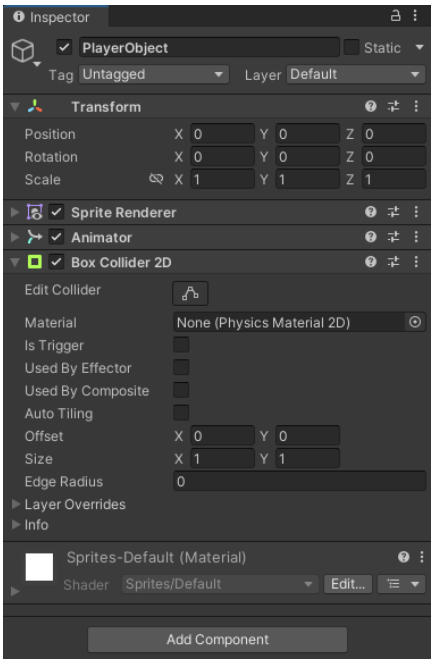
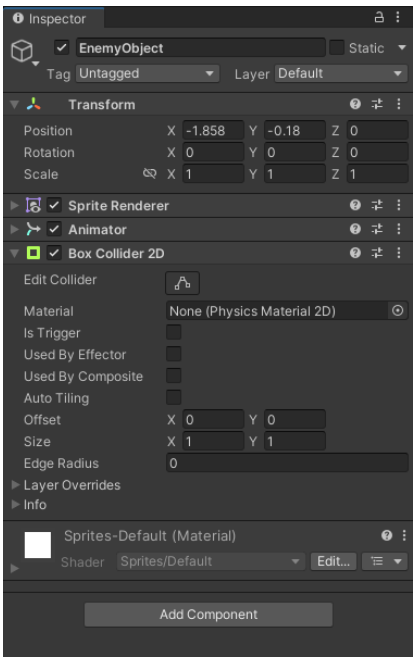
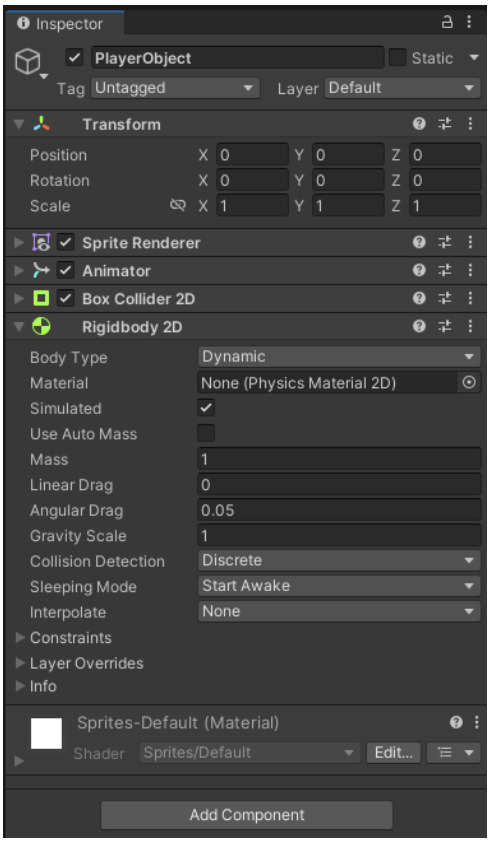
Lección 2 – Paso a paso

