

Desafío 4: Creación de scripts de fútbol

Enlace video:

<https://drive.google.com/file/d/1Az8Bwkysk6bJdFBzrSYEsNC7wBmwFxZ/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 está 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.

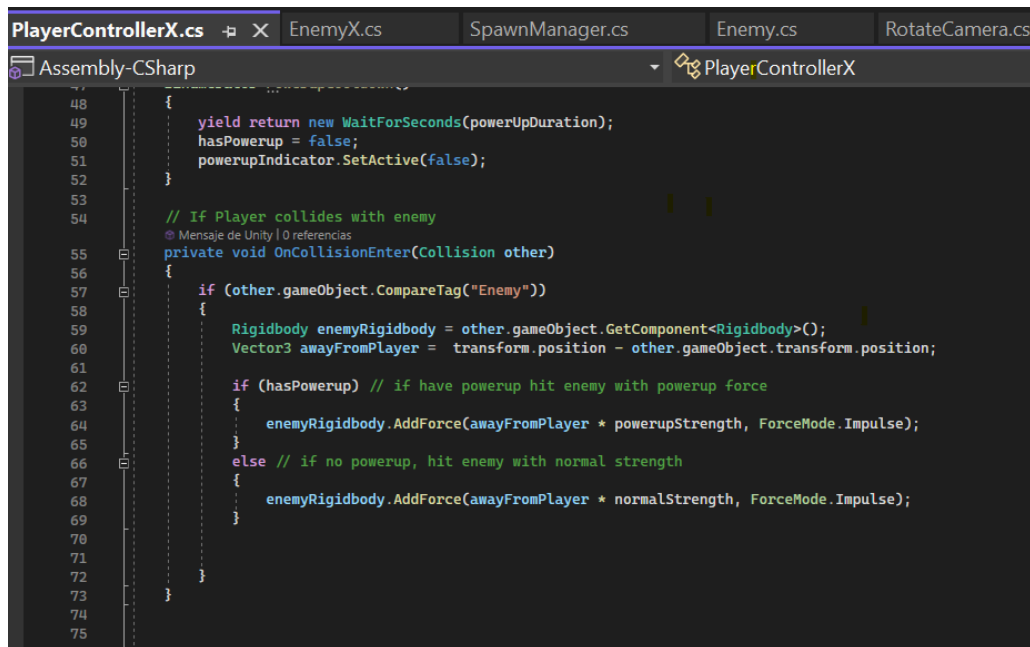


Ilustración 1. Antes del reto 1.

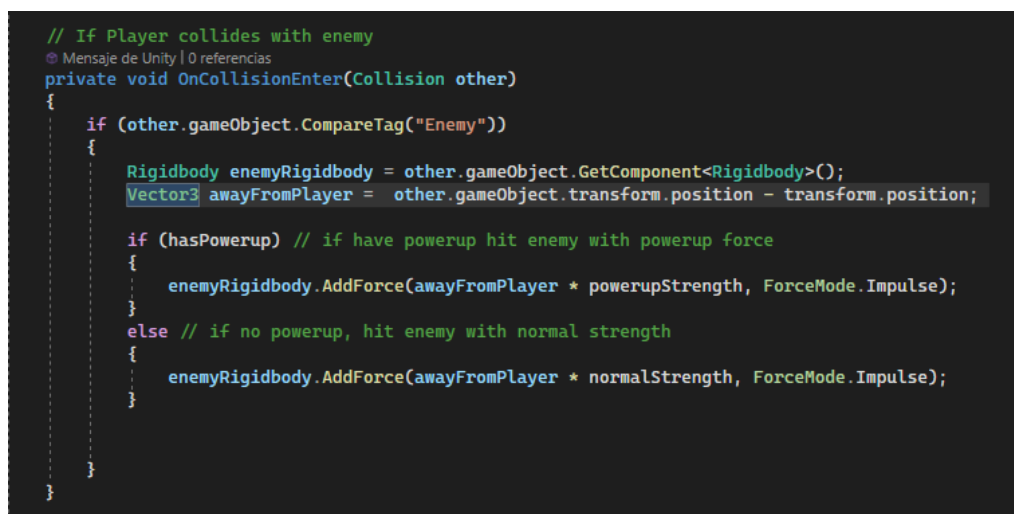


Ilustración 2. Después del reto 1.

