

Reporte de documentación: Space Wars!

CC3501-1 Modelación y Computación Gráfica para Ingenieros

Sebastián Olmos Hernández, 9 de mayo de 2020

La arquitectura del programa se puede ver simplificado en el diagrama de la Figura 1, donde al inicio se prepara e inicializa el programa en el *setup()*, aquí se cargan las imágenes y se crea el grafo de escena que se ve en la Figura 2 creando las referencias a los nodos principales. Luego el programa se actualiza en el *Update()* forzando a que se actualiza con los *FPS* indicados para una mayor estabilidad, aquí se actualizan todos los objetos y sus estados (lo que vendría siendo el modelo), a la vez que recibe el input del usuario (controlador) y por ultimo dibuja toda la escena (vista).

Cabe señalar que en el juego hay tres tipos de enemigos, los cuales aparecen en grupos de 5 a 7, donde cada grupo aparece arriba de la pantalla y se trasladan hasta quedar en la parte superior de la pantalla, donde realizan su movimiento y disparan. Solamente aparecerá otro grupo de enemigos cuando se elimine a todos las naves del grupo actual, esto para darle tiempo al jugador de eliminar a los enemigos. El primer tipo de enemigo (rojo) dispara constantemente mientras realizar un pequeño movimiento diagonal, teniendo un pequeño lapso de tiempo donde no dispara (aprovechar este tiempo para eliminarlo, notar que este espacio donde no se dispara se va desplazando hacia los lados). El segundo enemigo (naranja) se mueve en un rectángulo mientras dispara aleatoriamente con un tiempo de recarga entre posible disparo. El ultimo enemigo (rosa) gira en un ovalo, mientras dispara aleatoriamente, notar que cada vez que se elimina un enemigo de este grupo, el resto gira más rápido.

Para organizar mejor el código, se tienen varios archivos *.py* para tener separados las distintas tipos y clases que realizan determinada función.

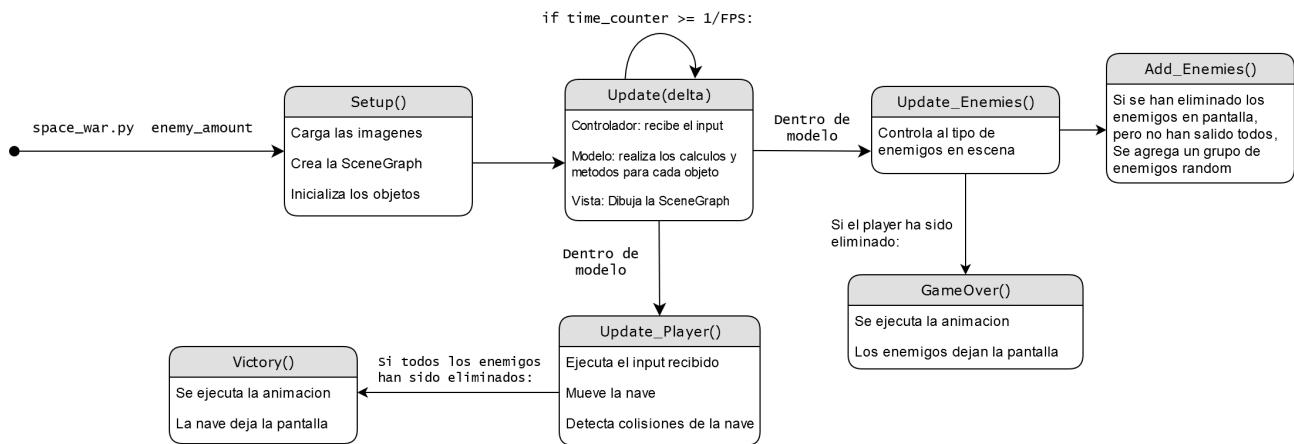


Figura 1: Diagrama de flujo.

En la Figura 2 se puede observar el grafo de escena que se crea en el *setup* y se actualiza en el *update*, como se observa, se dividen en tres grupos, el primero son los elementos de fondo que no interactúan, los cuales son objetos con texturas o animaciones que se desplazan formando capas con distintas velocidades para dar una sensación de profundidad, además dichos elementos se generan aleatoriamente para generar un fondo infinito y que no se sienta repetitivo.

Luego están los elementos con colisiones que interactúan entre ellos. Por ultimo se tiene a los elementos de interfaz, que son los corazones, indicando la vida que le queda a la nave del jugador.

