Mundos tridimensionales

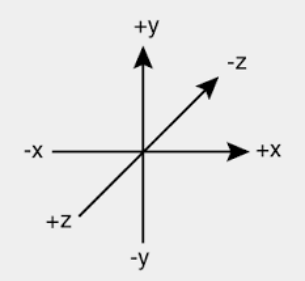
Introducción

En este post voy a introducir algunos conceptos básicos sobre los mundos tridimensionales. Vamos a enfocarlo des del punto de vista de los motores gráficos para juegos.
Los principales temas que vamos a tratar son:

- Coordenadas
- Espacio local (local space) y mundos (world space)
- Vectores
- Camaras
    - Projection mode (3D vs 2D)
- Plygons, edges, vertices and meshes
- Materials, textures and shader
- Rigidbody dynamics
    - Detección de colisiones

Coordenadas

En los mundos 3D se definen 3 coordenadas:



- Z-axis -> profundidad
- Y-axis -> altura
- X-axis -> anchura

Se representan de la siguiente manera (x,y,zy):

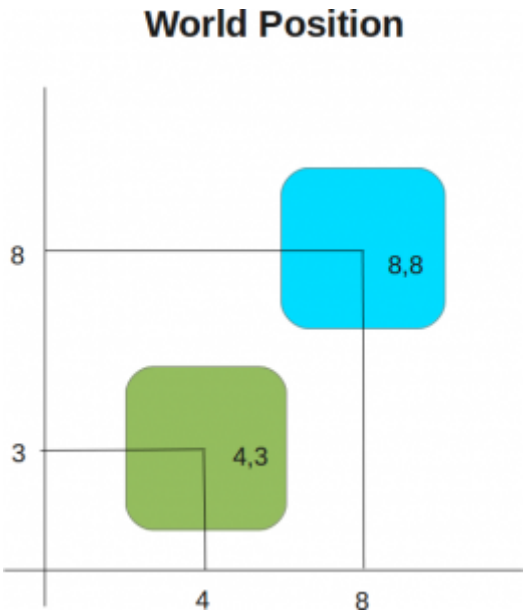
Local space (espacio local) y world space (mundo)

Es muy importante tener claras las diferencias entre local space y world space.

World space

En cualquier entorno 3D el world space será infinito por lo cual para poder referenciar la posición de los objetos deberemos marcar un punto llamado ‘origin’ o ‘world zero’. Este punto será la representación de la posición (0,0,0) .

Como podemos ver en esta imagen los dos objetos tienen sus respectivas posiciones en verso el punto (0,0), esto es igualmente aplicable a escenarios 3D.

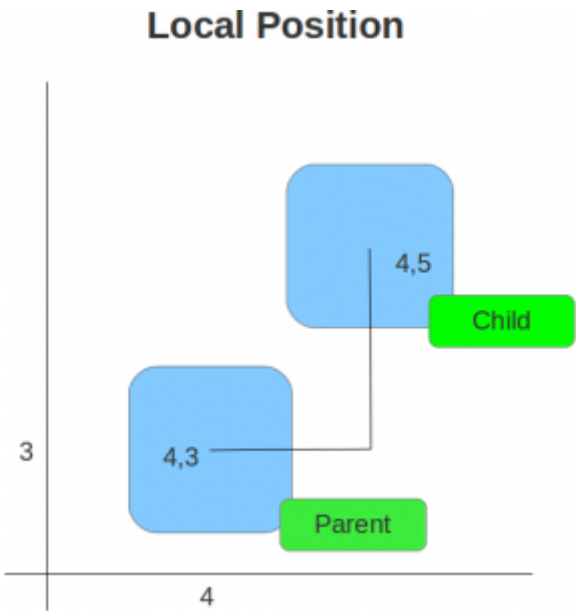


Local space

Los objetos en un escenario 3D están posicionados en relación al world zero. Pero, para hacer las cosas un poco más fáciles usamos local space (o también llamado object space) para definir la posición de un objeto respecto de otro. Esta relación se conoce como parent-child. Local space asume que cada objeto tiene su posición (0,0,0), desde este punto, normalmente el centro del objeto, se expandirá su volumen. La

relación parent-child nos permitirá comparar las posiciones de diferentes objetos entre ellos a través de las relaciones.

