IMAGEN DIGITAL

PLANTAS VS ZOMBIES

Proyecto Unity

Jesús Moruno Muñozo

Curso 2024/2025

ÍNDICE

1. INTRODUCCIÓN

Para este proyecto, he colaborado con mi compañero, Adrián Ramos Caballero, para desarrollar nuestra versión personalizada del icónico juego "Plants vs. Zombies" utilizando la plataforma Unity.

Nuestra intención con este proyecto ha sido no solo rendir homenaje a este popular título, sino también aprender y aplicar conocimientos fundamentales sobre desarrollo de videojuegos. Durante el proceso, hemos explorado y utilizado diversas herramientas y técnicas de diseño, programación y gestión de proyectos. A lo largo del desarrollo, nos hemos enfrentado a múltiples retos, desde la implementación de mecánicas de juego como el desplazamiento de los enemigos y la interacción con las plantas, hasta la creación de gráficos y efectos visuales que reflejen el espíritu del juego original.

Además, este proyecto nos ha brindado una oportunidad única de trabajar en equipo, dividir responsabilidades y combinar nuestras habilidades para obtener un resultado final que sea tanto educativo como entretenido. Mientras Adrián se enfocaba en el diseño del nivel y la programación de los zombis, yo me centré en el desarrollo de las mecánicas de las plantas y en los efectos visuales. Ambos colaboramos estrechamente para integrar estas partes y garantizar que el juego tuviera un flujo coherente y una jugabilidad satisfactoria.

Hemos aprendido mucho sobre Unity, desde el uso del motor gráfico y la implementación de scripts en C#, hasta el diseño de interfaces y la optimización del rendimiento del juego. Cada etapa del proyecto, desde el planteamiento inicial hasta las pruebas finales, ha sido una experiencia valiosa que nos ha permitido desarrollar habilidades técnicas y creativas.

En este documento, describiré en detalle cada paso del proceso de desarrollo de mi parte, los desafíos enfrentados y las soluciones implementadas, así como nuestras reflexiones finales sobre el resultado obtenido. Esperamos que esta experiencia sirva como un ejemplo de lo que se puede lograr con dedicación y trabajo en equipo.

2. CÓDIGO GIRASOL.CS

El girasol es un elemento fundamental del juego, cuya función es generar soles cada cierto tiempo. Estos soles son objetos que el jugador puede recoger para obtener recursos necesarios para colocar nuevas plantas en el campo de juego. Cada sol aparece cerca de la posición del girasol y tiene un tiempo limitado antes de desaparecer automáticamente.

Este script controla la lógica detrás de la generación de soles, determinando su frecuencia, posición aleatoria alrededor del girasol, y el tiempo que permanecen visibles en la escena.

2.1. Explicación del código

1. **Declaración de variables principales**:
    - frecuenciaDeSoles: Define el intervalo en segundos entre la aparición de cada sol.
    - Sol: Hace referencia al prefab que representa un sol en el juego.

2. **Inicio del comportamiento del girasol**:
   El método Start() permite realizar tareas de forma secuencial con intervalos de espera. En este caso, el girasol hace una pausa durante el tiempo especificado por frecuenciaDeSoles antes de crear un nuevo sol.
3. **Generación de soles**:
   - Los soles se instancian en una posición cercana al girasol, con una leve variación aleatoria hacia arriba, la izquierda o la derecha.
   - Una vez creados, los soles tienen un tiempo limitado (10 segundos) antes de ser eliminados automáticamente si no han sido recogidos por el jugador.
4. **Ciclo infinito**:
   El bucle while (true) asegura que los soles sigan generándose indefinidamente mientras el girasol permanezca activo en el juego.

Imagen 1. Explicación del funcionamiento del girasol.