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"

```

12 private float spawnZMax = 25; // set max spawn Z
13
14 public int enemyCount;
15 public int waveCount = 1;
16
17 public GameObject player;
18
19 // Update is called once per frame
20 // Mensaje de Unity | 0 referencias
21 void Update()
22 {
23     enemyCount = GameObject.FindGameObjectsWithTag("Powerup").Length;
24
25     if (enemyCount == 0)
26     {
27         SpawnEnemyWave(waveCount);
28     }
29
30 }
31
32 // Generate random spawn position for powerups and enemy balls
33 // 2 referencias
34 Vector3 GenerateSpawnPosition ()
35 {
36     float xPos = Random.Range(-spawnRangeX, spawnRangeX);
37     float zPos = Random.Range(spawnZMin, spawnZMax);
38     return new Vector3(xPos, 0, zPos);
39 }

```

Ilustración 3. Antes reto 2.

```

// Update is called once per frame
// Mensaje de Unity | 0 referencias
void Update()
{
    enemyCount = GameObject.FindGameObjectsWithTag("Enemy").Length;

    if (enemyCount == 0)
    {
        SpawnEnemyWave(waveCount);
    }
}

```

Ilustración 4. 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().

```

28 playerKB.AddForce(transform.forward * verticalInput * Speed * Time.deltaTime
29
30 // Set powerup indicator position to beneath player
31 powerupIndicator.transform.position = transform.position + new Vector3(0, -0.6f, 0);
32
33 }
34
35 // If Player collides with powerup, activate powerup
36 // Mensaje de Unity | 0 referencias
37 private void OnTriggerEnter(Collider other)
38 {
39     if (other.gameObject.CompareTag("Powerup"))
40     {
41         Destroy(other.gameObject);
42         hasPowerup = true;
43         powerupIndicator.SetActive(true);
44     }
45
46 // Coroutine to count down powerup duration
47 // 0 referencias
48 IEnumerator PowerupCooldown()
49 {
50     yield return new WaitForSeconds(powerUpDuration);
51     hasPowerup = false;
52     powerupIndicator.SetActive(false);
53 }

```

Ilustración 5. Antes reto 3.

```

// If Player collides with powerup, activate powerup
// Mensaje de Unity | 0 referencias
private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("Powerup"))
    {
        Destroy(other.gameObject);
        hasPowerup = true;
        powerupIndicator.SetActive(true);
        StartCoroutine(PowerupCooldown());
    }
}

// Coroutine to count down powerup duration
// 1 referencia
IEnumerator PowerupCooldown()
{
    yield return new WaitForSeconds(powerUpDuration);
    hasPowerup = false;
    powerupIndicator.SetActive(false);
}

```

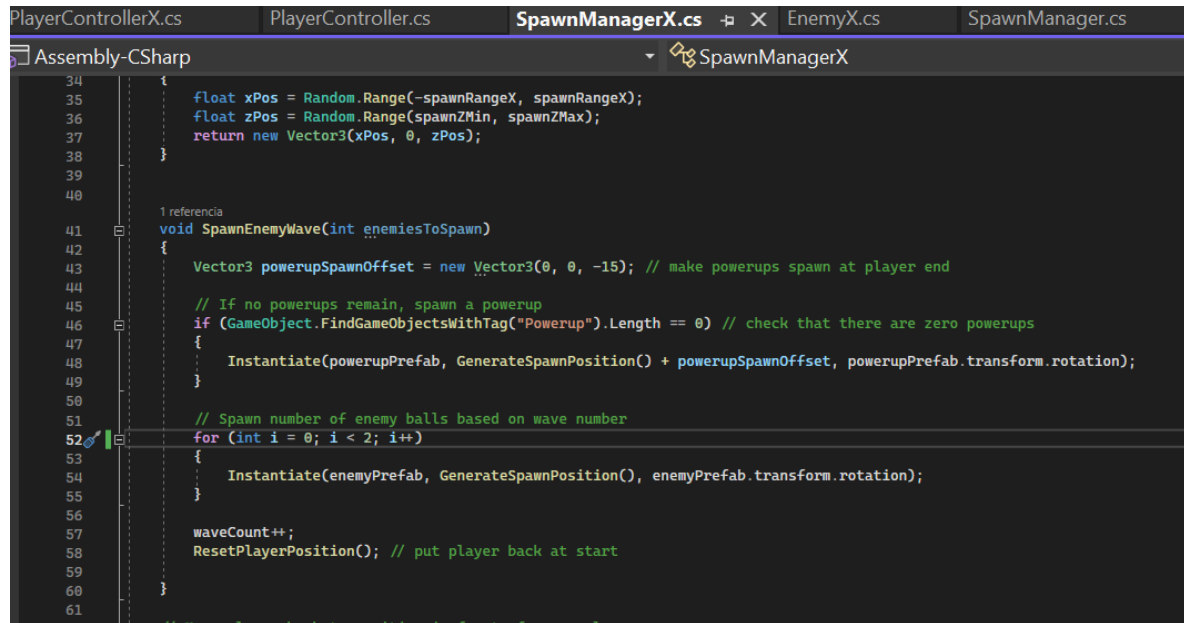
Ilustración 6. Después reto 3.

