### **Lab 4 : Animating a Character**

By: Mariam Salim (622952)

### **Lab Overview:**

This lab involves setting up an animated character using Unity's Animator system. You will configure an animation state machine, create transitions between animations, and modify the character script to control the animations based on user input.

### **Step-by-Step Solution:**

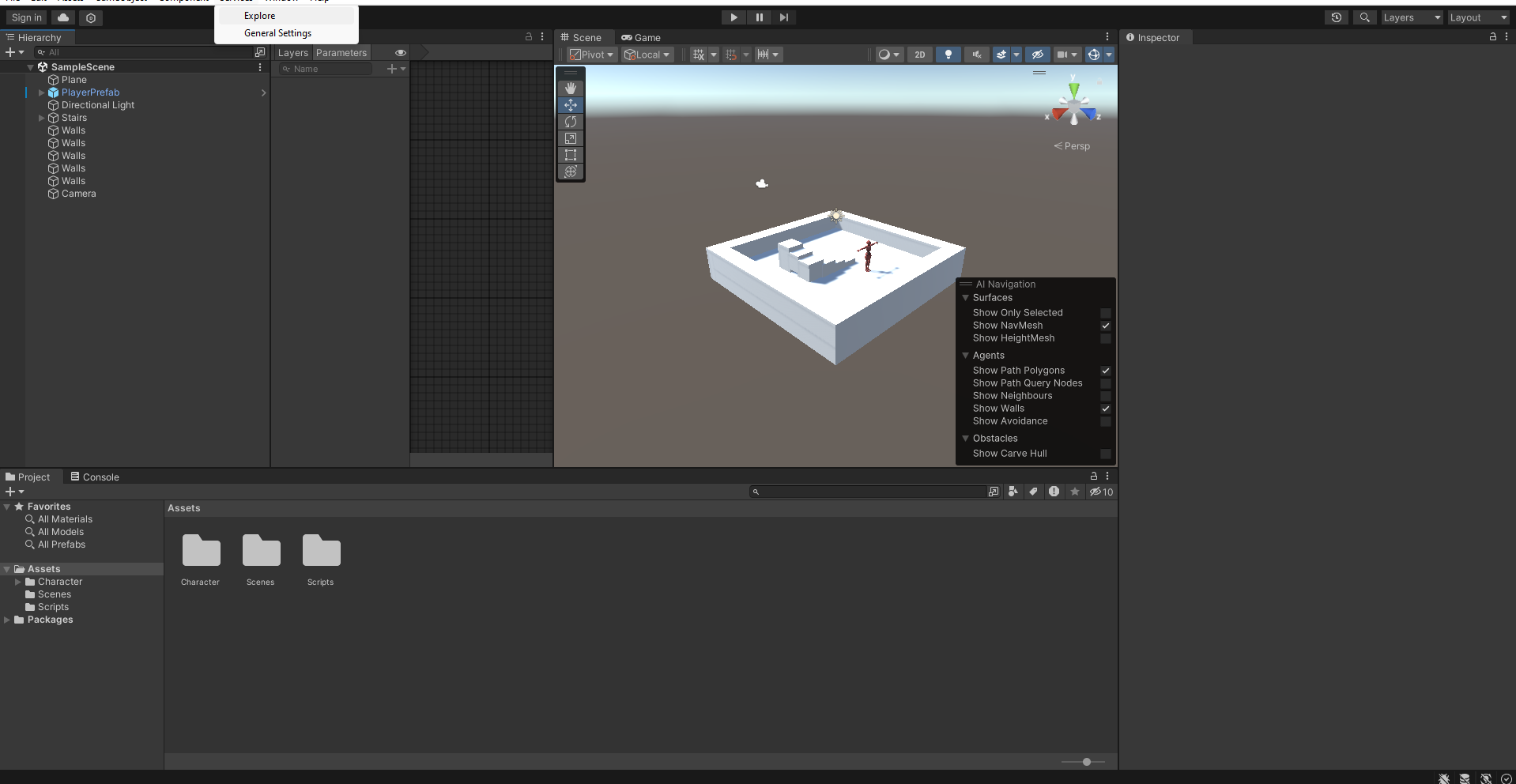
### **Task 1: Setup the Animator and Avatar for Both the Character and Animations**

1. **Open the github classroom assignment and download the repo :**

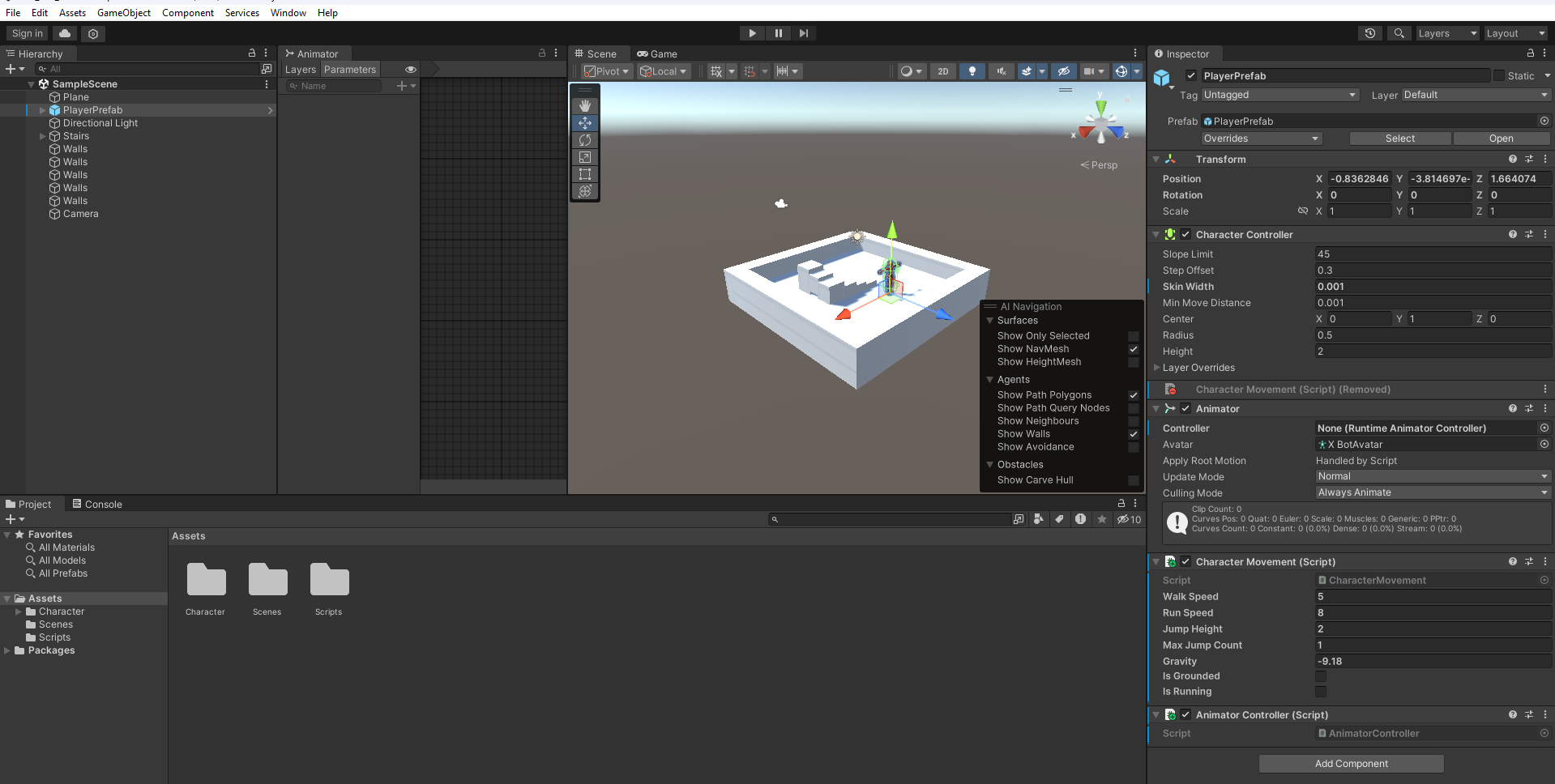
<https://classroom.github.com/a/TmlJ0Rzc>

