Creación de Videojuego 3D

Luzquiños Agama, Steve Anthony
sluzquinosa@uni.pe

Vicente Alva, Gabriel
gvicentea@uni.pe

September 7, 2020

Universidad Nacional de Ingeniería, Facultad de Ciencias
Curso:

CM3M2 - Matemática Computacional
Proyecto formativo

Abstract

Modern video games face a number of different challenges, from planning to deployment. A big number of them are related to the interactions between the game and the user, and how this interaction triggers many kind of actions in the video game, for instance: movement, collisions, rotations, illumination, shades, response time, difficulty, and etc. This work will cover the theoretical fundamentals about how to overcome this difficulties. Using many topics we have learned in the course, mainly linear transformations.

Keywords: *video games*, *linear transformations*.

Índice

---

1 Fundamento teórico

La principal acción llevada a cabo en cualquier videojuego es el movimiento, este se llevará a cabo usando transformaciones lineales, específicamente traslaciones.

1.1 Transformaciones

Las transformaciones las realizamos usando el plano proyectivo. El plano proyectivo añade puntos "en el infinito" a un plano, haciendo que dos líneas paralelas siempre tengan una intersección. Podemos hacer esto con $\mathbb{R}^2$ y $\mathbb{R}^3$ añadiendo una coordenada que valdrá 1 para cada punto. En los puntos infinitos valdrá 0.

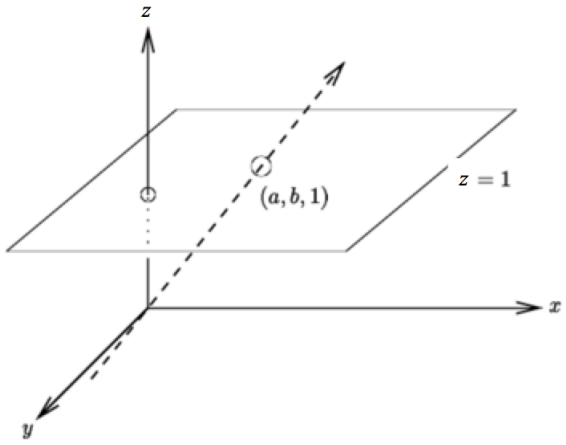

Fig. 1: Plano proyectivo

Con esto las funciones de traslación, escala, rotación y otras serán transformaciones lineales. Por ejemplo, la transformación de traslación estará dada por:

$$\begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Esta matriz se multiplca con el vector a trasladar.

1.2 Iluminación

Flat shading: En cada polígono se calcula la luz que llega al observador según la dirección (vector normal) del polígono. Según esto la cara se ilumina u oscurece uniformemente. Este método será sufieciente para nuestros modelos de pocos polígonos.

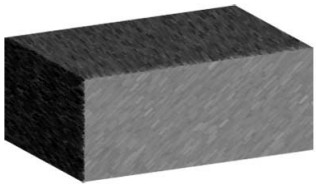

Fig. 2: *Flat shading*

2 Planteamiento del problema

El objetivo final del presente trabajo es la realización de un videojuego 3D, basado en el conocido videojuego *Star Fox*, para esto hemos optado por realizar un *endless runner* donde nuestro personaje principal será una nave espacial que "enfrentará" una serie de personajes secundarios, los cuales tendrá que esquivar o destruir, con el fin de ganar el juego.

Tal y como se muestra en las imágenes, nuestro personaje principal estará "atrapado" en mundo sin fin. Cabe mencionar que las imágenes no corresponden con el resultado final planificado, solo sirven para ser utilizadas de analogía.

Veamos los aspectos técnicos del problema:

- Los objetos tridimensionales pueden ser definidos a través de vectores y se les puede aplicar transformaciones. Sin embargo, graficar dichos objetos requerirá tomar otros factores en cuenta. Por ejemplo, estos

