



# Art and Animation (Part 1)

Introduction to Game Development in  
Unity  
Spring 2026, 98-127, Lecture 3

Instructors: Jingxuan Chen, Dario Quintero, Shangyi Zhu, Jeffrey Wang



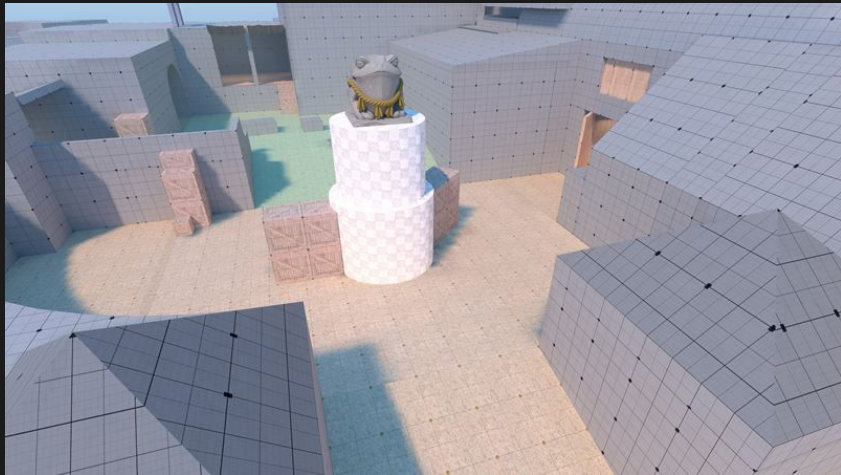
# Where do Art and Animation appear in games?

- Everywhere!
  - Character Art
  - Environment Art
  - UI art



# Art and Game Logic

- Up until this point we have been working with just a cube.
- It is possible to make a functioning game entirely with default primitives (cubes, spheres, etc)
- The art is just a visual wrapper for our game
- So how do we "wrap" our game? What does this process look like



# Asset Creation

- We use dedicated software to create 2D and 3D assets
  - 2D Programs:
    - Photoshop, Illustrator, Clip Studio Paint
  - 3D Programs:
    - ZBrush, Maya, Blender, Substance
- We then export assets from these programs into Unity



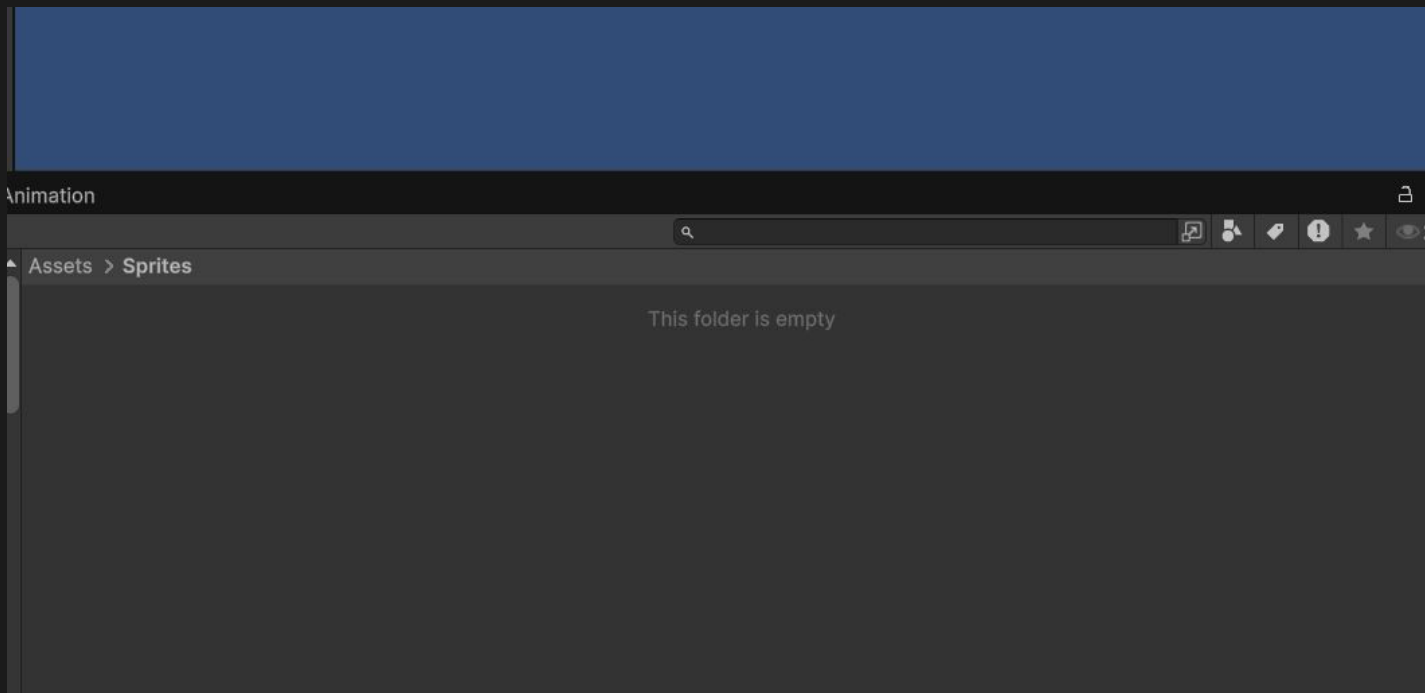
# Bringing Art into Unity

- 2D art and 3D art have different workflows
- In this lecture we will show you the 2D asset pipeline
- This includes:
  - Importing sprites and sprite sheets
  - Extracting animations from sprite sheets
  - Playing animations



# Importing Sprites

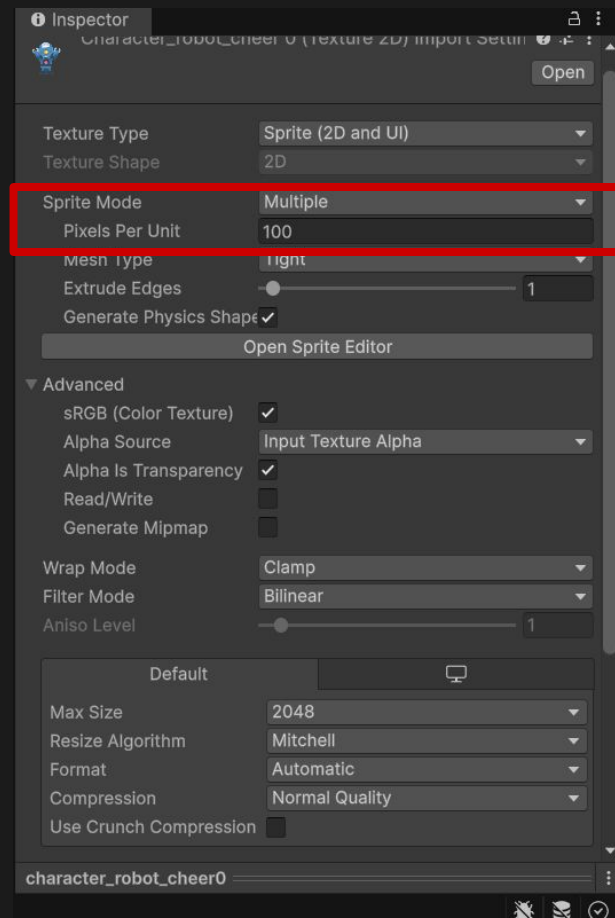
- Right click in Project Tab
- Click Import New Asset
- Select your file



