Documento PostMortem

## Por: Esteban Ricardo Rodriguez Moreno

## Proyecto: Practica Final

## Bogotá D.C

## 2020

PostMortem

# Logros

## Implementación GameObjects:

Se implementa clase abstracta GameObject de la cual puede heredar cualquier objeto nuevo creado en juego, esta clase realiza el render de los objetos y solicita que cualquier objeto credo tenga un método update y onCollision.

## Implementación FBO:

Implementación de clase FBO, des la cual se puede administrar la creación, eliminación, generación de FBO’s.

## Implementación Scene:

Se implementa la clase Scene, en esta clase se realiza el render de todos los objetos que se encuentren dentro de la lista de objetos de la escena; por último, también contiene los métodos para limpiar los FBO al terminar la ejecución.

## Implementación Light:

Se crea la clase global Light la cual debería funcionar como clase padre para cualquier tipo de luz que se desee crear (spot light, point light, direccional light); contiene el método de render y los parámetros para instanciarla en el Shader.

## Collisiones de Objetos:

Se implementa método onCollision el cual se ejecuta cada vez que se sobrepone a otro collider y envía el objeto con el que colisiono; los collider implementados son todos circulares.

# Complicaciones

## Sombras:

Luego de agregar las texturas normales al shader, dejaron de funcionar las sombras sobre los objetos creados, se intentó varias configuraciones, pero no se lograron implementar de forma exitosa.

## Múltiples Luces:

A pesar de que la arquitectura del programa está pensada para la implementación de múltiples luces, por falta de tiempo no se logra implementar la lógica de spot lights para su implementación.

## Arquitectura GameObjects:

La arquitectura del motor es funcional y sencilla para agregar nuevos objetos a las escenas y hacer que actualicen sus parámetros básicos (posición, rotación, escala) durante la ejecución del juego, aun así, no me parece que su implementación sea la correcta ya que era necesario generarlo después de la creación de un nuevo objeto.

## Control fuera de objeto:

Luego de implementar los game objects tuve problemas realizar el control de disparo, ya que los modelos bases se iniciaban en el main, no se corrige por tiempo; pero se deja de forma funcional; los controles de movimiento si se encuentran dentro del método update del GameObject

# Conclusiones

La computación grafica presenta una gran complejidad en sus conceptos, al ver tantos temas en tan poco tiempo es difícil llegar a manejarlos de forma fluida para la práctica final, en mi caso los shaders fueron los objetos que me causaron mas complejidad ya que en su implementación se encuentran cosas como texturas, luces, colores, sombras y postprocesado, esto las hace un poco más difícil de entender, teniendo en cuenta que en general solo devuelven un único valor al final.

Aun así, se aprenden otras cosas durante la ejecución de la clase, como el manejo de componentes(hardware) desde código optimizando así las funciones y métodos, esto se ve ayudado por la necesidad del motor grafico de optimizar todos sus componentes para no saturar el sistema. También arquitectura e implementación de patrones de diseño sobre código, se logró mejorar el manejo en el lenguaje C++ ya que, a pesar de ya manejar otros lenguajes, C++ en conjunto con la librería de GLM presentan cierta complejidad y mas cuando la prioridad en un motor grafico son los tiempos de computo.

Por ultimo el manejo tanto de recursos de la CPU como de la GPU para redendizar los objetos además de la importación y formatos de modelos complejos en 3D (obj, fbx), permite mayor dinamismo para los motores generados, los fundamentos del funcionamiento de un juego en 2D quedan comprendidos y los componentes básicos para la creación de un videojuego (Cámaras, Luces, Objetos etc.)