Instituto Tecnológico de Estudios Superiores de Monterrey

Campus Querétaro

Documentación Half-Term: Bullet Hell

Docentes

Oscar Hernández

Lizethe Pérez Fuertes

Presenta:

Alan Patricio González Bernal - A01067546

Querétaro, Querétaro México 23 de agosto del 2023

Índice

Conceptos iniciales

El proyecto fue desarrollado utilizando herramientas de Unity, además de varios assets ya hechos para las naves, las balas y el fondo principalmente.

El proyecto simula un videojuego tipo Bullet Hell, donde el jugador, la pequeña nave azul de la parte inferior de la pantalla, se enfrenta con una nave alienígena de forma muy extraña, la nave enemiga cuenta con un movimiento fijo de 200 y -200 en el eje X, y con varios patrones de disparo que se hablarán más adelante.

La nave del jugador cuenta con un tipo de disparo simple, el cual es más frecuente que el de la nave enemiga, al igual que una menor capacidad para recibir impactos, siendo esta cantidad de 3 impactos máximo, después del tercer impacto, la nave dejará de existir y será destruida y el mensaje de derrota se revela al jugador.

La nave enemiga es más grande, por lo mismo, cuenta con una mayor capacidad para recibir impactos, siendo este valor de 10, al pasar los 10 impactos, la nave enemiga es destruida y el mensaje de victoria es revelado para el jugador.

Lógica detrás del proyecto

Fondo/Background

El fondo se compone de 3 objetos, el espacio siendo dos de ellos y una nebulosa que también forma parte del fondo del proyecto. Estos fondos tienen un efecto de rotación el cual simula el movimiento del espacio en una galaxia lejana y desconocida. Los fondos solo brindan esa incertidumbre y por la paleta de colores, los proyectiles de la nave enemiga se pueden mezclar con el fondo, lo cual permite que no todo sea fácil para el jugador.

Nave del Jugador

La nave del jugador es un preset recuperado de la tienda de assets de Unity, siendo este un Asset gratuito. La nave está apuntando hacia arriba, como si estuviera viajando en el espacio mencionado anteriormente. La nave tiene un objeto vacío justo en la punta de la nave, este es el cañón de la nave, aunque es invisible para el jugador, se ve que de ahí se disparan las balas y esto da el efecto que el cañón está debajo de la nave, en la parte que el jugador no puede ver.

Las balas que dispara el jugador son diferentes a las del enemigo. Tienen el mismo tamaño, pero las del jugador se disparan a mayor velocidad y son de color rojizo.

La lógica del disparo de la nave del jugador hace que después de 3 segundos, las balas del jugador desaparezcan. La lógica de la nave del jugador es más básica que la de la nave enemiga.

Nave enemiga

La nave enemiga, que a partir de ahora será referida como Boss, tiene varias propiedades. Cuenta con una velocidad de 50, dispara una cantidad de balas definida, siendo este valor de 10. El Boss dispara a forma de ráfaga, teniendo tres tipos de ráfagas diferentes:

1. Medio círculo aleatorio
2. Círculo completo
3. Estrella giratoria

Más adelante se explicará cada tipo de disparo de manera individual.

El Boss está funcionando gracias a varios componentes, los cuales se conectan de la siguiente forma:

Layers del Boss

Primero se tiene la bala del Boss, esta es específica para el Boss ya que tendrá un valor de daño igual al Player, pero impactará únicamente con el player, ignorando los colliders de sí mismas y de las balas del player, esto para dar un efecto más cercano al de los Bullet Hells.

Una vez que las balas del Boss existen, se generó un objeto vacío llamado BulletPool, de este objeto vacío se generarán las balas que requiere el Boss, para aumentar el rendimiento del juego y evitar consumo de recursos innecesarios, el BulletPool aparece y desaparece las balas que necesita, de esta forma, se evita el gasto de recursos en balas que no afectan al player una vez pasan de él o salen de la cámara.

Tipos de movimiento

Los tres métodos de disparo del Boss son definidos en FireMovement. Este script tiene 3 clases diferentes, siendo cada una el tipo de disparo. Está programado para que cada 10 segundos, cambie el tipo de disparo entre los 3 posibles. Cada tipo de disparo tiene una función matemática que genera ese tipo de movimiento

Circle

El tipo de disparo de círculo es matemáticamente sencillo:

```
 angleStep = (360f / bulletsAmount);
                Quaternion rotation = Quaternion.AngleAxis(angleStep,
Vector3.forward);
```

La cantidad de balas que se van a disparar se dividen entre 360, la cantidad de grados en un círculo, el resultado es la cantidad de espaciado que habrá entre cada bala para dispararse. El Quaternion hace que siempre se disparen hacia adelante de donde sean generadas las balas, de manera que parece que el círculo se agranda con cada segundo.

