Guía de Laboratorio 9

Primera parte
Crear movimiento básico NPC

1. Preparar el Sprite

1. En el **Panel del Proyecto**, ubica el sprite en la ruta: Assets/Images/TinySwords/Factions/Knights/Troops/Pawn/Blue/Pawn_Blue.png
2. Selecciona el sprite y, en el **Inspector**, establece el valor de **Pixels per Unit** en **64**.
3. En el campo **Sprite Mode**, elige la opción **Multiple** para indicar que contiene varias imágenes.
4. Haz clic en el botón **Sprite Editor**.
    - En el editor, selecciona la opción **Slice → Grid By Cell Count**.
    - Configura **Rows (filas)** en **6** y **Columns (columnas)** en **6**.
    - Aplica los cambios presionando el botón **Slice**.
5. Finalmente, guarda los ajustes con el botón **Apply** en la esquina superior derecha del **Inspector**.

Repite el procedimiento para “Pawn/Puerple”, “Pawn/Red” y “Pawn/Yellow”.

2. Crear el objeto en el escenario

1. En el **Panel del Proyecto**, abre la carpeta Assets/Images/TinySwords/Factions/Knights/Troops/Pawn/Blue/.
2. Arrastra el sprite **Pawn_Blue_0** hacia la **Jerarquía** para colocarlo en el escenario.
3. Asigna el nombre **Pawn_Blue** al nuevo objeto en la jerarquía para mantener un orden coherente.

3. Asignar componentes

1. En la jerarquía, selecciona el objeto **EnemyTorchRed**.
2. En el **Inspector**, toma nota de los siguientes componentes:
    - **Box Collider 2D**
    - **Rigidbody 2D**
    - **Animator**
    - **Nav Mesh Agent**
3. Regresa al objeto **Pawn_Blue** y, uno por uno, agrega los mismos componentes desde **Add Component** (si lo prefieres, también puedes copiarlos directamente desde EnemyTorchRed y pegarlos en Pawn_Blue).

4. Configurar el Animator

1. En el componente **Animator** del objeto **Pawn_Blue**, localiza el campo **Controller**.
2. Haz clic en el círculo a la derecha del campo y selecciona **None** para dejarlo vacío.
   (Más adelante se asignará un controlador de animaciones personalizado para este personaje.)

5. Crear y asignar el script de comportamiento

1. En el **Panel del Proyecto**, crea una nueva carpeta (si no existe) en la ruta:
   Assets/Scripts/Npc/
2. Dentro de esa carpeta, crea un nuevo script llamado **Neutral.cs**.
3. Abre el script y copia el código del archivo **EnemyTorchRed.cs** (ubicado en Assets/Scripts/Enemy/ o la ruta que corresponda).
4. Guarda los cambios y vuelve a Unity.
5. Selecciona el objeto **Pawn_Blue** y arrastra el script **Neutral.cs** hacia el Inspector para asignarlo al objeto.

Segunda Parte
Animaciones del NPC

1. Organización de carpetas

1. En el **Panel del Proyecto**, navega a la ruta:
   Assets/Animations/Pawn/
2. Dentro de esa carpeta, crea las siguientes subcarpetas para organizar las animaciones de cada color:
   - PawnBlue
   - PawnPurple
   - PawnRed
   - PawnYellow
3. Estas carpetas servirán para guardar las animaciones correspondientes a cada tipo de Pawn.

2. Creación de animaciones

a. Pawn Blue

1. En el **Panel del Proyecto**, abre la carpeta:
   Assets/Images/TinySwords/Factions/Knights/Troops/Pawn/Blue/
2. Selecciona los sprites desde **Pawn_Blue_0** hasta **Pawn_Blue_5** (usa Shift para seleccionar varios a la vez).
3. Arrastra esta selección hacia la **Jerarquía** o directamente al **Panel de Proyecto** para crear una nueva animación.
4. Cuando Unity solicite un nombre y ubicación, asígnale:

- **Nombre:** Idle
- **Ubicación:** Assets/Animations/Pawn/PawnBlue/

5. Repite el procedimiento con los sprites **Pawn_Blue_6** a **Pawn_Blue_11** para crear la animación **Run**, guardándola en la misma carpeta (PawnBlue).
6. Una vez generadas las animaciones, elimina los objetos temporales creados automáticamente en la **Jerarquía** (por ejemplo, "Pawn_Blue_0").

b. Repetir para los demás Pawn

Repite exactamente el mismo proceso anterior para los siguientes personajes:

- **PawnPurple** → sprites Pawn_Purple_0 a Pawn_Purple_11, carpeta Assets/Animations/Pawn/PawnPurple/
- **PawnRed** → sprites Pawn_Red_0 a Pawn_Red_11, carpeta Assets/Animations/Pawn/PawnRed/
- **PawnYellow** → sprites Pawn_Yellow_0 a Pawn_Yellow_11, carpeta Assets/Animations/Pawn/PawnYellow/

Cada uno debe tener sus dos animaciones: **Idle** y **Run**.