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 usa 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.



```
PlayerControllerX.cs | PlayerController.cs | SpawnManagerX.cs | EnemyX.cs | SpawnManager.cs
Assembly-CSharp
1 referencia
void SpawnEnemyWave(int enemiesToSpawn)
{
    Vector3 powerupSpawnOffset = new Vector3(0, 0, -15); // make powerups spawn at player end

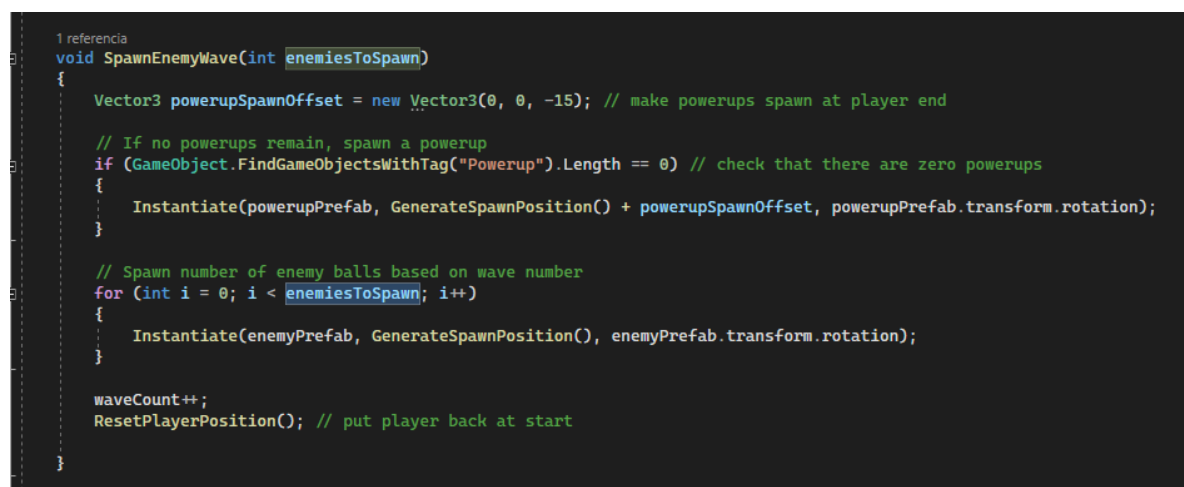
    // If no powerups remain, spawn a powerup
    if (GameObject.FindGameObjectsWithTag("Powerup").Length == 0) // check that there are zero powerups
    {
        Instantiate(powerupPrefab, GenerateSpawnPosition() + powerupSpawnOffset, powerupPrefab.transform.rotation);
    }

    // Spawn number of enemy balls based on wave number
    for (int i = 0; i < 2; i++)
    {
        Instantiate(enemyPrefab, GenerateSpawnPosition(), enemyPrefab.transform.rotation);
    }

    waveCount++;
    ResetPlayerPosition(); // put player back at start
}

// Move player back to position in front of enemy unit
```

Ilustración 7. Antes reto 4.



```
1 referencia
void SpawnEnemyWave(int enemiesToSpawn)
{
    Vector3 powerupSpawnOffset = new Vector3(0, 0, -15); // make powerups spawn at player end

    // If no powerups remain, spawn a powerup
    if (GameObject.FindGameObjectsWithTag("Powerup").Length == 0) // check that there are zero powerups
    {
        Instantiate(powerupPrefab, GenerateSpawnPosition() + powerupSpawnOffset, powerupPrefab.transform.rotation);
    }

    // Spawn number of enemy balls based on wave number
    for (int i = 0; i < enemiesToSpawn; i++)
    {
        Instantiate(enemyPrefab, GenerateSpawnPosition(), enemyPrefab.transform.rotation);
    }

    waveCount++;
    ResetPlayerPosition(); // put player back at start
}
```