° Agregar un Box Collider 2D al PlayerObject y EnemyObject



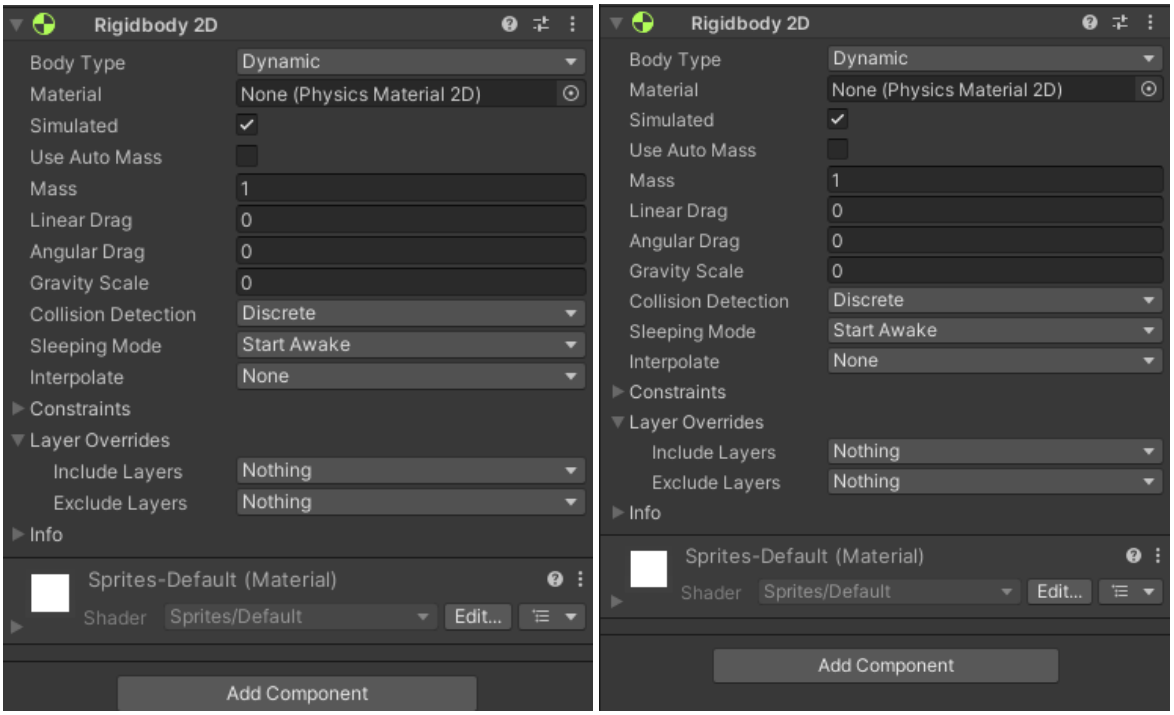


° Agregar un Box Rigidbody 2D al PlayerObjetc

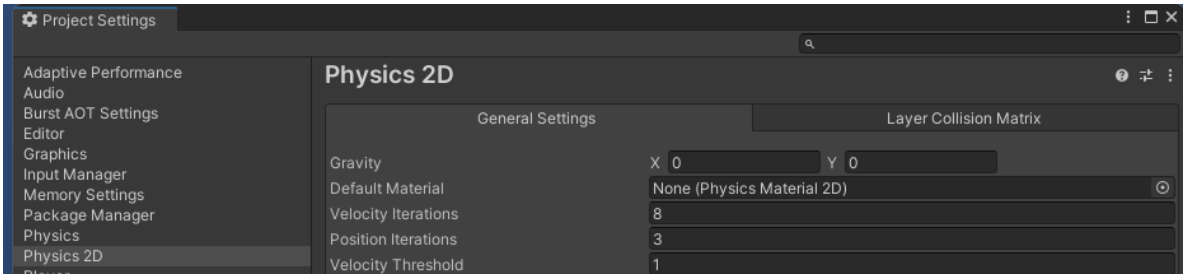


Agregar en el menú de RigidBody 2D los siguientes valores para las diferentes propiedades

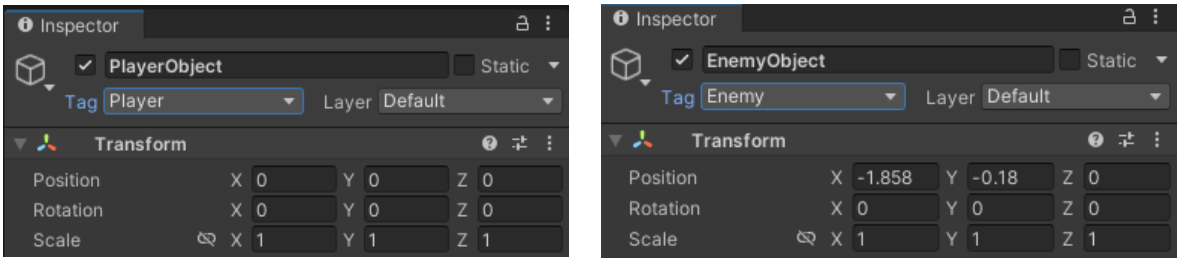
- Body Type Dynamic
- Mass 1
- Linear Drag 0
- Angular Drag 0
- Gravity Scale