3. Creación del controlador de animaciones (Animator Controller)

a. Configuración base

1. En el **Panel del Proyecto**, dentro de la carpeta Assets/Animations/Pawn/PawnBlue/, haz clic derecho y selecciona:
   Create → Animator Controller
2. Asigna el nombre: PawnBlueController.
3. Abre el controlador con doble clic para editarlo en el **Animator Window**.

b. Agregar estados de animación

1. Arrastra las animaciones Idle y Run al panel del **Animator**.
2. Verifica que Idle quede como **estado predeterminado** (indicado por la flecha naranja).
3. Crea dos transiciones:
   - De **Idle → Run**
   - De **Run → Idle**

c. Configurar transiciones

1. Selecciona cada transición y, en el **Inspector**:
   - Desactiva la opción **Has Exit Time**.
   - Establece el valor de **Transition Duration** en **0** (para un cambio inmediato).
2. Crea un parámetro que controlará la animación:
   - En el panel **Parameters**, haz clic en el ícono **+ → Bool**.
   - Nombra el parámetro: isRunning.
3. Define las condiciones de transición:
   - **Idle → Run**: condición isRunning = true

- **Run → Idle**: condición isRunning = false

d. Repetir para los demás Pawn

Sigue los mismos pasos anteriores para crear y configurar los controladores de animación:

- PawnPurple_Controller
- PawnRed_Controller
- PawnYellow_Controller

Cada uno debe tener sus animaciones **Idle** y **Run**, con las mismas condiciones y parámetros configurados.

Tercera Parte
Configuración de Skins para NPCs

1. Preparar el código

1. Abre el script **Neutral.cs** ubicado en la ruta:
   Assets/Scripts/Npc/Neutral.cs
2. En la parte superior del código, asegúrate de incluir las siguientes dependencias:

```
using UnityEditor.Animations;
using UnityEngine;
using UnityEngine.AI;
```

3. Dentro de la clase Neutral, agrega una etiqueta de encabezado y un **array de controladores de animación** para gestionar las distintas skins.
   Copia y pega la siguiente línea dentro de la clase:

```
[Header("Skin")]
public AnimatorController[] animatorControllers;
```

Esta etiqueta permitirá visualizar en el **Inspector** un grupo ordenado llamado “Skin”, donde se podrán asignar los distintos controladores de animación.

2. Configurar los Animator Controllers en el Inspector

1. Guarda el script y regresa a **Unity**.
2. Selecciona el objeto **Pawn_Blue** (o el NPC que utilice este script).
3. En el **Inspector**, dentro del componente **Neutral**, verás una nueva sección llamada **Skin** con el campo **Animator Controllers (Size)**.
4. Cambia el valor de **Size** a **4**, lo que creará cuatro espacios dentro del array.
5. Asigna los controladores de animación correspondientes en el siguiente orden:

1. **PawnBlue_Controller**
2. **PawnPurple_Controller**
3. **PawnRed_Controller**
4. **PawnYellow_Controller**

El orden es importante, ya que se utilizará para referenciar las skins según el índice en el array.

3. Implementar la selección de Skin

Agrega la siguiente sección de código dentro de la clase Neutral, debajo de la declaración del array:

```
    public NPCSkin selectedSkin;
    public enum NPCSkin
    {
        Blue,
        Purple,
        Red,
        Yellow
    }
```

Esto permitirá seleccionar fácilmente el color del personaje desde un menú desplegable en el Inspector.

