1. Modo de juego e instrucciones

El objetivo del juego es conseguir la mayor puntuación posible sobreviviendo el mayor tiempo posible y destruyendo todos los asteroides y naves enemigas posibles al mismo tiempo que evitamos ser golpeados por los mismos o tocar los bordes de la pantalla, exceptuando el superior.

Los controles del juego son lo siguientes:

- Z: para inicar del juego en el menú
- X: para salir del juego en el el menú
- W: para desplazarte hacia arriba.
- A: para desplazarte hacia la izquierda
- S: para desplazarte hacia abajo.
- D: para desplazarte hacia la derecha.
- Espacio: para disparar y poder eliminar los asteroides.

2. Ficheros del código

Asteroid.h: declara las funciones necesarias para poder: generar, eliminar, desplazar y mostrar en pantalla los asteroides del juego, además de su propio struct con el que declaramos su gráfico y su velocidad.

Asteroid.cpp: crea las variables necesarias para poder definir las funciones declaradas en su respectivo .h, siendo: la generación individual de cada asteroide, limitando su tiempo de aparición y la cantidad máxima de los mismos, concretando su gráfico y la zona por la que podrán aparecer y las velocidades a las que se desplazarán, además de poder colisionar con otros elementos definidos (el jugador y los disparos del mismo), y ser desactivados una vez salgan de pantalla. El movimiento de los asteroides vendrá definido por la velocidad asignada de forma individual a cada uno pero siempre desplazándose por el eje Y. Los asteroides que salgan de la pantalla, dejarán de contarse y serán transportados a un punto aleatorio.

Stars.h: declara las funciones necesarias para poder: generar, desplazar y mostrar en pantalla las estrellas, además del struct con el que declaramos su gráfico, su color y su posición.

Stars.cpp: crea las variables necesarias para poder definir las funciones declaradas en su respectivo .h, siendo: la generación individual de cada estrella, limitando la cantidad máxima de las mismas, concretando su gráfico y su color y la zona por la que podrán aparecer de forma aleatoria.

Player.h: declara las funciones necesarias para poder: generar, desplazar, mostrar en pantalla y “matar” al jugador, y generar, eliminar, desplazar y mostrar el disparo que puede

realizar el jugador, además de los structs para el jugador y el disparo, con ellos declaramos sus gráficos, las velocidades de ambos, y el estado de muerte y las posibles colisiones por parte del jugador.

Player.cpp: crea las variables necesarias para poder definir las funciones declaradas en su respectivo .h, siendo: la generación del jugador, donde ajustamos dónde aparecerá, sus diferentes gráficos, la velocidad a la que se podrá desplazar, además de poder colisionar con otros elementos definidos (los asteroides). Los controles se asignan a unas teclas que definen su estado y así decidir qué hace cada tecla presionada. También definimos el rango que utilizamos para determinar si se sale de la pantalla y así eliminar al jugador. Este mismo archivo crea las variables necesarias para poder definir las funciones del disparo declaradas en el Player.h, siendo: la generación del disparo, donde ajustamos dónde aparecerá, su gráfico, la velocidad a la que se desplazará, la cantidad de disparos máximos en un tiempo fijado, además de poder colisionar con otros elementos definidos (los asteroides).

Enemies.h: declara las funciones necesarias para poder: generar, desplazar, mostrar en pantalla, y generar, eliminar, desplazar y mostrar los disparos que pueden realizar las naves enemigas, además de los structs para los enemigos y sus disparos, con ellos declaramos sus gráficos, las velocidades de ambos, y la vida y los puntos que proporcionan al ser eliminados.

Enemies.cpp: crea las variables necesarias para poder definir las funciones declaradas en su respectivo .h, siendo: la generación de los enemigos, donde ajustamos dónde aparecerán, sus diferentes gráficos, la velocidad a la que se podrá desplazar, además de poder colisionar con otros elementos definidos (los disparos del jugador). Este mismo archivo crea las variables necesarias para poder definir las funciones del disparo declaradas en el Enemies.h, siendo: la generación del disparo, donde ajustamos dónde aparecerá, su gráfico, la velocidad a la que se desplazará, la cantidad de disparos máximos en un tiempo fijado, además de poder colisionar con otros elementos definidos (el jugador).

Menu.h: declara las funciones necesarias para poder: mostrar y generar la pantalla de inicio, además de su propio struct, con el que declaramos su sprite y su lista de estados.

Menu.cpp: crea las variables necesarias para poder definir las funciones declaradas en su respectivo .h, siendo: la generación del menú, donde definimos el gráfico que utilizaremos para mostrarlo y donde concretamos su ubicación con otra función. También definimos sus estados, los cuales son asignados a diferentes teclas para poder acceder a ellos, los cuales nos permitirán jugar al juego o salir de él.

Engine.h: declara las funciones necesarias para poder: iniciar, finalizar y cerrar el juego, además de su propio struct, con el que declaramos el tamaño del mundo, el punto central y la zona de aparición de enemigos.

Engine.cpp: crea las variables necesarias para poder definir las funciones declaradas en su respectivo .h, siendo: el proceso que sigue el funcionamiento del juego y la finalización del mismo. Calcula el punto central de la pantalla y la zona de aparición de los asteroides y

naves enemigas. También define el sistema de colisión entre las diferentes entidades del juego.

game.cpp: contiene el bloque del programa, el cual llama la función que ejecuta todo el juego y la función que declara su final.

3. Estructuras principales y para que sirven

Uno de los structs más importantes del programa es el del jugador. En él declaramos los diferentes gráficos utilizados (cuando se desplaza hacia arriba, cuando se desplaza hacia abajo y cuando se desplaza hacia los laterales, está quieto o dispara), también incluye las variables para declarar la velocidad normal y su velocidad al desplazarse en forma diagonal, otras variables añadidas son su estado de muerte, y sus tres posibles factores de muerte: salir fuera los límites del mundo y colisionar con un asteroide o un enemigo. Finalmente incluimos los diferentes estados del jugador, siendo sus ocho posibles direcciones de movimiento, mantenerse quieto en el sitio, disparar y morir.

Otro struct importante es el del propio motor del juego. En él definimos el tamaño que tendrá el mundo del juego, el punto central del mismo con el utilizamos para posicionar al jugador cuando empieza el juego, y la zona de aparición de enemigos, también incluye la variable que define si el juego está cerrado o no.

4. Flujo del programa