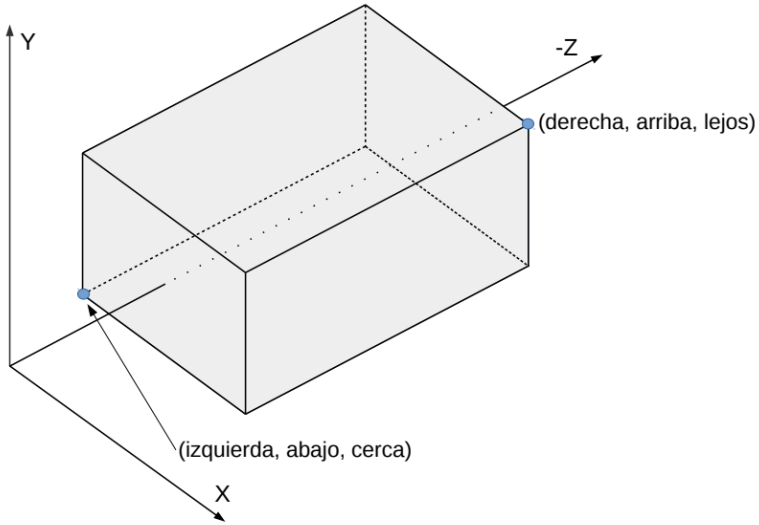
De la misma manera en la que ubicamos una cámara en Unity lo haremos mediante código con la matriz de transformación de la cámara que sitúa la cámara en la posición y orientación requeridas.

Transformacion de proyección

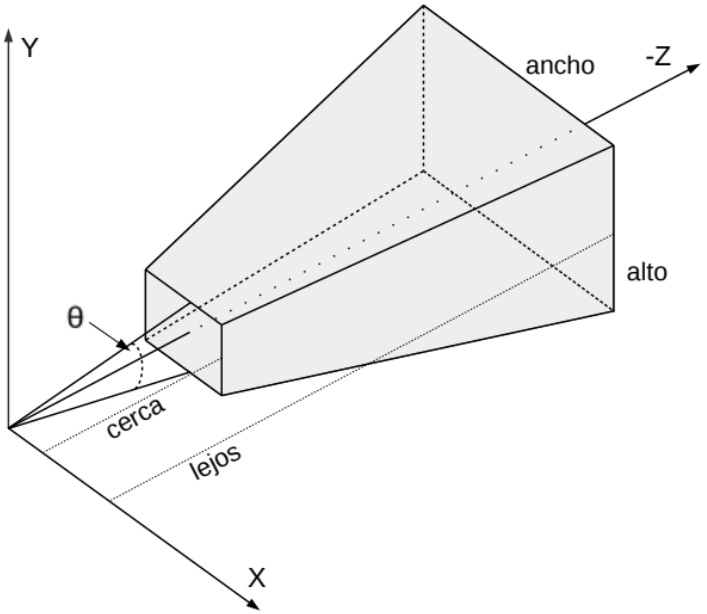
El volumen de la vista determina la parte del mundo 3D que puede ser vista por el observador. La forma y dimensión de este volumen depende del tipo de proyección:

- Proyección perspectiva. Es similar a como funciona nuestra vista y se utiliza para generar imágenes más fieles a la realidad en aplicaciones como videojuegos, simulaciones o, en general, la mayor parte de aplicaciones gráficas.
- Proyección paralela. Es la utilizada principalmente en ingeniería o arquitectura. Se caracteriza por preservar longitudes y ángulo

Transformacion de proyección



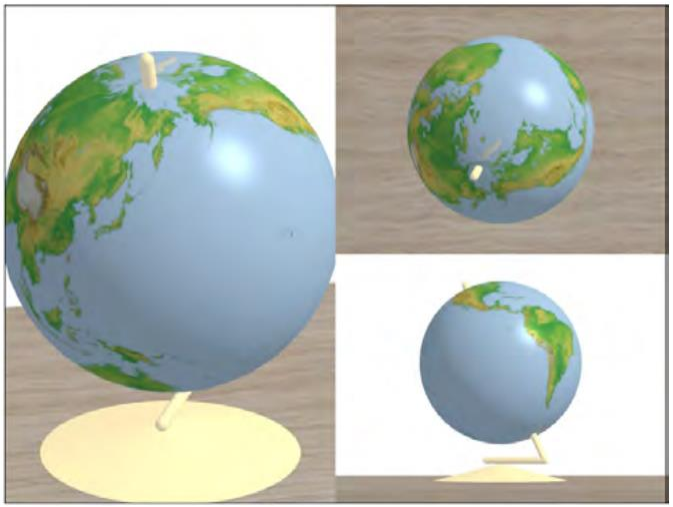
Proyección paralela



Proyección perspectiva

Transformacion al area de dibujo

El área de dibujo, también conocida por su término en inglés viewport, es la parte de la ventana de la aplicación donde se muestra la vista 2D. La transformación al viewport 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.