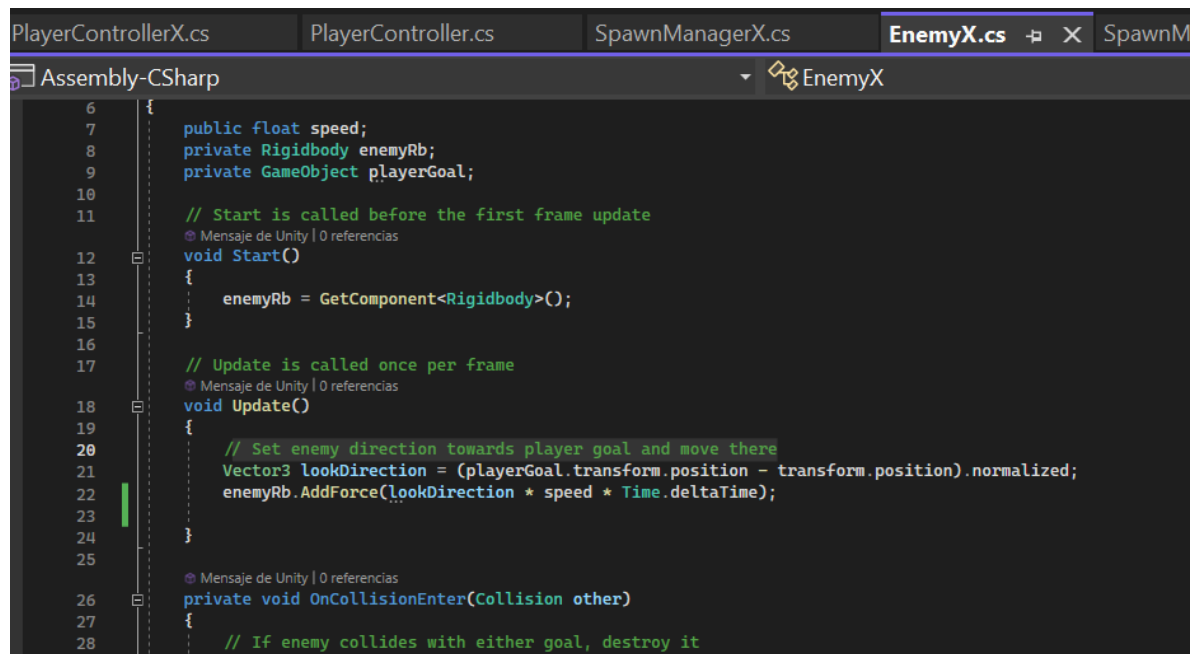
Ilustración 8. Después reto 4.

Las pelotas enemigas no se mueven hacia ningún lado

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

Solución.

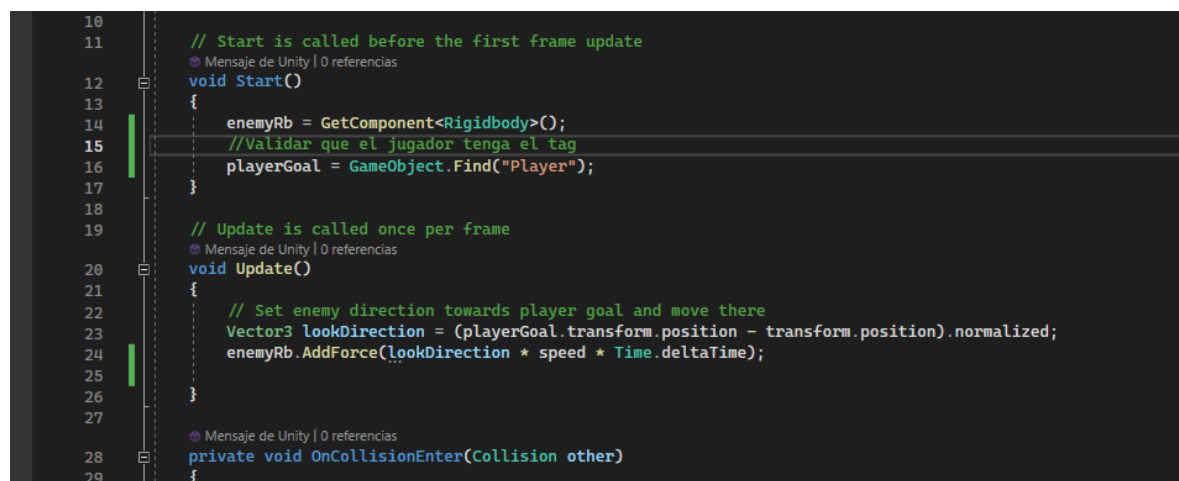
En la clase EnemyX.cs se contempla en el método Update() la asignación de dirección y movimiento de los enemigos hacia el jugador, sin embargo en Start() nunca se llega a instanciar el GameObject playerGoal. La solución consiste en asignar a la variable playerGoal el objeto del juego con el tag "Player"



```

6  {
7      public float speed;
8      private Rigidbody enemyRb;
9      private GameObject playerGoal;
10
11     // Start is called before the first frame update
12     void Start()
13     {
14         enemyRb = GetComponent<Rigidbody>();
15     }
16
17     // Update is called once per frame
18     void Update()
19     {
20         // Set enemy direction towards player goal and move there
21         Vector3 lookDirection = (playerGoal.transform.position - transform.position).normalized;
22         enemyRb.AddForce(lookDirection * speed * Time.deltaTime);
23     }
24
25     private void OnCollisionEnter(Collision other)
26     {
27         // If enemy collides with either goal, destroy it
28     }
  
