Guía de Laboratorio 6

Primera parte

Adición y configuración de un enemigo en Unity 2D

1. Selección y preparación del sprite del enemigo

1.1 Selecciona el sprite

- Busca el archivo

Assets/Images/TinySwords/Factions/Goblins/Troops/Torch/Red/Torch_Red.png

- Selecciónalo y, en el **Inspector**, configura:
    - **Pixels Per Unit:** 64 (o el valor usado para el resto del juego)

1.2 Editor de Sprites

1. Haz clic en **Sprite Editor** → **Slice** → **Grid by Cell Count** (7x5).
2. Guarda los cambios.
3. Arrastramos el Torch_Red_0.png a la escena.
4. Renombramos el sprite como EnemyTorchRed.

2. Configuración del objeto enemigo

2.1 Añadir componentes

Con el objeto Enemy seleccionado:

- **Rigidbody 2D**
    - Body Type: **Kinematic**
    - Gravity Scale: **0**
- **Box Collider 2D**
    - Ajusta su tamaño al cuerpo del goblin.

3. Creación del script de movimiento del enemigo

3.1 Crear el script

- Crea una nueva carpeta en Assets/Scripts/Enemy
- Crea un nuevo script llamado Enemy.cs

3.2 Código del script

Abre el script y copia el siguiente código:

```
using UnityEngine;

public class EnemyMovement : MonoBehaviour
{
    public float speed = 7;
```

```
    public Transform targetTransform;
    private Rigidbody2D rb2d;

    void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        rb2d.MovePosition(Vector2.MoveTowards(transform.position,
targetTransform.position, speed*Time.deltaTime));
    }
}
```

3.3 Asignar el script

1. Arrastra el script Enemy.cs al objeto EnemyTorchRed.
2. Verás dos campos públicos: **Target Transform** y **Speed**.

4. Creación de la posición objetivo

4.1 Crear el objeto vacío

1. En el **Hierarchy**, haz clic derecho → **Create Empty**.
2. Nómbralo EnemyTargetPosition
3. Colócalo en el punto del mapa hacia donde quieras que se mueva el enemigo.

4.2 Asignar al script

1. Selecciona el objeto Enemy.
2. En el campo **Target Transform** del script, arrastra el objeto EnemyTarget.

5. Prueba del movimiento

1. Presiona el botón **Play**.
2. Observa cómo el enemigo se mueve suavemente hacia la posición del EnemyTargetPosition.
3. Ajusta la variable Speed hasta obtener un movimiento fluido.

Segunda Parte
Añadir las Animaciones

1. Realizar la animación de Inactivo (Idle)

- En el Panel Project, selecciona los sprites Torch_Red_0 hasta Torch_Red_6. o Para seleccionar en orden, haz clic en el primero, mantén presionada la tecla Shift, y haz clic en el último.
- Arrástralos directamente a la Scene o al Panel Hierarchy.
- Unity abrirá automáticamente un cuadro de diálogo para guardar la animación.
- Dentro de la carpeta Assets/Animations.
- Dentro de Animations, crea otra carpeta llamada GoblinTorchRed.
- Guarda la animación con el nombre Idle.
- Al hacer esto, Unity creará dos cosas automáticamente:
    - El archivo de animación Idle.anim.
    - Un Animator Controller y un objeto del personaje en la escena.

2. Realizar la animación de Correr (Run)

- Repite los pasos anteriores, pero esta vez selecciona los sprites Torch_Red_7 hasta Torch_Red_12.
- Arrástralos a la escena.
- En el cuadro de diálogo, guarda el archivo de animación en la carpeta GoblinTorchRed con el nombre Run.

3. Configuración del controlador

- Elimina uno de los controladores
- Renombra uno de los controladores a TorchRedController y abrelo
- Arrastra la animación de Run al Controlador TorchRedController
- Elimina las dos animaciones de Hierarchi
- Agrega el componente Animator a EnemyTorchRed
- Asigna el TorchRedController a la propiedad Controller de EnemyTorchRed

Tercera Parte

IA de Enemigo 2D con NavMesh (AI Navigation + NavMeshPlus)

1) Preparación del proyecto

1. **Instalar AI Navigation**
   - Abre **Window → Package Manager**.
   - Abre el menú desplegable **Packages → Unity Registry**.
   - Busca **AI Navigation** e **instálalo**.
2. **Instalar NavMeshPlus**
   - Descarga desde GitHub: https://github.com/h8man/NavMeshPlus (ZIP).
   - Extrae y **copia la carpeta NavMeshComponents** a **Assets/** del proyecto.