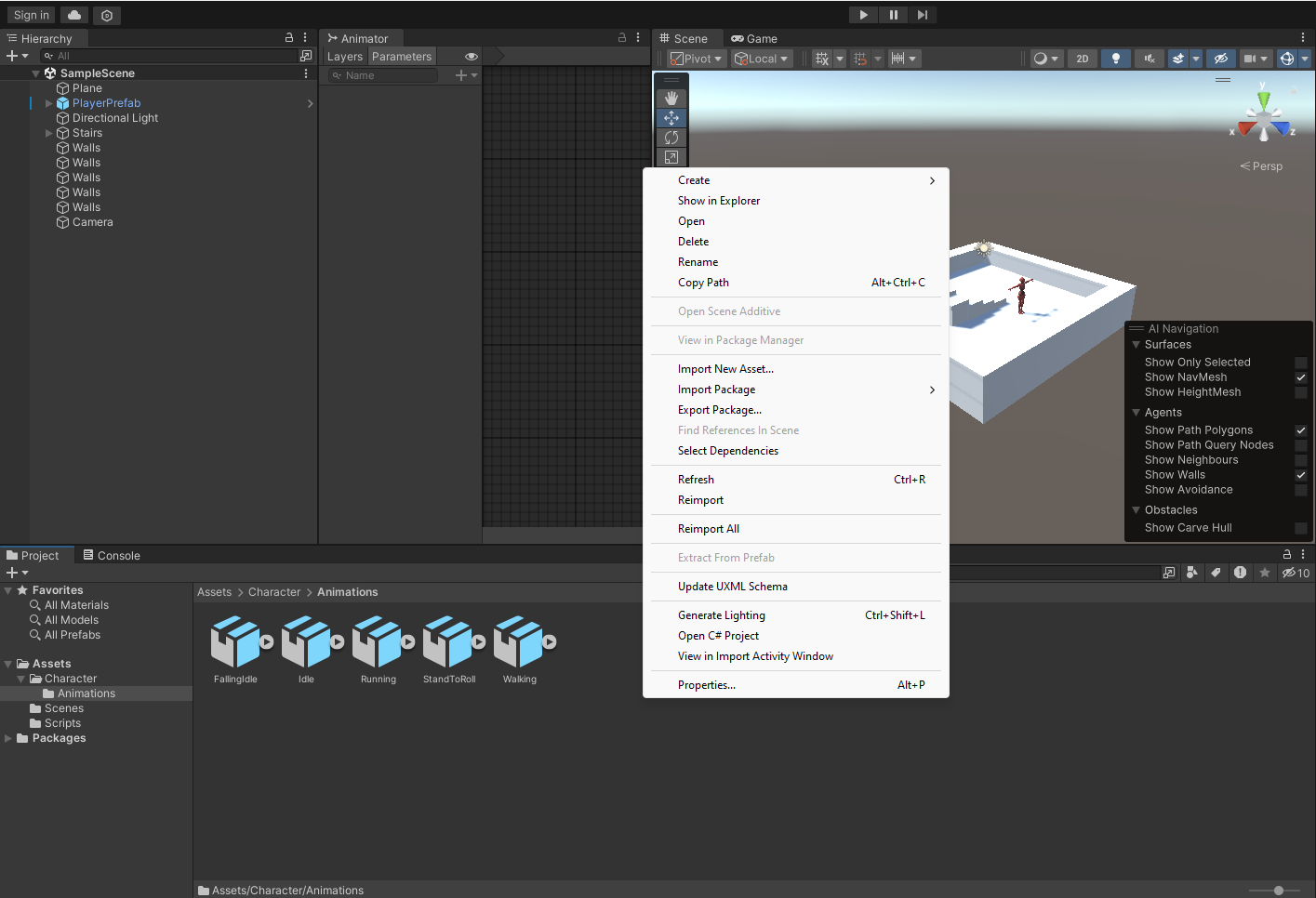
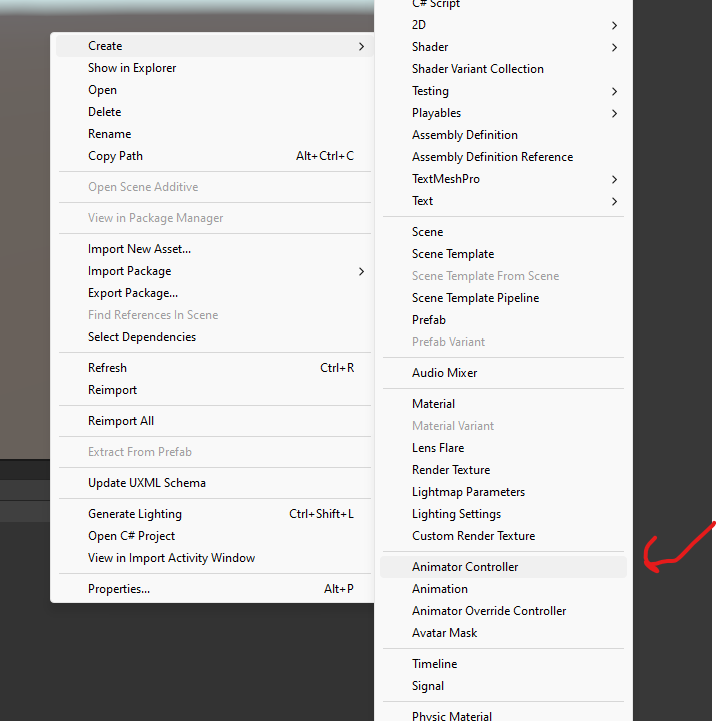
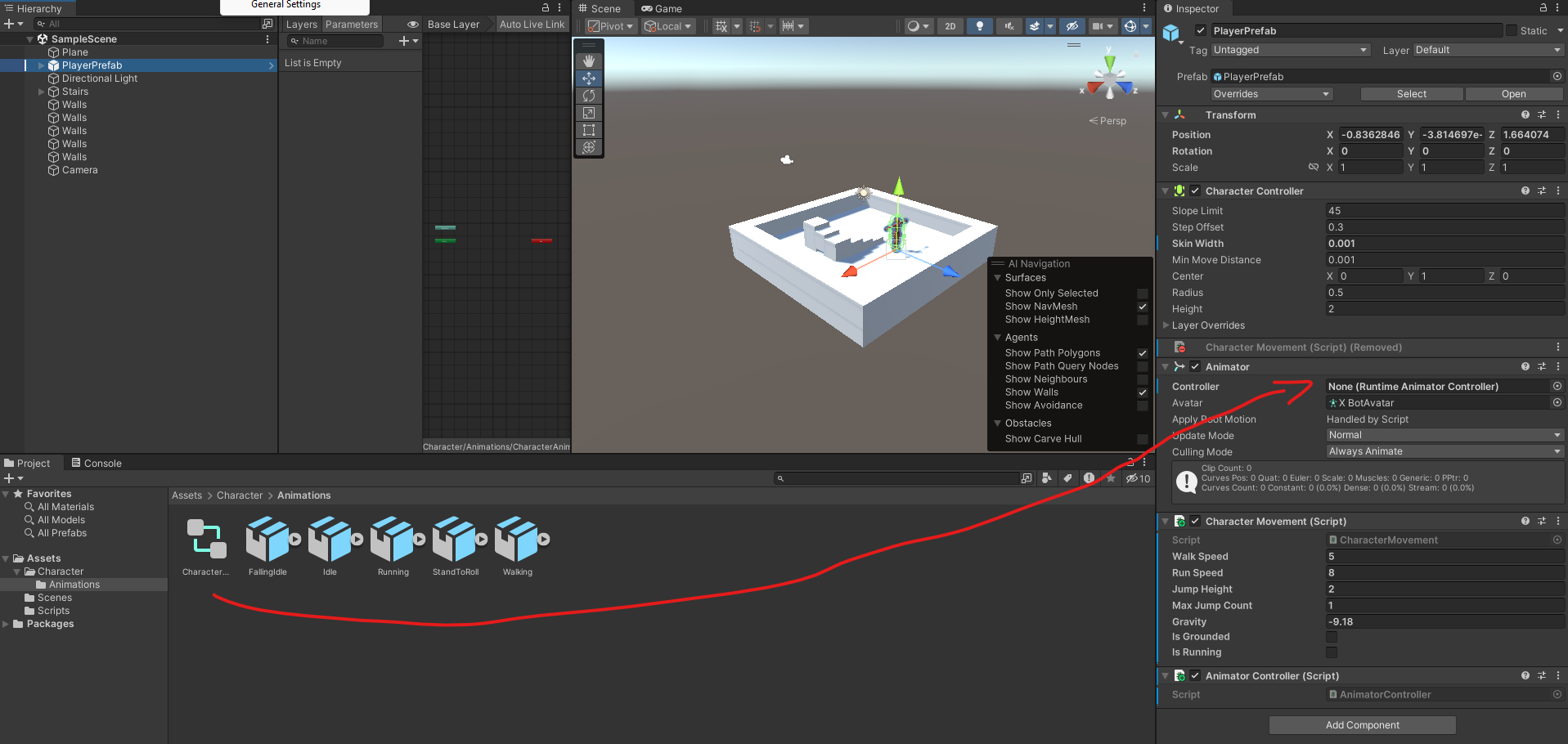
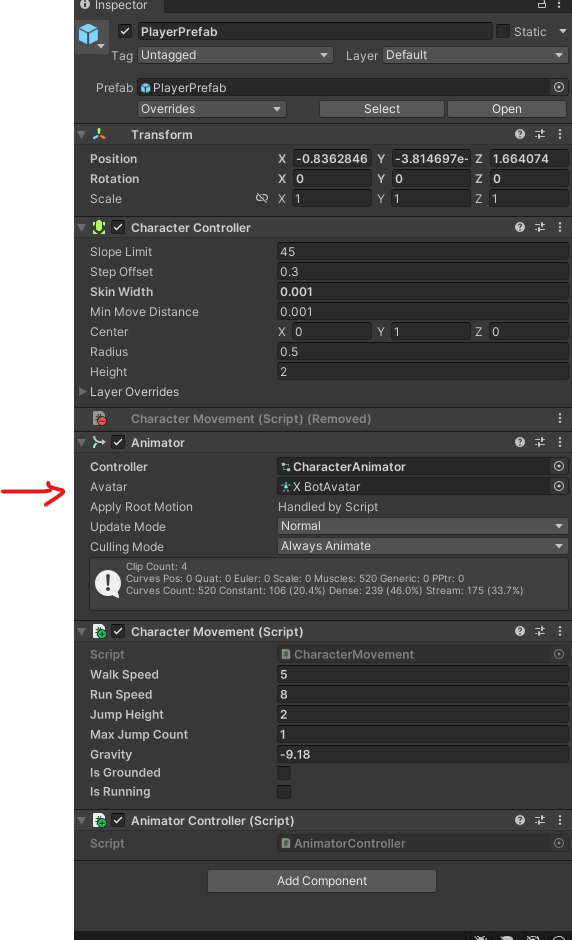
2. Once the repo is clone , Open the Sample Scene

