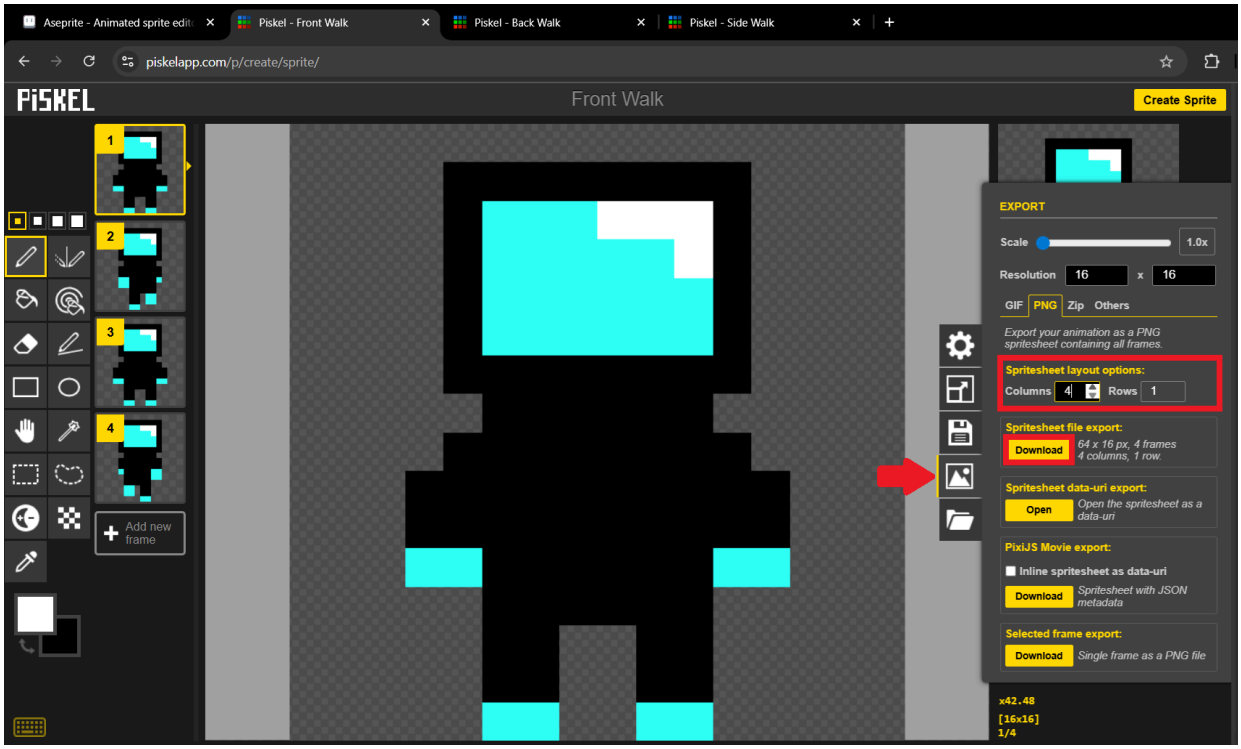
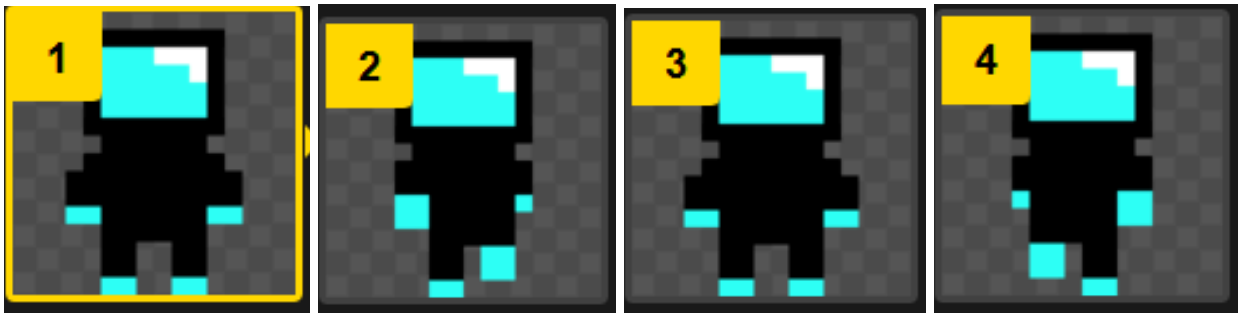
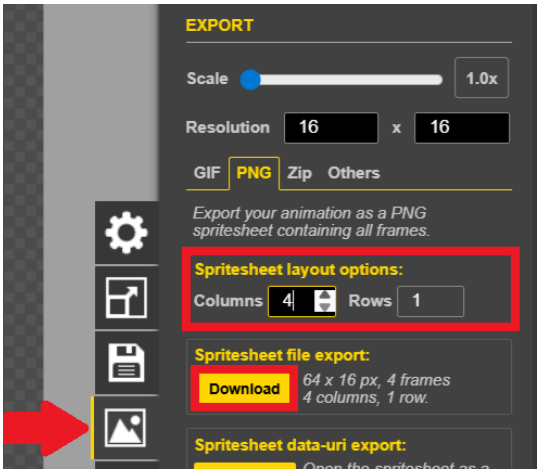
Front Walking Animation