2.2. Código

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Girasol : MonoBehaviour {

    public float frecuenciaDeSoles = 1;
    public GameObject Sol;

    IEnumerator Start()
    {
        while (true)
        {
            //------------------------------------------------ APENAS INICIE EL JUEGO
            yield return new WaitForSeconds(frecuenciaDeSoles);
//------------------------------------------------ LANZA EL SOL LUEGO DE LA FRECUENCIA
            GameObject go = Instantiate(Sol, transform.position + Vector3.up *
```

```
            Random.Range(0f, 1f) + Vector3.left * Random.Range(-1f, 1f),
            Sol.transform.rotation) as GameObject;
            Destroy(go, 10);

        }
    }
}
```

3. CÓDIGO LANZAGUISANTES.CS

El **lanzaguisantes** es una planta ofensiva que dispara guisantes periódicamente hacia los zombis para dañarlos. Este script controla su comportamiento: verifica si hay un zombi en su rango de ataque antes de disparar, crea el proyectil (guisante) desde la posición del cañón, y elimina el guisante automáticamente después de un tiempo si no impacta nada.

3.1. Creación del guisante

El guisante es un proyectil disparado por el **LanzaGuizantes** que se mueve hacia la derecha para impactar contra los zombis y causar daño. Este script controla su velocidad de movimiento y define el daño que inflige al colisionar con un enemigo.

Explicación del código:

1. **Declaración de variables principales**:
    - velocidad: Determina la rapidez con la que el guisante se desplaza hacia adelante.
    - daño: Especifica la cantidad de daño que el guisante inflige al impactar contra un zombi.
2. **Movimiento del guisante**:
    - En el método Update() (que se ejecuta una vez por fotograma), el guisante se mueve continuamente hacia la derecha.
    - La posición del guisante se actualiza sumando un desplazamiento proporcional a su velocidad y al tiempo transcurrido desde el último fotograma (Time.deltaTime), lo que garantiza un movimiento fluido y dependiente del tiempo.

Código

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Guisante : MonoBehaviour {

    public int velocidad = 10;
    public int daño = 1;
```

```
    void Update()
    {
        transform.position += Vector3.right * velocidad * Time.deltaTime;
    }
}
```

3.2. Explicación del código del lanzaguisantes

1. **Declaración de variables principales**:
    - cadencia: Determina el intervalo en segundos entre cada disparo.
    - Guisante: Prefab del proyectil que se dispara (un guisante).
    - cañon: Punto desde el cual se generan los guisantes.
    - layerZombie: Define la capa que identifica a los zombis como objetivos válidos para los disparos.
2. **Inicio del ataque del lanzaguisantes**:
    - Se ejecuta el método Start() una vez que el objeto está activo en el juego.
    - El lanzaguisantes espera un tiempo definido por la variable cadencia antes de intentar disparar.
3. **Detección de zombis en el rango**:
    - Mediante un **rayo** (Raycast), el script verifica si hay zombis dentro de un rango de 12 unidades hacia la derecha del lanzaguisantes.
    - Si el rayo detecta un zombi (es decir, choca con un collider en la capa layerZombie), el lanzaguisantes dispara un guisante.
4. **Creación y eliminación de guisantes**:
    - Un nuevo guisante se genera en la posición del cañón con la orientación del prefab.
    - El guisante es eliminado automáticamente después de 10 segundos para evitar que objetos innecesarios consuman recursos.

Código

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LanzaGuizantes : MonoBehaviour {

    public float cadencia = 3;
    public GameObject Guisante;
    public Transform cañon;
    public LayerMask layerZombie;

    IEnumerator Start()
    {
        while (true)
        {
            //------------------------------------------------ APENAS INICIE EL JUEGO
            yield return new WaitForSeconds(cadencia);
                 //------------------------------------------------ LANZA EL SOL LUEGO DE LA
FRECUENCIA
```

```
                    RaycastHit2D hit = Physics2D.Raycast(cañon.position, Vector3.right, 12,
layerZombie);
            if (hit.collider != null)
            {
                                    GameObject go = Instantiate(Guisante, cañon.position,
Guisante.transform.rotation) as GameObject;
                Destroy(go, 10);
            }