# Sprite Settings

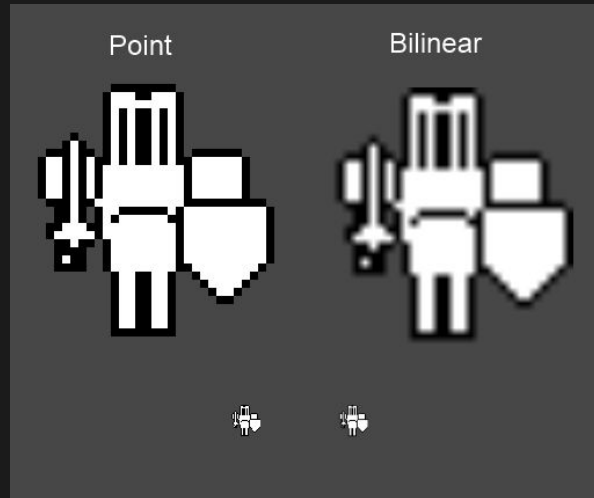
Clicking on sprite in project brings up this inspector panel. Here are some notable fields:

- **Sprite Mode:**
  - Single: “The Default”
  - Multiple: For sprite sheets
  - Polygon: For Special Cases (I’ve never used this)
- **Pixels Per Unit:**
  - 100 PPU => A 100x100 pixel image is the size of 1 unit in the scene



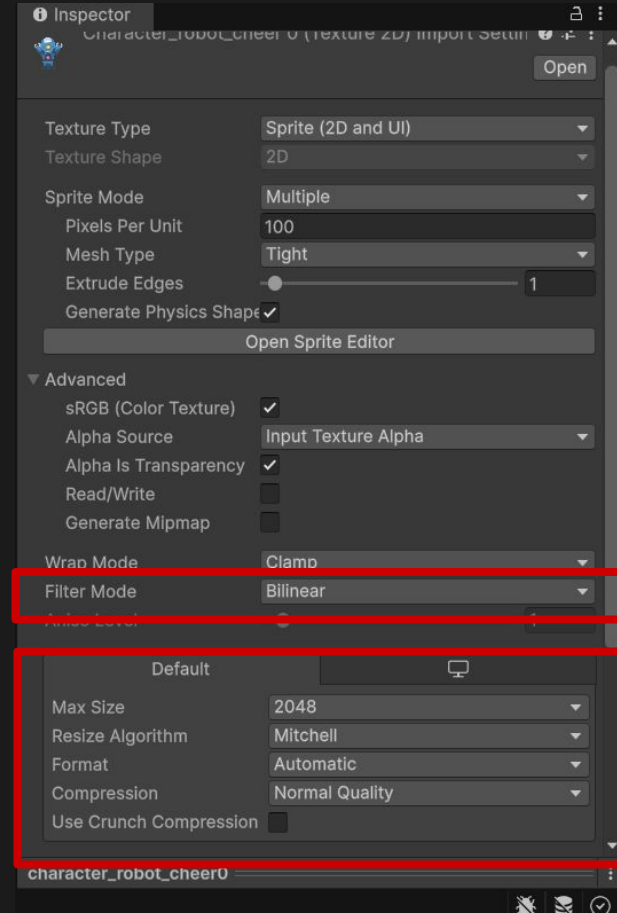
# Sprite Settings

- Filter mode:



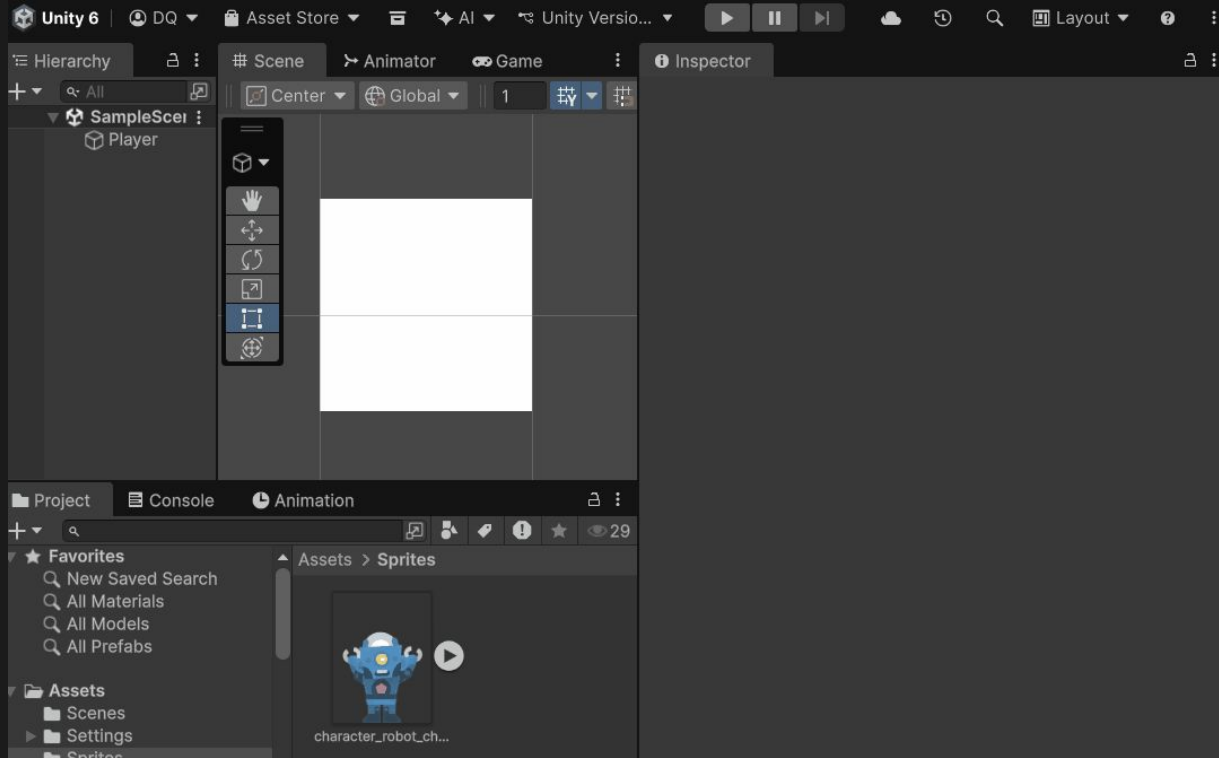
- Compression:

- Image compression to reduce file sizes
- We want to disable this when we are using pixel
- 