4. Aplicar el Skin en tiempo de ejecución

Agrega el siguiente método dentro del script para asignar el controlador de animación correcto al iniciar el juego:

```
    public void ApplySkin()
    {
        int skinIndex = (int)selectedSkin;
        animator.runtimeAnimatorController = animatorControllers[skinIndex];
    }
```

Luego, dentro del método Start(), añade la llamada al método ApplySkin() justo después de obtener las referencias de los componentes:

```
    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;

        ApplySkin();
    }
```

5. Código final del script Neutral.cs

Tu archivo completo debería quedar así:

```
using UnityEditor.Animations;
using UnityEngine;
using UnityEngine.AI;

public class Neutral : MonoBehaviour
{
    private Rigidbody2D rb2d;
    public Transform targetTransform;
    NavMeshAgent navMeshAgent;
    Animator animator;

    [Header("Skin")]
    public AnimatorController[] animatorControllers;
    public NPCSkin selectedSkin;
    public enum NPCSkin
    {
        Blue,
        Purple,
        Red,
        Yellow
    }
    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;

        ApplySkin();
    }

    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);

    }

    public void ApplySkin()
    {
        int skinIndex = (int)selectedSkin;
        animator.runtimeAnimatorController = animatorControllers[skinIndex];
    }
}
```

Cuarta Parte

Inicialización y Configuración de Rutas del NPC

1. Agregar la configuración de movimiento al script

1. Abre el archivo **Neutral.cs** en la ruta:

   Assets/Scripts/Npc/Neutral.cs
2. Dentro de la clase Neutral, agrega el siguiente bloque de código:

```
    [Header("Movement Type")]
    public MovementType movementType;
    public enum MovementType
    {
        Path,
        RandomMovement
    }

    [Header("Path")]
    public Transform[] pathPoints;
    public float waitTimeInPoint = 3;
    private int indexPath = 0;
```

2. Agregar el método de seguimiento de ruta

1. Abre el script **Neutral.cs** en la ruta:
   Assets/Scripts/Npc/Neutral.cs
2. Dentro de la clase Neutral, agrega el siguiente método al final del archivo (antes de la última llave }):

```
    IEnumerator FollowPath()
    {
        while (true)
        {
            if (pathPoints.Length > 0)
            {
                navMeshAgent.SetDestination(pathPoints[indexPath].position);

                while (!navMeshAgent.pathPending && navMeshAgent.remainingDistance >
0.1f)
                {
                    yield return null;
                }

                yield return new WaitForSeconds(waitTimeInPoint);

                indexPath = (indexPath + 1) % pathPoints.Length;
            }
            yield return null;
        }
    }
```

3. Activar la corrutina

Para que el NPC comience a moverse al iniciar el juego, se debe llamar a la corrutina desde el método Start().

Agrega la siguiente línea dentro de Start() **después** de la llamada a ApplySkin():

```
    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;

        ApplySkin();
```

```
        if (movementType == MovementType.Path)
        {
            StartCoroutine(FollowPath());
        }
    }
```

4. Creación de los puntos de ruta en la escena

Paso 1: Crear el grupo principal

1. En la jerarquía de Unity, haz clic derecho → **Create Empty**.
2. Nombra este objeto **Paths**.
   (Este servirá como contenedor principal de todas las rutas del escenario.)

Paso 2: Crear el grupo para el Pawn azul

1. Haz clic derecho sobre el objeto **Paths** → **Create Empty**.
2. Nómbralo **PathsPawnBlue**.
   (Este grupo contendrá los puntos específicos de la ruta del Pawn azul.)

Paso 3: Crear los puntos de recorrido

1. Haz clic derecho sobre **PathsPawnBlue** → **Create Empty**.
   - Nombra el primer punto **Point1**.
2. Repite el proceso para crear **Point2**.
3. Coloca los puntos **Point1** y **Point2** en el escenario moviéndolos con la herramienta **Move (W)**:
   - **Point1:** posición inicial del recorrido.
   - **Point2:** posición final o destino del NPC.

Puedes ver su ubicación activando el "gizmo" del objeto Empty; Unity mostrará un pequeño ícono en su posición.

5. Asignar los puntos de ruta al NPC

1. Selecciona el objeto **Pawn_Blue** en la jerarquía.
2. En el **Inspector**, busca la sección **Movement Type**.
   - Asegúrate de que el campo **Movement Type** esté configurado como **Path**.
3. En la sección **Path**, cambia el tamaño del arreglo **Path Points** a **2**.
4. Arrastra los objetos **Point1** y **Point2** desde la jerarquía hasta los espacios del arreglo en este orden:
   Element 0 → **Point1**
   Element 1 → **Point2**
5. Ajusta el campo **Wait Time In Point** (por ejemplo, **3 segundos**).