        }
    }
 }
```

Imagen 2. Funcionamiento del lanzaguisantes

4. CÓDIGO PLANTAS.CS

Este script gestiona las características básicas de las plantas en el juego. Define atributos esenciales como la carta visual de la planta, su costo en soles y su cantidad de vida. También incluye una función para reducir su vida cuando son mordidas por los zombis, destruyendo la planta si su vida llega a cero.

Explicación del código:

1. **Declaración de variables principales**:
    - cartaAsignada: Es un sprite que representa la carta asociada a la planta, utilizado posiblemente en la interfaz para seleccionar la planta.

- precioSoles: Indica el costo en soles para plantar esta unidad en el campo de juego.
- vida: Define la cantidad de vida que tiene la planta. Este valor disminuye cuando la planta es atacada por los zombis.

2. **Método Morder**:
   - Este método se llama cuando la planta es atacada por un zombi.
   - Reduce en 1 la vida de la planta cada vez que se ejecuta.
   - Si la vida de la planta llega a 0 o menos, se destruye el objeto de la planta en la escena con Destroy(gameObject).

Código

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Plantas : MonoBehaviour
{
    public Sprite cartaAsignada;
    public int precioSoles;
    public int vida;

    void Morder()
    {
        vida--;
        if (vida <= 0)
            Destroy(gameObject);

    }
}
```

5. EXPLICACIÓN DEL TERRENO DE JUEGO

Como vemos, el terreno de juego está estructurado en un **tablero de varias filas y columnas** que simula un jardín o patio. Este espacio es donde el jugador organiza las plantas para defenderse de las oleadas de zombis.

Cada celda del tablero es un espacio donde el jugador puede plantar una unidad defensiva o de soporte, como el **girasol** o el **lanzaguisantes**.

Los **zombis** avanzan de **derecha a izquierda**. El objetivo del jugador es **evitar que los zombis crucen el tablero** y lleguen al borde izquierdo, donde está la casa.

El diseño del terreno fomenta una estrategia basada en la colocación adecuada de plantas según la amenaza de cada fila. Los zombis avanzan por filas específicas, por lo que el jugador debe observar constantemente todo el tablero para adaptarse y proteger las líneas más vulnerables.

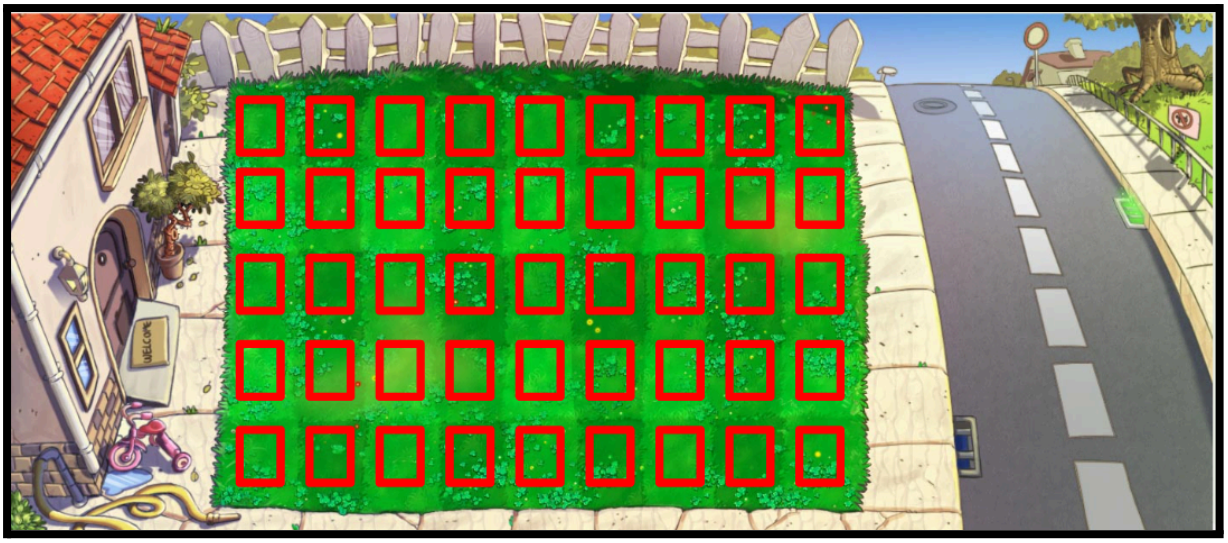

Imagen 3. Mapa del juego Plantas vs Zombies, en rojo vemos las parcelas del jardín donde se pueden plantar plantas