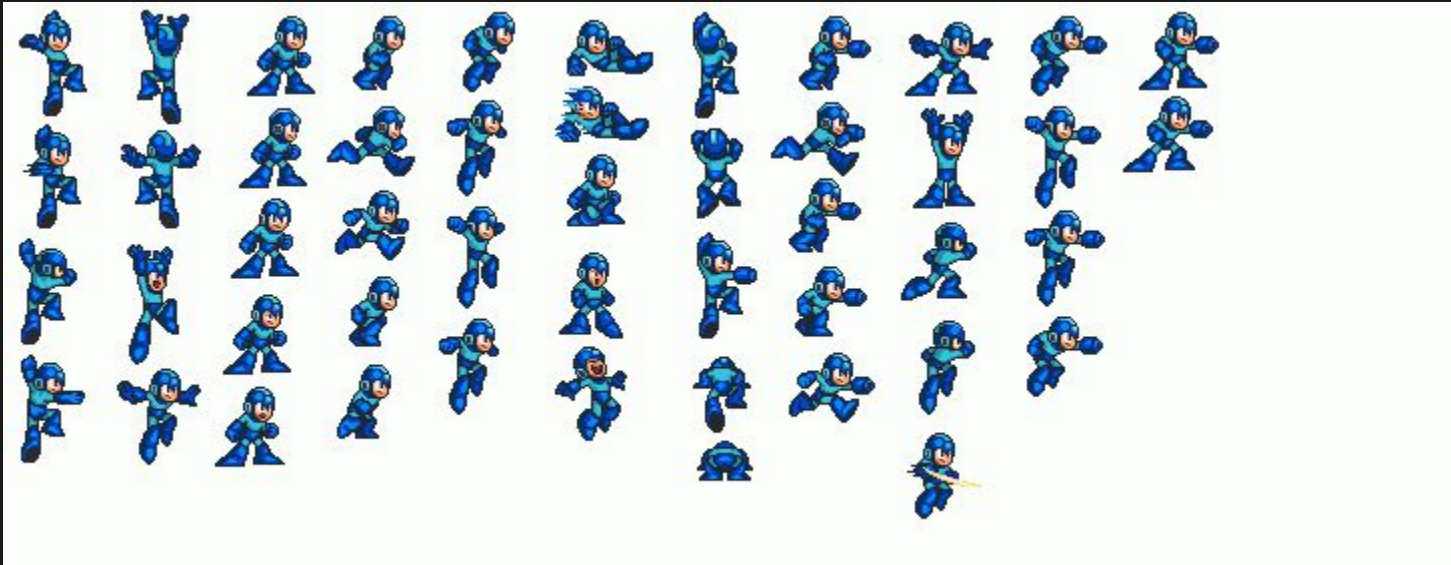
# Assign Sprites

- Select Game Object
- Drag sprite into Sprite tab of Sprite Renderer



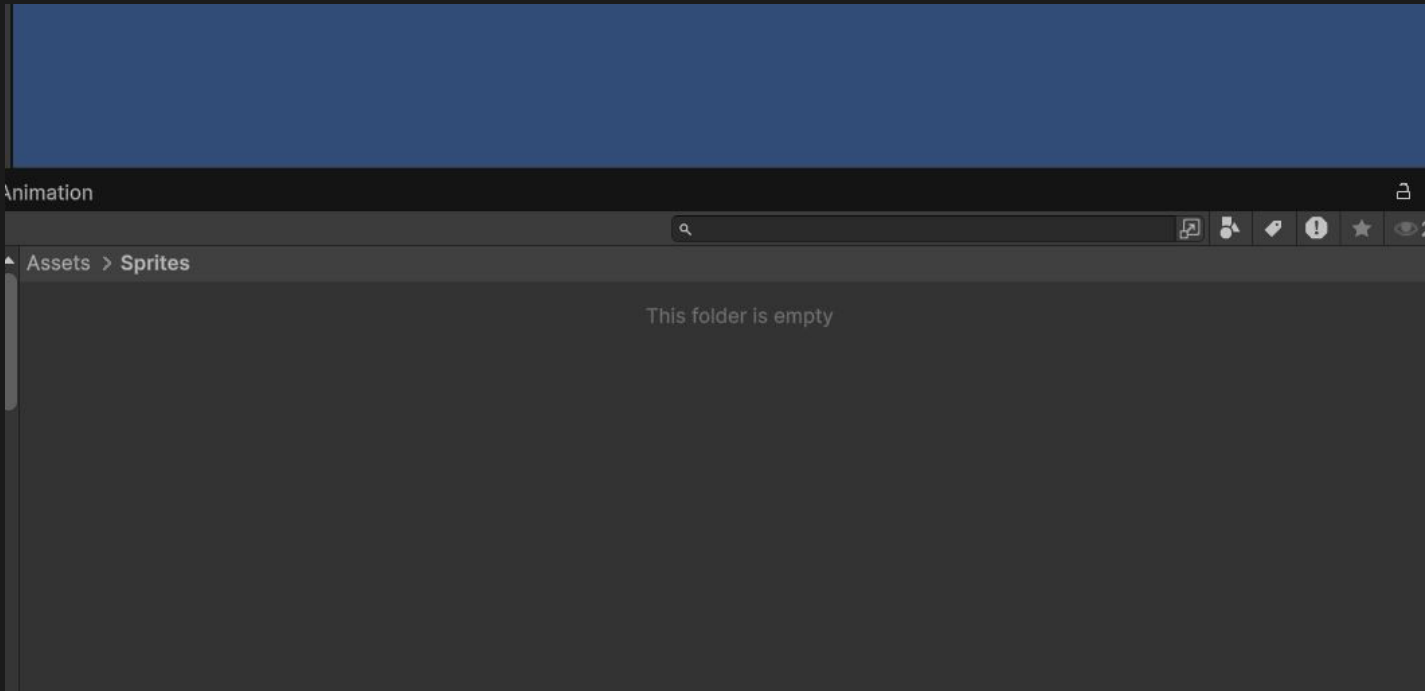
# Animation

- 2D Animations are stored as sprite sheets
- A sprite sheet consists of all frames of an animation organized in rows and columns
- All information in a single file, more efficient than loading multiple files



# Importing Sprite Sheets

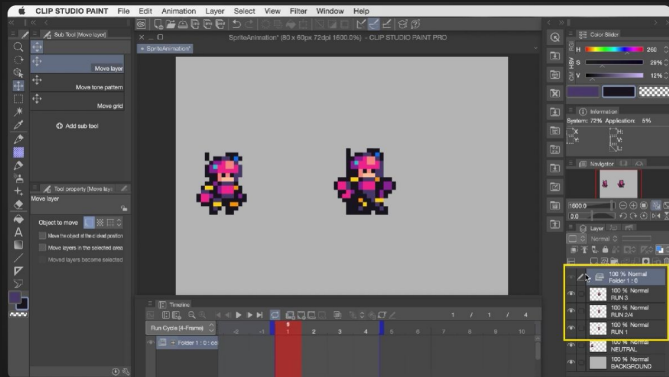
- Import sprite sheet as you would any other image



# Resources on Creating Sprite Sheets

Many image manipulation programs have features that make exporting sprite sheets easy.

