Práctica Unity Motores

Roselló Morell, Sergio
sergio.rosello@live.u-tad.com

April 16, 2018

Contents

1 Inspiración e idea generál

El juego en el que me he decidido inspirar es: 1942, un juego desarrollado por Capcom para máquinas recreativas en 1984. Mi juego debería ser capaz de tener un loop completo, en el que fuese posible jugar como en las máquinas recreativas. El jugador solamente va a jugar en una escena con scroll semi-infinito. La máxima puntuación se mantiene entre partidas, de forma que un jugador puede ver si ha superado su máximo o no. Al final de las oleadas de aviones pequeñas, va a llegar la oleada del boss, en la que simplemente aparecerá el boss en la pantalla contra el jugador.

2 Estructura de código

El código está planteado de la forma más sencilla posible, en la que solamente existe lo esencial para que funcione. A continuación explicaré los distintos scripts usados y por qué los he planteado así.

3 Controladores de movimiento y comportamiento

3.1 BehaviourController

Este script es la base de cualquier actor, ya sea enemigo o personaje principal. Se encarga de actualizar la velocidad del objeto, gestionar su vida y adquirir los componentes necesarios para el correcto funcionamiento de las clases hijas (Enemigos y Actor principal).

3.1.1 Enemigos

Los enemigos adquieren el comportamiento de BahaviourController. Una opción a valorar era que el enemigo básico (BasicEnemyController) tuviese los comportamientos añadidos a BehaviourController para los enemigos y que los enemigos más complejos extendiesen de este pero finalmente decidí que, al ser enemigos básicos no tenía sentido hacerlo por tanto, cada enemigo hereda directamente de BehaviourController, luego ya añado las características de cada enemigo.
Actualmente tengo dos enemigos:

- **BasicEnemyController**
  Aparece de arriba de la pantalla y llega hasta abajo para dar la vuelta y salir por donde ha aparecido.
- **SteerEnemy**
  Aparece en una posición aleatoria de los bordes de la pantalla y hace un circulo.

Ambos enemigos son destruidos si salen de la visión de la cámara.

3.1.2 Actor principal

Se encarga de gestionar la velocidad del actor,). Los disparos también los controlo desde aquí, con un contador, puedo cambiar el intervalo de disparo.

- **Mantener al jugador entre los límites de la pantalla**
  Gestiona que el actor no pueda salir del mapa, usando Screen.width y Screen.height calculo el aspect ratio de la pantalla, para averiguar el tamaño de la pantalla en X. Una vez hecho esto, me aseguro de que el actor se quede entre esas coordenadas. El método se llama keepPlayerWithinBounds()

- **Gestión del movimiento**
  Usa el método DetermineDirection para asignar los axis sobre los que controlar el jugador. Este método se ejecuta al llamar a base.Update en su propio update. Actualiza la variable velocity con la nueva velocidad cada frame.

- **Gestión de los disparos**
  Los disparos usan un temporizador para poder regular la cadencia de disparo.

- **Gestión de la animación**
  Asigna los booleanos necesarios para gestionar las animaciones del jugador.

- **Gestión de la vida**
  Cuando el BoxCollider2D colisiona con cualquier objeto, se le reduce la vida al jugador, si colisiona con un enemigo, le quita la vida al enemigo también. Si solo tiene una vida, el enemigo muere.