Proyecto Final – Carrera de naves

Antonio García Ulises Eduardo

Computación Gráfica Avanzada

Grupo 1

Repositorio: <https://github.com/gitloot/ComputacionGraficaAvanzada>

Índice

[Objetivo 1](#_Toc63702661)

[Hipótesis 1](#_Toc63702662)

[Introducción 1](#_Toc63702663)

[Plan de trabajo – Cronograma 2](#_Toc63702664)

[Desarrollo 3](#_Toc63702665)

[Manual de usuario 22](#_Toc63702666)

[Costo del proyecto. 23](#_Toc63702667)

[Licenciamiento. 23](#_Toc63702668)

[Principales retos y dificultades, mejoras, trabajo a futuro. 23](#_Toc63702669)

[Conclusión 25](#_Toc63702670)

# Objetivo

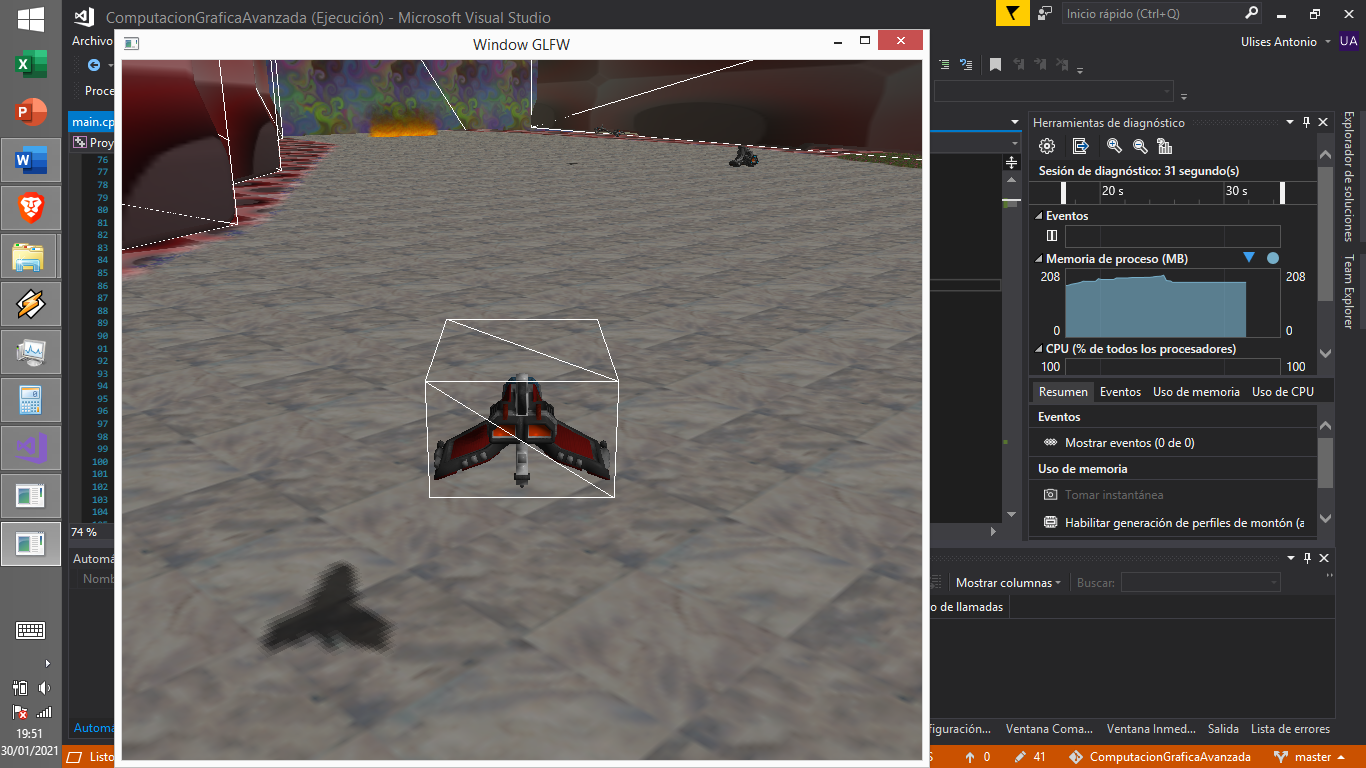
Con la realización de este proyecto se buscó poner en práctica el conocimiento adquirido en clase haciendo un videojuego con las herramientas de OpenGL, además de cumplir con una tasa de frames de al menos 30, así como reforzar los conceptos adquiridos al hacer un repaso en el proceso.

# Hipótesis

A partir del conocimiento adquirido en un curso de Computación Gráfica, más el aprendizaje de un bagaje de otras materias relacionadas con las matemáticas e ingeniería se puede hacer un videojuego sencillo en un semestre.

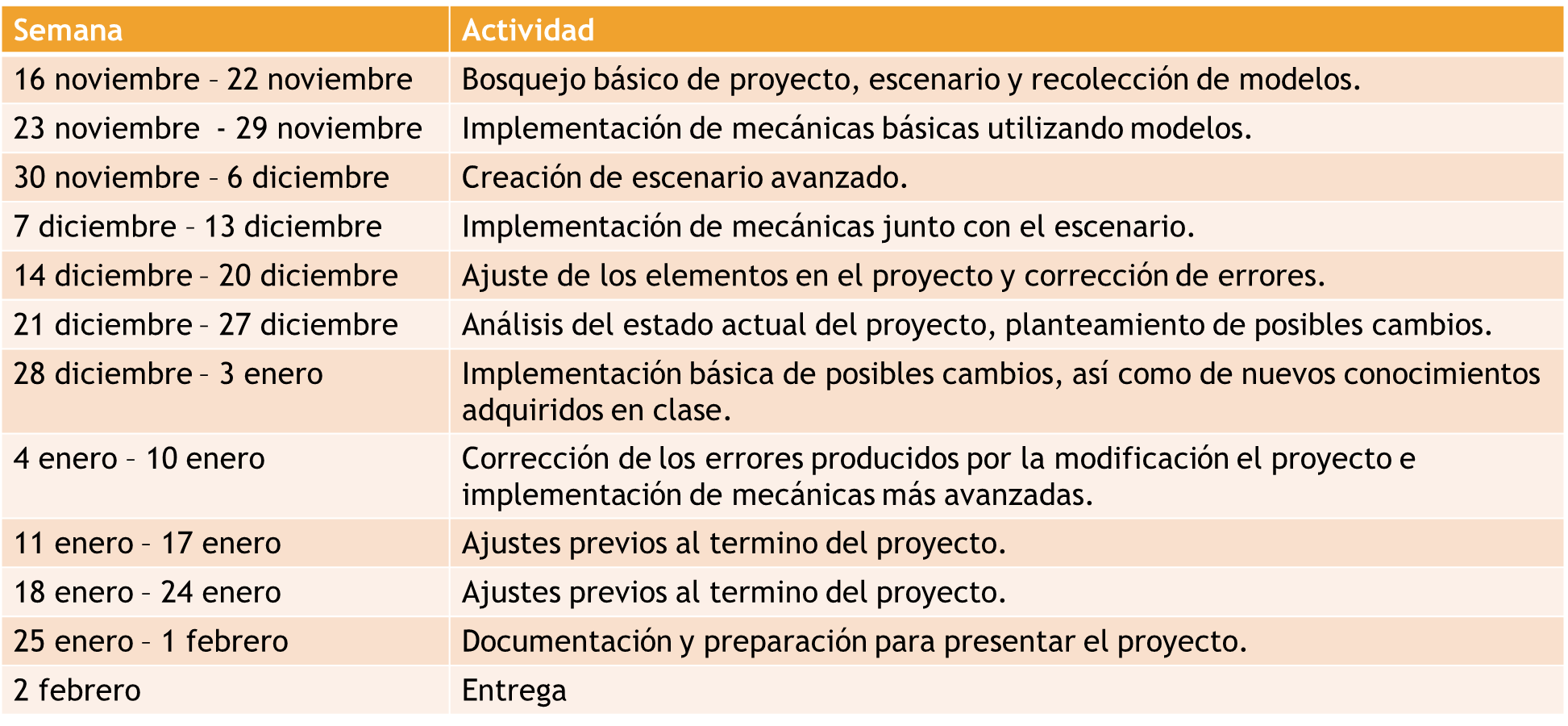
# Introducción

El concepto básico de este proyecto es el de un videojuego de carreras con aeronaves en donde el jugador debe recorrer un circuito tratando de arrebatar y ganar una carrera contra otros tres contrincantes.