- For pixel art, I recommend Aseprite (paid), Piskel (free web app), We also have Clip Studio passes for this class
- [Clip Studio Tutorial](#)

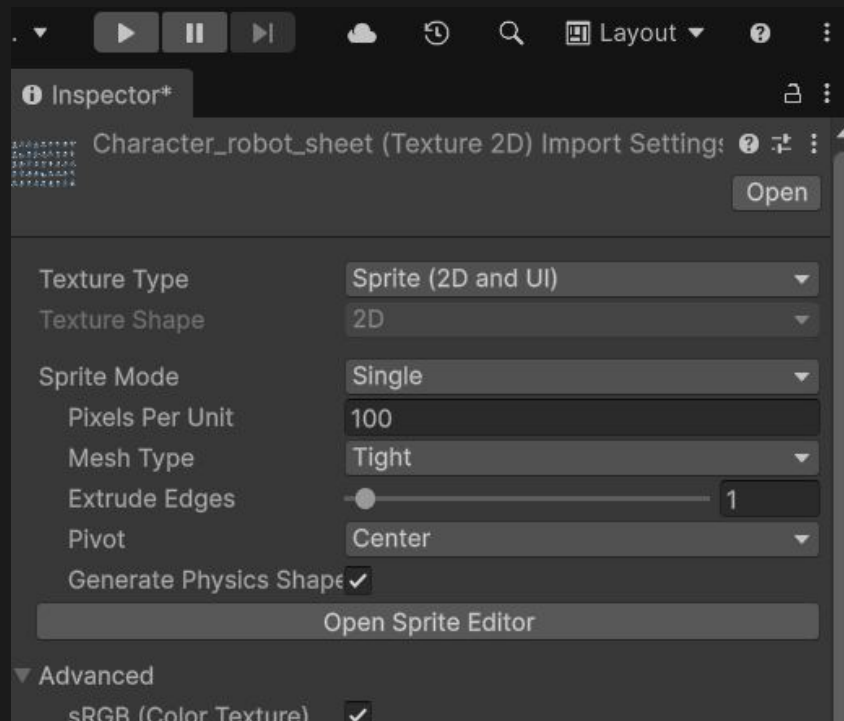


How to animate sprites in Clip Studio Paint! | Brandon James Greer



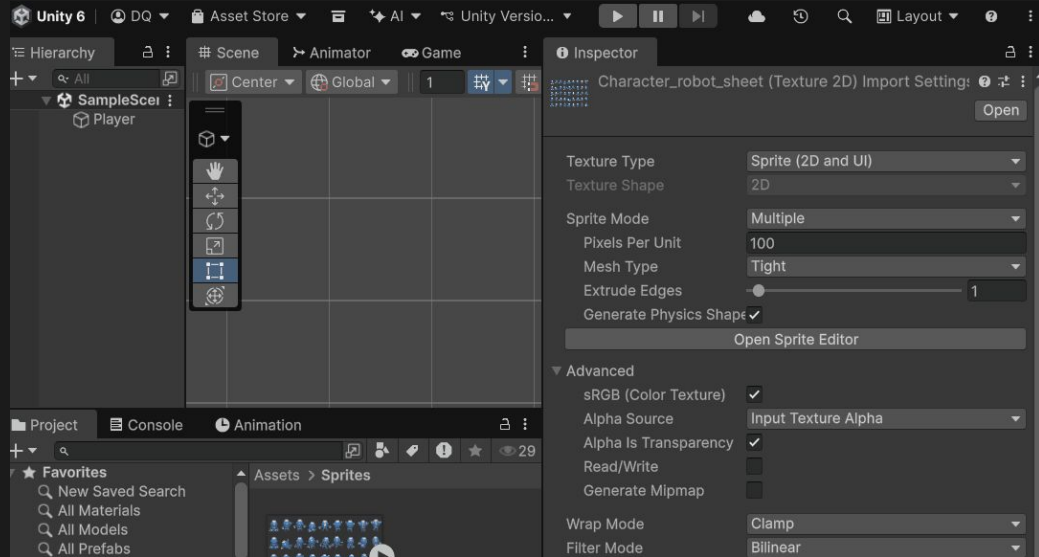
# Importing Sprite Sheets

- Set Sprite Mode to Multiple

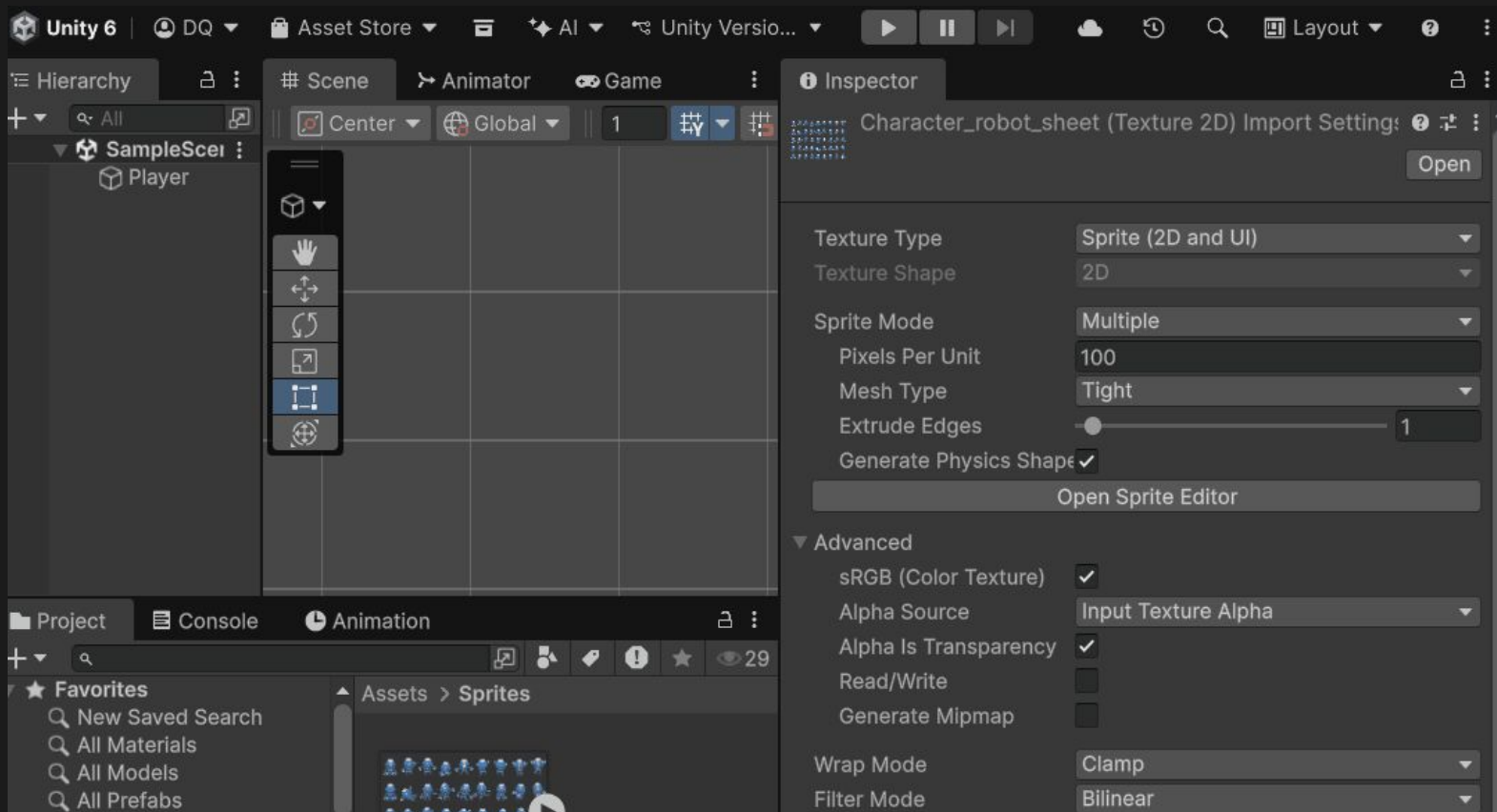


# Importing Sprite Sheets

- Open the Sprite Editor
- Click Slice dropdown
- Make sure type is Grid by Cell Count (feel free to try others though)
- Specify number of rows and columns
- Click Slice Button
- Click Apply