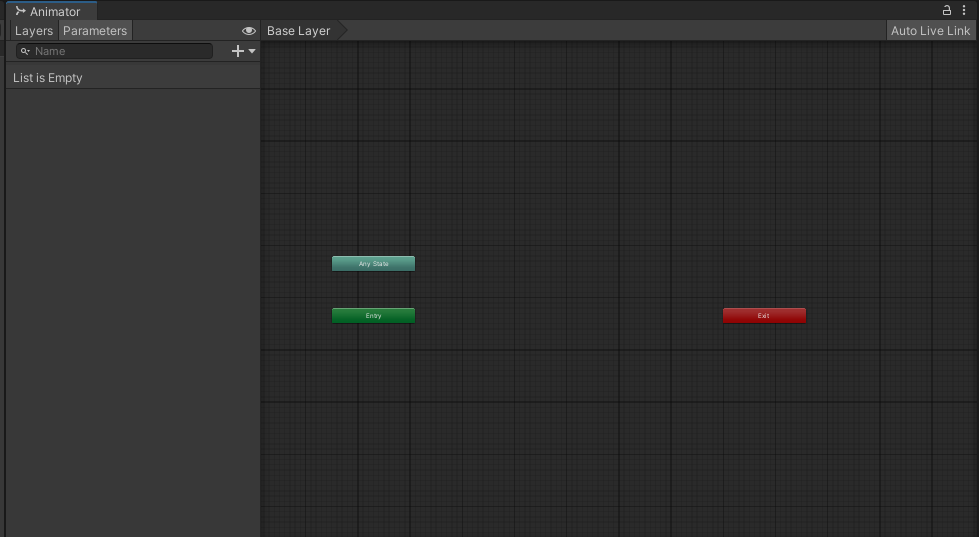
* + In the scene folder, Open the sample scene . You should see something like this