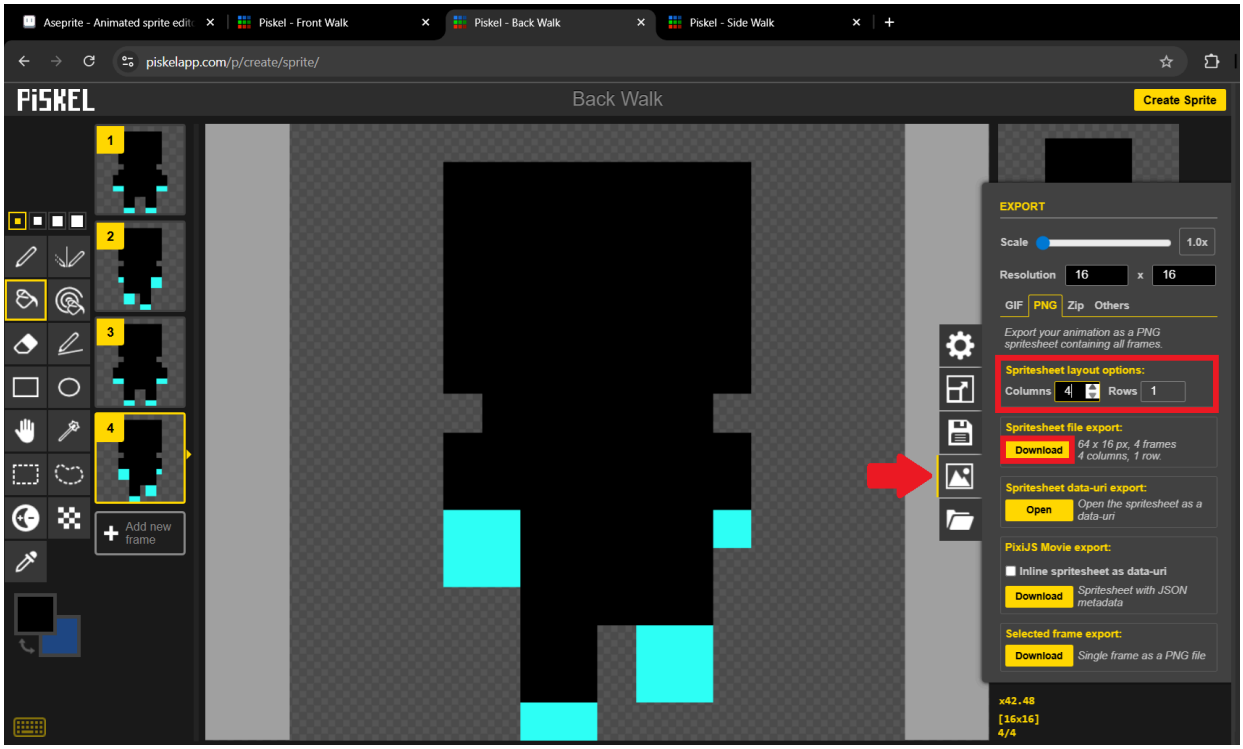


How to Export:

1. Select the **Image** Icon in the Right-Side Menu
2. Select **PNG**
3. Ensure that you have **1 Row**
4. Under **Spritesheet file export**, click **Download**



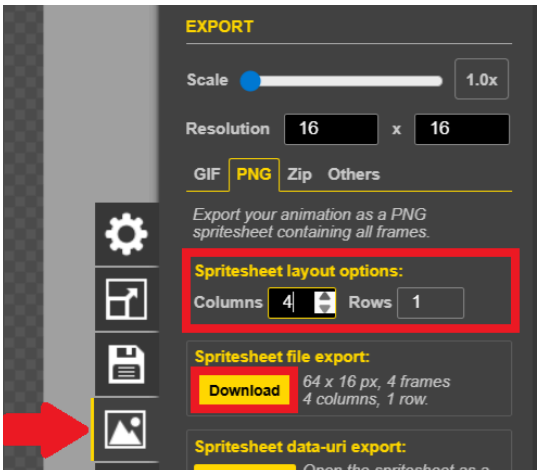
Back Walking Animation



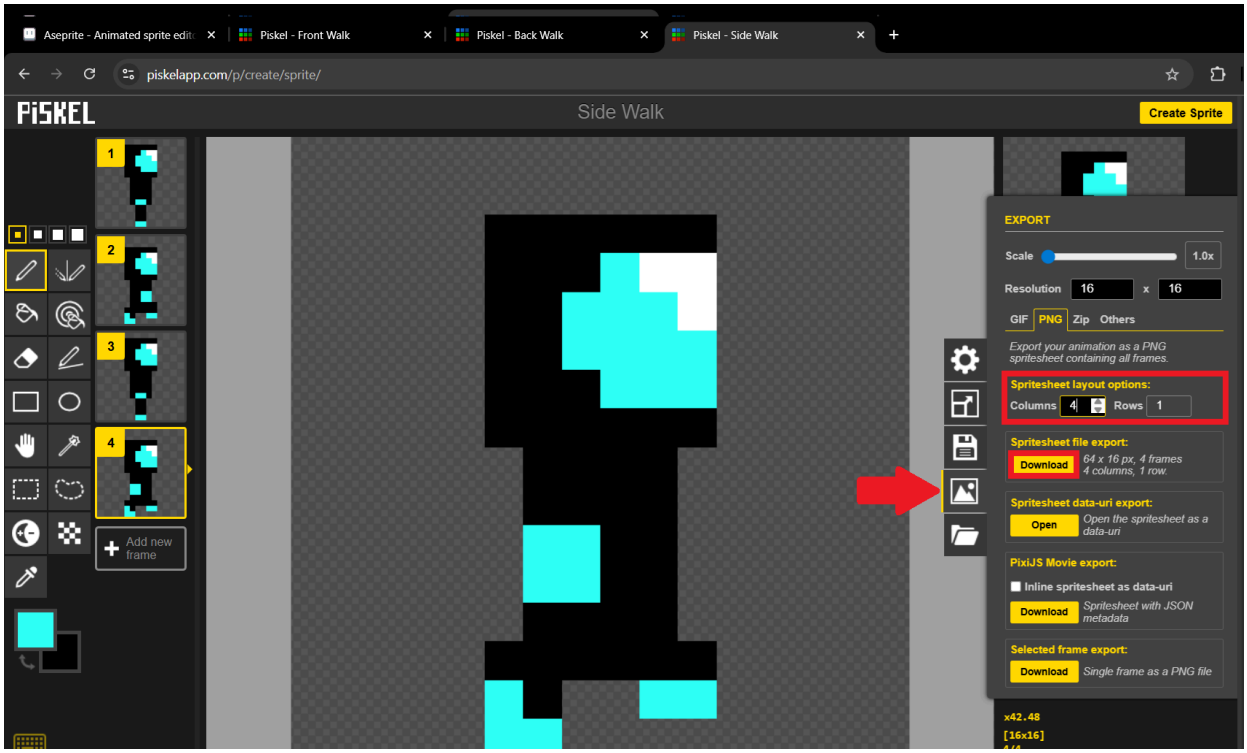
*Simply Color over the Front Walking Animation