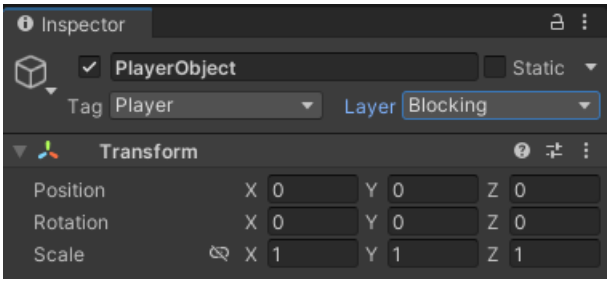
° Ajustar el valor de la gravedad Y de –9,81 a 0.



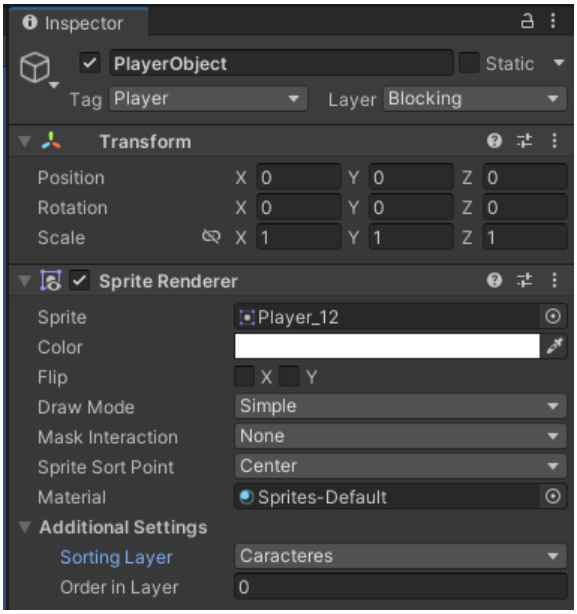
° Cambiar el nombre del tag del objeto PlayerObject a lo siguiente **Player** y EnemyObject a lo siguiente **Enemy**