Star

```
angleStep = 360f / 5;
                startAngle = Random.Range(0f, 360f);
```

En este caso, el ángulo de la estrella nos lo da el dividir el círculo entre 5, esto nos da el valor de los vértices de la estrella, siendo estos los puntos en los que se tiene una bala. Su rotación se da de forma aleatoria, de manera que sea más desafiante para el jugador.

Current

```
angleStep = 200f / (bulletsAmount) - 5 ;
                startAngle = Random.Range(0f, 180f);
```

Este tipo de disparo es el que se selecciona como Current, debido a que con este tipo de disparo se inicia el juego. Este efecto es un poco más complejo de explicar que los anteriores. En este tipo de disparo, las balas se dispersarán en efecto de disparo de escopeta, como medio círculo, pero gracias al startAngle, el ángulo de disparo es aleatorio entre 0 y 180, de esta forma me aseguro que el disparo está hecho de manera que el jugador puede ser alcanzado por una bala.

El angleStep es el que indica la separación de las balas, este valor fue seleccionado así ya que de otra forma, el espaciado es demasiado entre las bala.

Puntos en común de los tipos de disparo

Los tres tipos de disparo tienen el resto del código en común o muy similar uno con otro, el funcionamiento es redundante:

```
for (int i = 0; i < bulletsAmount; i++)
                {
                    Vector2 bulDir = Quaternion.AngleAxis(startAngle,
Vector3.forward) * Vector2.up;

                    GameObject bul =
BulletPool.bulletPoolInstance.GetBullet();
                    if (bul != null)
                    {
                        bul.transform.position =
bossFirePoint.position;
```

```
                        bul.SetActive(true);

bul.GetComponent<Bullet>().SetMoveDirection(bulDir);
                    }

                    startAngle += angleStep;
                }
                break;
```

Este ciclo for, lo que hace es que por la cantidad de balas que tiene el pool, o sea, 15, se genera la dirección de bala, un Vector2 que calcula la dirección de cada bala. El Quaternion.AnglkeAxis me permite rotar el Vector2 a lo largo del eje Z, dando la rotación y haciendo que el vector apunte en dirección correcta.Después el GameObject bul me permite reutilizar las balas creadas del pool, para que así haya un mejor rendimiento a lo largo del proyecto. El bul.transform.position es la posición desde donde se realizará el disparo, siendo este el punto BossFirePoint, un punto frente al Boss para evitar problemas con la colisión del jefe en sí.

El bul.SetActive(true) permite que la bala sea visible para el jugador y que funcione, el bul.GetrComponent… Configura la dirección de la bala, el cual fue calculado con bulDir. Startangle += angleStep incrementa el ángulo para que las balas se distribuyan en el patrón que se busca, y por último el break que finaliza el bloque en el que se encuentre, ya sea Star, Circle o Current.

TextMeshes

Existen 3 TextMeshes en el juego, los cuales se activan dependiendo de lo que suceda en el juego; si el player elimina al boss, el Text de Win se activa, al igual que el Text de créditos del autor. Si el Boss elimina al player, el Text de Lose se activa. Cada Text está en posiciones diferentes y se activan gracias a su conexión con los scripts de bossHealth y PlayerHealth, los cuales dictan los valores de salud de cada nave, de igual forma, en ese código está la condición de destrucción del objeto, lo cual activa el Text correspondiente.

Video del proyecto

En este espacio, pondré mi video del proyecto funcionando.
https://youtu.be/EOz5KtcXmBY

Video del gameplay del proyecto
https://youtu.be/4zhdw-PGf48

Reflexión Final

Este proyecto me gustó bastante porque me retó a investigar bastante por mi cuenta sobre el funcionamiento y aplicaciones de varias propiedades de Unity. También me permitió desarrollar un juego tipo Bullet Hell de manera funcional y que en lo personal me hace sentir orgullo, esto debido a la complejidad y al avance que demostré en mi mismo cuando hace 3 semanas ni siquiera había usado Unity. Sin duda es un proyecto que reta al estudiante, aunque sin duda me hubiera gustado que los laboratorios fueran un poco más relacionados con este proyecto, de esta forma, los consideraría más útiles.