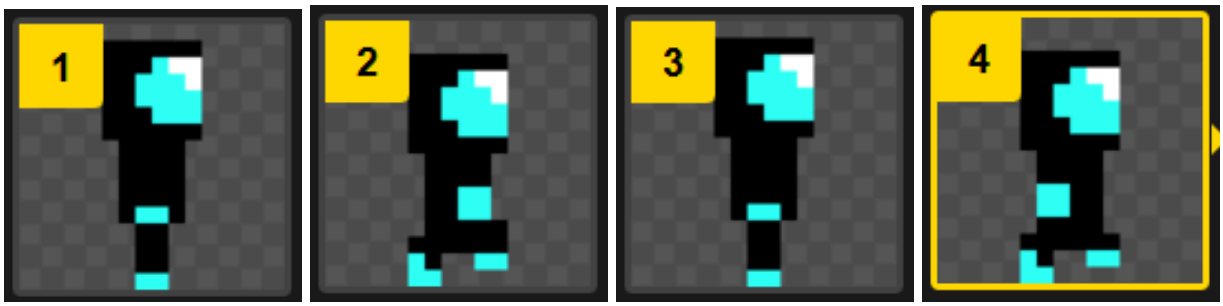
How to Export:

1. Select the **Image** Icon in the Right-Side Menu
2. Select **PNG**
3. Ensure that you have **1 Row**
4. Under **Spritesheet file export**, click **Download**



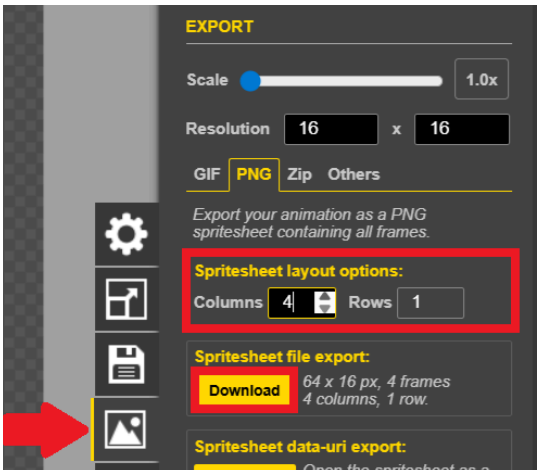
Side Walking Animation



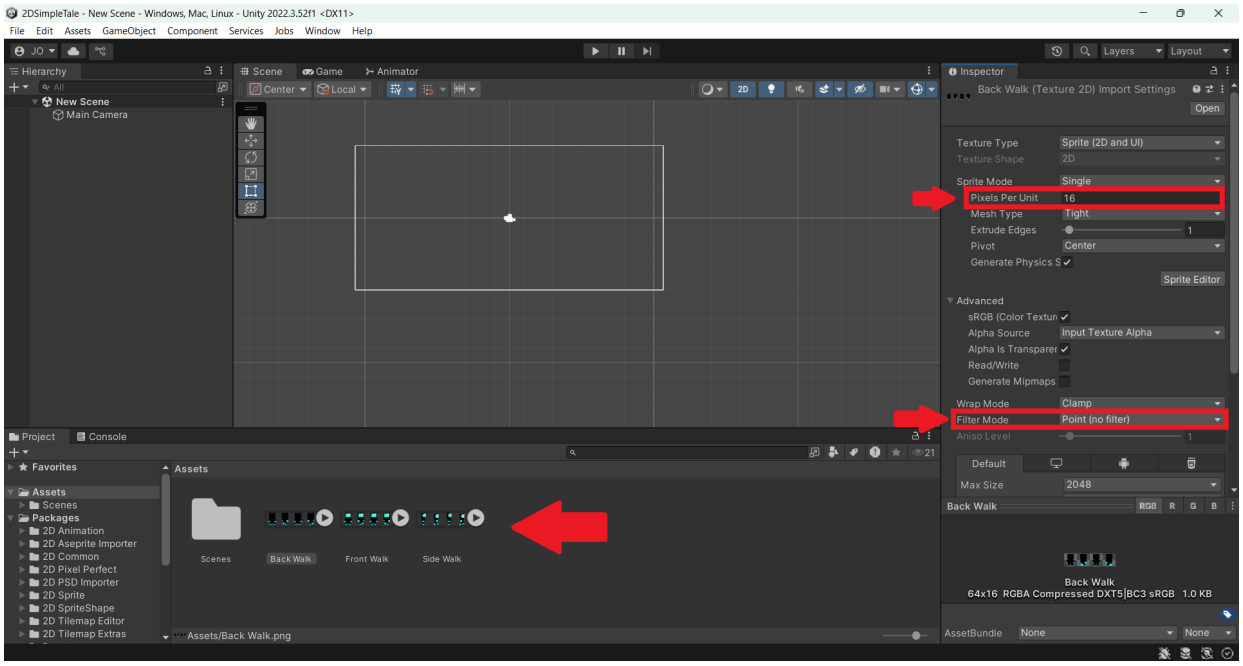


How to Export:

1. Select the **Image** Icon in the Right-Side Menu
2. Select **PNG**
3. Ensure that you have **1 Row**
4. Under **Spritesheet file export**, click **Download**



Import Animations into Unity



How to Import:

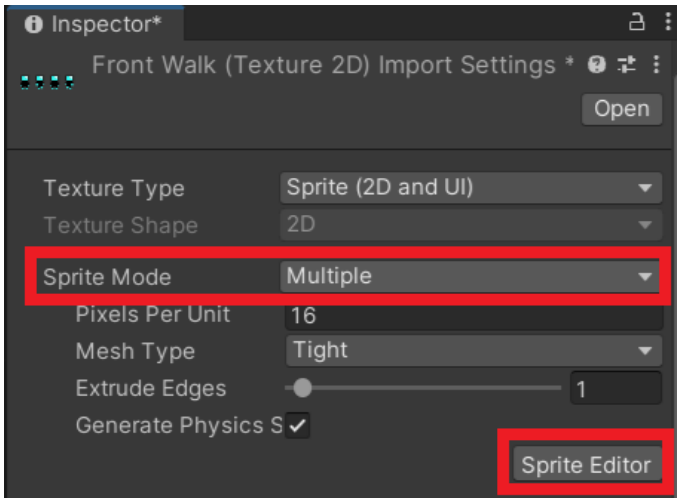
1. Right-Side inside the **Assets** section
2. Select **Import New Asset…**
3. Select all **Spritesheets** you made
4. Click **Import**