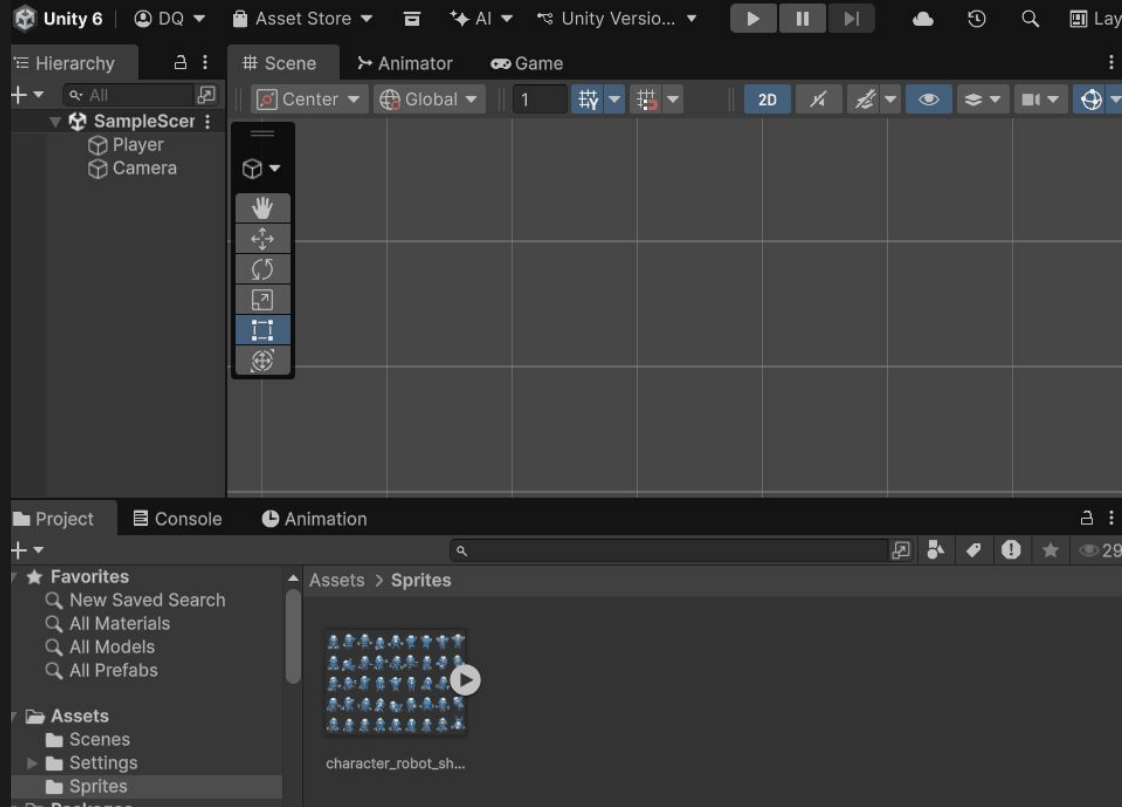


# Importing Sprite Sheets



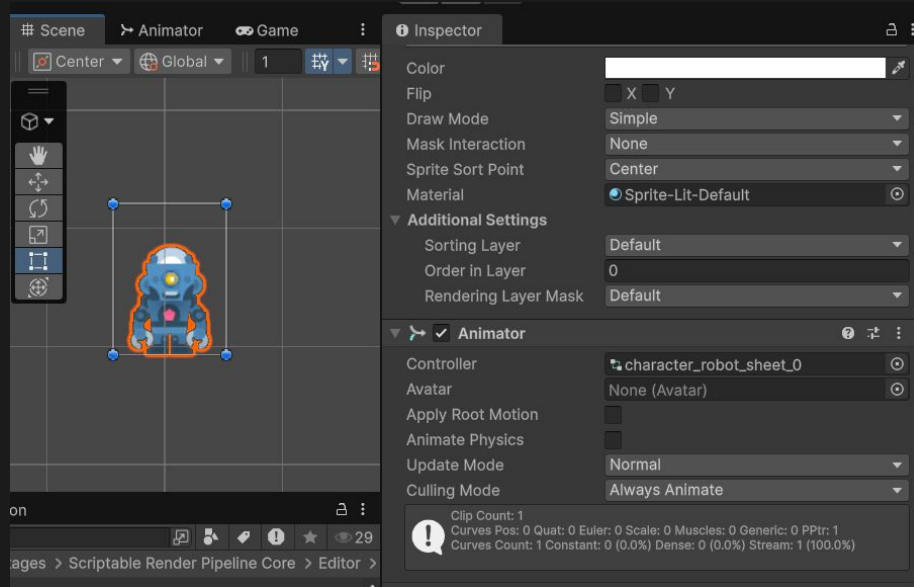
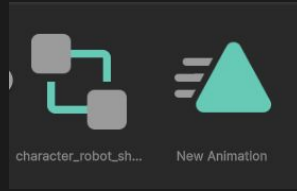
# Importing Sprite Sheets

- Now you can drag this sliced sprite into the scene and automatically see its animations playing



# What is happening here?

- Two assets have been created:
  - A Controller asset
  - An Animation asset
- A Game Object is created in the scene with an Animator Component
- What we've shown is a shortcut for having animated objects in your scene, we will now show the whole proper workflow