# Plan de trabajo – Cronograma

A lo largo del desarrollo de este proyecto se planeó seguir con el siguiente esquema.



A lo largo del trayecto se pudo cumplir con un porcentaje favorable de dicho plan de trabajo. En donde se presentaron atrasos debido a otros proyectos y practicas de clase y a la corrección de errores.

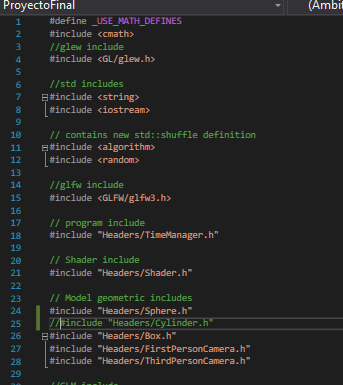
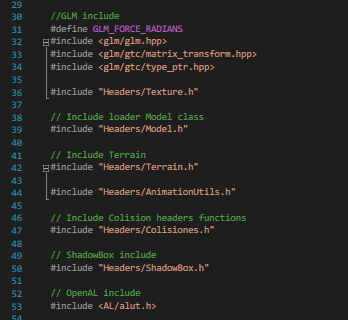
# Desarrollo

Utilizando como base la practica 14 que es la última que se realizó en clase se implementaron definiciones de funciones, así como de variables parecidas a las que aparecen ahí.

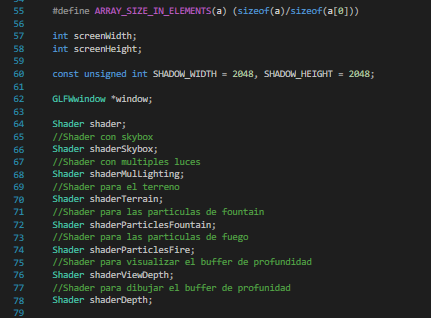
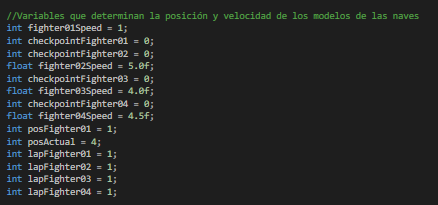
Aunque se descartaron algunas funciones para hacer correr mejor el programa y porque no había una forma más fácil de darles utilidad, entre aquellas que se descartaron fueron las que generan partículas de agua y el uso de keyframes.

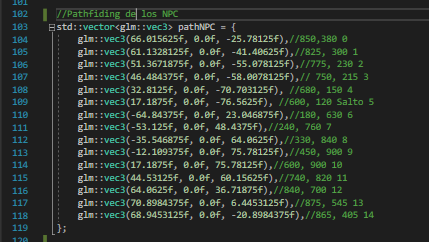
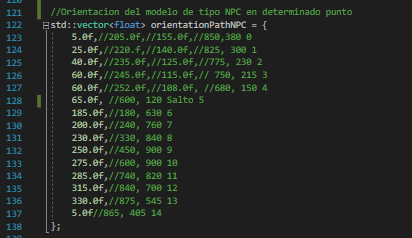
El código del archivo main del proyecto se encuentra estructurado de la siguiente forma:

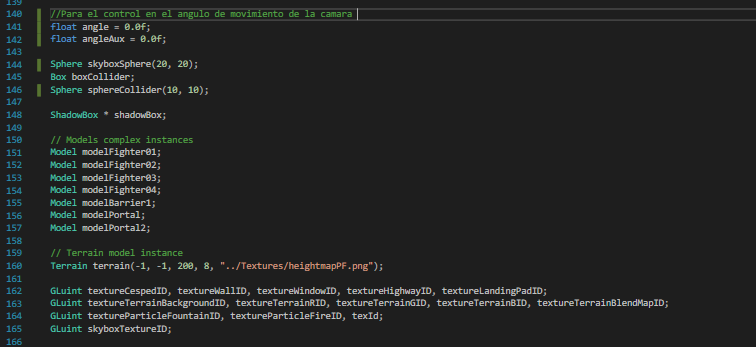
* En el inicio se establecen las librerías que se están utilizando, algunas provistas por el lenguaje de cpp, OpenGL, OpenAL, y otras que se crearon para el curso y fueron proporcionadas por el profesor.

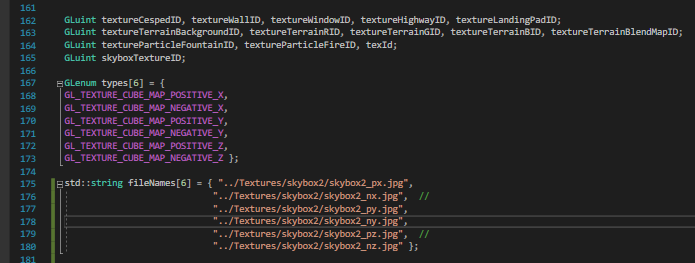
 

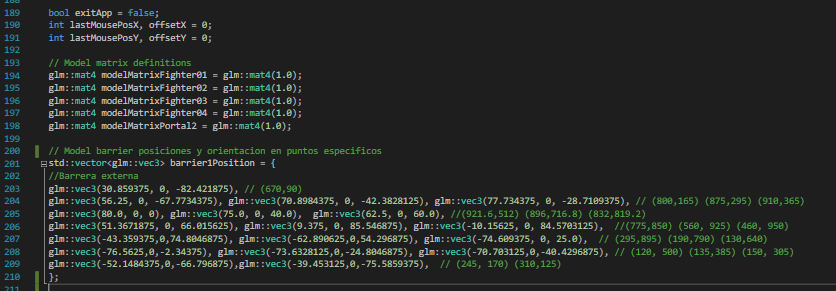
* Enseguida se encuentran las variables globales para todos los elementos que se crean y son utilizados por las distintas funciones, asimismo como la declaración de estos procedimientos.

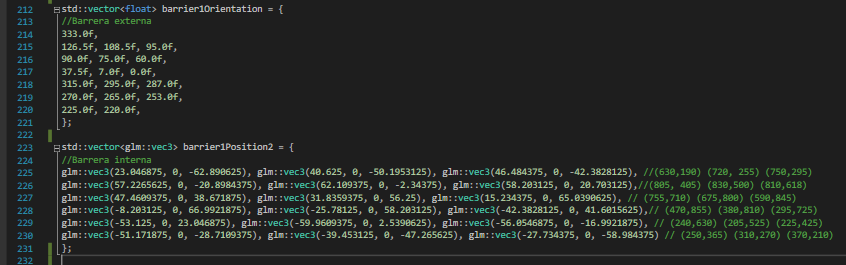
 

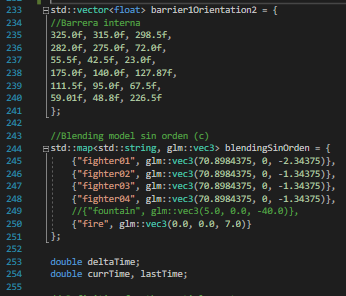
 

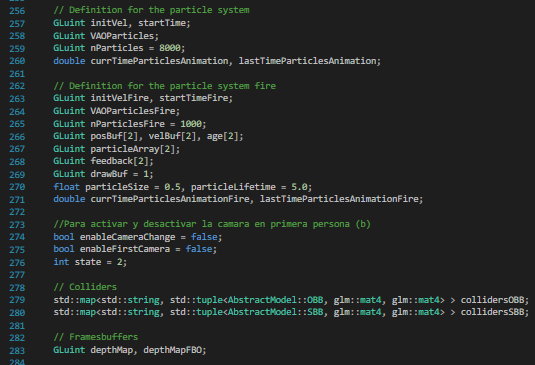


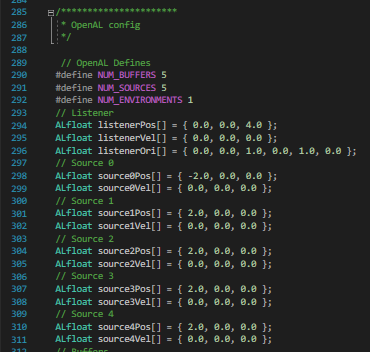
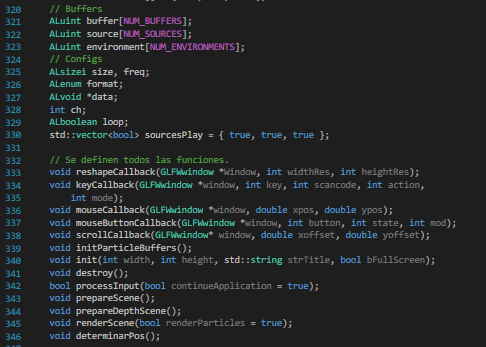