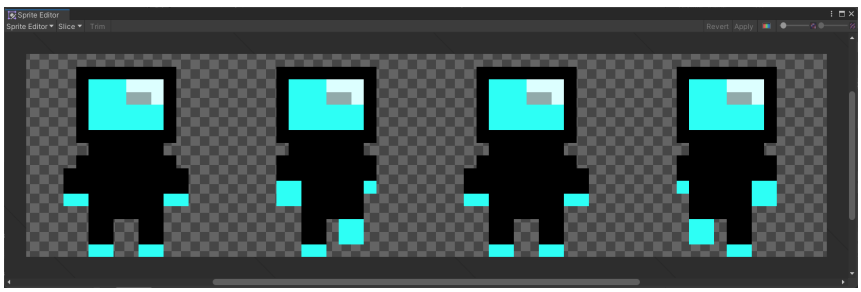
How to Remove Blur:

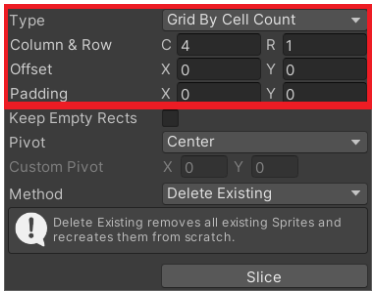
1. Select the **Spritesheets** in the **Assets**
2. In the **Inspector** window:
   - Pixels Per Unit: **16**
   - Filter Mode: **Point (no filter)**
3. Select **Apply**

How to Slice Spritesheet:

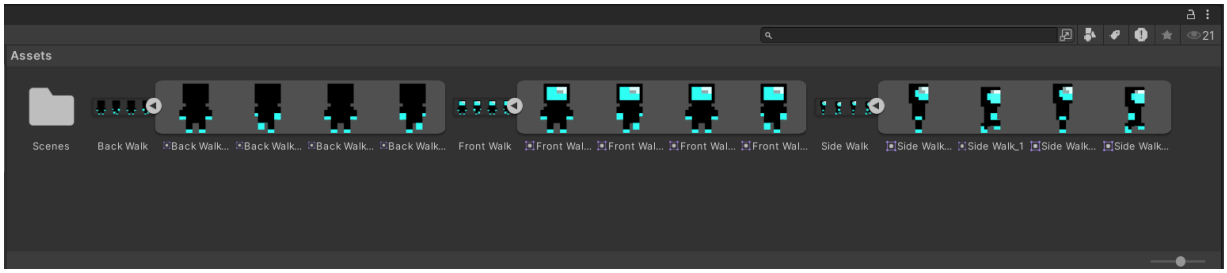
1. Select the **Spritesheets** in the **Assets**
2. In the **Inspector** window:
    - Sprite Mode: **Multiple**
3. Select the **Sprite Editor**
4. Select **Slice**
    - Type: **Grid By Cell Count**
    - C: **4** (Columns)
    - R: **1** (Rows)
5. Select **Apply**





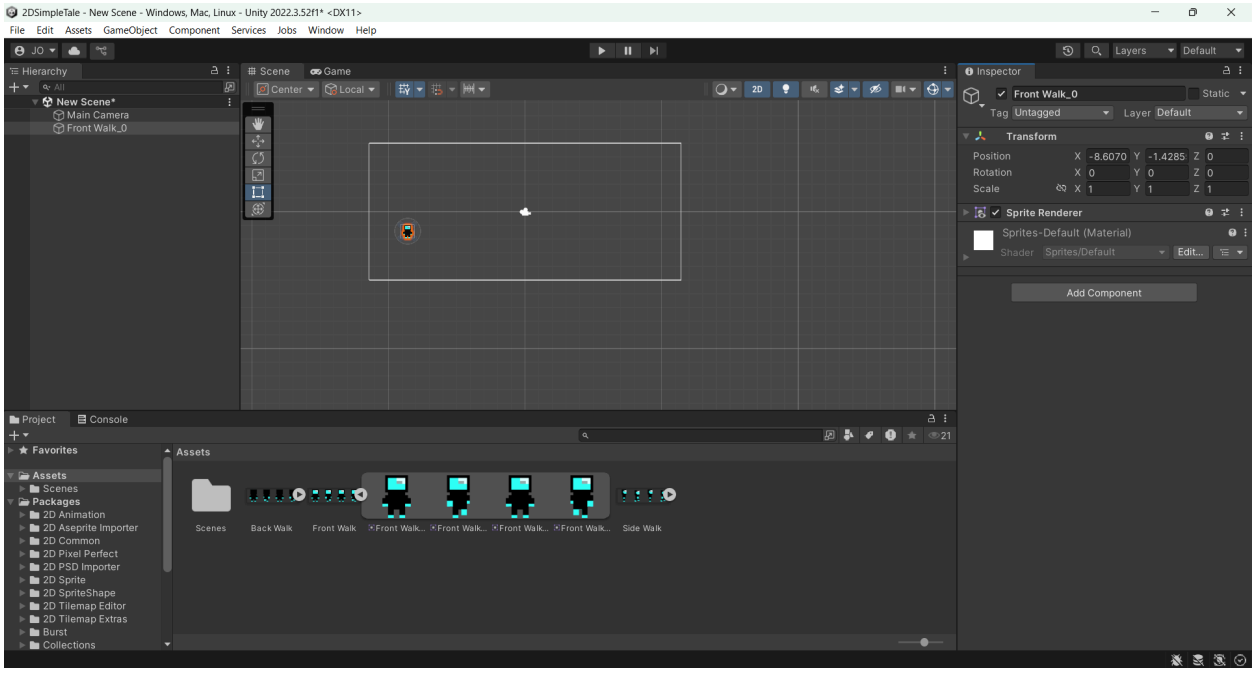


This is how your Spritesheets should look now.

