José Manuel Matehuala Avilés **GDGS3101-E**
Diego Torres Pérez

Desafío 3: Globos, bombas y booleanos

Enlace video:

https://drive.google.com/file/d/12Z8EcKWG-6FmvIlSPARn5HaqaIjk2blU/view?usp=sharing

Descripción general:

Aplica tus conocimientos de física, fondos con desplazamiento lateral y efectos especiales a un globo que flota a través de un pueblo mientras recolecta tokens y esquiva explosivos. Tendrás que resolver muchos problemas en este proyecto porque tiene una gran cantidad de errores.

Resultado del desafío:

- El globo flota hacia arriba cuando el jugador mantiene presionada la barra espaciadora.
- El fondo se repite sin errores y simula el movimiento del globo.
- Se generan bombas y tokens monetarios de manera aleatoria a intervalos.
- Cuando colisionas con los tokens monetarios, hay un efecto de sonido y partículas.
- Cuando colisionas con la bomba, hay una explosión y el fondo se detiene.

Retos

El jugador no puede controlar el globo

- El globo debe flotar hacia arriba cuando el jugador presiona la barra espaciadora.

Solución

Para hacer que el globo flote hacia arriba cuando el jugador presiona la barra espaciadora, se deben realizar algunos cambios en el script ***PlayerControllerX.cs*** que está asignado al ***GameObject Player***. Lo que falta es inicializar la variable ***playerRb***, que es el ***Rigidbody*** del objeto que debería flotar. Para ello, en el ***método start()*** a la variable ***playerRb*** se le asigna al componente ***Rigidbody*** usando un método llamado ***GetComponent*** seguido de los signos **<>** que representan que dan un tipo de algo, es decir, sirven para encontrar diferentes componentes, en este caso se obtiene el ***Rigidbody*** del ***GameObject Player***.

Finalmente se colocan los paréntesis porque es una llamada que se hace como método. Ver Ilustración 1.

```
void Start()
{
    Physics.gravity *= gravityModifier;
    playerAudio = GetComponent<AudioSource>();
    playerRb = GetComponent<Rigidbody>();

    // Apply a small upward force at the start of the game
    playerRb.AddForce(Vector3.up * 5, ForceMode.Impulse);

}
```

Ilustración 1. Reto 1.

El fondo solo se mueve cuando el juego termina

- El fondo debe moverse al inicio y *detenerse* cuando el juego termine.

Solución

Para hacer que el fondo no solo se mueva cuando el juego termina, se necesita realizar un cambio en el script ***MoveLeftX***. Actualmente, el fondo se mueve cuando ***playerControllerScript.gameOver*** es ***true***. En lugar de eso, se debe hacer que el fondo se mueva al principio y se detenga cuando el juego termine. Para ello en la condición se debe negar el ***playerControllerScript.gameOver*** para que el programa lea que si el valor de ***gameOver*** que proviene de la clase ***PlayerControllerX.cs*** el falso, se tiene que mover el fondo. Ver Ilustración 2.

```
// If game is not over, move to the left
if (!playerControllerScript.gameOver)
{
    transform.Translate(Vector3.left * speed * Time.deltaTime, Space.World);
}
```

Ilustración 2. Reto 2.

No se genera ningún objeto

- Haz que se generen objetos bomba o monetarios cada cierto segundo.

Solución

El error se debe a un error relacionado con la ortografía en la llamada a ***InvokeRepeating*** en la clase ***SpawnManagerX.cs***. Es necesario asegurarse de que el nombre del método coincida con el nombre del método al que se desea invocar. En este caso, el método se llama ***SpawnObjects***, no ***PrawnsObject***. Además, se necesita especificar el intervalo entre las invocaciones. Ver Ilustración 3.

```
void Start()
{
    InvokeRepeating("SpawnObjects", spawnDelay, spawnInterval);
    playerControllerScript = GameObject.Find("Player").GetComponent<PlayerControllerX>();
}

// Spawn obstacles
0 referencias
void SpawnObjects ()
{
    // Set random spawn location and random object index
    Vector3 spawnLocation = new Vector3(30, Random.Range(5, 15), 0);
    int index = Random.Range(0, objectPrefabs.Length);

    // If game is still active, spawn new object
    if (!playerControllerScript.gameOver)
    {
        Instantiate(objectPrefabs[index], spawnLocation, objectPrefabs[index].transform.rotation);
    }

}
```

Ilustración 3. Reto 3.

Los fuegos artificiales aparecen a un costado del globo

- Haz que los fuegos artificiales se muestren en la posición del globo.

Solución

Para hacer que los fuegos artificiales se muestren en la posición del globo cuando éste choca con un signo de pesos y no más alejados de este ***GameObject*** se debe cambiar el valor del ***eje x***, el cual representa el desplazamiento lateral del globo desde la vista de la cámara durante la ejecución del juego, en la sección ***Inspector*** de Unity. El valor que se le debe asignar a los fuegos artificiales (***FX_Fireworks_Yellow_Small***) que se ubican dentro del ***GameObject Player*** es igual a 0 como sucede con la ubicación de la explosión al chocar con una bomba. Ver Ilustración 4.