2) Configuración del NavMesh 2D

1. **Crear un contenedor del NavMesh**
   - Crea un **Empty GameObject** en la jerarquía y nómbralo **NavMesh**.
   - En **Transform**, pulsa **Reset** (posición 0,0,0; escala 1).
2. **Agregar los componentes de superficie y recolección 2D**
   - Con NavMesh seleccionado:
     - **Add Component → Navigation Surface**.
     - **Add Component → Navigation CollectSources2d**.
   - En **Navigation Collect Sources 2D**, **activa**:
     - **Rotate Surface To XY** (esto orienta el NavMesh al plano 2D XY).
3. **Marcar áreas Walkable y No Walkable**
   - En tu **Grid** (o el Tilemap base de piso, por ejemplo Grid/Flat):
     - Selecciona el **Tilemap o GameObject** que representa el piso (caminar).
     - **Add Component → Navigation Modifier**.
     - Marca **Override Area** y elige **Area: Walkable**.
   - En tu **Tilemap de obstáculos** (por ejemplo Grid/Elevations, muros o zonas bloqueadas):
     - **Add Component → Navigation Modifier**.
     - Marca **Override Area** y elige **Area: Not Walkable**.
4. **Hornear (Bake) el NavMesh**
   - Selecciona el objeto **NavMesh** con **NavMesh Surface**.
   - Pulsa **Bake** en el componente.
   - Deberías ver la **superficie azul** (Walkable) generada sobre tu Tilemap.

3) Preparar el enemigo con NavMeshAgent (2D)

1. **Configurar el Enemigo**

- Selecciona **EnemyTorchRed** en la jerarquía.
- **Add Component → Nav Mesh Agent**.
- Ajustar Base Offset, Radius y Height

4) Script de movimiento (mejorado)

```
using UnityEngine;
using UnityEngine.AI;

public class EnemyMovement : MonoBehaviour
{
    public float speed = 7;
    public Transform targetTransform;
    private Rigidbody2D rb2d;
    NavMeshAgent navMeshAgent;

    void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
        navMeshAgent = GetComponent<NavMeshAgent>();
        // Evita que el enemy rote para ir directo al punto
        navMeshAgent.updateRotation = false;
        // Evita que se maneje el eje z
        navMeshAgent.updateUpAxis = false;
    }

    void Update()
    {
        //rb2d.MovePosition(Vector2.MoveTowards(transform.position,
targetTransform.position, speed*Time.deltaTime));
        navMeshAgent.SetDestination(targetTransform.position);
    }
}
```

Cuarta Parte

Añadir las Animaciones al Enemigo

1. Abrir el controlador de la animación del Enemy.
2. Crear la transición de Idle a Run y de Run a Idle.
3. Crear el parámetro isRunning de tipo bool.
4. Seleccionar cada transición y deshabilitar Has Exit Time y poner a 0 Transition Duration
5. Agregar las condiciones para cada transición, de Idle a Run isRunnig debe ser True y de Run a Idle isRunnig debe ser false.
6. Mdificar el código del Enemy.

```
using UnityEngine;
using UnityEngine.AI;

public class EnemyMovement : MonoBehaviour
{
    public float speed = 7;
    public Transform targetTransform;
    private Rigidbody2D rb2d;
    NavMeshAgent navMeshAgent;
    Animator animator;

    void Start()
    {
        rb2d = GetComponent<Rigidbody2D>();
        navMeshAgent = GetComponent<NavMeshAgent>();
        animator = GetComponent<Animator>();
        // Evita que el enemy rote para ir directo al punto
        navMeshAgent.updateRotation = false;
        // Evita que se maneje el eje z
        navMeshAgent.updateUpAxis = false;
    }

    void Update()
    {
        //rb2d.MovePosition(Vector2.MoveTowards(transform.position,
targetTransform.position, speed*Time.deltaTime));
        navMeshAgent.SetDestination(targetTransform.position);
        AdjustAnimationsAndRotation();
    }

    public void AdjustAnimationsAndRotation()
    {
        bool isMoving = navMeshAgent.velocity.sqrMagnitude > 0.01f;
        animator.SetBool("isRunning", isMoving);

        if (navMeshAgent.desiredVelocity.x > 0.01f)
```

```
            transform.localScale = new Vector3(1, 1, 1);

        if (navMeshAgent.desiredVelocity.x < -0.01f)
            transform.localScale = new Vector3(-1, 1, 1);

    }
}
```