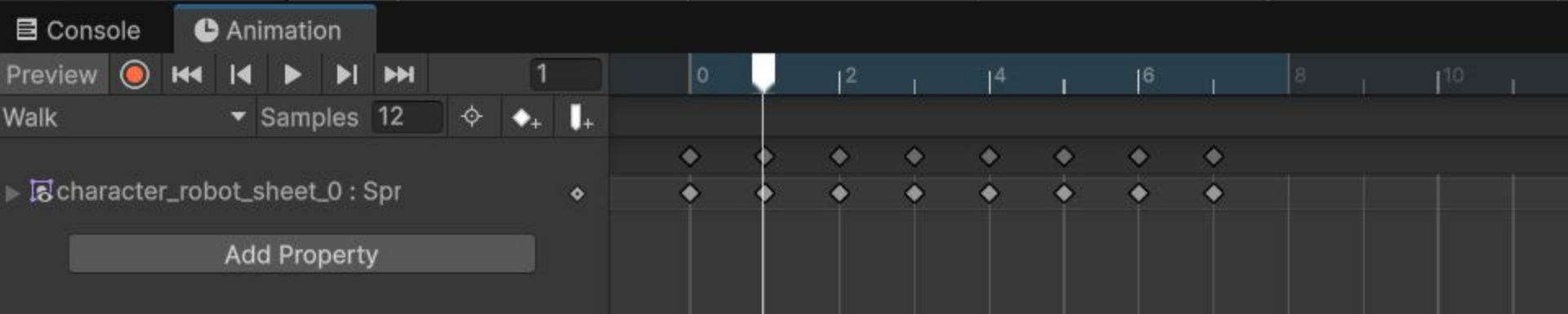


# Animations

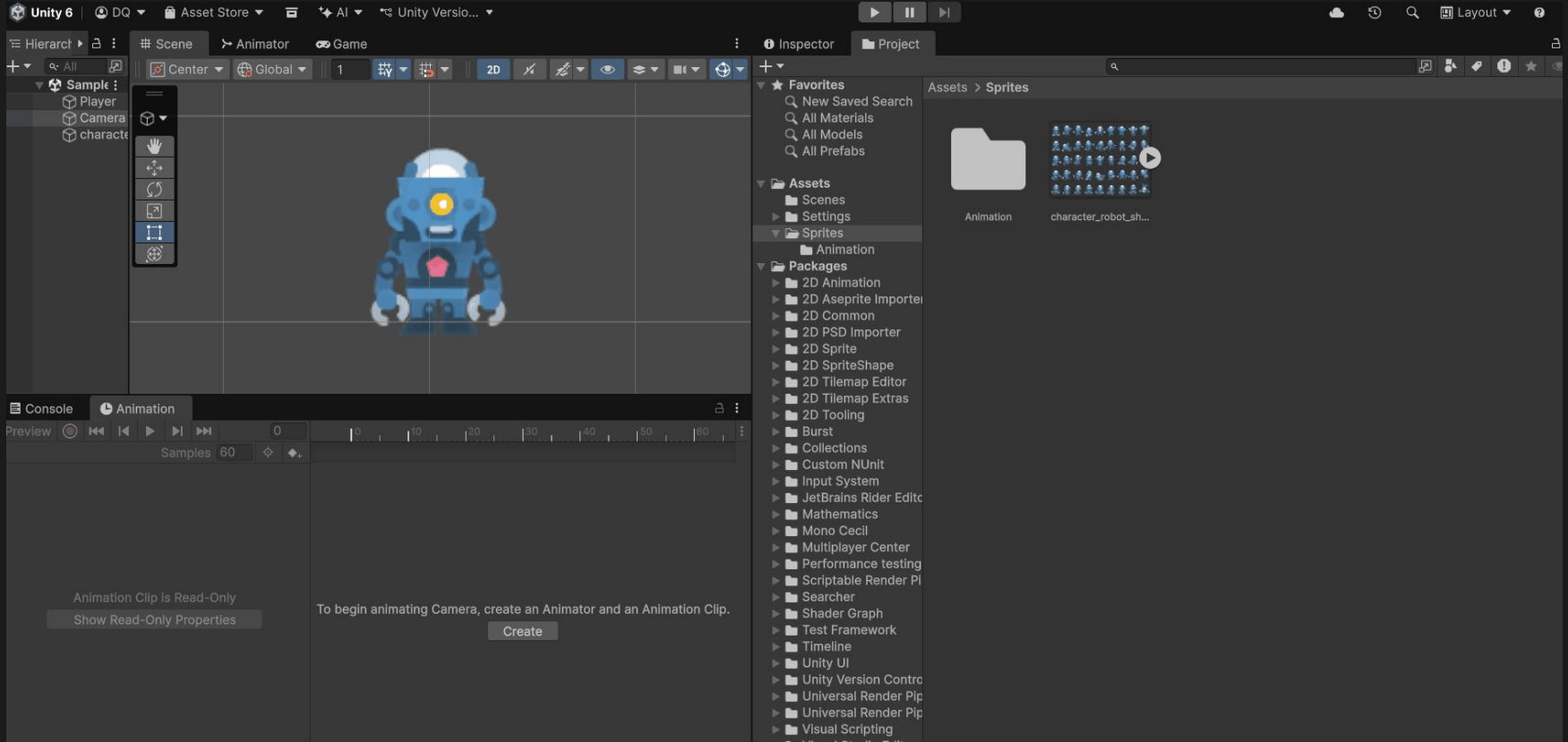
Each state in the animator has an associated animation asset. We create animations in the animation window.

In the animation window we define keyframes. These keyframes can be sprites, or fields such as position that are interpolated between frames.

By changing the samples field of the window we change the fps of the animation.