Eliminación de partes ocultas

La eliminación de partes ocultas consiste en determinar qué primitivas de la escena son tapadas por otras primitivas desde el punto de vista del observador. Aunque para resolver este problema se han desarrollado diversos algoritmos, el más utilizado en la práctica es el algoritmo conocido como z-buffer.

El **algoritmo z-buffer** se caracteriza por su simplicidad. Para cada píxel de la primitiva que se está dibujando, su valor de profundidad (su coordenada z) se compara con el valor almacenado en un buffer denominado buffer de profundidad. Si la profundidad de la primitiva para dicho píxel es menor que el valor almacenado en el buffer para ese mismo píxel, tanto el buffer de color como el de profundidad se actualizan con los valores de la nueva primitiva, siendo eliminado en cualquier otro caso.

Todo esto en WebGL

Hasta ahora, en los ejercicios realizados en los capítulos anteriores, se ha conseguido visualizar modelos sin tener que realizar ni la transformación de la cámara ni establecer un tipo de proyección. Esto ocurre porque la escena a visualizar se ha definido de manera que quedase dentro del volumen canónico de la vista.

Habitualmente, la matriz de la cámara se opera con la de transformación del modelo, creando la transformación conocida con el nombre de modelo-vista. Esta matriz y la de proyección se suministran al procesador de vértices, donde cada vértice v debe ser multiplicado por ambas matrices.

Todo esto en WebGL

Shader de vértices para transformar cada vértice

```
#version 300 es

uniform    mat4 projectionMatrix;   // perspectiva o paralela
uniform    mat4 modelViewMatrix;    // cameraMatrix * modelMatrix

in vec3 VertexPosition;

void main()  {

  gl_Position = projectionMatrix *
                modelViewMatrix *
                vec4(VertexPosition,1.0);

}
```

Todo esto en WebGL

GL Matrix

- mat4.lookAt (out, p, i, UP) --> matriz de transformacion de la camara
- mat4.ortho (out, izquierda, derecha, abajo, arriba, cerca, lejos) --> transforma el volumen de la vista a proyección paralela/ortográfica
- mat4.perspective (out, θ, ancho/alto, cerca, lejos) --> transforma el volumen de la vista a proyección perspectiva

Ejercicios

1. Carga la página camara.html. Comprueba que se ha añadido una cámara interactiva que puedes modificar utilizando el ratón. Comprueba también los cambios del contenido nuevo que se ha visto.
   Responde lo siguiente comentando en el código: ¿cuál es el punto de interés establecido? ¿qué tipo de proyección se utiliza?, ¿qué cambios harías para utilizar el otro tipo de proyección visto en este capítulo?

Ejercicios

2. Amplía la solución del ejercicio anterior para que se pueda cambiar de proyección paralela a perspectiva, y viceversa, y que sin modificar la matriz de transformación de la cámara se siga observando el modelo completo.

Ejercicios

3. Identifica donde se define el viewport (busca viewport). Busca en la documentación qué hace la función que encuentres.

Parte, por ejemplo, del modelo del tanque o del último ejercicio del tema de transformaciones que ya hiciste en el capítulo anterior y modifícalo para que en el canvas se muestren cuatro áreas de dibujo . Utiliza proyección paralela en las cuatro áreas pero haz que en tres de ellas la dirección de la vista coincida con uno de los ejes de coordenadas y que la cuarta sea en dirección (1,1,1). En todas ellas el modelo debe observarse en su totalidad.

Ejercicios

4. Amplía la solución del ejercicio 3 para que se realice la eliminación de las partes ocultas.

- Para observarlo mejor, utiliza un color diferente para cada primitiva y dibújalas como triángulos, no como líneas.
- Respecto a la eliminación de partes ocultas, WebGL implementa el algoritmo del z-buffer y la GPU comprueba la profundidad de cada fragmento de forma fija en la etapa de procesado del fragmento, pero después de ejecutar el shader de fragmentos. A esta comprobación se le denomina test de profundidad. Aunque esta operación está implementada en la GPU, el programador aún debe realizar dos tareas:

1. Habilitar la operación del test de profundidad (ya que por defecto no se rea liza): **gl.enable (gl.DEPTH_TEST);**
2. Inicializar el buffer a la profundidad máxima antes de comenzar a dibujar:

gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);