Como podemos ver en esta imagen el segundo objeto (child) con posición (4,5) esta situado respecto al objecto (parent) con posición (4,3), esto es igualmente aplicable a escenarios 3D.



Posicionamiento y diseño de assets

Es importante tener en cuenta que si diseñamos assets con alguna herramienta de modelado (por ejemplo blender) tenemos que asegurarnos que la posición en la que creamos el modelo sea la (0,0,0).

Vectores

Los vectores simplemente son lineas dibujada en un escenario 3D con una longitud y dirección. Se pueden mover en el world space sin que estos sufran modificaciones. Son muy útiles en el diseño de juegos 3D, nos permiten:

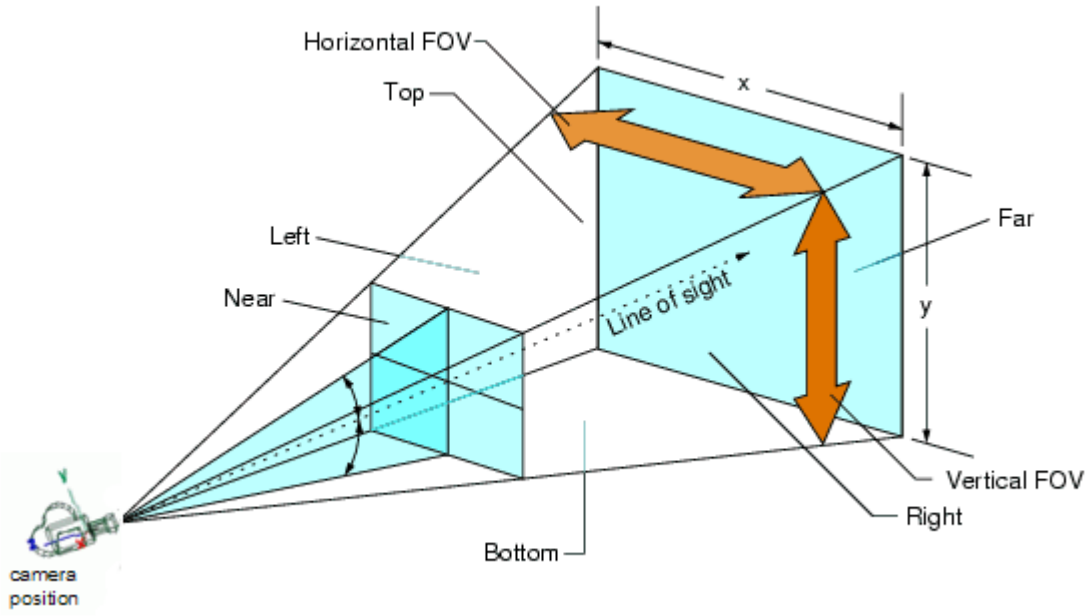
- Calcular distancias entre objetos
- Ángulos de posición entre ellos
- La dirección de estos

Cámaras

Las cámaras son esenciales en mundos 3D, ya que actúan como el punto de visión. Pueden ser posicionadas en cualquier punto del mundo, animadas, enlazadas a un objeto u objetos que forman parte de un escenario. Varias cámaras pueden existir en una escena particular, pero se asume que una siempre será la «main camera» (cámara principal) que renderizará lo que el usuario puede visualizar.

Projection mode (3D vs 2D)

El Projection mode (modo de proyección) de estados cámara se representa en 3D (perspectiva) o 2D (ortográfica). Por lo general, las cámaras están en el modo perspectiva de proyección, y como tal, tiene un campo de visión en forma de pirámide (FOV).