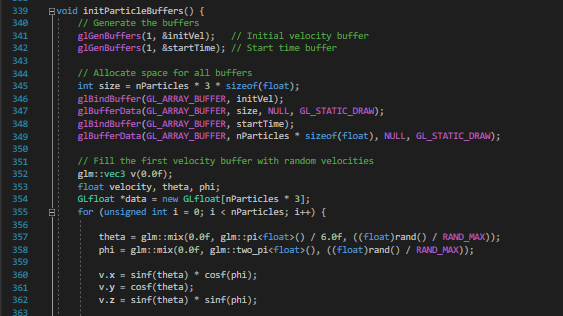
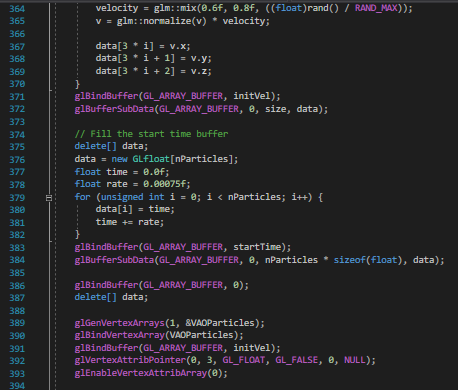




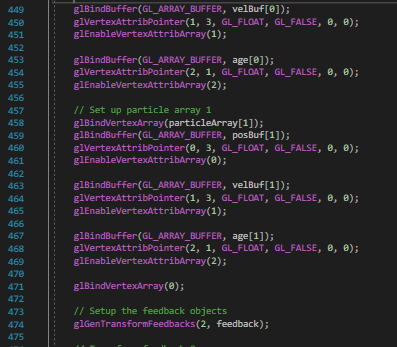
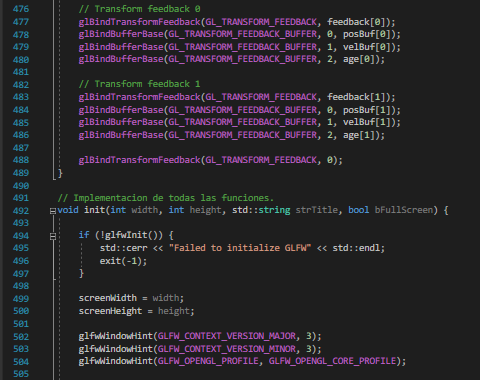


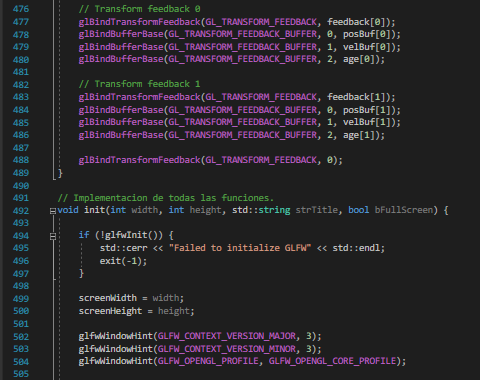
* Algunas de las funciones que se implementaron se mantuvieron tal y como estaban de prácticas anteriores, en específico aquellas que corresponden con la generación de partículas.

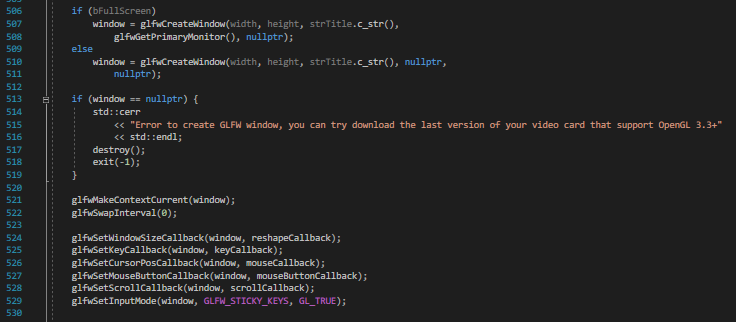
 

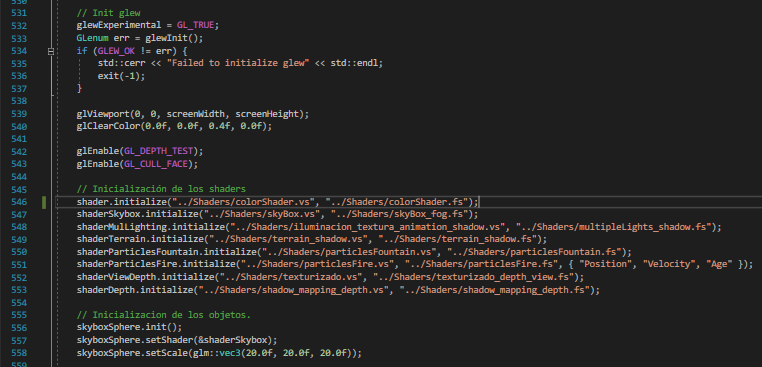
 

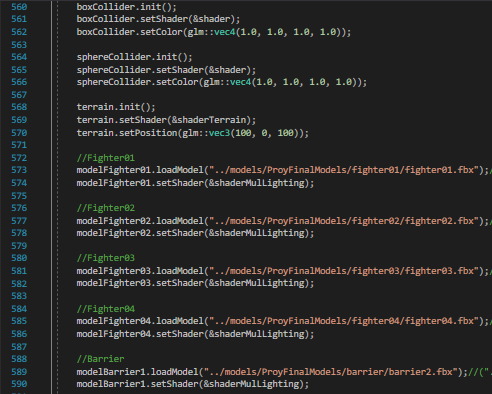
 

* En la función init se toma e inicializan los modelos.



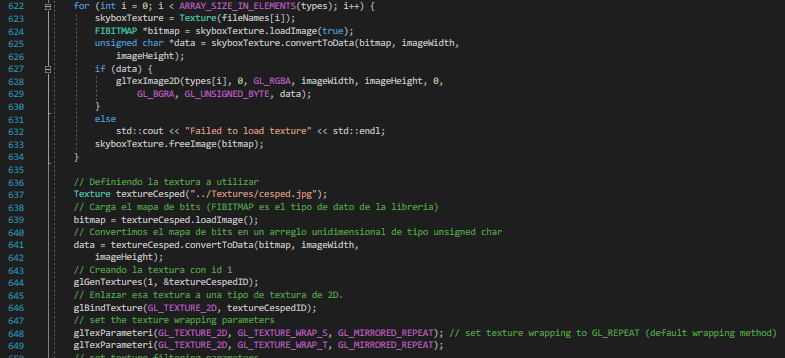




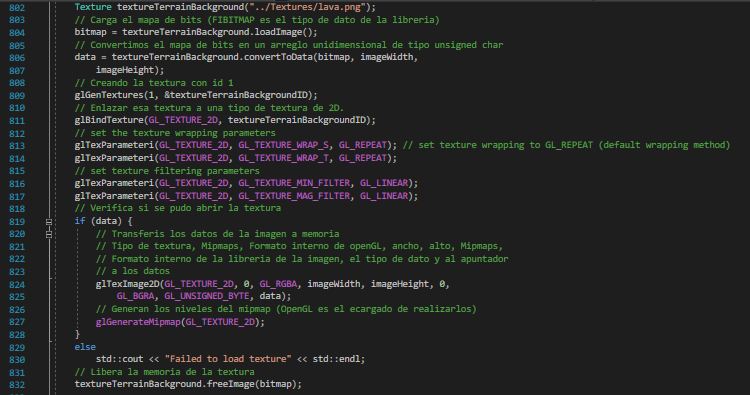
* Dentro de la misma función de inicio se encuentra la renderización del terreno, en este caso se muestra las imágenes utilizadas en cada caso.

Para el skybox.