En la Figura 3 se puede apreciar la estructura/grafó de escena que tienen los elementos con colisiones, donde su nodo principal es un objeto de clase `gameObject` u otra heredada de esta. Su primer hijo corresponde a una animación o una imagen, mientras su segundo hijo a la clase que guarda la información y métodos para detectar colisiones, a la vez que contiene una imagen por si se quiere apreciar su hitbox.

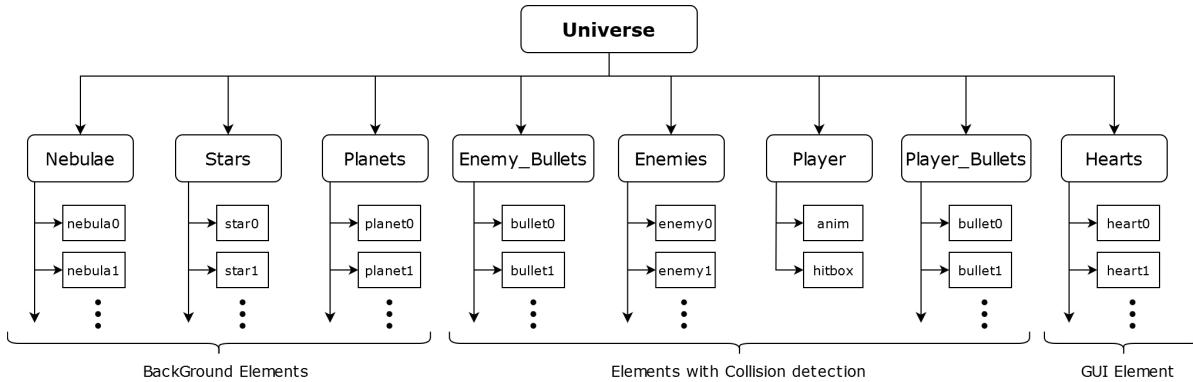


Figura 2: Grafo de escena.

Para las animaciones se usaron las funciones y variables que hice para la decima 6, y para las colisiones, se detectan mediante la verificación de si chocan círculos centrados en los objetos

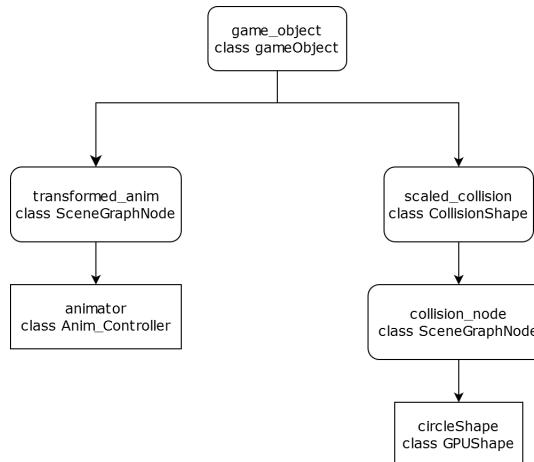


Figura 3: Estructura de los objetos que colisionan

Instrucciones de Ejecución

El programa se inicia con el comando `python space-war.py enemies_amount` donde `enemies_amount` es el número con la cantidad de enemigos a derrotar. Luego los controles de dirección son las teclas *W* (avanzar hacia arriba), *S* (abajo), *A* (izquierda) y *D* (derecha), estos botones dirigen a la nave con movimiento suave (lento al principio y final del movimiento) para darle una sensación de peso y de conservación del movimiento que hay en el espacio (por lo que para frenar rápido hay que apretar la dirección contraria).

Para el disparo, se realiza con la barra espaciadora. Los disparos tendrán un tiempo de recarga entre estos (para no spamear disparos) y si desea un disparo continuo puede mantener la barra espaciadora apretada.

Cuenta con la cantidad de vida en forma de corazones azules en la parte inferior izquierda para una mejor control de las vidas, si estas desaparecen, morirá, mostrando la animación de muerte (que incluye una modificación al fragment shader para darle un transición a escala de grises a la escena) y las naves enemigas seguirán su camino, saliendo por la parte inferior de la pantalla. De lo contrario, si elimina todos los enemigos, se mostrará una animación de victoria y su nave saldrá por la parte superior de la pantalla.

Además se incluyeron las teclas tabulador para poder ver las hitboxes de los objetos y la tecla *P* para pausar el juego.

