**Desafío 4: Creación de scripts de fútbol**

## **Enlace video:**

<https://drive.google.com/file/d/1Az8Bwkysk6bJdFBzrSYEsaNC7wBmwFxZ/view?usp=sharing>

## **Descripción general:**

## Utiliza las habilidades que aprendiste en el prototipo Sumo Battle en un contexto completamente distinto: un campo de fútbol. Al igual que en el prototipo, controlarás una pelota al rotar la cámara a su alrededor y aplicando una fuerza hacia adelante, pero en lugar de empujarlas por el borde, tu objetivo es lanzarlas dentro de la red del oponente mientras que él intenta hacer lo mismo en tu red. Al igual que en Sumo Battle, después de cada ronda se generará una nueva oleada con más pelotas enemigas para desafiar a tu defensa. Sin embargo, casi nada funciona en este proyecto. Es tu trabajo hacer que funcione correctamente.

## **Resultado del desafío:**

* Los enemigos se mueven hacia tu red, pero puedes golpearlos para desviarlos lejos de ella.
* Los potenciadores aplican un aumento de fuerza y desaparecen después de 5 segundos.
* Cuando no hay más pelotas enemigas, una nueva oleada se genera con 1 enemigo adicional.

## **Retos**

**Al golpear a un enemigo lo envías directamente hacia ti**

#### Cuando golpeas a un enemigo, se debe alejar del jugador.

#### **Solución**

El camino que sigue el enemigo al colisionar con el player se esta definiendo con la resta de la posición del player menos la del enemigo, por ello es por lo que este se envía hacia el player. La solución es cambiar la posición de las variables. Se resta a la posición del enemigo a la del player y esta se asigna al camino que seguirá el enemigo.

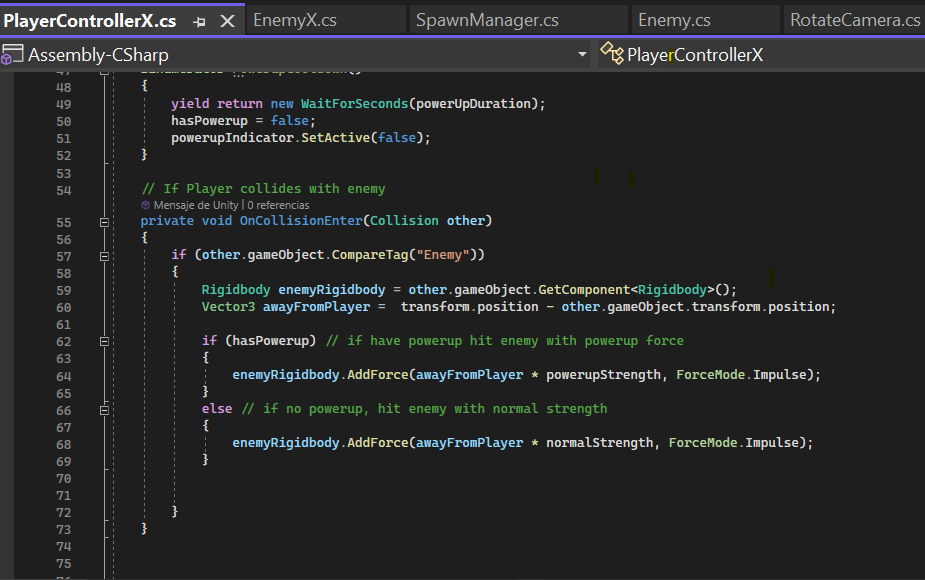


Ilustración . Antes del reto 1.

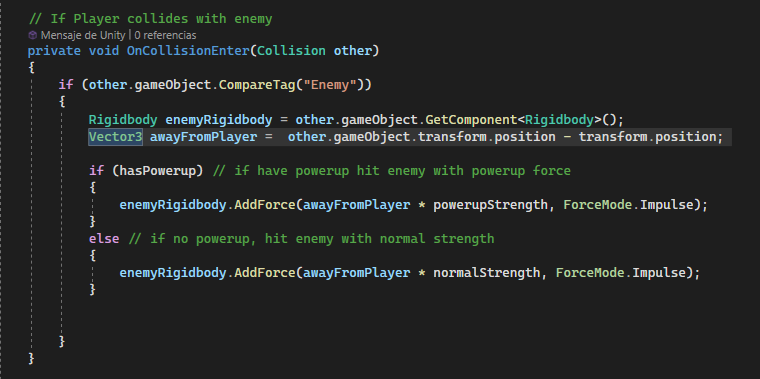


Ilustración . Después del reto 1.