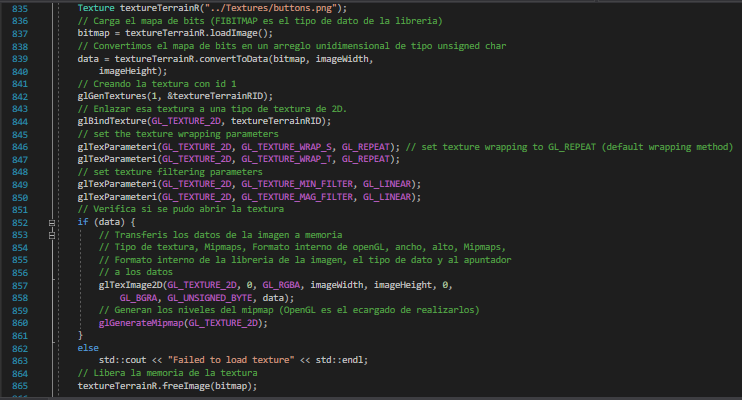
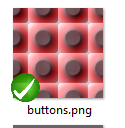




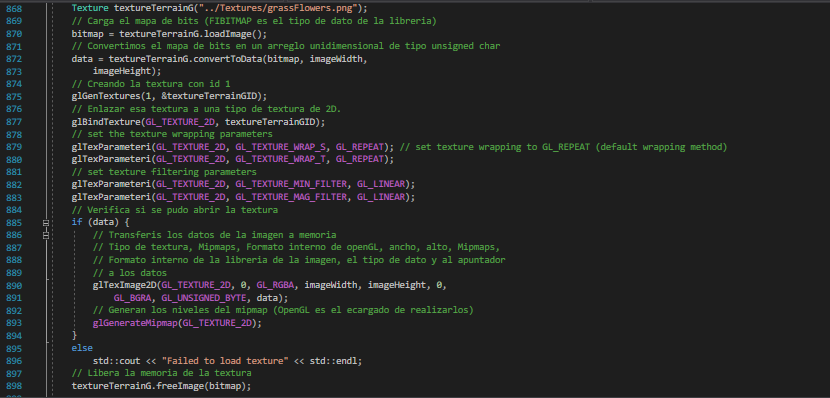
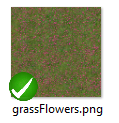
Para el terreno en su punto más bajo, en el blending map corresponde con la parte de color negro.

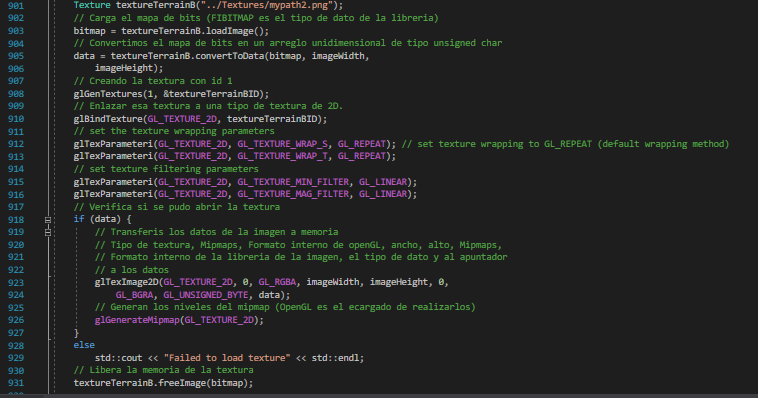
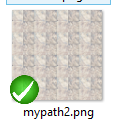
Para los bordes del camino, en el blending map es el área roja.

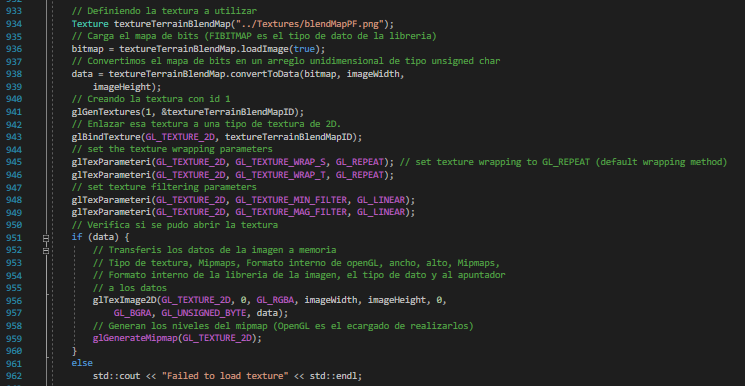
Para algunas partes en los bordes del camino y en el blending corresponde al color verde.

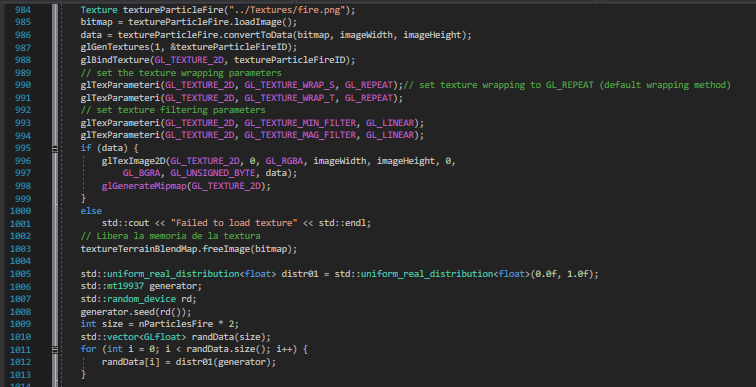
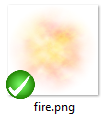
Para el camino, en el blending map corresponde con el área azul.

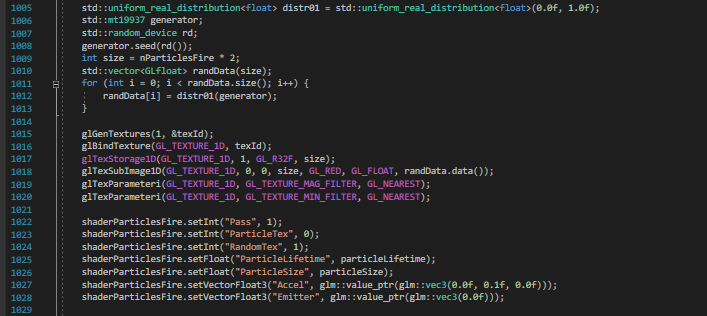
 

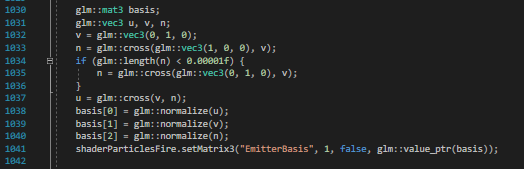
El mapa utilizado.

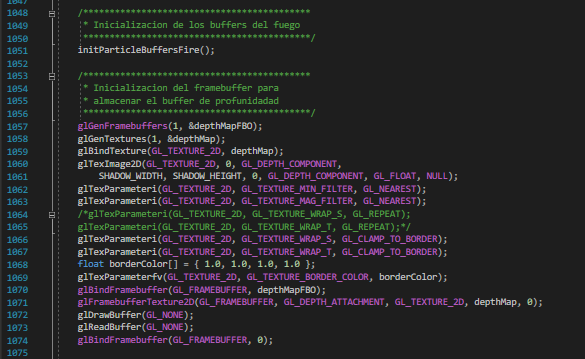
Partículas de fuego.

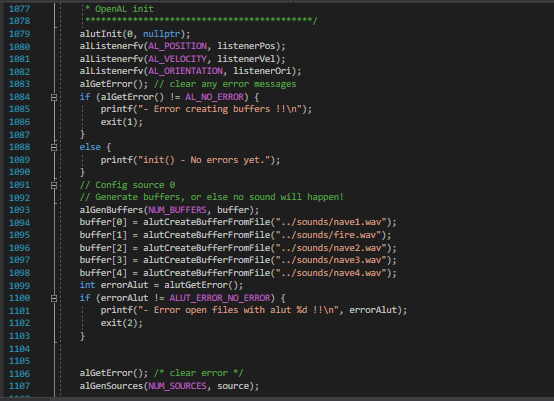
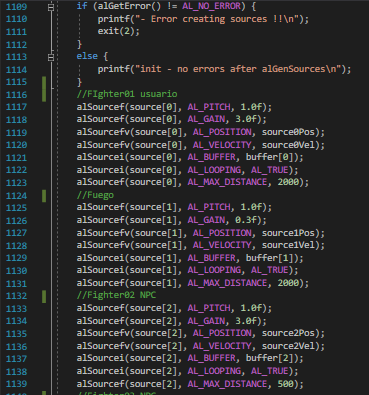


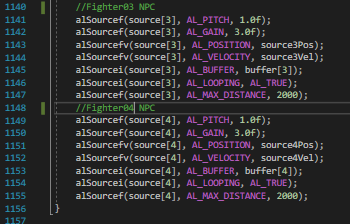


* En init inicialización de los buffers del fuego y profundidad.