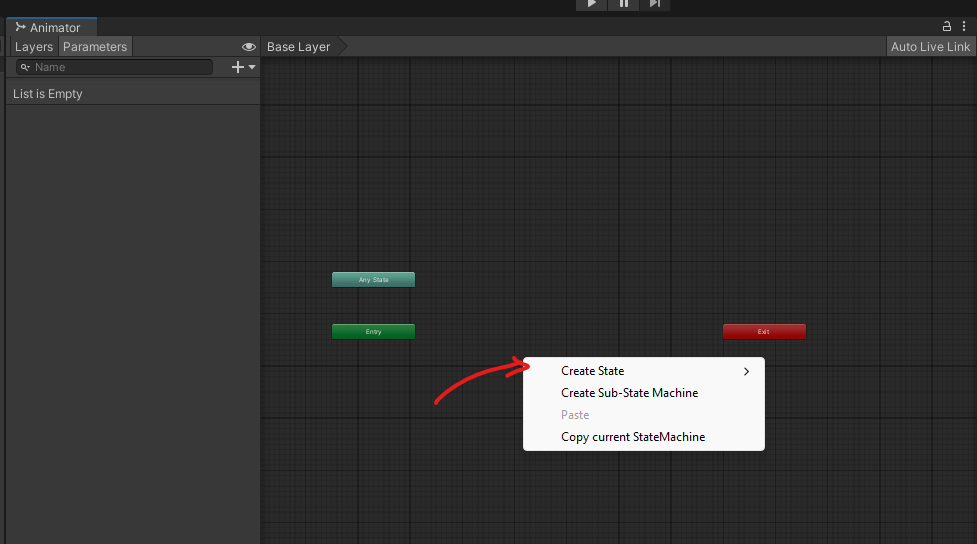
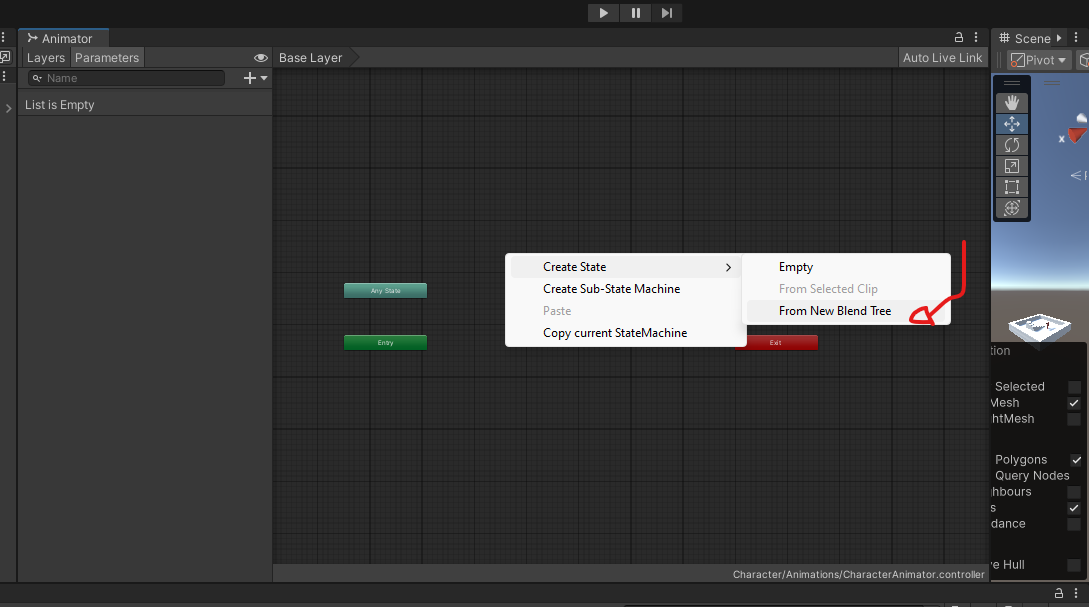
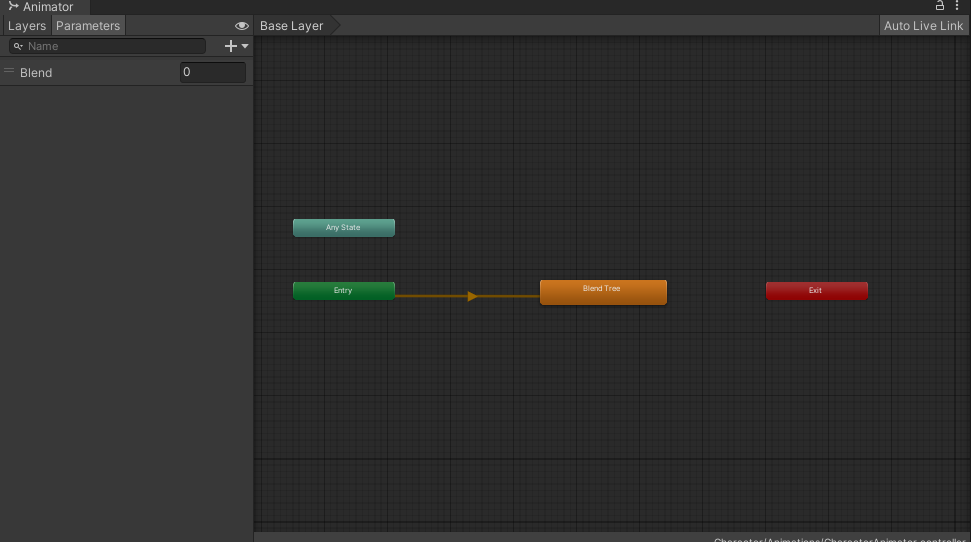
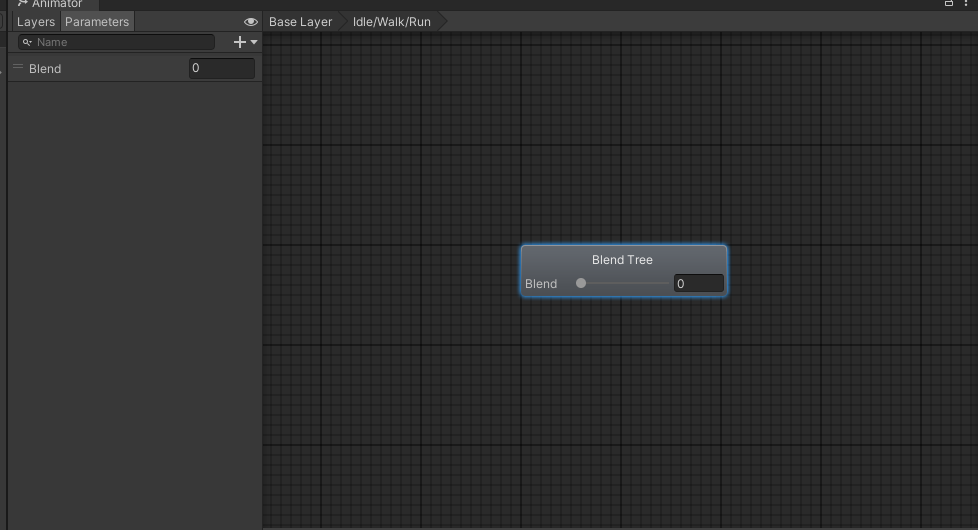
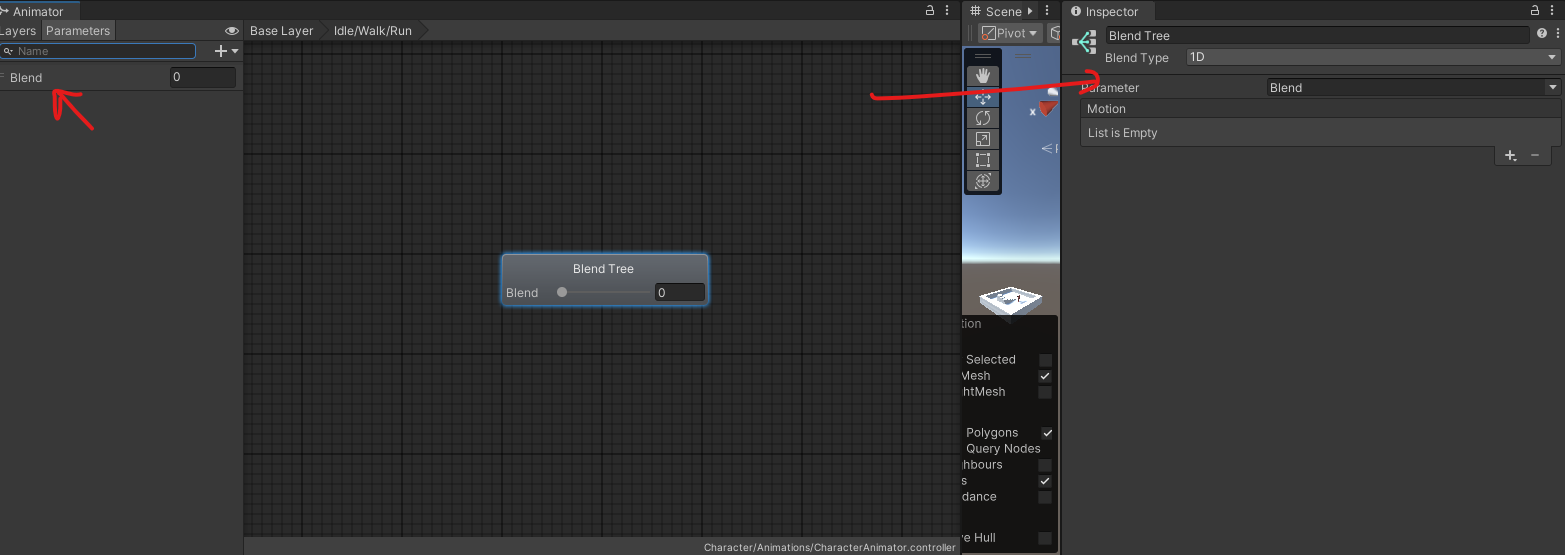
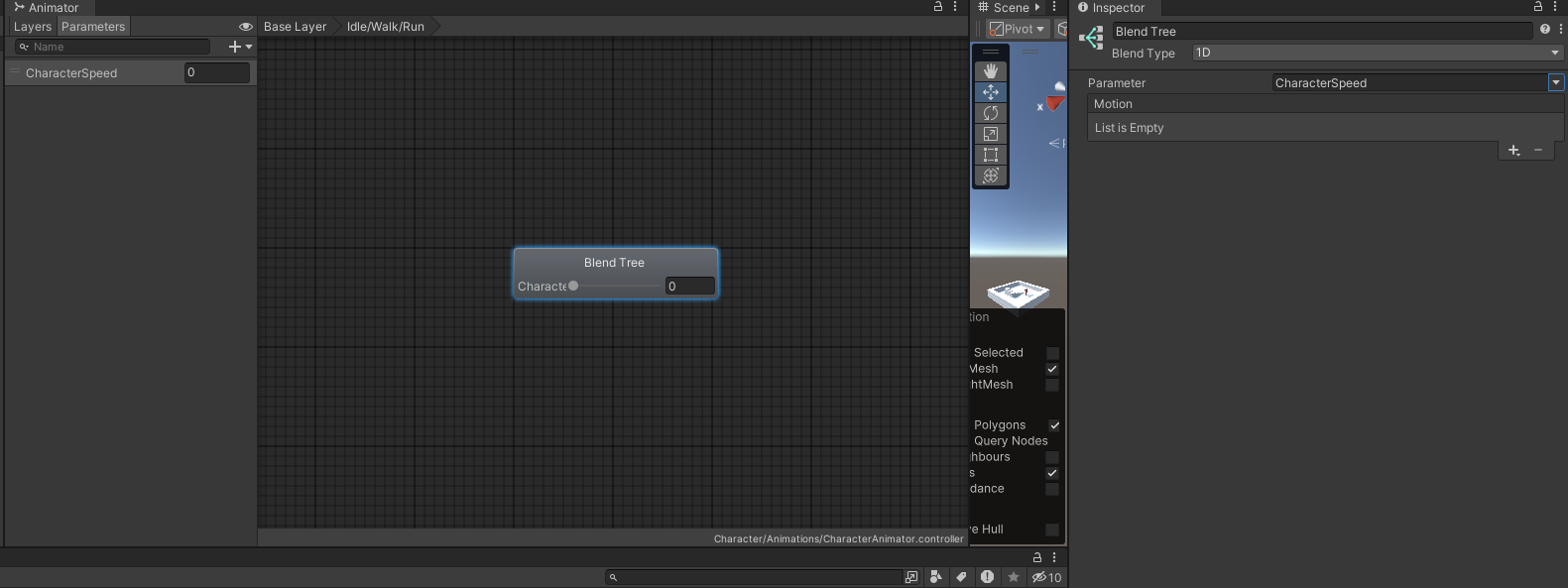
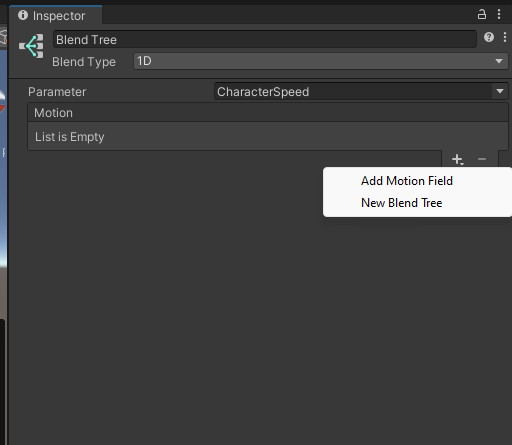
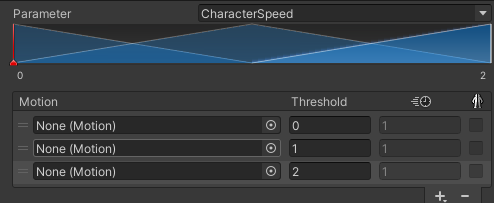
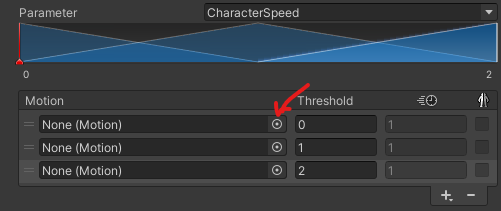
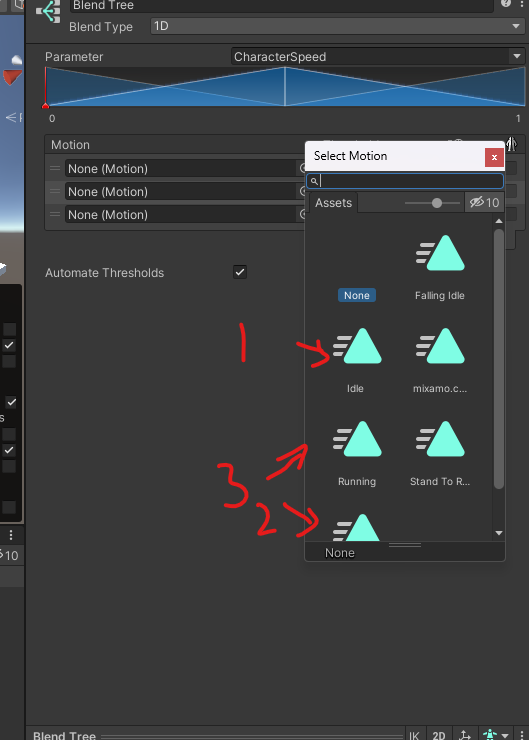
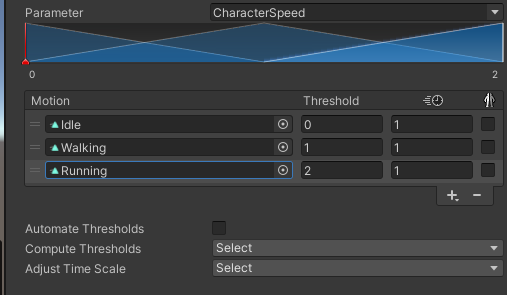
1. **Select Your Character:**
   * In Unity, go to the **Hierarchy** window and locate your Player prefab.