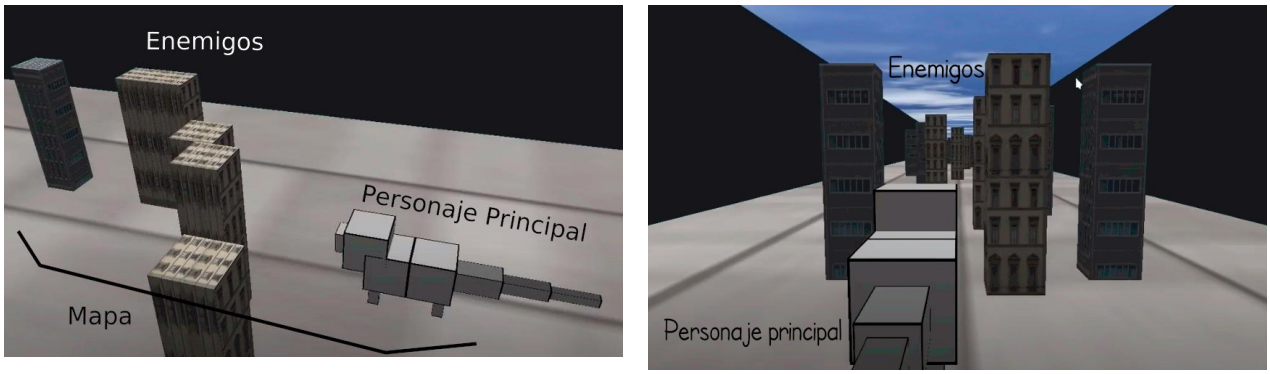

Fig. 3: Visualización de la escena del *endless runner*

necesitarán un color (serie de colores) o una textura (serie de texturas) para su renderizado, además de que estos usarán transformaciones lineales para poder desplazarse en la escena, finalmente, los objetos deben ser capaces de ser destruidos (desaparecer o reubicarse) por el personaje principal.

- Los polígonos deben ser proyectados en el plano, con esto se sabe qué pixeles de la matriz son "coloreados", estos colores también son calculados según la iluminación, y al graficar uno o varios objetos se debe elegir un método para graficar los polígonos sin que se solapen incorrectamente. Esto con el fin de que el jugador debe ser capaz de reconocer la distancia entre los objetos y el personaje principal.
- Para hacer un videojuego en 3d, en cada fotograma se calculan diversas cantidades, entre ellas por supuesto, la posición de los objetos. Todos estos variarán en el tiempo y por interacciones del jugador. Estos calculos deben ser optimizados para que la tasa de fotogramas no destruya la experiencia de usuario, ya que la experiencia nos indica que la CPU puede procesar una cantidad limitada de enemigos en la escena sin pérdida de paquetes o frames (justamente lo que se busca evitar).

3 Bosquejo de la solución del problema

3.1 Personaje principal

El gato mostrado líneas arriba será reemplazado por la siguiente nave espacial:

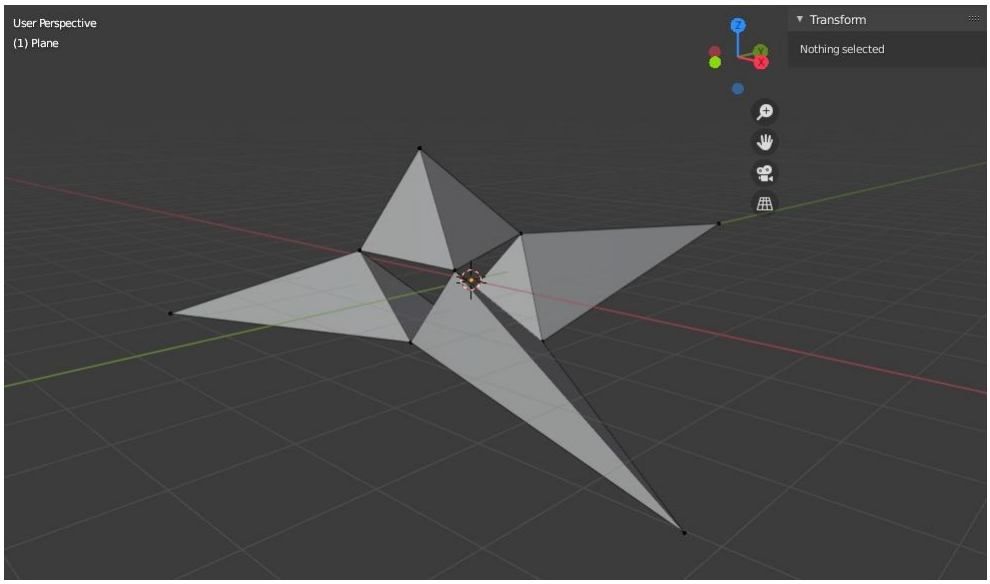

Fig. 4: Nuevo personaje principal

La cual se diseñó usando *Blender*.

3.2 Diseño de la escena

La nueva escena será desarrollada usando las siguientes librerías de Python: PyOpenGl, pygame y glfw.

3.3 Transformaciones

Las transformaciones, y en general las operaciones matemáticas, serán llevadas a cabo usando las siguientes librerías de Python: pyrr y numpy.

3.4 Renderizado

El renderizado de los objetos en la escena será llevada a cabo utilizando *shaders*.

3.5 Iluminación

Se utilizará la técnica de *flat shading* mencionada en el fundamento teórico.

4 Referencias Bibliográficas

1. Trabajo base: Miauzilla.
2. Librerías de Python:
   - PyOpenGL
   - pygame
   - glfw
   - pyrr
   - numpy
3. *Shaders*.

References

[1] Steven Janke, *Mathematical Structures for Computer graphics*. Wiley, 2014.

[2] Bui Tuong Phong, *Computer Generated Pictures* .