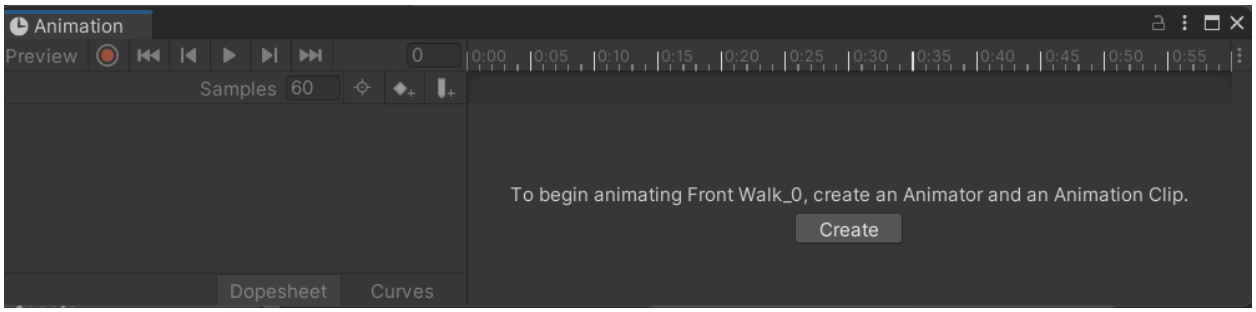


Set Up Animations

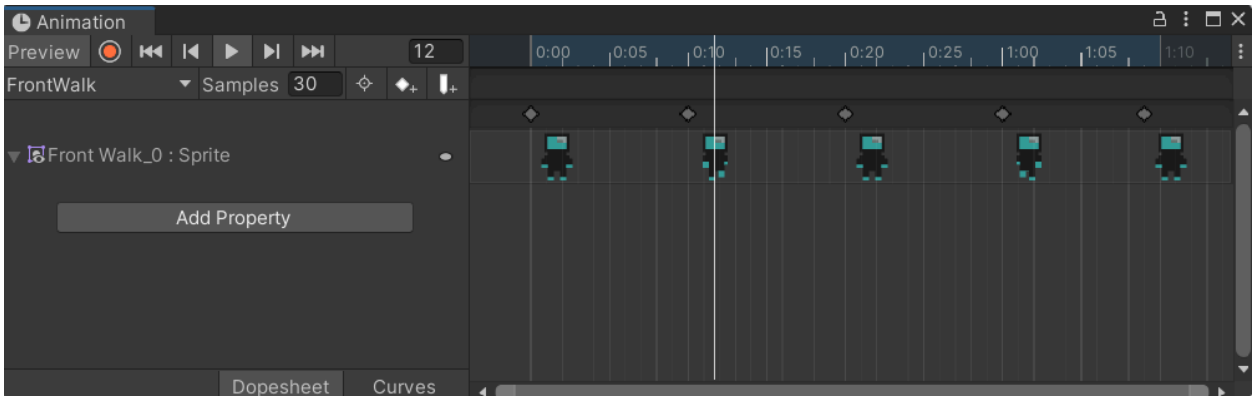
1. Click and Drag the First Front Walk Sprite into the Scene



2. Select the **Window** tab, Click **Animation**, then in the submenu click **Animation**

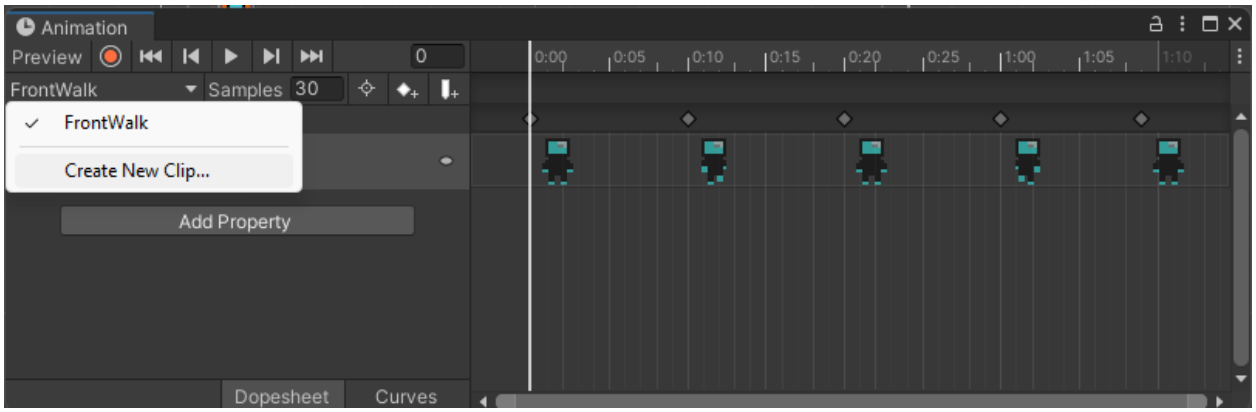


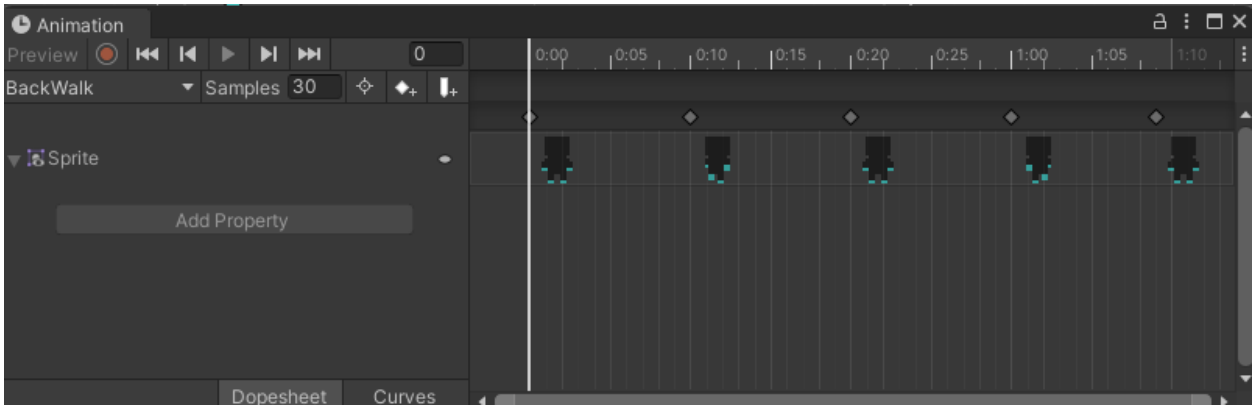
3. Select **Create,** then Name it “**FrontWalk”**
4. Click and Drag the **Front Walk Sprites** into the **Animation Window**