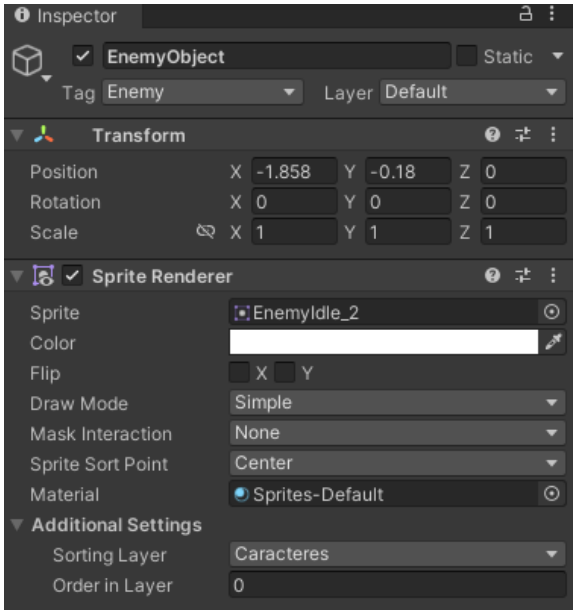


° Ajustar la capa de PlayerObject a **Blocking**

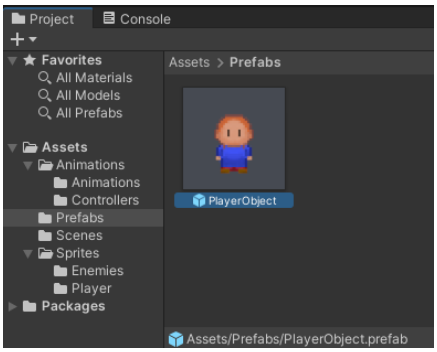


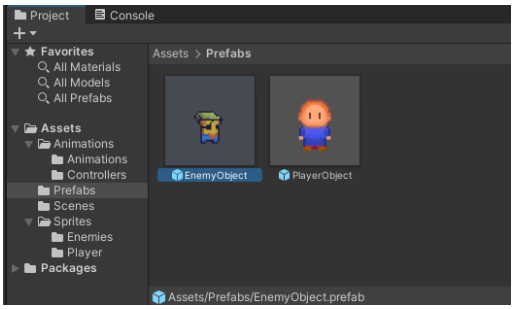
° agregar una capa de ordenamiento llamada **"Caracteres"** que usaremos para nuestro jugador y para los enemigos.



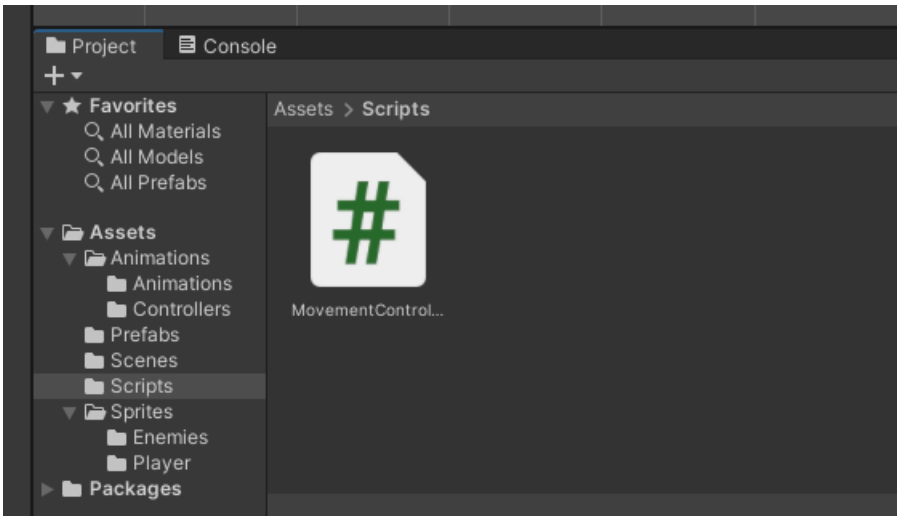


° Crear la carpeta de **prefabs** y mover ambos objetos a la capeta creada a su vez eliminar los objetos de la vista Hierarchy