#### **Una nueva oleada se crea cuando el jugador obtiene un potenciador**

* Se debe crear una nueva oleada cuando todas las pelotas enemigas han desaparecido.

#### **Solución**

En SpawnManegeX.cs en la función Update() se están buscando los objetos con tag “Powerup” por ello al tomarlo se genera otra ola de enemigos. Se debe cambiar el tag por “Enemy”

Texto

Descripción generada automáticamente

Ilustración . Antes reto 2.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración . Después reto 2.

#### **El potenciador nunca desaparece**

* El potenciador debe durar solo por un tiempo limitado y después desaparecer.

#### **Solución**

En el PlayerControllerX.cs se tiene PowerCooldown() sin embargo este no se usa. La solución es añadir en OnTriggerEnter() un método StartCoroutine() y dentro llamar a PowerCooldown().

Texto

Descripción generada automáticamente

Ilustración . Antes reto 3.

Texto

Descripción generada automáticamente

Ilustración . Después reto 3.

**Se generan 2 enemigos en cada oleada**

#### Se debe generar un enemigo en la oleada 1, dos en la oleada 2, tres en la oleada 3, etc.

#### **Solución**

En SpawnManagerX.cs en el método SpawnEnemyWave() se una un for para instancias a los clones de los enemigos por oleada, sin embargo no se está usando el parámetro enemiesToSpawn. Para solucionarlo, solo cambiamos el 2 en el for por enemiesToSpawn.

Texto

Descripción generada automáticamente

Ilustración . Antes reto 4.

Texto

Descripción generada automáticamente

Ilustración . Después reto 4.

#### **Las pelotas enemigas no se mueven hacia ningún lado**

#### Las pelotas enemigas deben moverse hacia el objeto «Player Goal».

#### **Solución.**

En la clase EnemyX.cs se contempla en el método Update() la asignación de dirección y movimiento de los enemigos hacia el jugador, sin embargo en Start() nunca se llega a instanciar el GameObject playerGoal. La solución consiste en asignar a la variable playerGoal el objeto del juego con el tag “Player”

Texto

Descripción generada automáticamente

Ilustración . Antes reto 5.

Texto

Descripción generada automáticamente

Ilustración . Después reto 5.

#### **Extra: El jugador necesita un potenciador turbo.**

#### El jugador debe obtener un potenciador de velocidad cuando presione la barra espaciadora y un efecto de partículas debe aparecer cuando lo utilice.

#### **Solución.**

La solución consiste en agregar dos variables al PlayerControllerX, se agregan powerSpeed con un valor superior a speed, que será la velocidad que tendrá el Player con al presionar Espacio. Se agrega una variable Transform playerTrasform para dar seguimiento a ala posición del player y colocar las partículas y una variable ParticleSystem para controlar las partículas.

Se crea el método PlayParticleEffect() para controlar la acción de las partículas y UpdateParticlePosition() que nos ayuda a controlar la posición de las partículas. Con ello, en Update() primero se posiciona las partículas sobre el jugador sin accionarlas y se crea un if para detectar cuando la tecla Espacio esta accionada, si es verdadero se accionan las partículas y la velocidad es mayor, si no los parámetros son los normales.

Texto

Descripción generada automáticamente

Ilustración . Extra 1.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Ilustración . Extra 1.

Texto, Chat o mensaje de texto

Descripción generada automáticamente

Ilustración . Extra 1.

Texto

Descripción generada automáticamente

Ilustración . Extra 1.

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media

Ilustración . Extra 1.

#### **Extra: Los enemigos nunca se vuelven más difíciles**

* La velocidad de los enemigos debe aumentar poco a poco con cada nueva oleada.

#### **Solución**

En SpawnManageX.cs se crea la variable speedWave y esta aumenta en 100 por cada ola que se genera. En EnemyX.cs se crea una variable de tipo SpawnManageX y a esta se le asignan todos los objetos del juego de tipo SpawnManageX que es solo el objeto vacio que gestiona el spawn, de esta variable se extrae la propiedad speedWave y se asigna a la variable local speed que se usa para la movilidad de los enemigos. De esta forma en cada nueva Ola los enemigos son más rápidos.

Texto

Descripción generada automáticamente

Ilustración . Extra 2.

Texto

Descripción generada automáticamente

Ilustración . Extra 2.