Resultados

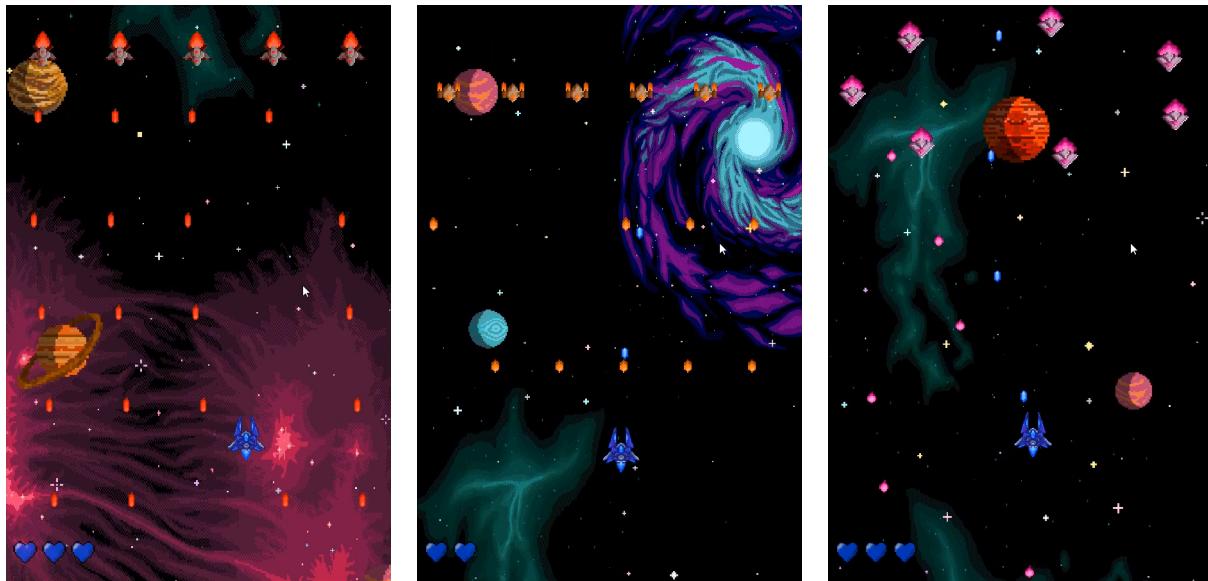


Figura 4: Tres diferentes tipos de naves enemigas.

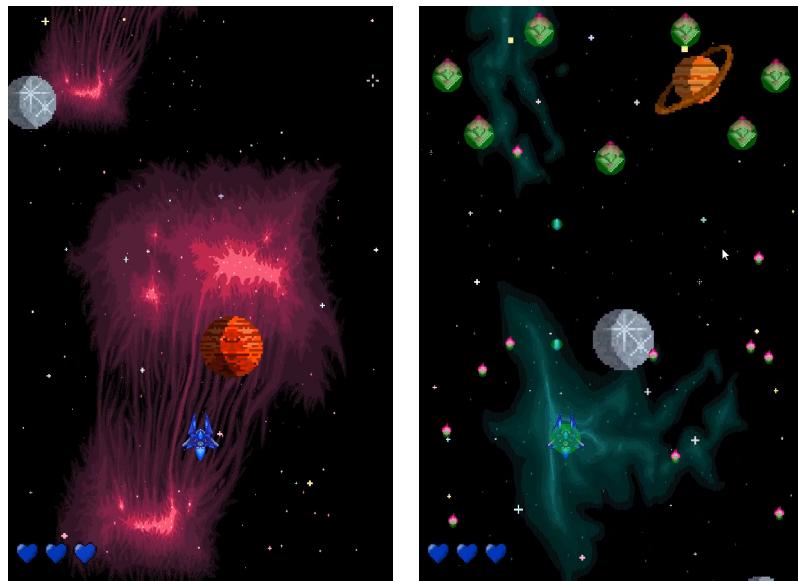


Figura 5: fondo aleatorio y vista de las hitboxes.