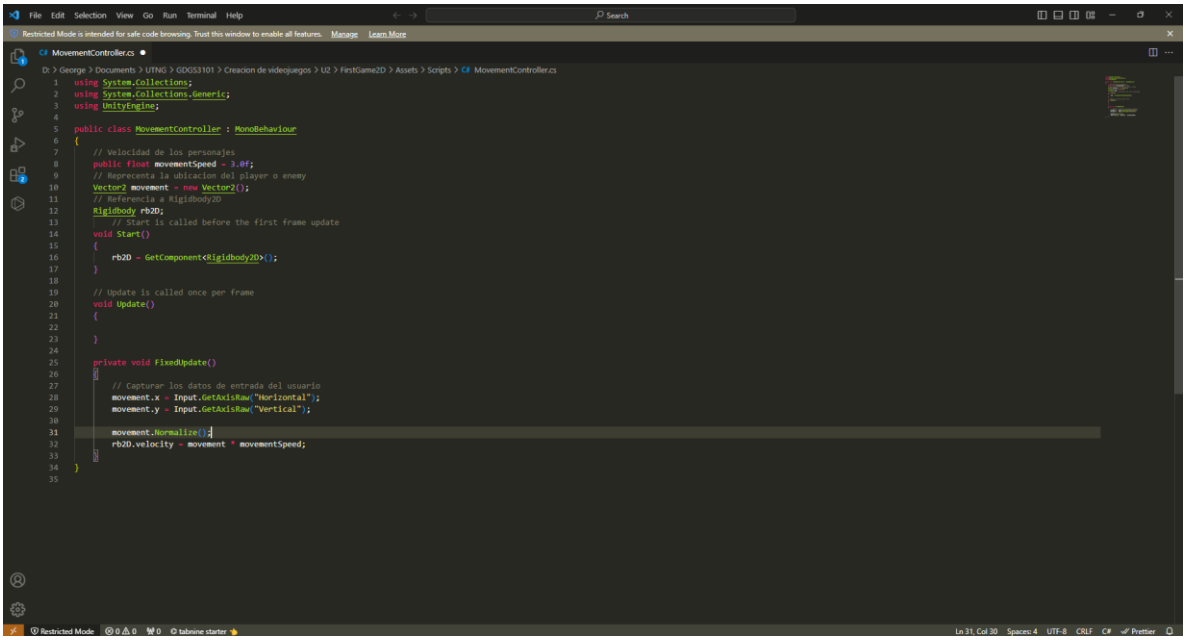




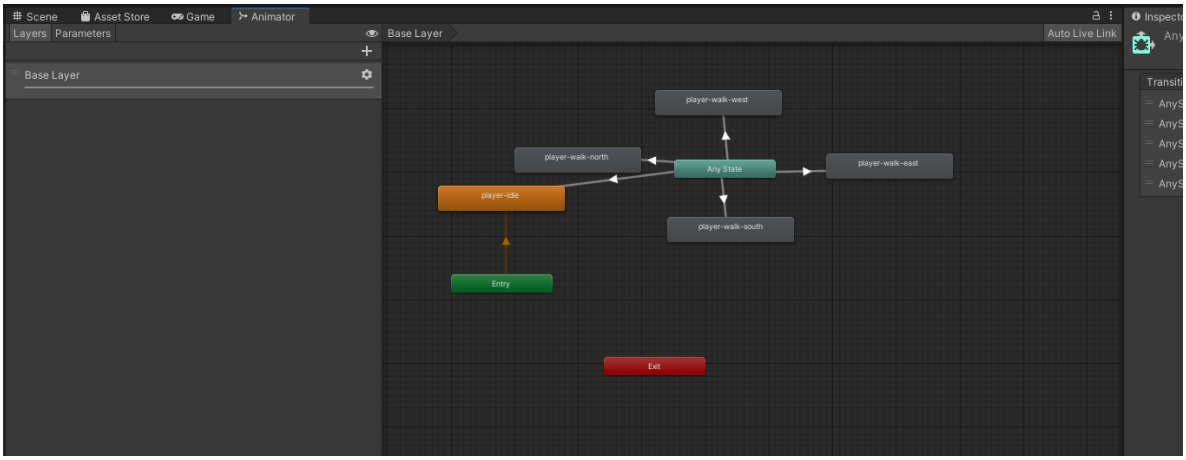
° Seleccione nuestro **PlayerObject** y cree un nuevo Script y nombrarlo como **"MovementController"** ahora Cree una nueva carpeta llamada **"Scripts"** y agregamos el script a nuestra carpeta de script



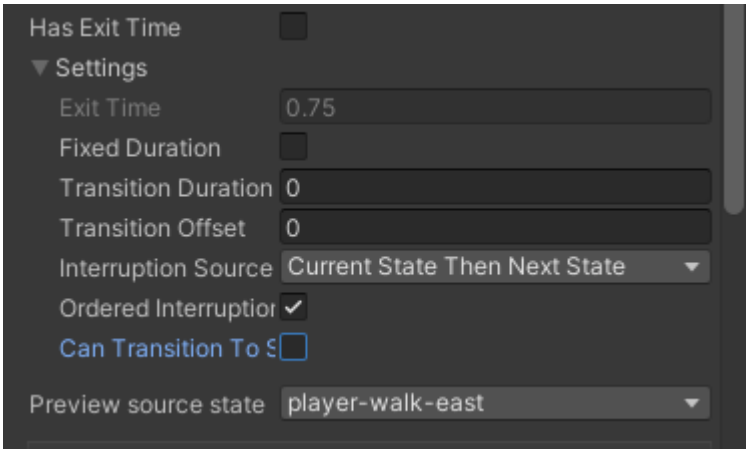
Agregamos el código correspondiente y agregamos la lógica para nuestro juego