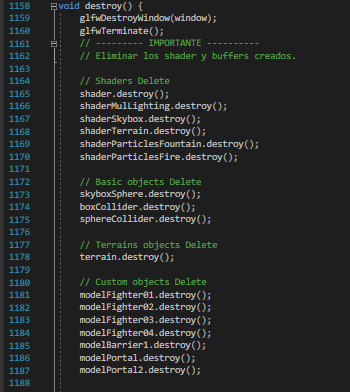
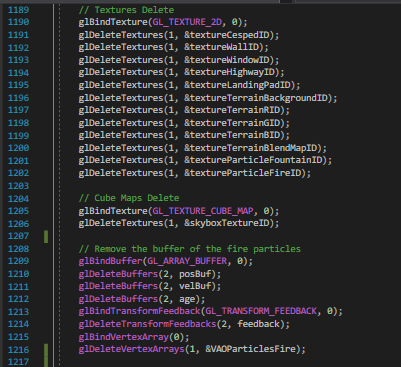


* En init algunas declaraciones para poder utilizar OpenAL. Entre ellas se encuentra la ruta de acceso a los sonidos que se usaron y la asociación de posición y velocidad al modelo o lugar al que fueron asignados.

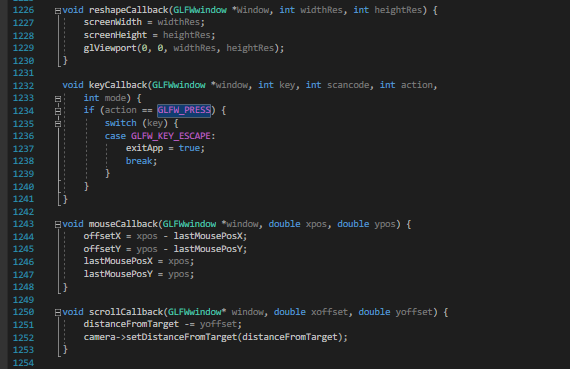
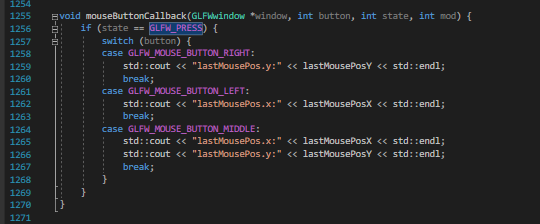
 



* La siguiente función corresponde con la liberación de memoria.

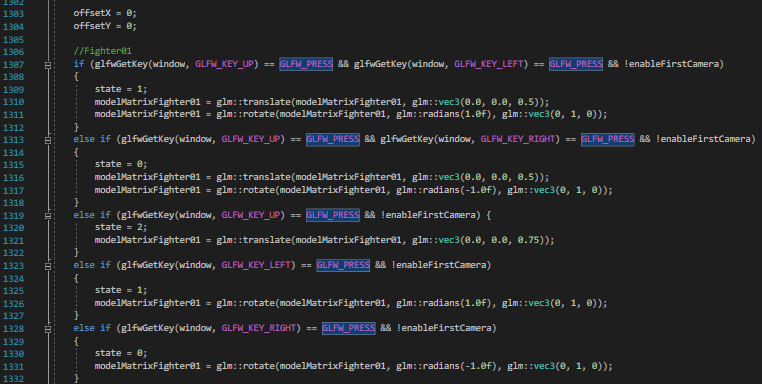
 

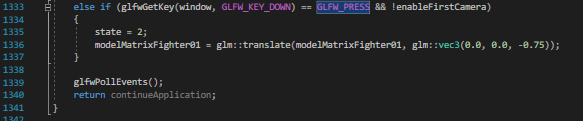
* Las siguientes funciones cumplen con la función de ajustar el tamaño de la pantalla y controlar eventos al usar el ratón y el teclado.

En el caso del teclado dependiendo del tipo de cámara que se utilice se podrá manipular el movimiento de esta o del modelo en tercera persona.





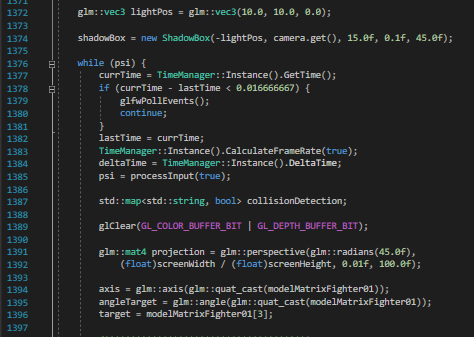


* En aplicationloop se manda a llamar otras funciones para el renderizado de modelos.

En esta primera imagen se posiciona la matriz de los modelos tanto del jugador como NPCs.

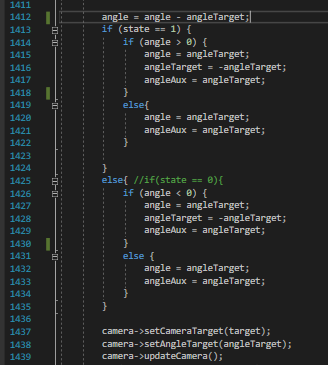
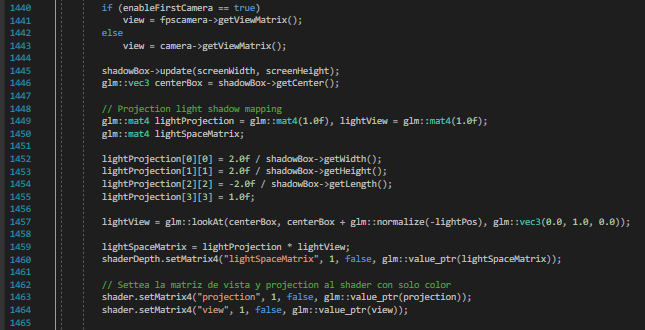


Siguiendo con otras variables para posicionar la luz y manipular los frames a los que corre el proyecto.

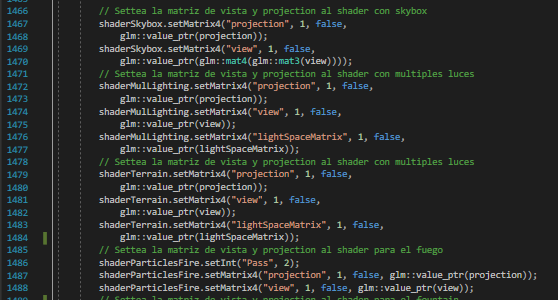


Se estableció una condición para controlar el giro en el ángulo de la cámara ya que en determinado momento al ir rotando sobre su mismo eje en la vista en tercera persona causaba un efecto que contraponía la vista del jugador respecto al modelo.

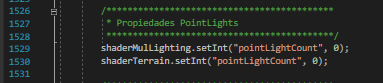
Aun así, no se pudo corregir por completo este problema y la solución que se tomo fue jugar con el escenario para saltar los puntos donde se produce este efecto implementando un portal en medio de la carrera.

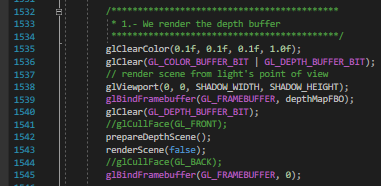
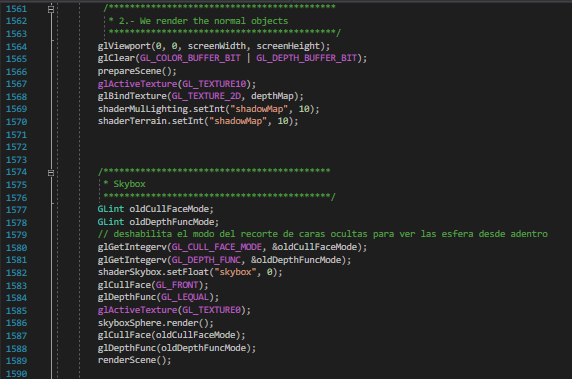
La siguiente sección fue para el setteó de matrices.



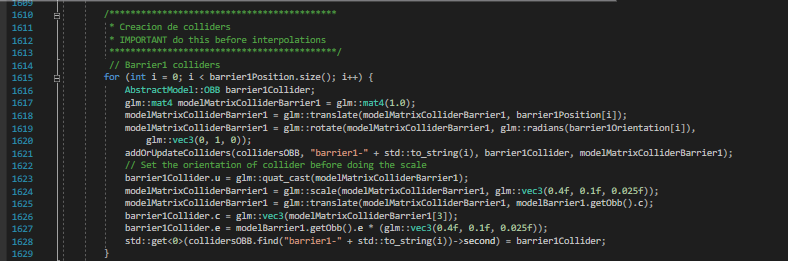
En esta fue para la asignación de parámetros para cada uno de los shaders.