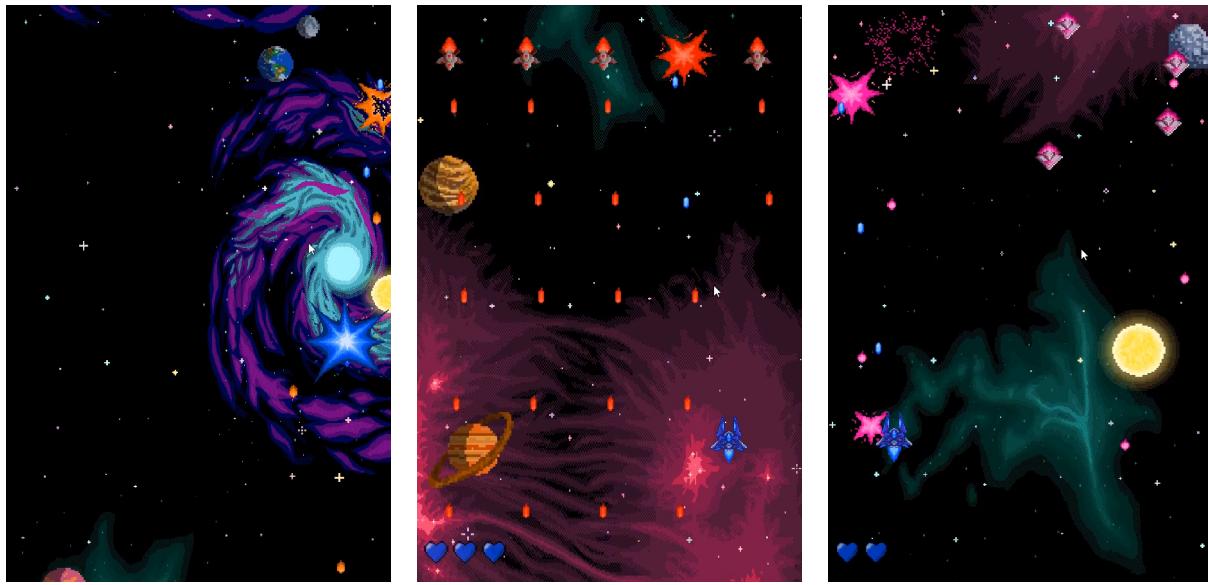


Figura 6: Explosiones de la nave, enemigo y de una bala respectivamente.

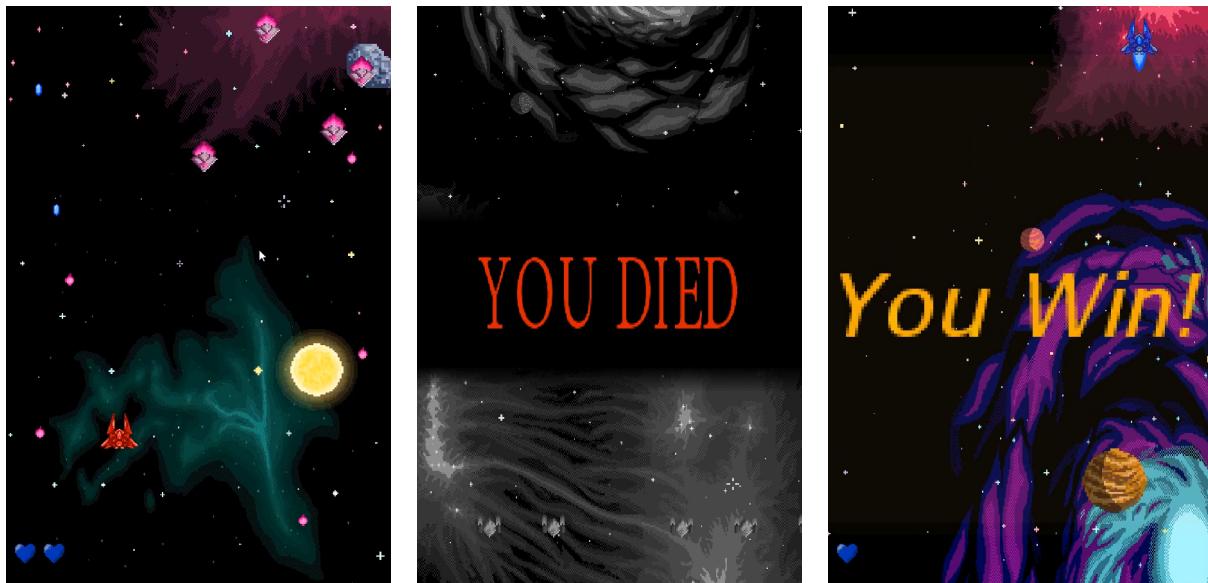


Figura 7: Animaciones de nave herida, game over y victory respectivamente