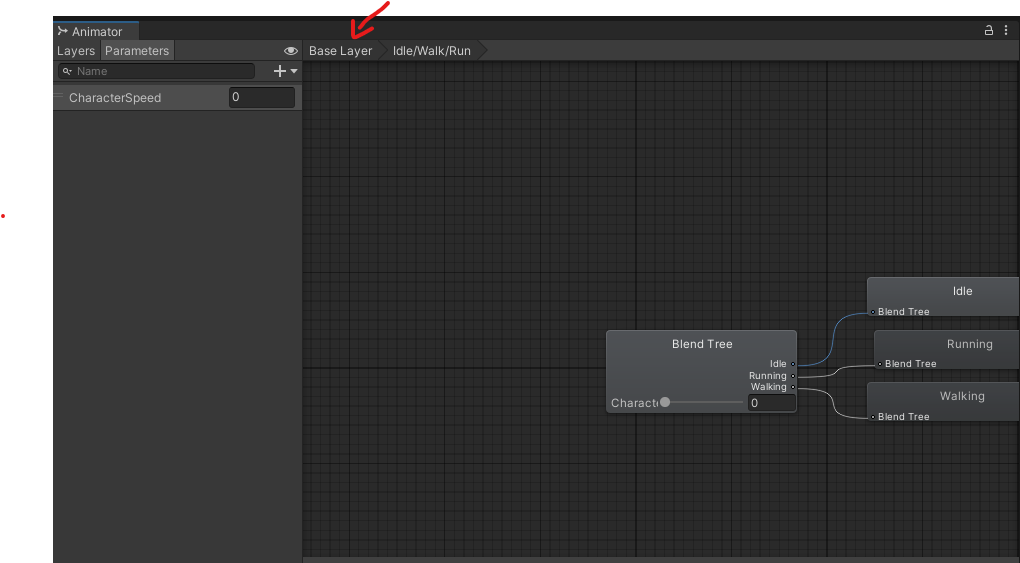
1. **Create an Animator Controller:**
   * In the **Assets** window (Project tab),open the **Character** Folder, then the **Animation** Folder
   * Then right-click and choose **Create > Animator Controller**.
   * Name the new Animator Controller, for example, CharacterAnimator.
   * ****
2. **Assign the Animator Controller to the Character:**
   * Select your character in the **Hierarchy** and look in the **Inspector** window.
   * Under the **Animator** component, assign the newly created CharacterAnimator controller by dragging it from the **Assets** window to the **Controller** slot in the **Animator** component.
   * 
3. **Ensure the Avatar is Assigned:**
   * In the **Animator** component, there is a field for **Avatar**. Ensure that the Avatar (skeleton) for your character is properly assigned here. Unity should auto-assign it when importing the character, but if it is missing, you can manually select the Avatar file associated with your character model.
   * 
4. **Open the Animator Controller:**
   * You should already have animations (Idle, Walking, Running, Rolling, Falling) in your **Animation** folder( Assets/Character/Animation).
   * Open the **Animator** window by going to **Window > Animation > Animator or by double clicking on the CharacterAnimator** in the assets view .
     1. You should see a window like this that pop up



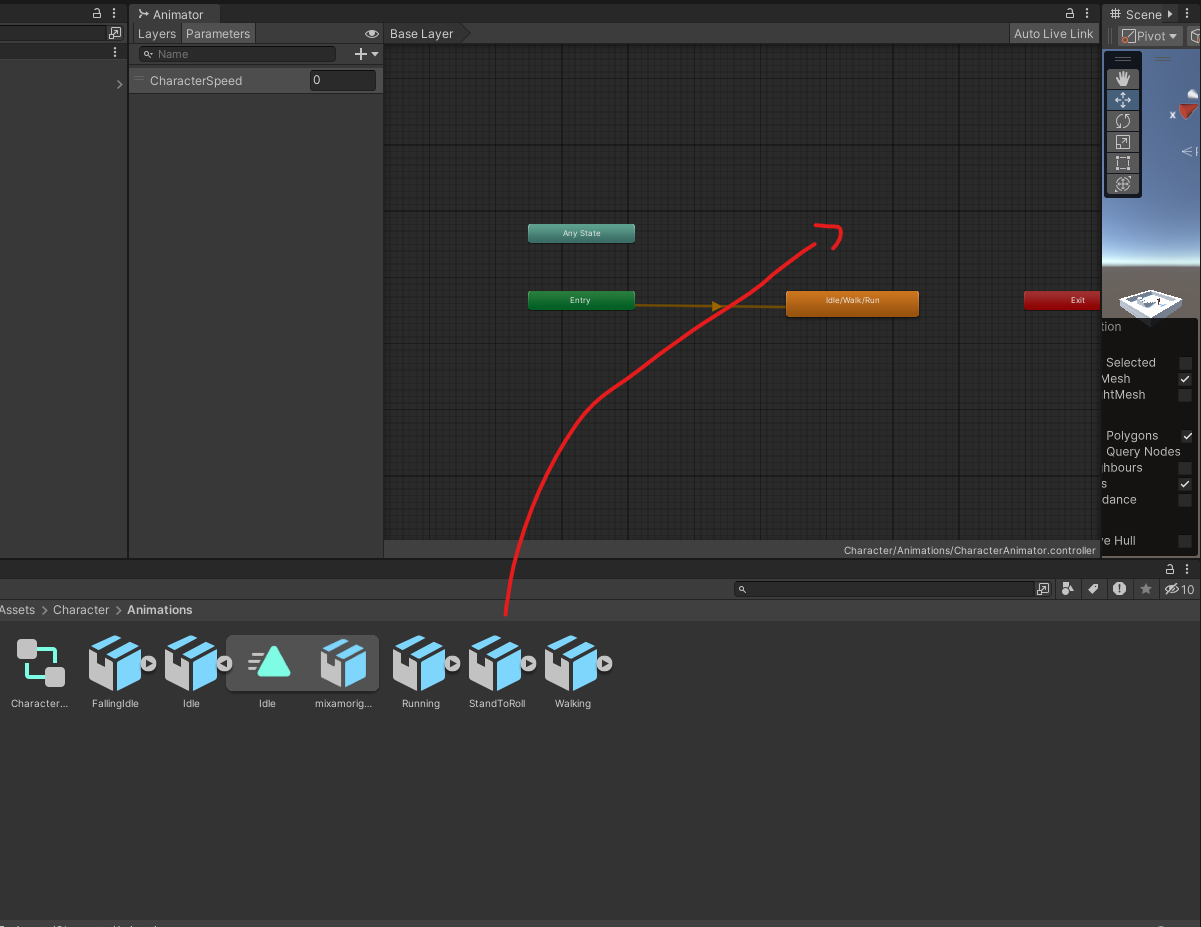
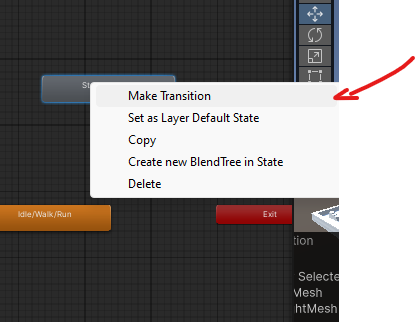
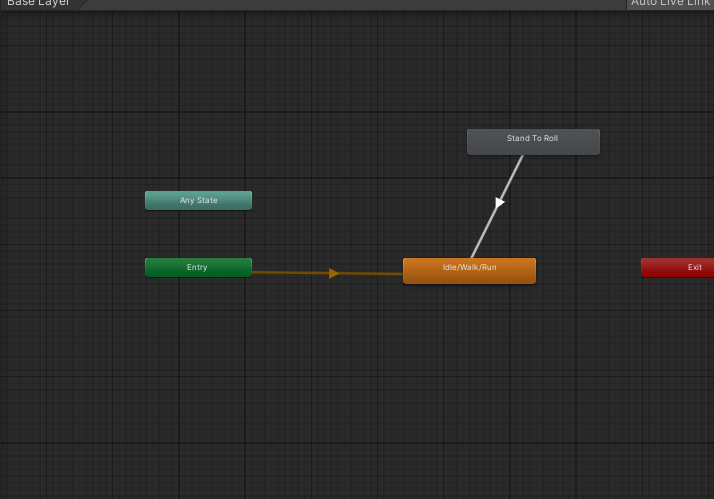
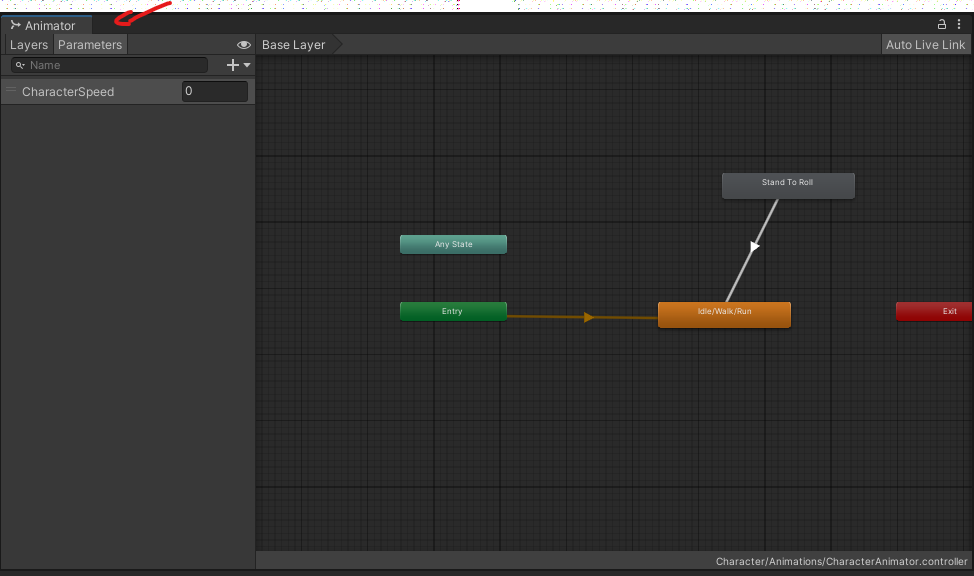
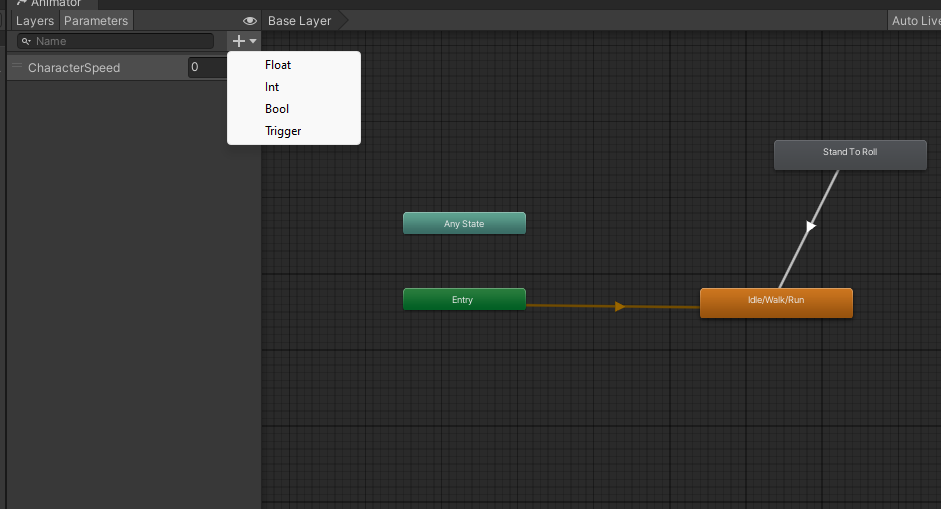
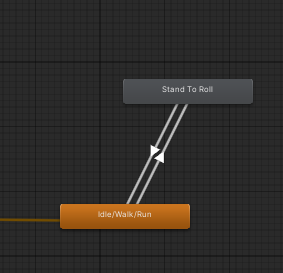
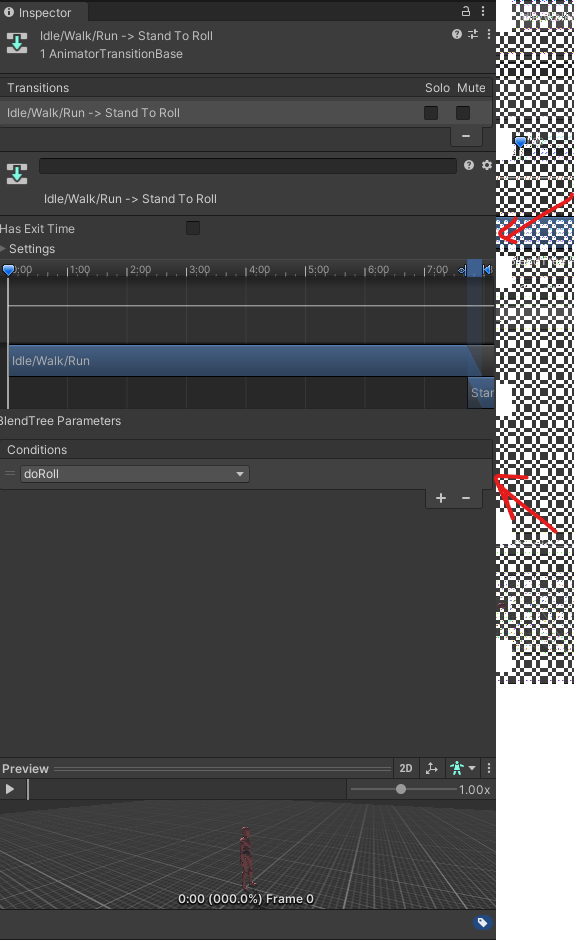
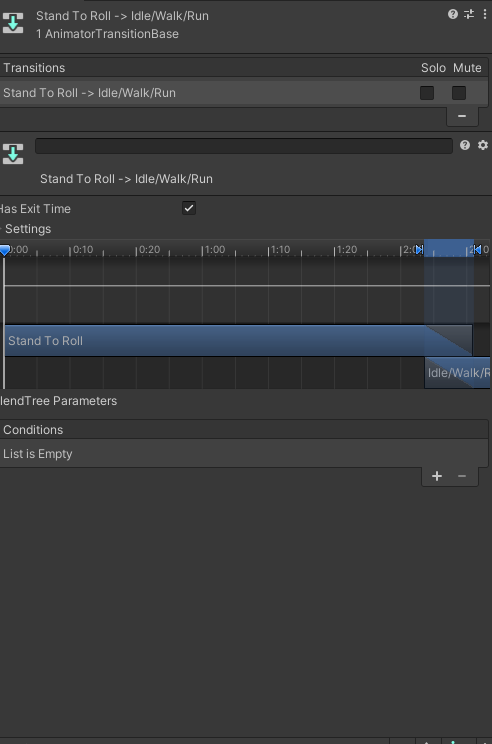
### **Task 2: Create a Blend Tree for Movement (Idle/Walking/Running)**