Se puede utilizar en una cámara principal rectangular para juegos 2D o como cámara secundaria en juegos 3D que sirve para hacer Heads Up Display (HUD) elementos como un mapa o una barra de energía. En los motores de juego los efectos como iluminación, desenfoques de movimiento y otros se aplican a la cámara para ayudar a crear una experiencia de juego más realista al ojo humano. Los juegos 3D más modernos utilizan

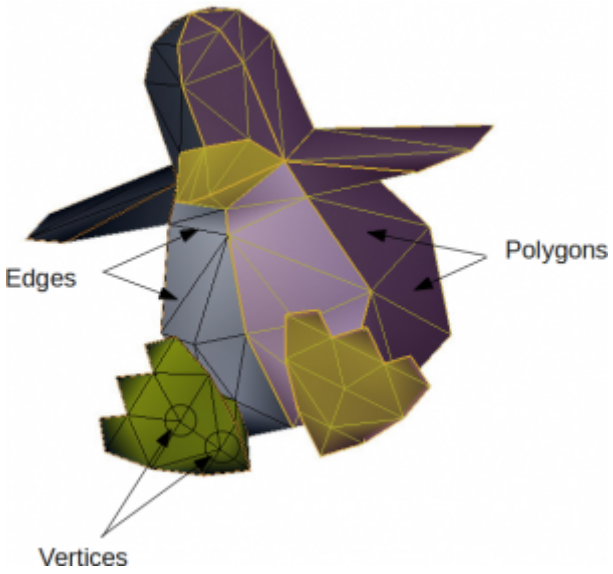
varias cámaras para mostrar las partes del mundo.

Plygons, edges, vertices and meshes

Las formas 3D estan compuestas de pequeñas formas 2D:

- Polygons: Formas 2D (por ejemplo en Unity son triángulos)
- Edges: Los triángulos están formados por edges (bordes) conectados entre si.
- Vertices: Punto en el que convergen los edges.
- Meshes: son formas complejas creadas a partir de la interconexión de muchos plygons. Por ejemplo el tux que tenemos bajo estas lineas.

Tux Mesh



Al conocer estos puntos y ploygons, los motores de juego son capaces de hacer cálculos sobre los puntos de impacto, conocidos como colisiones. Se utiliza la detección de colisiones complejas con Mesh Colliders (colisionadores de malla), por ejemplo en juegos de shooters para detectar la ubicación exacta en la que una bala impacta sobre un objeto. Los meshes pueden tener otros usos, por ejemplo se pueden utilizar para especificar una forma del objeto menos detallado, pero más o menos la misma forma. Esto puede ayudar a mejorar el rendimiento del motor de física evitando que este

compruebe un mesh muy detallado.

Materials, textures and shader

- Materials: Son un concepto común en los entornos 3D, ya que establecen la apariencia visual de un modelo 3D (mesh). Desde colores básicos a reflejos en imágenes que representan superficies, los materiales lo manejan todo.
- Textura: una o más imágenes que se pueden aplicar al material para mejorar la apariencia de este.
- Shaders: un material trabaja con un shader, que no es más que un script encargado de estilizar el renderizado.

Cuando se crean texturas con programas de diseño como Photoshop o GIMP, hay que tener en cuenta la resolución. Grandes texturas aseguran un mayor detalle pero es más costoso renderizarlas.

Rigidbody physics

Cuando se trabaja con motores de juego, los motores de física proporcionan la posibilidad de que los objetos en los juegos simulen respuestas similares a las del mundo real.
En los motores de juego, por defecto un objeto no debe verse afectado por la física, porque esto requiere una gran cantidad de potencia de procesamiento, y porque simplemente no hay necesidad de hacerlo. Por ejemplo, en un juego de de carreras en 3D tiene sentido
que los coches esten bajo la influencia del motor de física, pero no la pista o alrededores (árboles, paredes…) que se mantengan estáticos durante la partida.
Los motores de física para juegos utilizan el sistema Rigidbody dynamics para crear movimientos realistas. Esto significa que en lugar de tener objetos estáticos en un entorno 3D, estos pueden tener propiedades tales como la masa, gravedad, velocidad y fricción. A medida que la potencia del

hardware y software incrementa, Rigidbody physics es cada vez más usado, ya que ofrece la posibilidad simular entornos 3D más variados y realistas.

Detección de colisiones

La detección de colisiones es la forma en la que analizamos nuestro mundo 3D. Al aplicar un Collider component a un objeto, estamos poniendo una red invisible a su alrededor. Esta red, por lo general, imita su forma y es la encargada de informar de cualquier colisión con otros Colliders, haciendo que el motor de juego de responder en consecuencia.

Observaciones
Bien, este ha sido el primer post. Se han introducido los conocimientos básicos para poder trabajar en un entrono 3D. En el siguiente post haremos una aproximación al motor Unity.

Ruben.