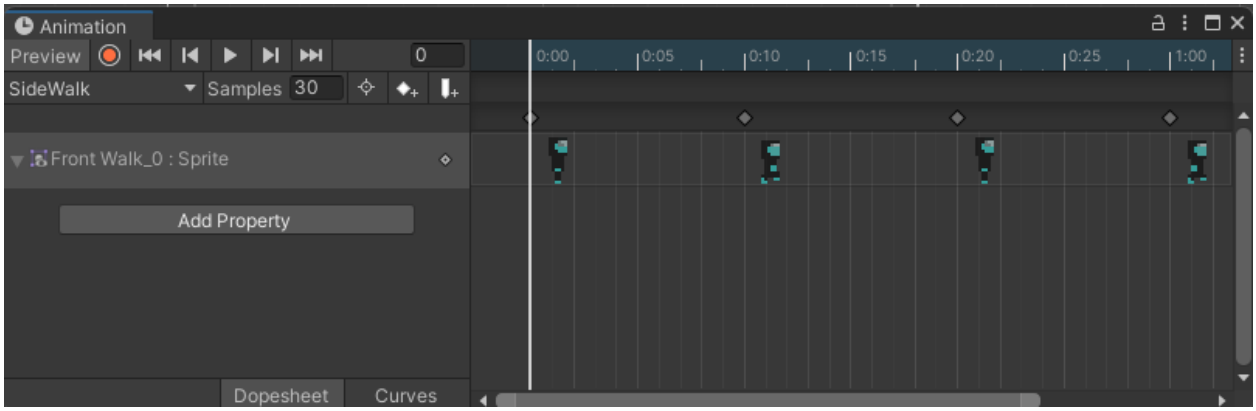


5. You can adjust the speed by changing the **Samples** value or moving the **Sprites**

6. Once finished, select **Create a New Clip…** and make the **BackWalk** and **SideWalk** animations



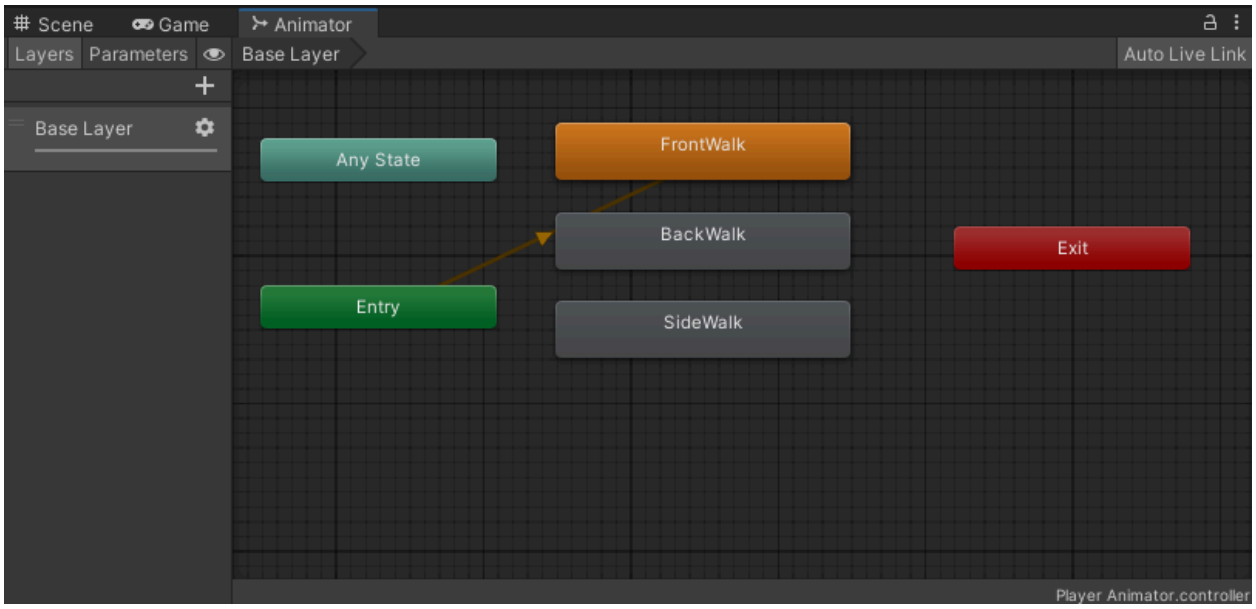




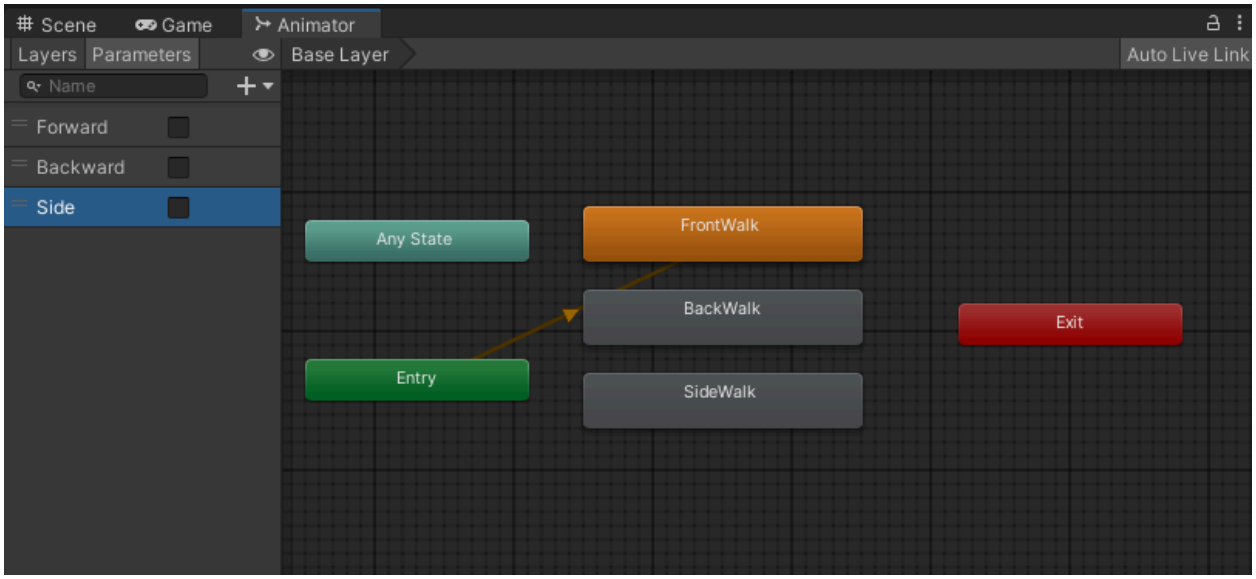
Set Up the Animator Controller