1. **Create a Blend Tree State:**
   * In the **Animator** window, right-click in an empty space and choose **Create State > From New Blend Tree**.
   * 
   * 
   * 
     + In order to rename it. Right click on the Orange box and details of that stat should show up in the inspector
     + 
     + You can now rename the State at the top of the Inpsector
     + Rename this blend tree Idle/Walking/Running to represent the different movement states.
   * By default, this state should appear in **orange**, meaning it is the entry state (the default state when the game starts).
2. **Open the Blend Tree:**
   * Double-click on the Idle/Walking/Running state to open the blend tree editor.
   * 
3. **Set the Blend Parameter:**
   * In the **Inspector** panel (while the blend tree is selected), you will see a **Blend** section.
   * 
   * Set the **Blend Parameter** to a new parameter called CharacterSpeed. This parameter will control which animation (Idle, Walking, or Running) plays based on the character’s speed.
4. **Add Animations to the Blend Tree:**
   * In the **Inspector** panel, click **+ Add Motion Field** to add animation slots to the blend tree.
   * Do that 3 times 
   * 
   * 
   * Click the little dot in the entry of the List of Motion,then select
   * 
   *   
     **Idle** to the first slot , **Walking** to the second, and **Running** to the third slot.
   * Uncheck the Compute threshold checkbox.
   * 
   * Your blend tree all setup should look like the picture above
5. **Configure Speed Thresholds:**
   * Set the speed values for each animation to correspond with different character speeds:
     + Idle: Speed = 0
     + Walking: Speed = 5
     + Running: Speed = 7
   * 
   * As the Character Speed parameter increases from 0 to 2, the blend tree will interpolate between these animations based on the speed of the character.

Go back to the base layer in the Animator by clicking on the Base Layer Button :

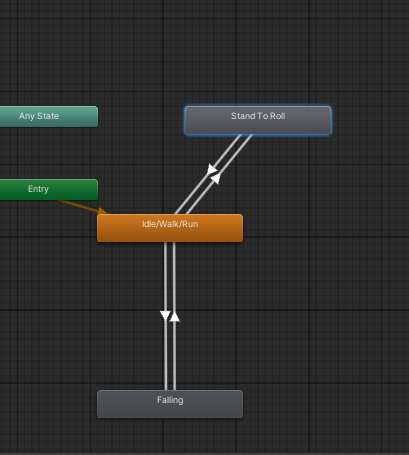


### **Task 3: Create Transitions for the "Stand to Roll" Animation**

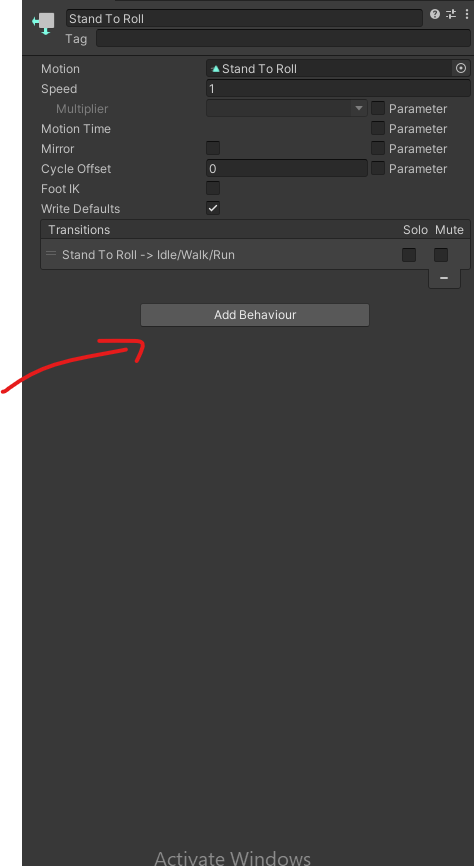
1. **Add the Rolling Animation:**
   * Drag the **Stand to Roll** animation from your **Assets** folder into the **Animator** window, creating a new state.
2. **Create a Transition from Idle/Walking/Running to Rolling:**
   * Right-click on the Idle/Walking/Running blend tree and select **Make Transition**.
   * Drag the arrow to the **Stand to Roll** state.
   * 
3. **Create a Transition from Rolling back to Idle/Walking/Running:**
   * Right-click on the **Stand to Roll** state, choose **Make Transition**, and drag the arrow back to the Idle/Walking/Running blend tree.
   * 
   * 
4. **Setup the "IsRolling" Parameter:**
   * In the **Animator** window, click the **Parameters** tab (next to Layers).
   * 
   * Add a new **Trigger** parameter called doRoll.
   * 
   * ****
   * By the end your stand to roll should be connected like the previous picture
5. **Configure Transition Conditions:**
   * Select the transition from Idle/Walking/Running to **Stand to Roll**.
     1. Add the parameter doRoll in the condition
     2. Uncheck Has Exit time
     3. 
   * Select the transition back to Idle/Walking/Running make sure the HasExitTime is still checked to make sure the roll animation plays fully before going back to the Idle/Walk/Run state
   * 