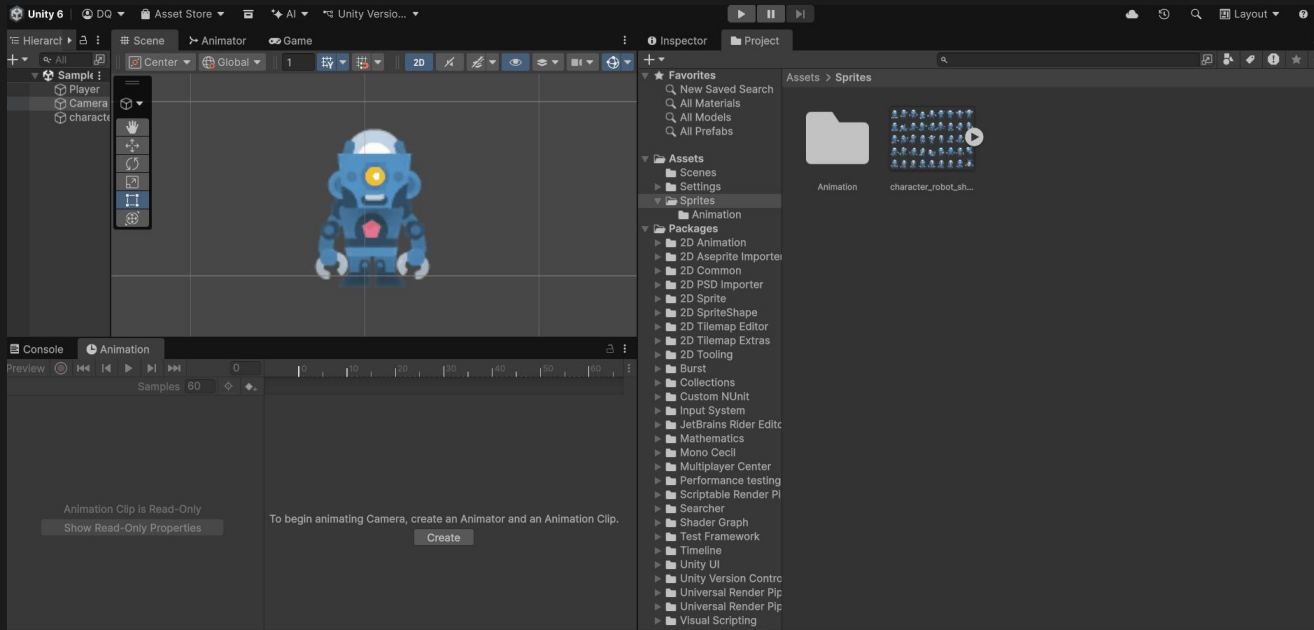


# Making Animations



# Making Animations

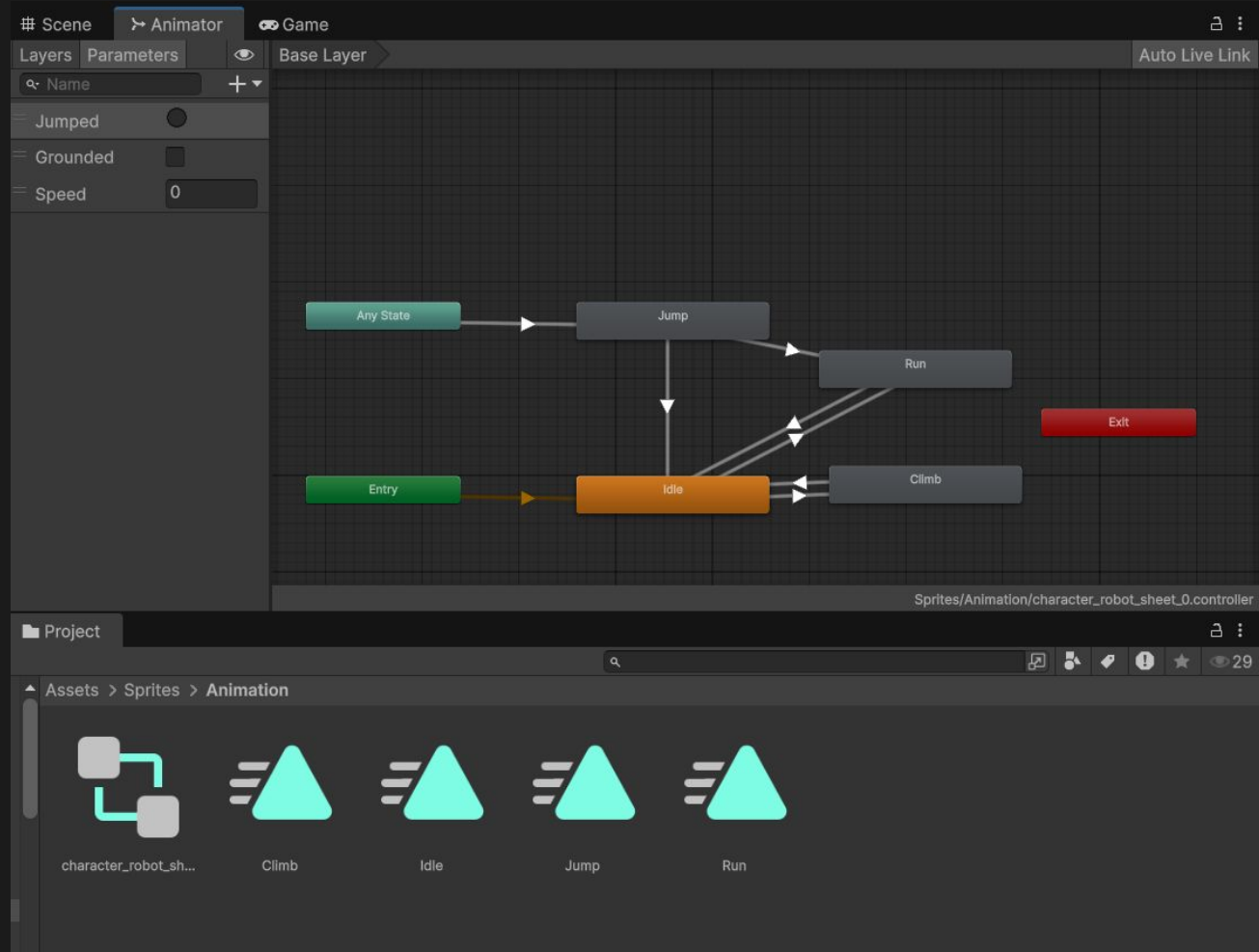
- Open the animation window. (Window->Animation->Animation)
- Click on your object in the scene
- Create a new animation clip in that window
  - Give clip an appropriate name and location
- Drag our frames into the animation timeline



# The Animator

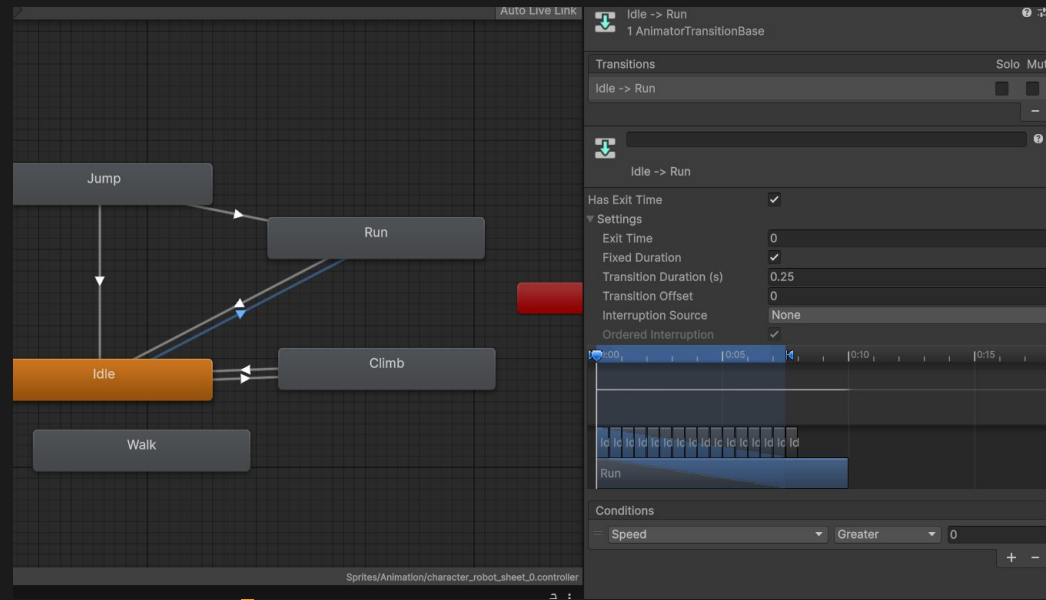
A state machine graph where states (animations) are connected by transitions between states that we define.

We use parameters to trigger transitions.