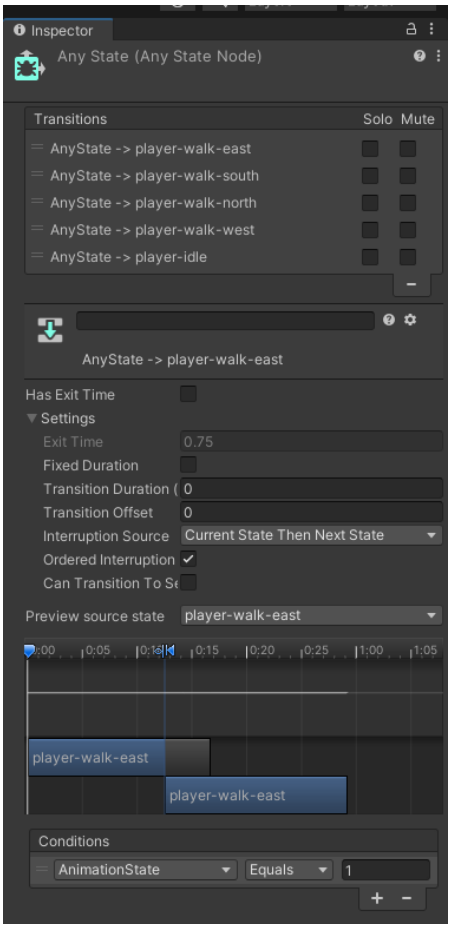
Debemos crear un total de cinco flechas de transición blancas que apunten desde Any State a los cuatro estados de animación de caminata del jugador y al jugador inactivo player-idle.



Agregamos los parámetros para la animacion



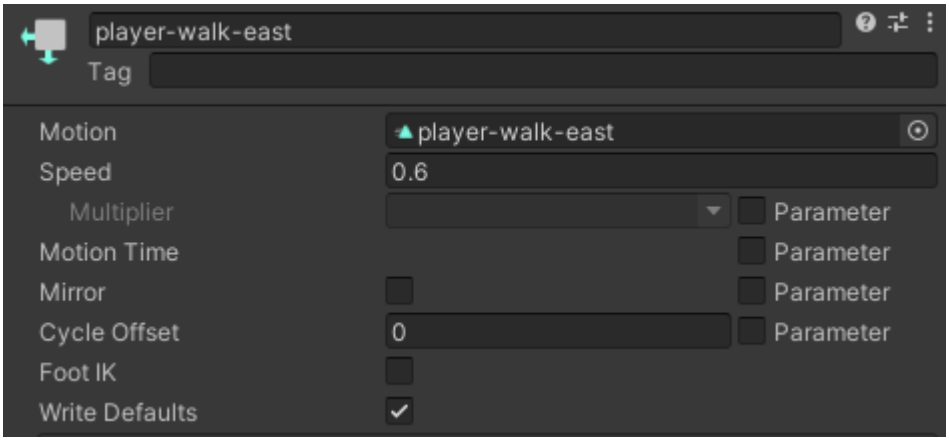
Seleccione la línea de transición blanca que conecta cualquier estado con el estado en la Inspector, cambie la configuración para que coincida y en la parte inferior del inspector, verá un área titulada "Conditions". Haga clic en el símbolo + en la parte inferior derecha y seleccione AnimationState ➤ Equals, e ingrese 1