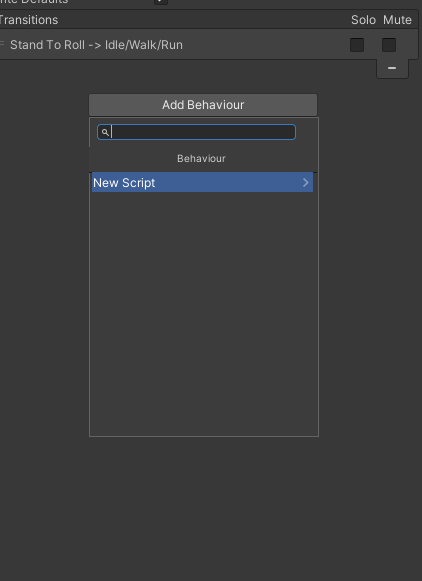
Adding a behaviour to the roll.

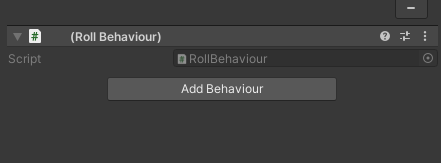
Click on the state Stand to roll



On the left panel you should see this window click on the Add behaviour button.



Click new Script   


Call it **Roll Behaviour   
**

**Update the script you just created this script. This will enable and disable rootmotion at the begining and end of the roll animation so that only the roll is affected by root motion.**

**using System.Collections;**

**using System.Collections.Generic;**

**using UnityEngine;**

**public class RollBehaviour : StateMachineBehaviour**

**{**

**// OnStateEnter is called when a transition starts and the state machine starts to evaluate this state**

**override public void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)**

**{**

**animator.applyRootMotion = true;**

**}**

**// OnStateExit is called when a transition ends and the state machine finishes evaluating this state**

**override public void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int layerIndex)**

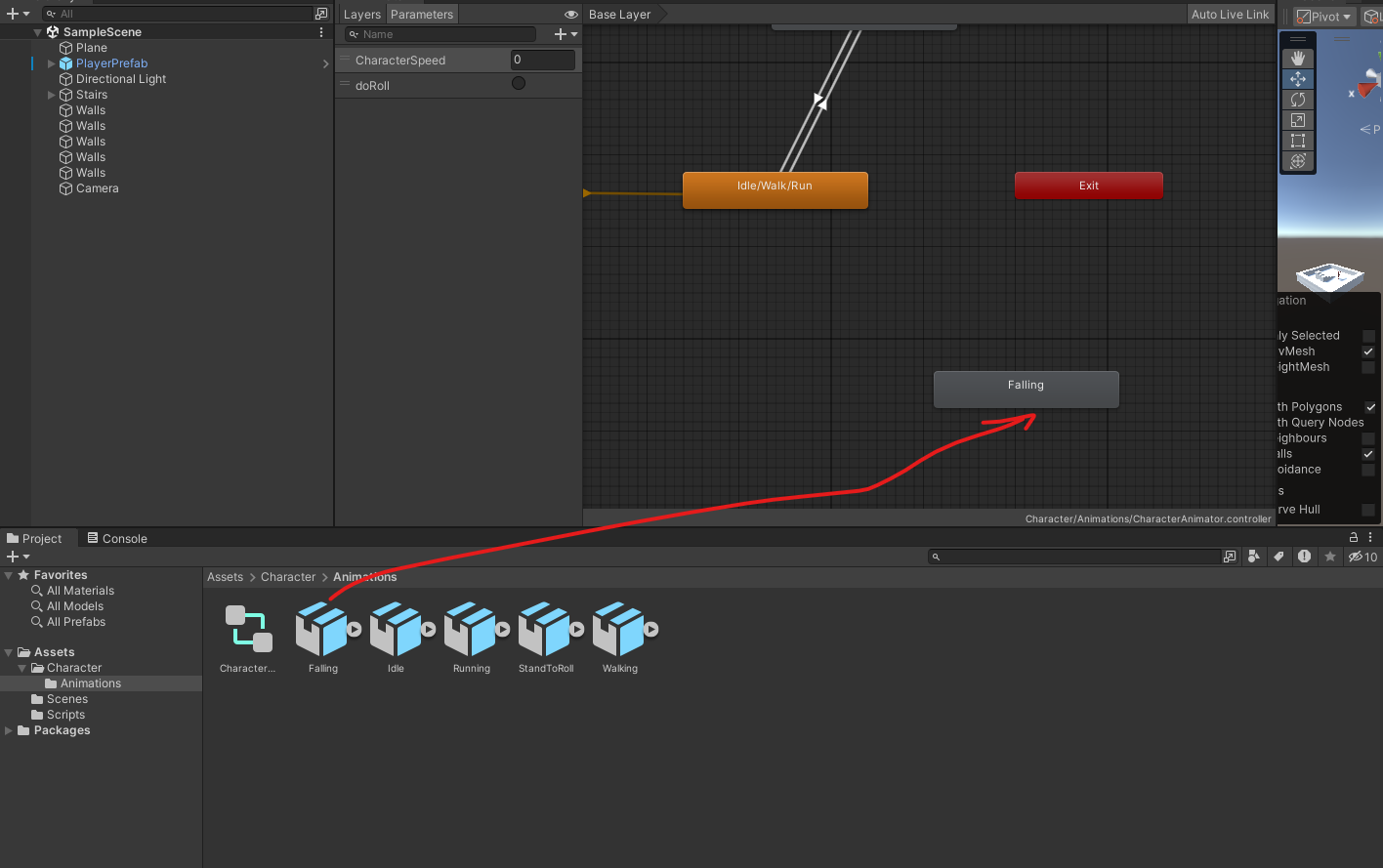
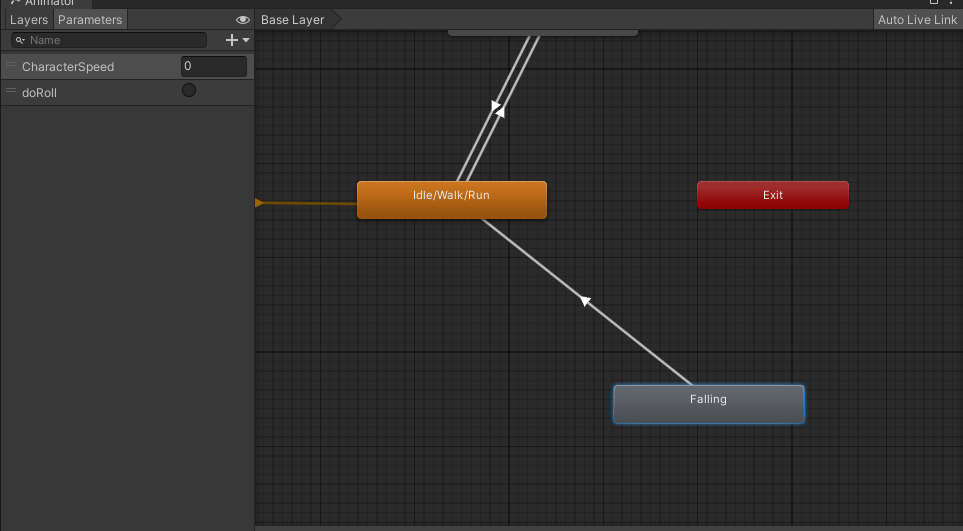
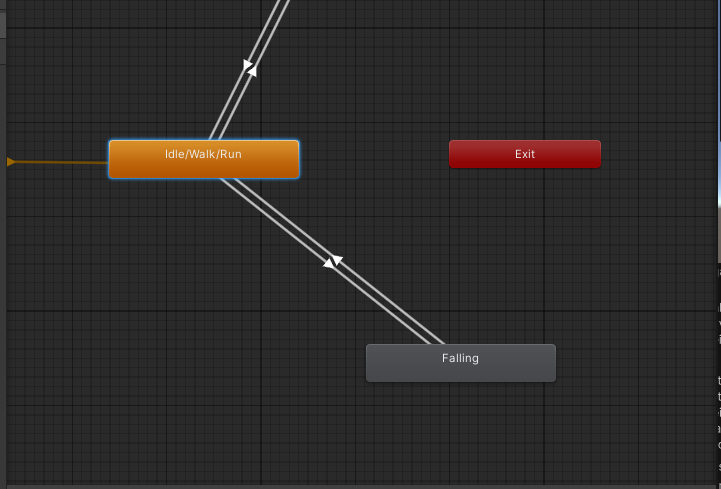
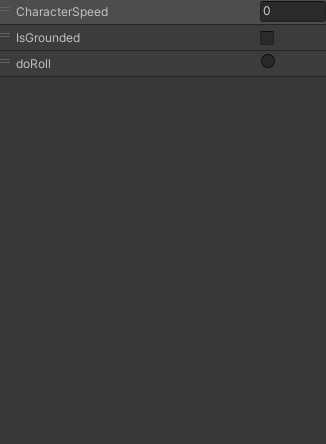
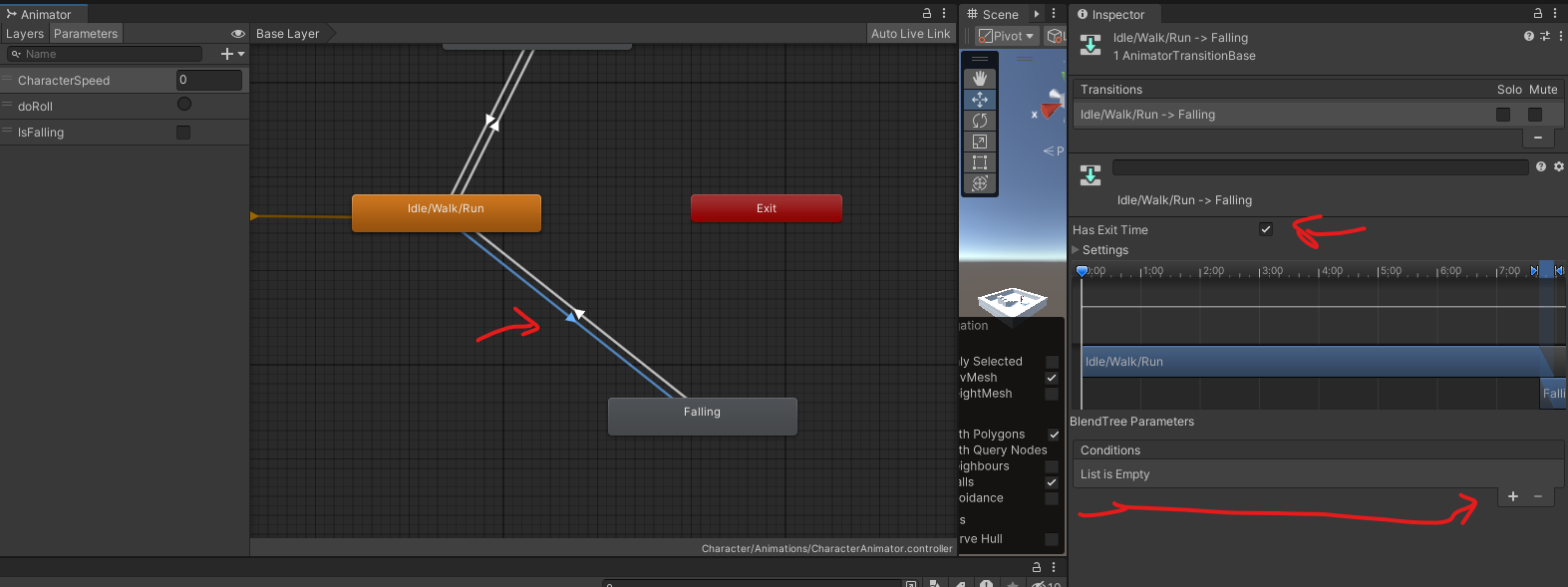
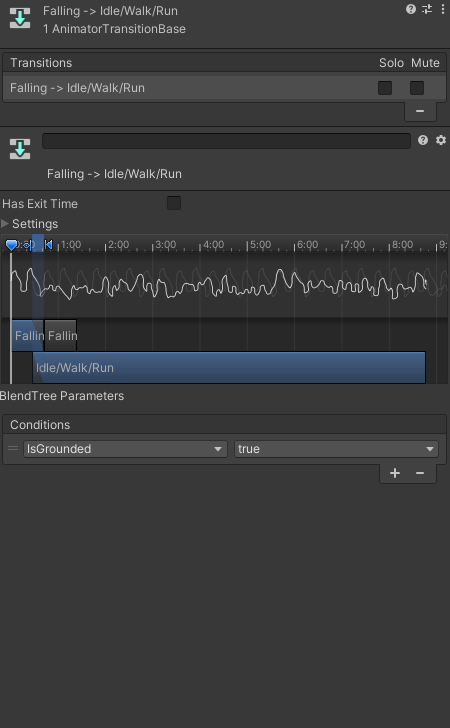
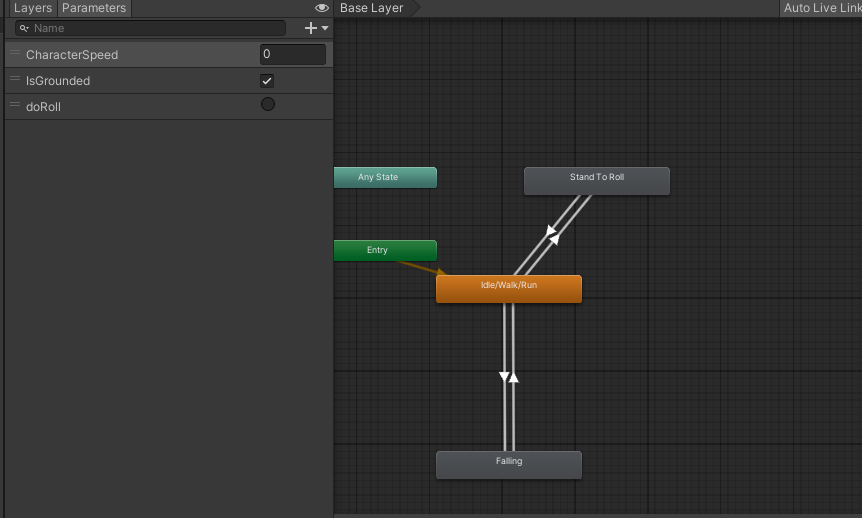
**{**