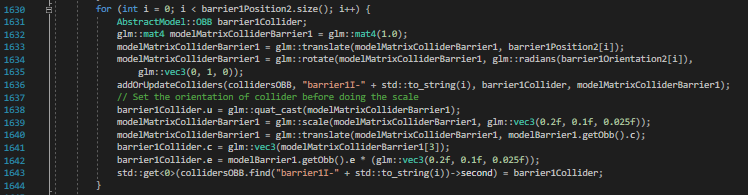
 

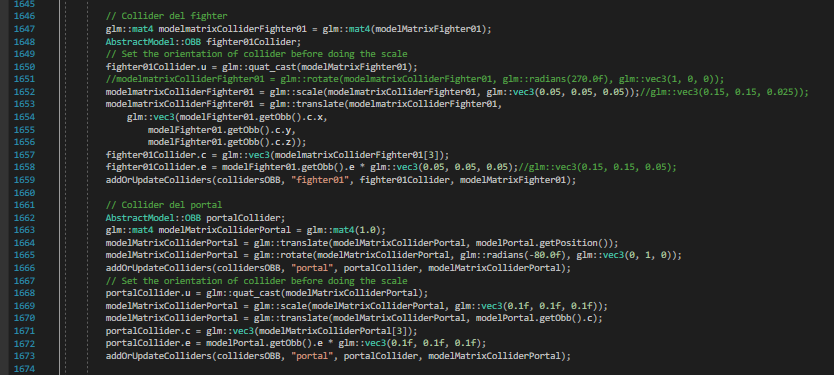
Posteriormente se encuentra la parte del código dedicada a renderizar el buffer de profundidad, el skybox y se manda a llamar la función de renderizado del resto de los modelos.

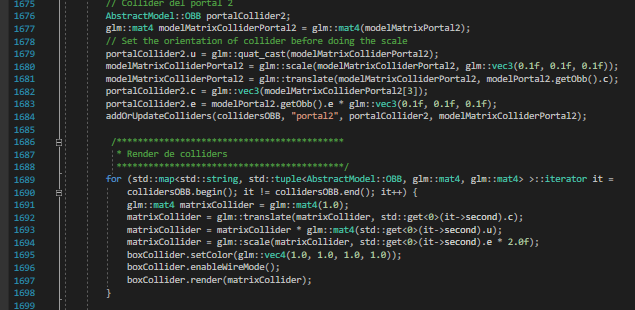
 

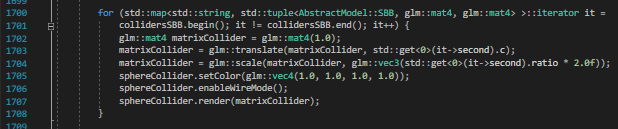
También se crean los colliders.

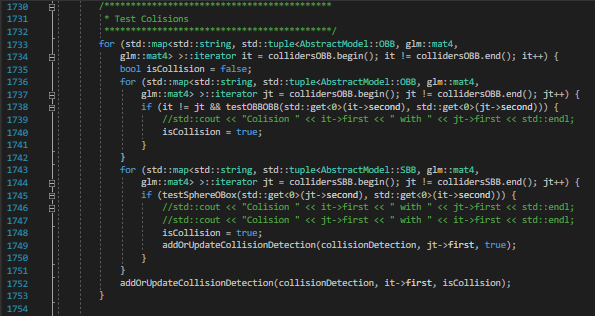


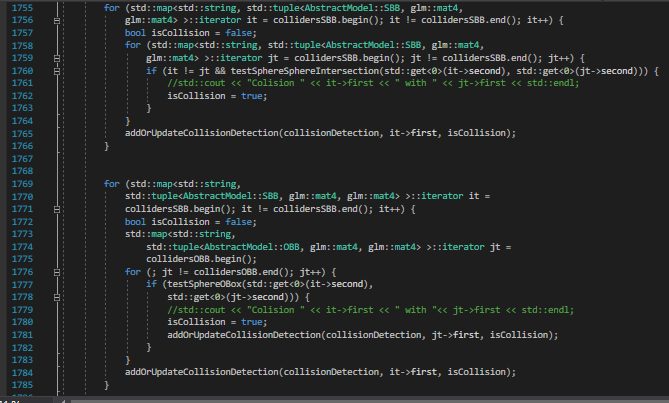


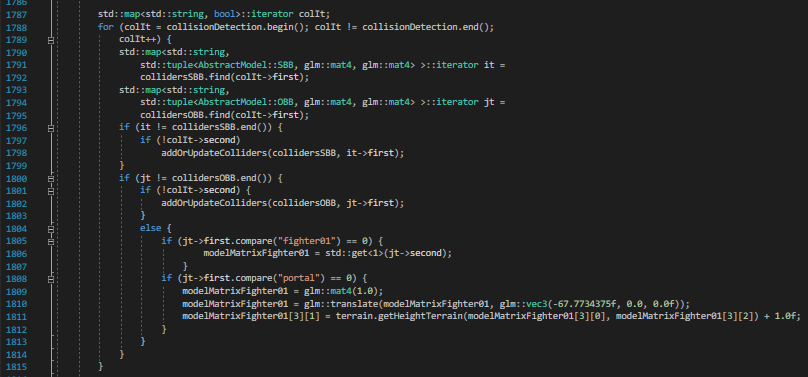






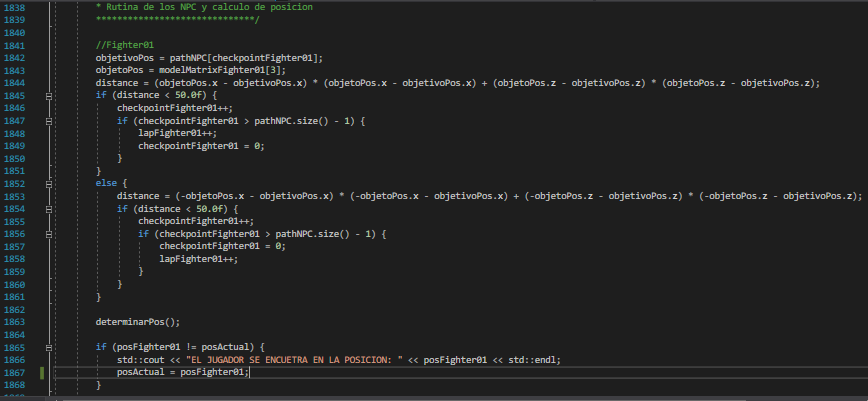


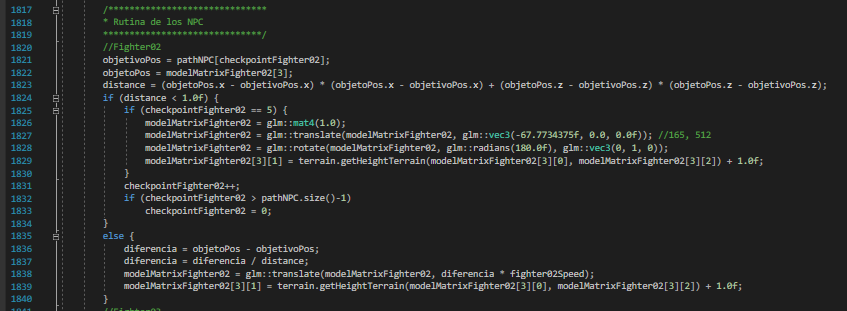


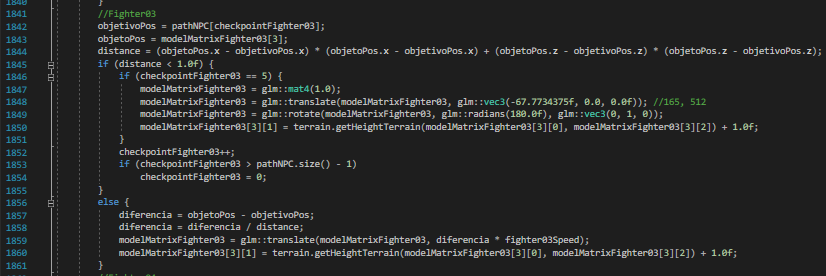


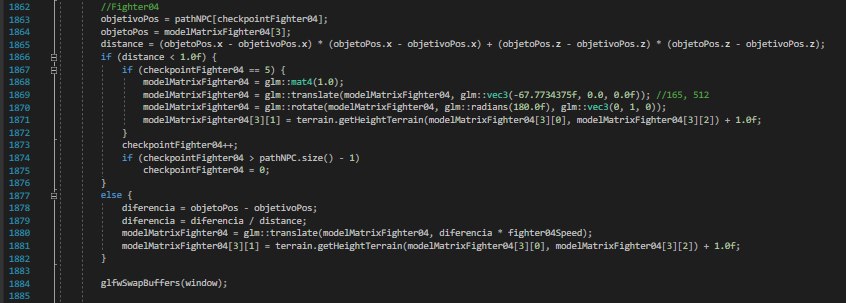
También se creó una “máquina de estados” para establecer la ruta que deben seguir los NPC sobre la pista, utilizando la distancia entre dos puntos se calcula una diferencia que sirve para identificar en qué momento cambian su dirección a otro punto.