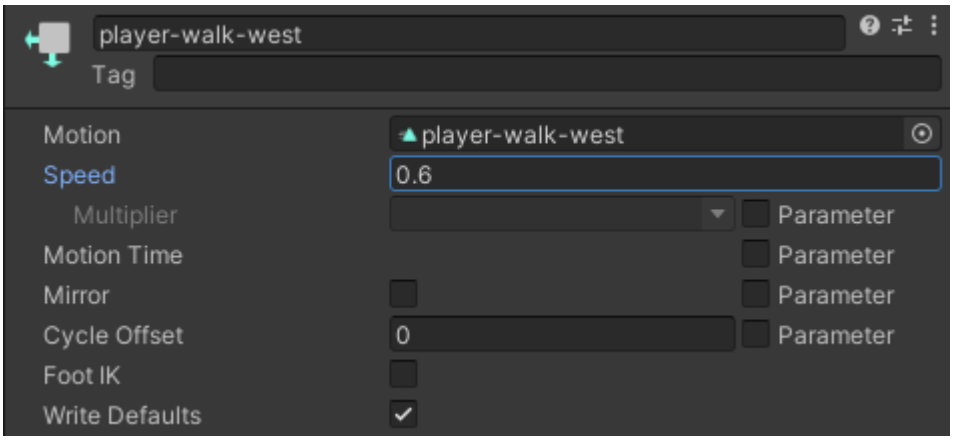


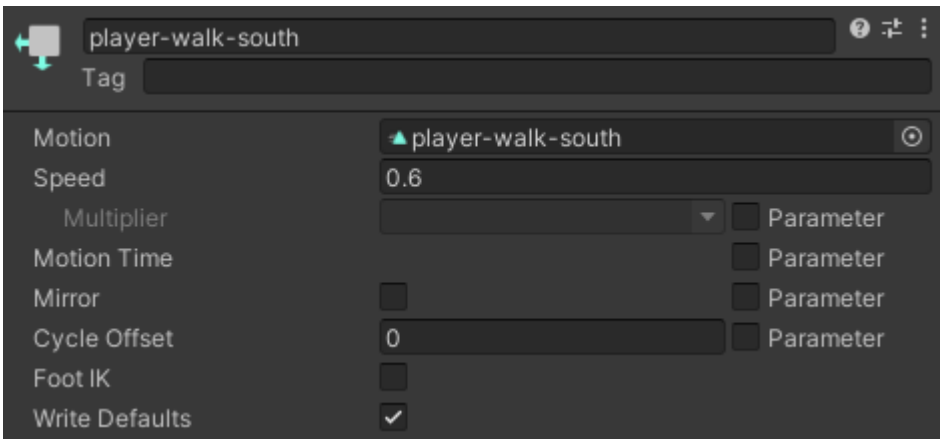
Lo siguiente que vamos a hacer es establecer que el parámetro AnimationState igual a 1 en nuestro script. Regrese a Visual Studio y abra nuestro Script MovementController.cs.

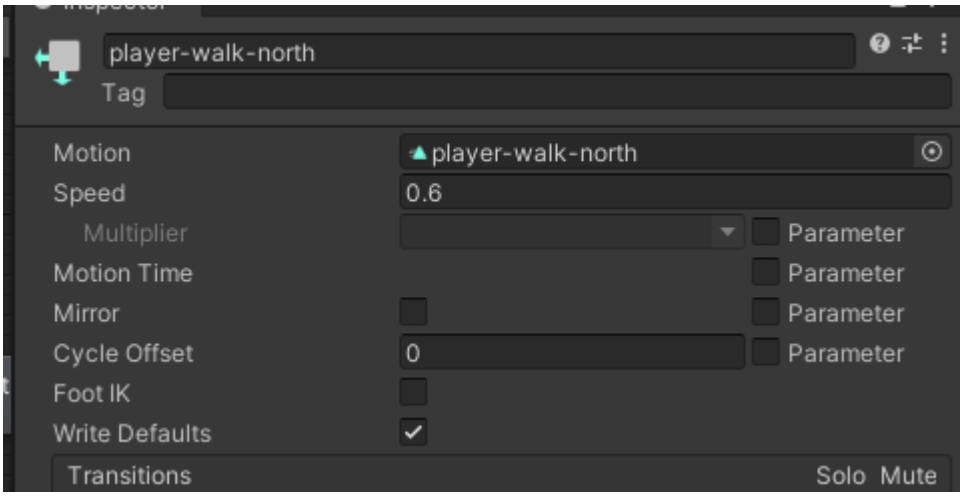
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MovementController : MonoBehaviour
{
    // Velocidad de los personajes
    public float movementSpeed = 3.0f;
    // Representa la ubicación del player o enemy
    Vector2 movement = new Vector2();
    // Referencia a Rigidbody2D
    Rigidbody2D rb2D;

    Animator animator; // Referencia a componente animator
    string animationState = "AnimationState";

    // Enumeracion de los estados
    enum CharStates
    {
        walkEast = 1,
        walkSouth = 2,
        walkWest = 3,
        walkNorth = 4,
        idleSouth = 5
    }