**animator.applyRootMotion = false;**

**}**

**}**

### **Task 4: Create Transitions for Falling Animation**

1. **Add the Falling Idle Animation:**
   * Drag the **Falling** animation from your **Assets** folder into the **Animator** window, creating a new state.
   * 
2. **Create a Transition from Idle/Walking/Running to Falling:**
   * Right-click on the Idle/Walking/Running state, select **Make Transition**, and drag the arrow to the **Falling** state.
   * 
3. **Create a Transition from Falling back to Idle/Walking/Running:**
   * Right-click on the **Falling** state, select **Make Transition**, and drag the arrow back to Idle/Walking/Running.
   * 
4. **Setup the "IsGrounded" Parameter:**
   * In the **Parameters** tab, add a new **Bool** parameter called IsGrounded.
   * 
5. **Configure Transition Conditions:**
   * Select the transition from Idle/Walking/Running to **Falling**.
   * In the **Inspector**, set the condition to **IsGrounded== false** and make sure **HasExitTime** is checked off. ( We want to transition as soon as IsFalling Becomes true
   * 
   * 
   * Select the transition from **Falling Idle** back to Idle/Walking/Running and set the condition to **IsGrounded== true** and we also want to make sure HasExitTime is checked off.
   * 
6. Make sure the parameter IsGrounded is set to true by default
7. 

### **Task 5: Create an PlayerAnimatiorController Script to Control Animation Parameters**

Create a script called **PlayerAnimatiorController** on the Player use this code by click add component and creating a new script on the player game object

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class PlayerAnimatorController : MonoBehaviour

{

private Animator animator;

private CharacterMovement movement;

private Rigidbody rb;

public void Start()

{

animator = GetComponent<Animator>();

movement = GetComponent<CharacterMovement>();

rb = GetComponent<Rigidbody>();

}

public void Update()

{

animator.SetFloat("CharacterSpeed", rb.velocity.magnitude);

animator.SetBool("IsGrounded", movement.IsGrounded);

if (Input.GetButtonUp("Fire1"))

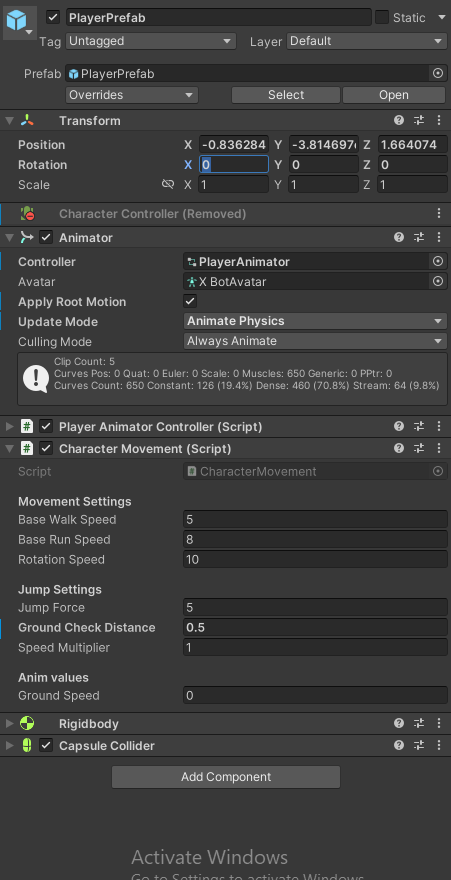
{

animator.SetTrigger("doRoll");

}

}

}



### **Final Steps:**

1. **Test Your Project:**
   * Press **Play** in Unity to test the character’s animations. Ensure the transitions between Idle, Walking, Running, Rolling, and Falling work as expected based on your inputs.
2. **Submission:**
   * Submit your assignment through github