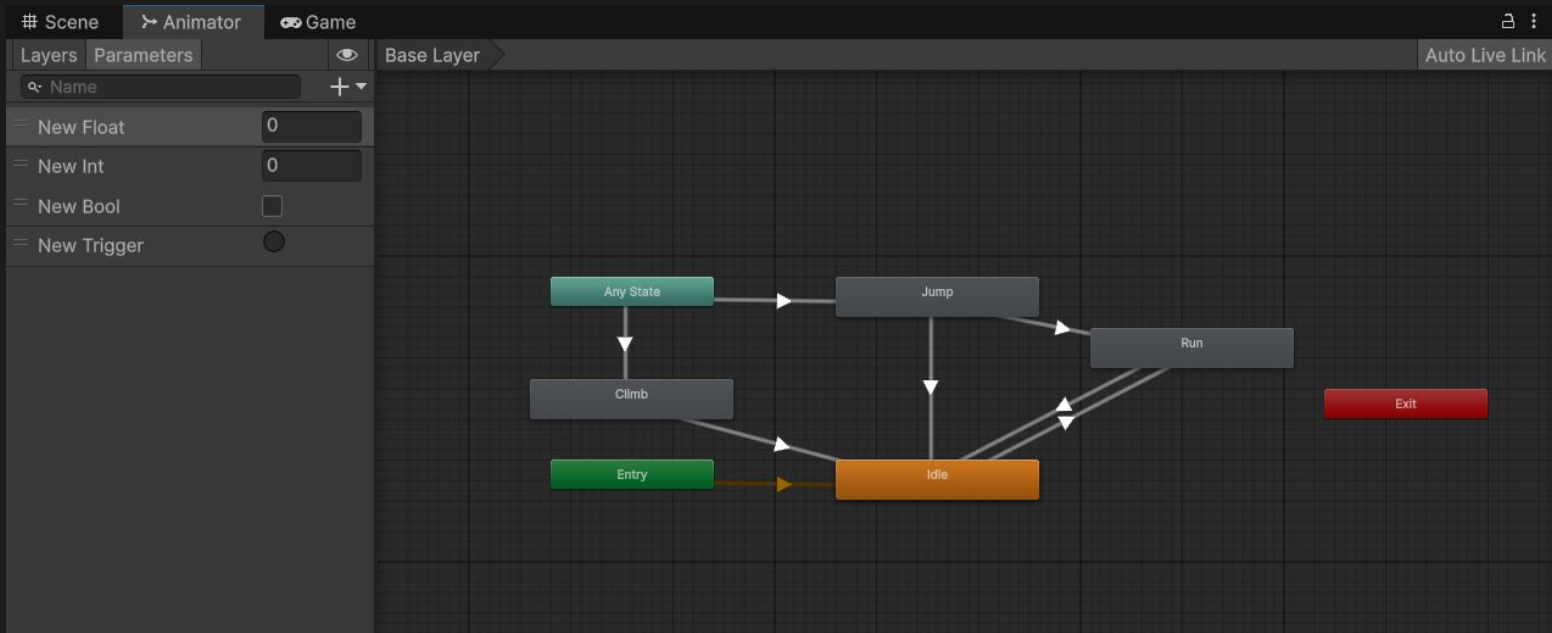
# Transitions

- Has Exit Time: Does previous Animation finish before transitioning to next animation
- Transition Duration: How long we do we blend animations
- For sprite animation we want Has Exit Time = false, Transition Duration = 0
- Conditions use parameters



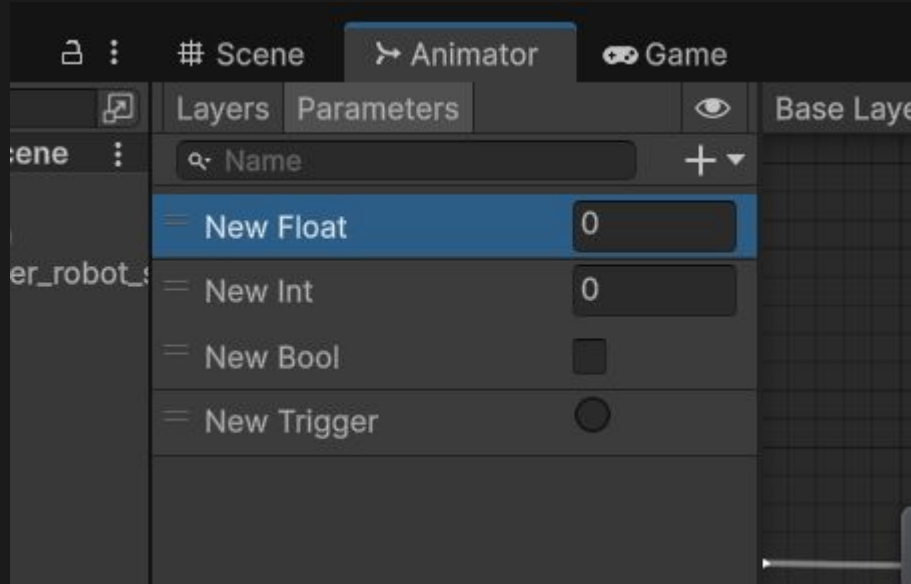
# Transitions

- Orange state is your default (initial) state
- Transitions from “Any State” happen from any state
- Entry and Exit are used to transition between state machines



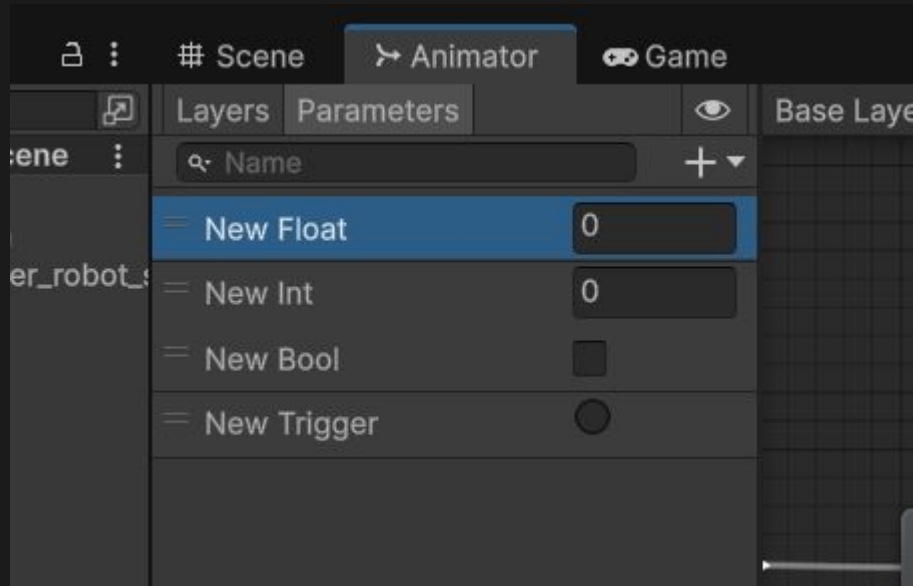
# Parameters

- We use these to begin transitions
  - Transition when float > 0, when bool = true, etc
- Type can be Float, Int, Bool, Trigger
  - A Trigger is a bool that is true for one frame then turns false



# Changing Parameters

- We use these to begin transitions
  - Transition when float > 0, when bool = true, etc
- Type can be Float, Int, Bool, Trigger
  - A Trigger is a bool that is true for one frame then turns false



# Changing Parameters

- Use SetFloat, SetInt, SetBool, SetTrigger to change parameters
- Arguments are (name, value)

```
0 references | ☒ Unity Script
public class Player : MonoBehaviour
{
    3 references
    private Animator anim;
    0 references | ☒ Unity Message
    void Start()
    {
        anim = GetComponent<Animator>();
    }
    0 references
    public void Jump(InputAction.CallbackContext context)
    {
        anim.SetTrigger("Jump");
    }
    0 references
    public void Walk(InputAction.CallbackContext context)
    {
        Vector2 input;
        input = context.ReadValue<Vector2>();
        anim.SetFloat("Speed",input.x);
    }
}
```



# Overall Workflow

- Create your animator
- Create your animations
- Bring your animations into the animator
- Create your animator parameters
- Create your animation transitions using those parameters
- Write code to modify your parameters