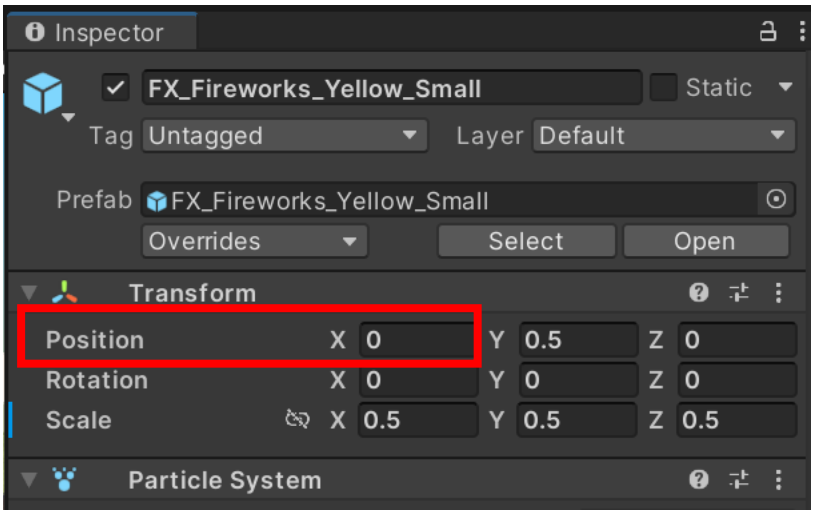

Ilustración 4. Reto 4.

El fondo no se repite correctamente

- Haz que el fondo se repita sin errores.

Solución

Para hacer que el fondo se repita correctamente, se debe ajustar la lógica en el script ***RepeatBackgroundX.cs***. Como se puede observar en la codificación, la variable ***repeatWidth*** debe ser la mitad del ancho del fondo en lugar de la mitad del alto. Para ello, al obtener el tamaño del componente ***BoxCollider*** mediante el método ***GetComponent*** se tiene que conseguir el tamaño del ancho (**x**) y obtener la mitad de éste (**/2**). Ver Ilustración 5.

```
Mensaje de Unity | 0 referencias
private void Start()
{
    startPos = transform.position; // Establish the default starting position
    repeatWidth = GetComponent<BoxCollider>().size.x / 2; // Set repeat width to half of the background
}
```

Ilustración 5. Reto 5.

Extra: El globo puede flotar demasiado alto

- Evita que el jugador pueda hacer flotar su globo demasiado alto.

Solución

En la clase ***PlayerControllerX.cs*** se deben declarar dos variables una de tipo float y otra bool; la de tipo float llamada ***limite*** que nos permite delimitar el límite hasta donde puede subir el globo y la bool para saber si el globo puede o no subir, el límite lo inicializamos en 7. En el método ***update()*** se establece una condición en la que se obtiene la ***position*** en ***y*** del globo mediante el componente ***transform*** y se específica que si la posición en ***y*** que es la que indica la subida del globo es menor al límite entonces la variable ***puedeSubir*** será verdadera, en caso contrario será falsa. Finalmente, se específica que si se presiona la tecla de espacio y el juego no ha terminado y ***puedeSubir*** es verdadero entonces el globo puede subir y continuar su trayecto. Ver Ilustración 6.

```
private float limite = 7;
private bool puedeSubir;

// Start is called before the first frame update
Mensaje de Unity | 0 referencias
void Start()
{
    Physics.gravity *= gravityModifier;
    playerAudio = GetComponent<AudioSource>();
    playerRb = GetComponent<Rigidbody>();

    // Apply a small upward force at the start of the game
    playerRb.AddForce(Vector3.up * 5, ForceMode.Impulse);

}

// Update is called once per frame
Mensaje de Unity | 0 referencias
void Update()
{
    if (transform.position.y < limite)
    {
        puedeSubir = true;
    } else
    {
        puedeSubir = false;
    }

    // While space is pressed and player is low enough, float up
    if (Input.GetKey(KeyCode.Space) && !gameOver && puedeSubir)
    {
        playerRb.AddForce(Vector3.up * floatForce);
    }
```

Ilustración 6. Reto 6.

Extra: El globo puede atravesar el suelo

- Haz que parezca que el globo rebota en el suelo para evitar que salga por la parte inferior de la pantalla. También debe haber un efecto de sonido cuando esto ocurra.

Solución

Para resolver, primero en el ***Inspector*** se debe crear y asignar una nueva etiqueta al piso, en este caso llamada ***"Piso"***. Posteriormente, en la clase ***PlayerControllerX.cs*** se debe crear una condicional en la que se establezca que si la etiqueta corresponde a "Piso" durante una colisión entre el globo y el piso se aplicará una fuerza multiplicada por 5 y adicionalmente se le agrega un impulso para que el efecto sea inmediato. Para incluir el audio al momento que rebote el globo es preciso definir una variable public para que se pueda acceder desde el ***Inspector*** y ahí colocar el audio ***Boing*** del tipo ***AudioClip*** llamada ***reboteSound***. Finalmente, cuando haya colisión entre el globo y el suelo se reproduce por medio de la variable del tipo ***AudioSource playerAudio*** el audio con el método ***PlayOneShot*** que hace que se ejecute una sola vez y que recibe como parámetros el audio que se desea reproducir, así como el volumen al que se desea escuchar. Ver Ilustración 7, 8 y 9.

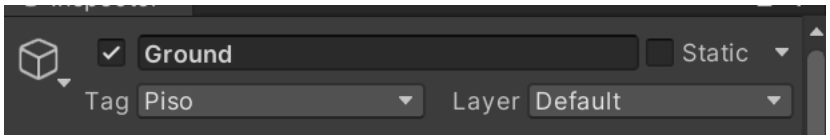

Ilustración 7. Tag Piso.

```
public AudioClip reboteSound;
```

Ilustración 8. Variable Tipo AudioClip.

```
} else if (other.gameObject.CompareTag("Piso"))
{
    playerRb.AddForce(Vector3.up * 5, ForceMode.Impulse);
    playerAudio.PlayOneShot(reboteSound, 1.0f);
}
```

Ilustración 9. Lógica al haber colisión entre el globo y el piso.