1. In the Assets, look for the **Animator Controller** and rename it to **“Player Animator”**

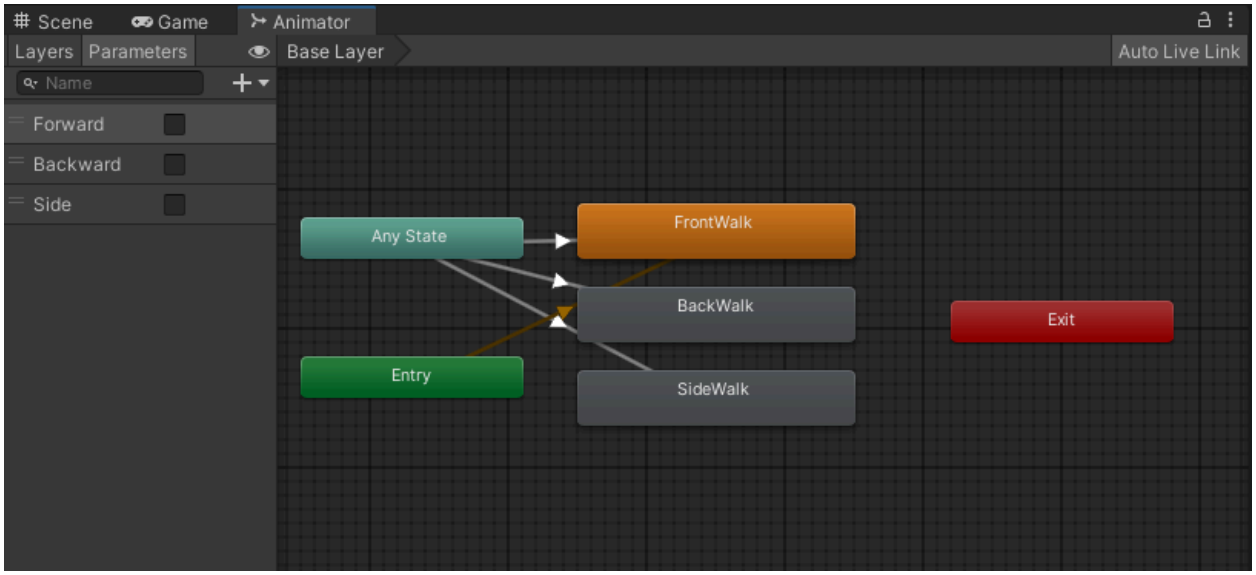
2. Double-Click the **“Player Animator”** to open the **Animator Window**.



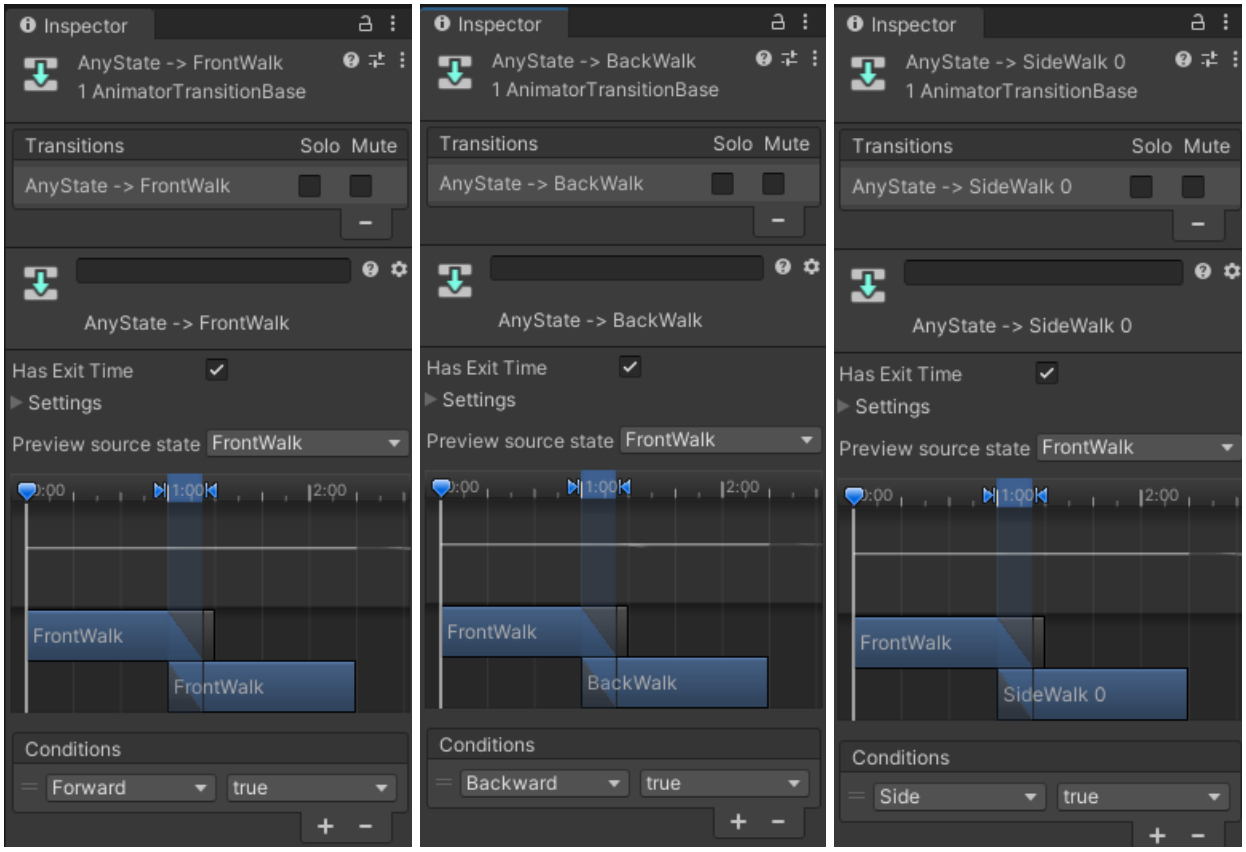
3. Click the Parameters Tab and create 3 Boolean Parameters and name them:
    - “Forward”, “Backward”, and “Side”