```

Ilustración 9. Antes reto 5.



```

10
11     // Start is called before the first frame update
12     void Start()
13     {
14         enemyRb = GetComponent<Rigidbody>();
15         //Validar que el jugador tenga el tag
16         playerGoal = GameObject.Find("Player");
17     }
18
19     // Update is called once per frame
20     void Update()
21     {
22         // Set enemy direction towards player goal and move there
23         Vector3 lookDirection = (playerGoal.transform.position - transform.position).normalized;
24         enemyRb.AddForce(lookDirection * speed * Time.deltaTime);
25     }
26
27     private void OnCollisionEnter(Collision other)
28     {
29     }
  
```

Ilustración 10. 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 playerTransform para dar seguimiento a la 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.

```
2 using UnityEngine;
3 using UnityEngine;
4
5 public class PlayerControllerX : MonoBehaviour
6 {
7     private Rigidbody playerRb;
8     private float speed = 500;
9     private float powerSpeed = 1500;
10    private GameObject focalPoint;
11
12    public bool hasPowerup;
13    public GameObject powerupIndicator;
14    public int powerUpDuration = 5;
```

Ilustración 11. Extra 1.

```
13    public GameObject powerupIndicator;
14    public int powerUpDuration = 5;
15    public Transform playerTransform;
16    private float normalStrength = 10; // how hard to hit enemy without powerup
17    private float powerupStrength = 25; // how hard to hit enemy with powerup
18    public new ParticleSystem particleSystem;
19    void Start()
```

Ilustración 12. Extra 1.

```
23 }
24 void PlayParticleEffect()
25 {
26     if (particleSystem != null)
27     {
28         // Activa el sistema de partículas
29         particleSystem.Play();
30     }
31 }
```

Ilustración 13. Extra 1.

```
32 void Update()
33 {
34     UpdateParticlePosition();
35     // Add force to player in direction of the focal point (and camera)
36     float verticalInput = Input.GetAxis("Vertical");
37     if (Input.GetKey(KeyCode.Space))
38     {
39         playerRb.AddForce(focalPoint.transform.forward * verticalInput * powerSpeed * Time.deltaTime);
40         PlayParticleEffect();
41     }
42     else
43     {
44         playerRb.AddForce(focalPoint.transform.forward * verticalInput * speed * Time.deltaTime);
45     }
46 }
47
48 // Set powerup indicator position to beneath player
49 powerupIndicator.transform.position = transform.position + new Vector3(0, -0.6f, 0);
50
51 }
```

Ilustración 14. Extra 1.

```
51 }
52 void UpdateParticlePosition()
53 {
54     if (particleSystem != null && playerTransform != null)
55     {
56         // Actualiza la posición del sistema de partículas para que siga al jugador
57         particleSystem.transform.position = playerTransform.position;
58     }
59 }
60 // If Player collides with powerup, activate powerup
```

Ilustración 15. 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 vacío 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.


```
14 public int enemyCount;  
15 public int waveCount = 1;  
16 public int speedWave = 10;  
17  
18 public GameObject player;  
19
```

Ilustración 16. Extra 2.

```
54 Instantiate(enemyPrefab, GenerateSpawnPosition(), enemy  
55 }  
56  
57 waveCount++;  
58 speedWave = speedWave + 100;  
59 ResetPlayerPosition(); // put player back at start  
60  
61 }  
62  
63 // Move player back to position in front of own goal
```

Ilustración 17. Extra 2.