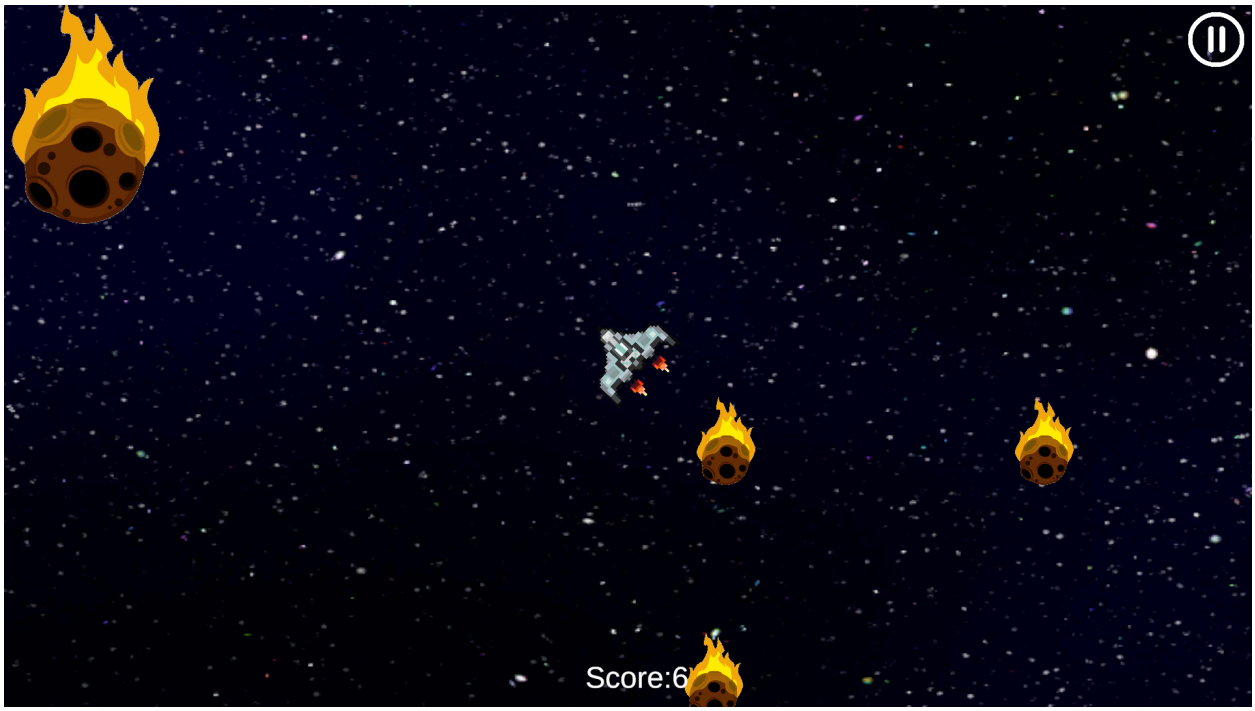
ASTEROID

Proyecto de minijuego para Fundamentos de los Videojuegos



Daniel F[co] Lanzas Larrañeta

Apartado gráfico

Escenas

Dado que no se ha requerido el uso de múltiples escenas para completar este proyecto, hemos utilizado únicamente la escena por defecto que proporciona Unity (*SampleScene*).

Assets

Un *asset* es cualquier elemento que puede ser utilizado dentro de un proyecto de Unity, para simplificar vamos a explicar cada tipo de los utilizados en el proyecto por separado.

Sprites

Los *sprites* son objetos gráficos en 2D, esto abarca desde texturas hasta el propio fondo del juego. Los utilizados en este proyecto son:

Bullet: sprite utilizado para las balas.	*Meteor*: sprite usado para los asteroides.	*Space*: fondo del videojuego.	*Ship*: nave que manejamos a lo largo del videojuego.	*boton-de-pausa-de-video*: usado para botones e imágenes de pausa. [1]
				

Prefabs

Los *prefabs* son plantillas utilizadas para crear instancias de un mismo objeto de Unity. Para Asteroid hemos requerido 3 *prefabs*:

- *Bullet*: *prefab* usado para crear balas.
- *Meteor*: *prefab* usado para crear asteroides.
- *SmallMeteor*: *prefab* usado para crear asteroides pequeños.