Adicionalmente se usa para determinar en que posición se encuentra el jugador respecto a los NPC’s.







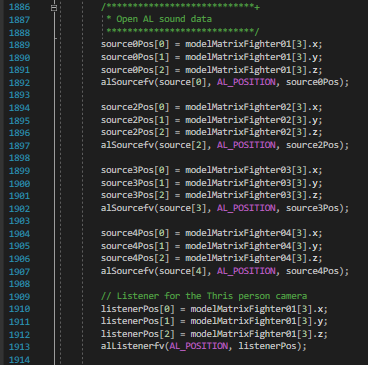
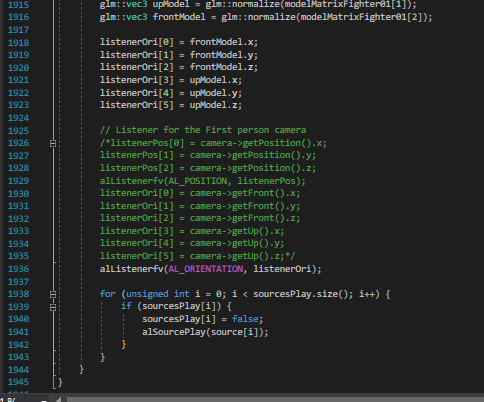


Como se puede ver en el caso del modelo que corresponde con la nave del jugador y a diferencia de las demás naves se deja un margen de error bastante alto entre la distancia que existe en un punto a otro porque el jugador no necesariamente tiene que pasar por el punto exacto, caso contrario a los NPC’s, sino “seguir la ruta” marcada y solo cuando lo hace avanza posiciones en la carrera.

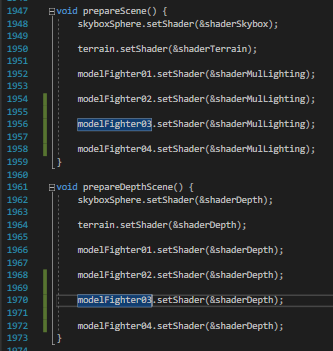
También debido al tiempo disponible para desarrollar, no se pudo corregir un detalle respecto al cálculo de distancia de un punto a otro para el jugador, por lo que se definieron dos ecuaciones para determinar si se encontraba cerca de un punto o no. Esto se produce ya que al momento de cruzar el portal la matriz del modelo se modifica para posicionarse adecuadamente.

Posteriormente se procede a calcular el lugar en el que se encuentra el jugador mandando a llamar una función que determina el marcador en base a una serie de condiciones sobre las variables de lapFighter y checkpointFighter.

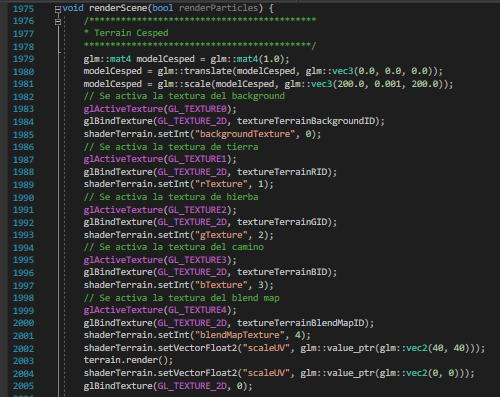
Por último, en la función de loop se encuentra la inicialización de los sonidos que se agregaron para ser utilizados con OpenAL, se consideró un sonido distinto para los NPCs, el jugador y el fuego.

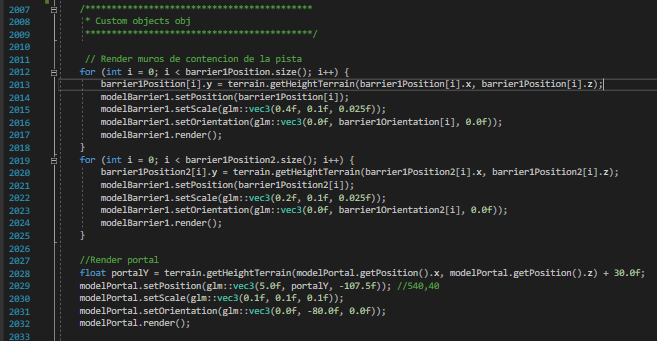
 

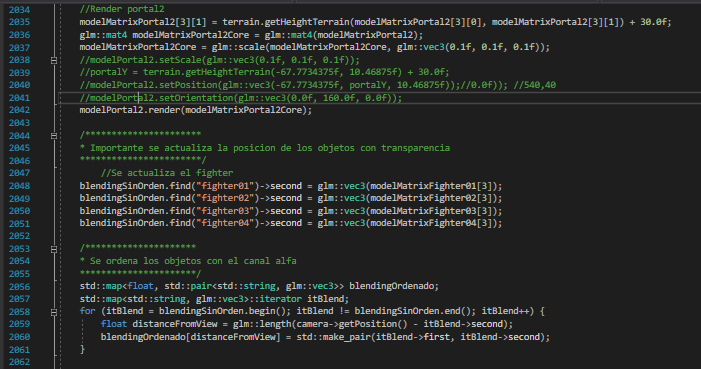
* Otras dos funciones importantes son las que se encargan de preparar la escena al asignar shader multilighting y profundidad de los modelos y terreno en para poder visualizar luces y sombras en ellos.



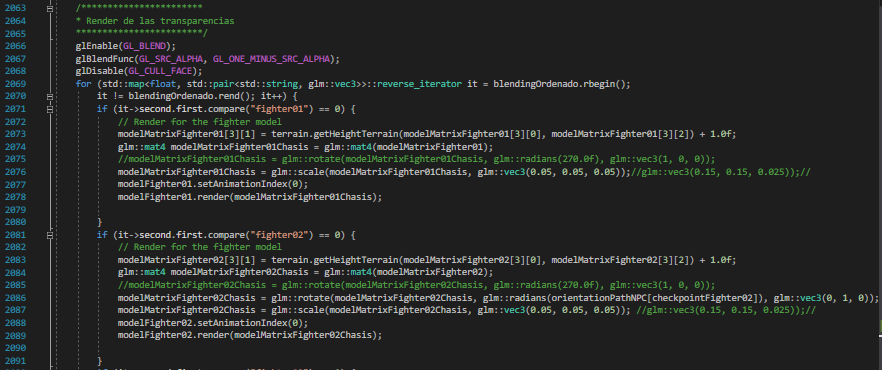
* En otro procedimiento se encuentra el renderizado de los modelos, que se manda a llamar desde la función de loop.