    // Start is called before the first frame update
    void Start()
    {
        // Establece el componente Rigidbody2D enlazado
        rb2D = GetComponent<Rigidbody2D>();
        // Establecer el valor del componente Animator del objeto ligado
        animator = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
        this.UpdateState();
    }

    private void UpdateState()
    {
        if (movement.x > 0) // ESTE
        {
            animator.SetInteger(animationState, (int)CharStates.walkEast);
        }
        else if (movement.x < 0) // OESTE
        {
            animator.SetInteger(animationState, (int)CharStates.walkWest);
        }
        else if (movement.y > 0) // NORTE
        {
            animator.SetInteger(animationState, (int)CharStates.walkNorth);
        }
        else if (movement.y < 0) // SUR
        {
            animator.SetInteger(animationState, (int)CharStates.walkSouth);
        }
        else
        {
            animator.SetInteger(animationState, (int)CharStates.idleSouth);
        }
    }

    private void FixedUpdate()
    {
        MoveCharacter(); // Método definido para ingresar la dirección
    }

    private void MoveCharacter()
    {
        // Capturar los datos de entrada del usuario
        movement.x = Input.GetAxisRaw("Horizontal");
        movement.y = Input.GetAxisRaw("Vertical");

        movement.Normalize();
        rb2D.velocity = movement * movementSpeed;
    }
}
```

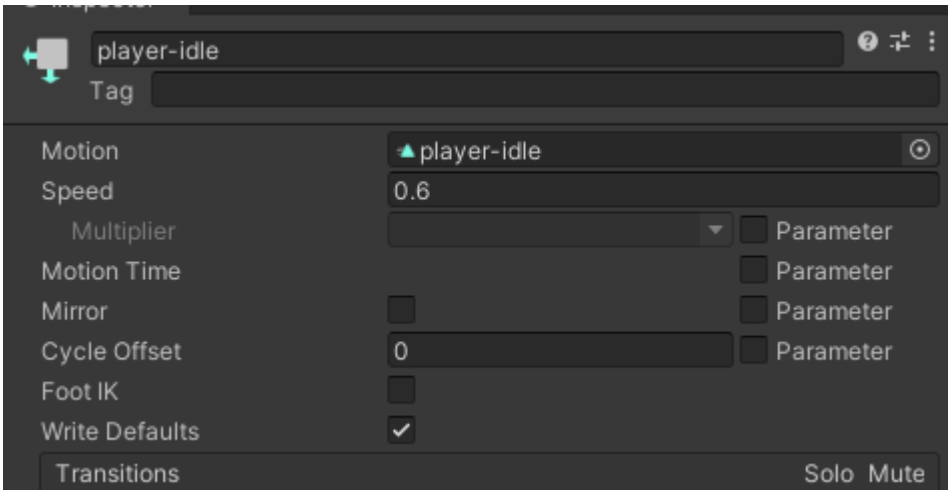
Seleccione cada estado de animación de caminata del jugador objeto y ajuste la velocidad a 0,6, y ajuste player-idle a 0.25. Esto hará que las animaciones de nuestros jugadores se vean bien.




player-walk-west
Tag
Motion
player-walk-west
Speed
0.6
Multiplier
Parameter
Motion Time
Parameter
Mirror
Parameter
Cycle Offset
0
Parameter
Foot IK
Write Defaults


player-walk-south
Tag
Motion
player-walk-south
Speed
0.6
Multiplier
Parameter
Motion Time
Parameter
Mirror
Parameter
Cycle Offset
0
Parameter
Foot IK
Write Defaults


player-walk-north
Tag
Motion
player-walk-north
Speed
0.6
Multiplier
Parameter
Motion Time
Parameter
Mirror
Parameter
Cycle Offset
0
Parameter
Foot IK
Write Defaults
Transitions
Solo
Mute


player-idle
Tag
Motion
player-idle
Speed
0.6
Multiplier
Parameter
Motion Time
Parameter
Mirror
Parameter
Cycle Offset
0
Parameter
Foot IK
Write Defaults
Transitions
Solo
Mute