Apartado jugable

Mecánicas

Asteroid es un juego simple con cuatro mecánicas principales, estas son mover y rotar la nave, disparar a meteoritos, división de meteoritos al impactar con balas y *object pooling*.

El movimiento y rotación de la nave se han implementado de manera diferente cada uno, mientras que el movimiento se ha implementado utilizando un componente *RigidBody* aplicando fuerzas, la rotación se ha implementado modificando la componente *transform* de la nave.

Para la mecánica de disparo se ha optado por usar un *trigger* en vez de una colisión de manera que evitamos interacciones no deseadas con la nave pero mantenemos la sensación de colisión con los meteoritos.

Los meteoritos tienen únicamente un componente *RigidBody* desde el cual podemos aplicarles gravedad. Para la división, al colisionar con una bala, aplicamos fuerzas en direcciones opuestas a dos objetos de tipo *SmallMeteor* en sus componentes *RigidBody* y les proporcionamos la posición del asteroide original.

El *object pooling* se ha implementado siguiendo un patrón *Singleton*. Las *pools* de las que se extraen los objetos necesarios son colas de estilo *FiFo* para evitar que tomemos un mismo objeto si necesitamos dos al mismo tiempo. El patrón *Singleton* nos permite acceder a estas colas de manera sencilla desde cualquier *script* al haber una única instancia de ellas.

Scripts

Nuestro proyecto está dividido en seis *scripts*:

- *Player.cs*: desde este *script* implementamos las colisiones con meteoritos, el disparo y el movimiento del jugador que incluye: aceleración y frenado, rotación, evitamos que el jugador y las balas no salgan fuera de los límites de la pantalla
- *Bullet.cs*: desde este *script* manejamos las colisiones de las balas con los asteroides y las actualizaciones pertinentes de la puntuación del jugador.
- *MeteorSpawn.cs*: desde este *script* manejamos el *spawn* de meteoritos y que no salgan de los límites de la pantalla.
- *MeteorDivide.cs*: este *script* es usado para manejar la división de asteroides.
- *PauseScript.cs*: *script* creado para manejar el menú de pausa.
- *ObjectPooling.cs*: desde este *script* hemos implementado *object pooling*.

Interacción

El método de entrada de Asteroid es el teclado, las asignaciones de teclas se pueden observar en la tabla siguiente:

Tecla	Acción
w | Flecha_arriba	Acelerar
a | Flecha_izquierda	Girar en sentido antihorario
s | Flecha_derecha	Girar en sentido horario
d | Flecha_abajo	Frenar
Espacio	Disparar
Esc	Menú de pausa

Además de mediante la asignación de teclas se puede acceder al menú de pausa mediante el ratón.

Desarrollo

El desarrollo del proyecto ha estado dividido en 3 partes. Comenzamos con el desarrollo de los apartados obligatorios del proyecto (movimiento de la nave, disparo, y *spawn* de meteoritos) así como del menú de pausa, esta primera parte no resultó demasiado complicada pudiendo completarse en el primer día de desarrollo.

La segunda parte a la que nos enfrentamos fue el diseño del *object pooling* que aplicamos tanto a las balas como a los dos tipos de asteroides que aparecen en el juego. La implementación del *object pooling* costó más de lo esperado. Primero fue necesario documentarse acerca de este patrón *software*, el primer acercamiento no resultó satisfactorio al no entender correctamente el funcionamiento del patrón *Singleton* que se quería utilizar lo que conllevó errores como añadir todos los objetos a la misma *pool*. Sin embargo; al final se ha conseguido implementar de forma satisfactoria.

El último punto desarrollado fue la división de meteoritos al colisionar con las balas. En este punto, con un mayor conocimiento de unity, se realizaron cambios como refactorización de código o cambios en el funcionamiento del *object pooling*. Este apartado si bien no fue el más complicado no estuvo exento de dificultades tales como coger dos veces el mismo objeto.

A pesar de todo se ha conseguido un juego con todos los aspectos solicitados, tanto obligatorios como opcionales, por lo que el grado de satisfacción con el resultado es bueno.

Repositorio

El repositorio de github del proyecto es: https://github.com/dflanzs/minijuego/tree/dev

Otros enlaces

[1] https://www.flaticon.es/icono-gratis/boton-de-pausa-de-video_16427