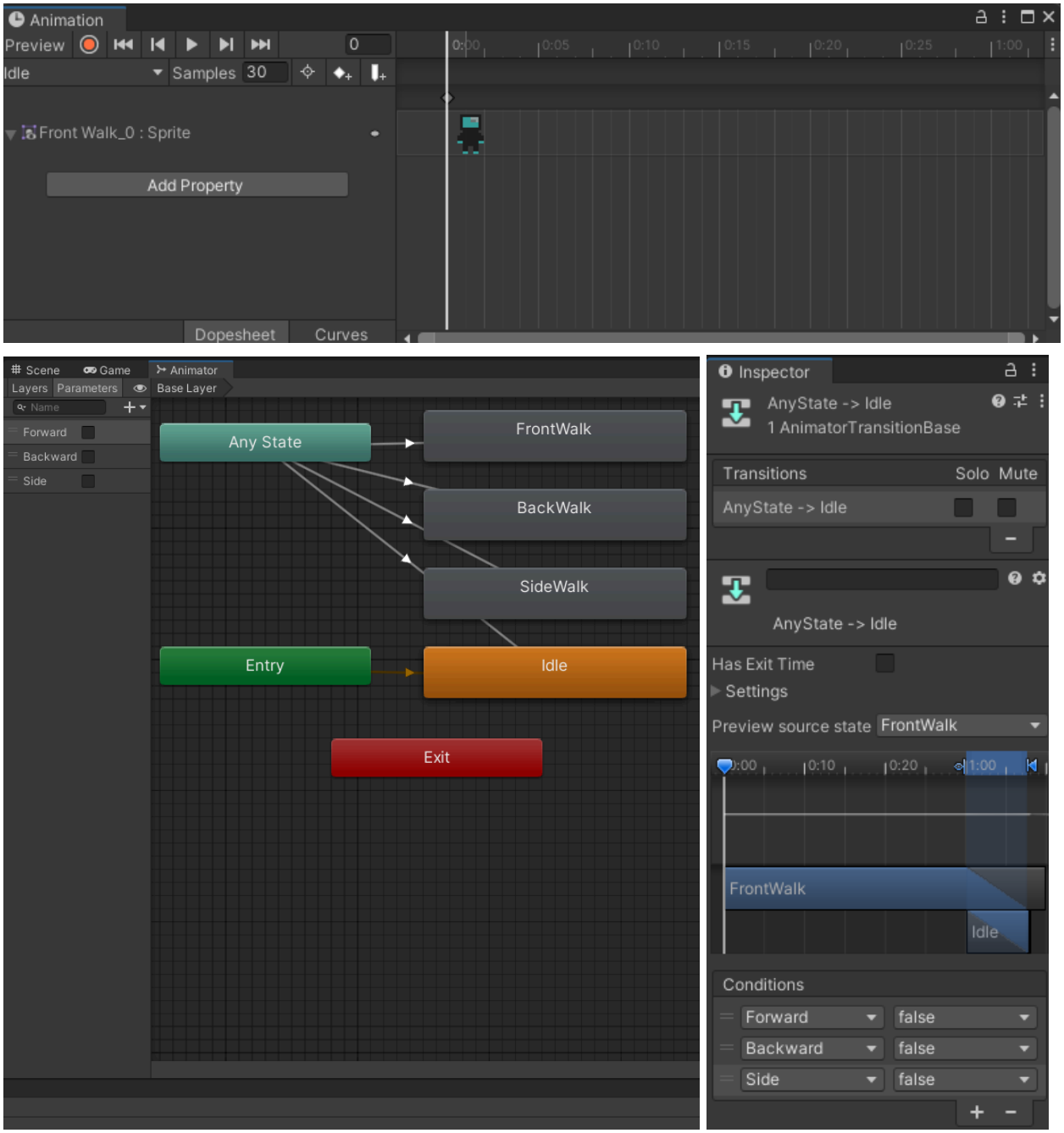
4. Right Click on “**Any State**” and create a **Transition** to each of the Animations



5. In the **Inspector Window**, check the **Has Exit Time** for all animations and set up the **Conditions** using the correct parameter



Review: Add the Player's Idle Animation



Code the Movement of the Player

Code the **Movement.cs** Script and **Add** this **Script** to the **Player** Game Object

```
public class Movement : MonoBehaviour
{
    // Movement Variables
    public float speed = 1f;

    // Component Variables
    Rigidbody2D rb;
    SpriteRenderer sr;
    Animator anim;

    // Start is called before the first frame update
    void Start()
    {
        // Get Components
        rb = GetComponent<Rigidbody2D>();
        sr = GetComponent<SpriteRenderer>();
        anim = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update()
    {
        // Get Inputs
        float h = Input.GetAxis("Horizontal");
        float v = Input.GetAxis("Vertical");

        // Call Movement and Animation Functions
        Move(h, v);
        Animate(h, v);
    }

    void Move(float h, float v)
    {
        // Get Direction of Movement
        Vector2 direction = new Vector2(h, v).normalized;
        rb.MovePosition(rb.position + direction * speed * Time.deltaTime);
    }

    void Animate(float h, float v)
    {
        // Fix Left and Right Direction
        if (h != 0)
        {
            sr.flipX = h < 0;
        }

        // Set Up Animation Parameters
        anim.SetBool("Horizontal", h != 0);
        anim.SetBool("Forward", v < 0);
        anim.SetBool("Backward", v > 0);
    }
}
```

(Optional) Make the Camera Follow the Player

Code the **CameraFollow.cs** Script and **Add** this **Script** to the **Camera** Game Object

```
public class CameraFollow : MonoBehaviour
{
    public float edgeBuffer = 2f; // Distance from screen edge before camera moves
    public float camSpeed = 5f;  // Speed of camera movement

    private Transform player;
    private Camera cam;


    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").transform;
        cam = GetComponent<Camera>();

    }

    void LateUpdate()
    {
        if (player == null) return;

        // Get screen bounds in world coordinates
        Vector3 playerScreenPos = cam.WorldToViewportPoint(player.position);

        // Calculate how much to move camera
        Vector3 cameraMove = Vector3.zero;

        // Check horizontal edges (0 = left, 1 = right)
        float edgeThreshold = edgeBuffer / (cam.orthographicSize * 2 * cam.aspect);

        if (playerScreenPos.x < edgeThreshold) // Too far left
            cameraMove.x = (playerScreenPos.x - edgeThreshold) * cam.orthographicSize * 2 * cam.aspect;
        else if (playerScreenPos.x > 1 - edgeThreshold) // Too far right
            cameraMove.x = (playerScreenPos.x - (1 - edgeThreshold)) * cam.orthographicSize * 2 * cam.aspect;

        // Check vertical edges (0 = bottom, 1 = top)
        float verticalThreshold = edgeBuffer / (cam.orthographicSize * 2);

        if (playerScreenPos.y < verticalThreshold) // Too far down
            cameraMove.y = (playerScreenPos.y - verticalThreshold) * cam.orthographicSize * 2;
        else if (playerScreenPos.y > 1 - verticalThreshold) // Too far up
            cameraMove.y = (playerScreenPos.y - (1 - verticalThreshold)) * cam.orthographicSize * 2;

        // Move camera smoothly
        transform.position += cameraMove * camSpeed * Time.deltaTime;
    }
}
```