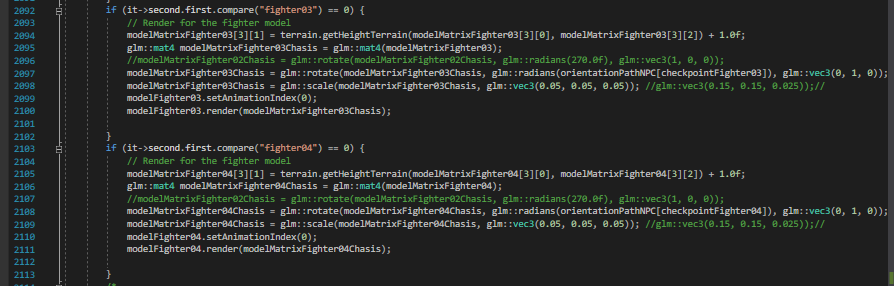




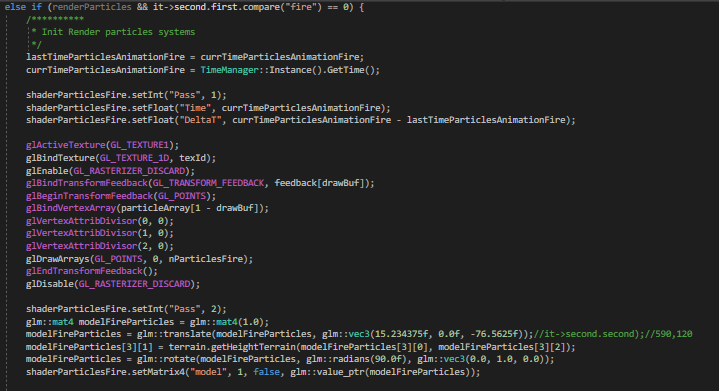


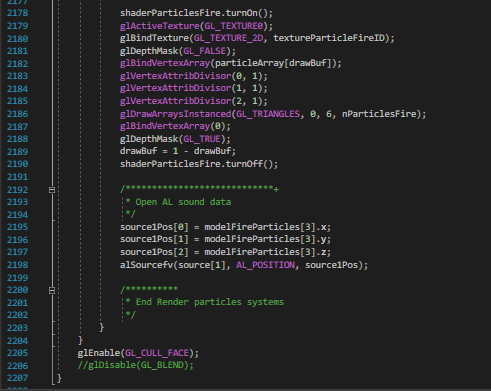
Las naves tienen transparencia en la ventana que tienen en frente.



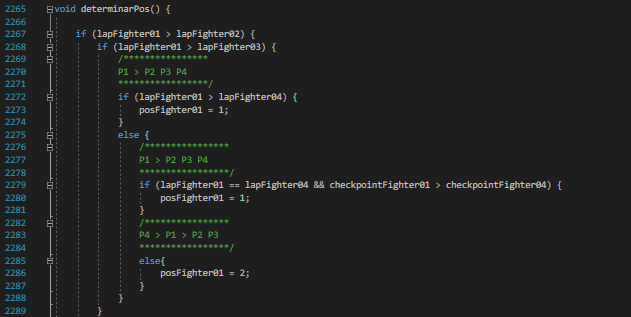


Se implementaron las partículas de fuego frente al portal.

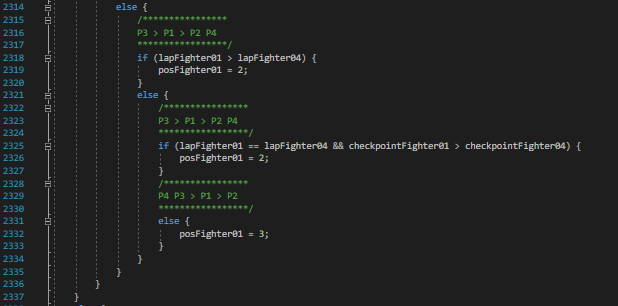




* Antes de llegar al main se encuentra la función para determinar en que lugar va el jugador respecto a sus competidores. Básicamente es una máquina de estados que dependiendo de los parámetros de vueltas por competidor y checkpoint marcado en la pista se encuentra la posición.

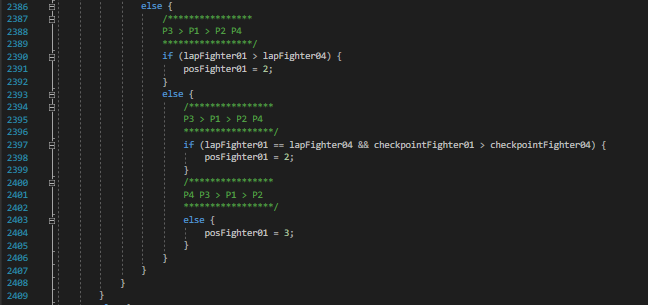






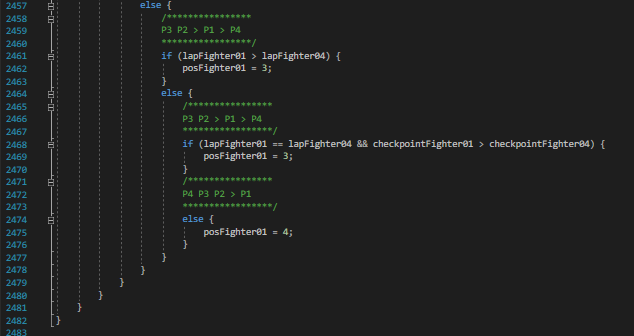




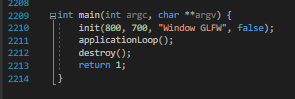








* Por último, se encuentra la función main que se encarga de iniciar toda la construcción del proyecto, también aquí se asigna el tamaño de la ventana y la liberación de memoria.



# Manual de usuario

El proyecto demo se trata de un videojuego de carreras de naves, los controles básicos son sencillos.

* Desplazamiento hacia el frente y hacia atrás con las respectivas teclas UP y DOWN del teclado.
* Rotación de la nave, con las teclas LEFT y RIGHT. Esta modalidad funciona en conjunto con la tecla UP, por lo que la nave puede avanzar e ir rotando, no es el mismo caso para la tecla DOWN que no tiene esta función implementada.
* La cámara se puede girar moviendo el ratón.
* Si se utiliza el scroll del ratón es posible alejar o aumentar la vista hacia el modelo de la nave del usuario.
* Existe una función de “PAUSA” que detiene a los NPC y los movimientos del jugador al presionar la tecla P, se reanuda el juego si se presiona nuevamente. Si se utiliza esta función se entrará en el modo de primera persona.
* En el modo en primera persona se tienen las mismas funciones de alejar y acercar la cámara y también girar, y se agrega la modalidad de desplazamiento por el escenario para poder apreciarlo utilizando las teclas direccionales UP, DOWN, LEFT, RIGHT.

# Costo del proyecto.

Tomando en consideración los siguientes factores: las clases en línea debido a la pandemia, horas de clase en el semestre, periodo de tiempo dedicado para el proyecto de 3 meses incluyendo clases y para determinar el costo por hora se consideran solo 26 días del mes con jornadas de 8 horas.



Se obtuvo un promedio de $26,700 por el desarrollo de este proyecto. Aunque podría llegar a $33375 si se considera una ganancia del 25%.

# Licenciamiento.

El estado actual del proyecto y el plan a futuro que se tiene implican que licenciar dicho trabajo no es necesario por lo que quedaría libre para que cualquier persona haga uso de él.

# Principales retos y dificultades, mejoras, trabajo a futuro

Se pueden mejorar en todos los aspectos con una buena inversión monetaria y de tiempo: visualmente, en sonido, diseño de niveles, mecánicas, motor o entorno de desarrollo.

En el aspecto visual, se pueden implementar mejores modelos, los que se utilizaron no venían texturizados ni animados, solo tuve tiempo para hacer el texturizado.

Al utilizar partículas se utilizan muchos recursos por lo que se utilizaron muy poco para mejorar el rendimiento del juego.

En el diseño de la pista de carreras no fue posible utilizar alguna prefabricada ya que estos modelos requieren de un collider mejor detallado que los que se vieron en clase que constaron de esferas y cajas.

Se utilizó mucho tiempo en determinar la ruta que siguen los NPC’s, por lo que se decidió utilizar un circuito con forma circular para ahorrar tiempo en determinar la posición de estos puntos.

En la parte de mecánicas uno de los problemas más significativos fue al momento de rotar la cámara junto con el modelo de la nave, ya que en determinado punto en el circuito la vista se antepone al modelo y en vez de mirar hacia el frente mira hacia atrás, aquí se hizo un poco de trampa saltando un tramo de la pista en donde se presenta este bug.

Aunque no hubo muchos problemas utilizando OpenGL, principalmente porque mucho del código fue reutilizado de las prácticas de la clase. En este sentido se puede mejorar utilizando un motor gráfico que sustituya a OpenGL.

# Conclusión

Con el resultado obtenido se pudo demostrar que un proyecto de este tipo es relativamente fácil de hacer, solo requiere más tiempo, gente experimentada para codificar y buenos diseñadores de audio, imagen y niveles para hacer que resalte al público en general.

Aunque no es tan probable que se pueda desarrollar un videojuego completo en un solo semestre, es posible hacer un demo.

Por otra parte, aunque no se pudieron implementar todos los conocimientos adquiridos durante el curso debido a la falta de tiempo y al nivel de procesamiento de la máquina con la que se contó al realizarlo, se utilizaron algunos elementos para hacer este demo.