Proyecto SNASA

Christian Capeáns Pérez

Proyecto de Computación Gráfica
christian.capeans@rai.usc.es

I. Introducción

Este proyecto ha sido realizado haciendo uso de la versión de openGL 3.3 y GLEW en OSX, con el editor de código XCode. Al iniciar la aplicación podemos ver una nave en tercera persona en un sistema solar remoto. Si nos movemos con los controles [W, S, D, A] podemos apreciar cómo los planetas rotan en torno a una estrella gigante. Estos planetas rotan en direcciones, velocidades y ángulos completamente distintos y es importante no interponernos en su trayectoria, ya que si nos golpean tendremos que volver a empezar. Pronto nos daremos cuenta de que hay una ciudad gigante que tendremos que proteger de las naves que se dirigen a ella. Estas naves enemigas pueden teletransportarse a pequeñas distancias para esquivar nuestras balas (que podremos disparar pulsando la tecla Enter). Además, nuestra nave dispone de un sistema de control de velocidad que nos permite movernos más rápido o más lento (pulsando la tecla espacio se va modificando la velocidad).
Tenemos que tener en cuenta que no podemos acercarnos demasiado al sol, ya que si lo hacemos la nave se fundiría y perderíamos la partida.

II. Cámara

Podemos ver nuestra nave espacial gracias a una cámara en tercera persona. Esta nave está situada en la parte trasera de la nave y un poco elevada en el eje Y. Apunta siempre a dónde está mirando la nave. Por otro lado, si pulsamos la tecla C, cambiamos a una cámara que está libre en la escena, esta la podemos controlar con las flechas del teclado y el ratón. Si queremos volver a ver la nave, simplemente volvemos a pulsar la tecla C. Esta última funcionalidad la puse después de la revisión porque no quería quedarme corto en este aspecto. Para realizarla tengo una clase cámara que gestiona las rotaciones, la sensibilidad de las teclas y la velocidad del movimiento.

III. Texturas

Para cargar las texturas cree una clase y así evito la duplicidad del código, ya que siempre es hacer lo mismo, como mis texturas se componen todas por imágenes en jpg, no incluí ningún método para cargar texturas con transparencias. Las texturas están situadas en *res/TEXTURAS*. Para las balas, que son muchos objetos instanciados en la escena, pero con una misma textura, cargo únicamente una y utilizo el identificador que genera con todas sus instancias. También cabe destacar que uso esta clase cuando estoy cargando modelos obj, cosa que comento en la sección siguiente.

IV. Modelos obj

Para poder crear un juego más elaborado, decidí utilizar la librería Assimp y así poder importar modelos .obj ya creados. Para ello cree las clases Model y Mesh. De esta forma solo tenia que instanciar un objeto Model en la escena y pasarle mediante el constructor la ruta a donde se encontraba el archivo .obj. Las naves espaciales están creadas a partir de Modelos obj y tienen sus propios shaders, tanto fragment como vertex. La ciudad también tiene su propio shader, ya que en vez de afectarle la luz que emite la estrella del Sistema Solar, cree una *spot light* en la nave que manejas que apunta a la ciudad, para que actúe como un foco de la nave.

V. Luces

Las luces que incluí son las del sistema solar, cuya estrella central emite luz y afecta a todos los modelos (excepto a la ciudad estelar), la luz focal de la nave y las balas, que también emiten luz. Para ello creé diferentes shaders, uno para la ciudad, otro para las balas, otro para los planetas y otro para las naves espaciales. Están incluidos los tres tipos de luces: ambiente, difusa y especular.

VI. movimiento

El movimiento que hay en el juego es, por un lado, el de los planetas que giran alrededor de la estrella gigante en ángulos y direcciones arbitrarias. Cabe destacar que no utilicé prácticamente nada (más allá de lo obvio) del sistema solar realizado en las prácticas de clase. Creé una clase para los astros y varias funciones que gestionan la carga de texturas, tamaño y movimiento de los planetas. Por otro lado está el movimiento de la nave, que se mueve hacia delante automáticamente a diferentes velocidades (seleccionadas por el usuario), hacia los lados, hacia arriba y abajo. Después está el movimiento de las balas que siguen la dirección a donde apunta la nave espacial y finalmente el movimiento de las naves enemigas, que se dirigen hacia la ciudad estelar, pero multipliqué el vector que utilizo para avanzar por un vector unitario con signos aleatorios para que su movimiento no sea predecible; creando así un efecto de tele transporte.

VII. Colisiones

Para mejorar la experiencia de juego, creé funciones de detección de colisiones en las clases en las que la necesitaba (en todas funciona con detección esfera a esfera). Metí detección de colisiones nave-planeta (solo afecta a la nave del jugador), bala-naves enemigas y naves enemigas-ciudad estelar. Para comprobar si las balas están chocando con algo, cada vez que disparo meto la bala en un vector de Misiles y en un lazo compruebo si se está detectando colisión en alguna componente de dicho vector. Hago lo mismo con los planetas. Tampoco permito instanciar balas infinitas, si no que limito el número de misiles que hay en la escena. Para ello calculo la distancia que hay entre la nave y la bala y cuando pasa de determinado valor la elimino del vector, por lo tanto, ya no se dibuja.

VIII. Otros

Cabe destacar que para cuando la nave se acerca mucho al sol, implementé un "sensor de temperatura" que provoca que la nave se "funda" y la partida se pierda. La temperatura de la nave se muestra cada cierto tiempo por la terminal gracias a un temporizador implementado con el tiempo entre frames. También se acumulan puntos cada vez que se elimina una nave espacial.

Por último, incluí un *cubemap* o *skybox* para representar el espacio. Tiene su propio shader y utilizo imágenes en .tga para realizarlo. El resultado es que, empleando un cubo, obtenemos la sensación de profundidad y de infinitud.

También es importante mencionar que hago uso de la librería SOIL para cargar texturas.