Guía de Laboratorio 3

Primera Parte
Cambiar entre las animaciones

Transiciones entre las animaciones

Para que el personaje cambie automáticamente entre las animaciones **Idle** y **Walk**, debemos configurar transiciones en el **Animator**.

1. Abre el **Animator Controller** del personaje:
    - En la carpeta **PlayerWarriorRed**, haz doble clic sobre **PlayerAnimController**.
    - Esto abrirá la ventana **Animator**, donde ya están las animaciones Idle y Walk.
2. En la ventana Animator, selecciona la pestaña **Parameters** (parte superior izquierda).
    - Presiona el ícono + y selecciona **Float**.
    - Crea dos parámetros con los nombres:
        - **Vertical**
        - **Horizontal**
3. Ahora debemos crear transiciones:
    - Haz clic derecho sobre el estado **Idle**, selecciona **Make Transition** y conéctalo a **Walk**.
    - Haz esto dos veces, porque configuraremos una transición para cada parámetro (Horizontal y Vertical).
4. Configuración de condiciones:
    - Selecciona la primera transición de **Idle → Walk**.
    - En el **Inspector**, busca la sección **Conditions** y añade una nueva condición Greater:
        - **Horizontal > 0.1**.
    - Selecciona la segunda transición de **Idle → Walk**.
    - Añade la condición Greater:
        - **Vertical > 0.1**.
5. Crear el retorno a Idle:
    - Haz clic derecho en **Walk**, selecciona **Make Transition** y conéctalo hacia **Idle**.
    - En el Inspector, añade la condición Less:
        - **Horizontal < 0.1**
        - **Vertical < 0.1**.

Codificación para usar las transiciones

Ahora programaremos la lógica para controlar esas transiciones desde el script del personaje.
Edita el script **Player.cs** con el siguiente código:

```
using UnityEngine;

public class Player : MonoBehaviour
{
    public float speed = 5;

    Rigidbody2D rb2D;

    Vector2 movementInput;

    Animator animator;

    void Start()
    {
        rb2D = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
    }

    void Update()
    {
        movementInput.x = Input.GetAxisRaw("Horizontal");
        movementInput.y = Input.GetAxisRaw("Vertical");

        movementInput = movementInput.normalized;

        animator.SetFloat("Horizontal", Mathf.Abs(movementInput.x));
        animator.SetFloat("Vertical", Mathf.Abs(movementInput.y));
    }

    private void FixedUpdate()
    {
        rb2D.linearVelocity = movementInput * speed;
    }
}
```

Explicación:

- **Mathf.Abs()** asegura que los valores sean positivos (nos interesa solo si hay movimiento, no la dirección).
- Con animator.SetFloat() enviamos esos valores a los parámetros creados en el Animator.

Mejorar la apariencia del cambio de animaciones

Si probamos el juego en este punto, notaremos que a veces la animación tarda un poco en cambiar. Lo solucionaremos modificando las transiciones.

1. Selecciona las transiciones que creaste (**Idle** → **Walk** y **Walk** → **Idle**) una por una.
2. En el **Inspector**, busca la sección **Settings**.
3. **Desactiva** la opción **Has Exit Time**.
    - Esto evita que Unity espere a que termine un ciclo de animación antes de cambiar.
4. Cambia el valor de **Transition Duration** a **0**.

- Esto hace que el cambio de animación sea inmediato, sin mezcla innecesaria.

Ejecución

1. Presiona **Play** en Unity.
2. El personaje estará en la animación **Idle** mientras no presiones teclas.
3. Al presionar las teclas de movimiento (flechas o WASD), el personaje cambiará a la animación **Walk** inmediatamente.
4. Al soltar las teclas, volverá a **Idle** sin demora.

Segunda Parte
Ajustar la dirección del personaje

¿Por qué necesitamos el Flip?

En muchos juegos 2D, los sprites del personaje están dibujados mirando solo hacia un lado (generalmente a la derecha). Para evitar tener que duplicar sprites para la izquierda, podemos usar una técnica llamada **flip horizontal**, que consiste en **espejar la escala en el eje X**.
De esta forma, el personaje "mirará" automáticamente hacia la derecha o la izquierda según la tecla que se presione.

Implementación del método CheckFlip()

Añadiremos una función llamada CheckFlip() en nuestro script **Player.cs**.

```
void CheckFlip()
    {
        // Verifica si el personaje se mueve a la derecha y está mirando a la
izquierda
        // o si se mueve a la izquierda y está mirando a la derecha
        if ((movementInput.x > 0 && transform.localScale.x < 0) ||
            (movementInput.x < 0 && transform.localScale.x > 0))
        {
            // Invierte el valor de la escala en X multiplicándolo por -1
            transform.localScale = new Vector3(
                transform.localScale.x * -1,  // Cambia el signo de X
                transform.localScale.y,      // Mantiene el valor de Y
                transform.localScale.z       // Mantiene el valor de Z
            );
        }
    }
```

Explicación paso a paso del código

1. movementInput.x > 0 → significa que el jugador se está moviendo hacia la derecha.

- Si al mismo tiempo la escala en X (transform.localScale.x) es negativa, quiere decir que el personaje está mirando a la izquierda, por lo que hay que voltearlo.

2. movementInput.x < 0 → significa que el jugador se está moviendo hacia la izquierda.
   - Si la escala en X es positiva, el personaje está mirando a la derecha, por lo que también hay que voltearlo.
3. transform.localScale = new Vector3(...) → aquí modificamos directamente la escala del objeto:
   - Multiplicamos la escala en X por -1 → el personaje se espeja horizontalmente.
   - La escala en Y y Z se mantienen igual para no distorsionar al personaje.

En resumen: **cuando el jugador cambia de dirección, el sprite se voltea automáticamente para mirar hacia ese lado.**

Uso de la función en el script

Para que esta función se ejecute correctamente, debemos llamarla en el método **Update()** del script Player:

```
void Update()
    {
        movementInput.x = Input.GetAxisRaw("Horizontal");
        movementInput.y = Input.GetAxisRaw("Vertical");

        movementInput = movementInput.normalized;

        animator.SetFloat("Horizontal", Mathf.Abs(movementInput.x));
        animator.SetFloat("Vertical", Mathf.Abs(movementInput.y));

        CheckFlip();
    }
```

5. Ejecución

1. Presiona **Play** en Unity.
2. Mueve al personaje a la derecha → el sprite mira hacia la derecha.
3. Mueve al personaje a la izquierda → el sprite se voltea y mira hacia la izquierda.
4. El cambio es inmediato y mantiene la coherencia de la animación Idle y Walk.