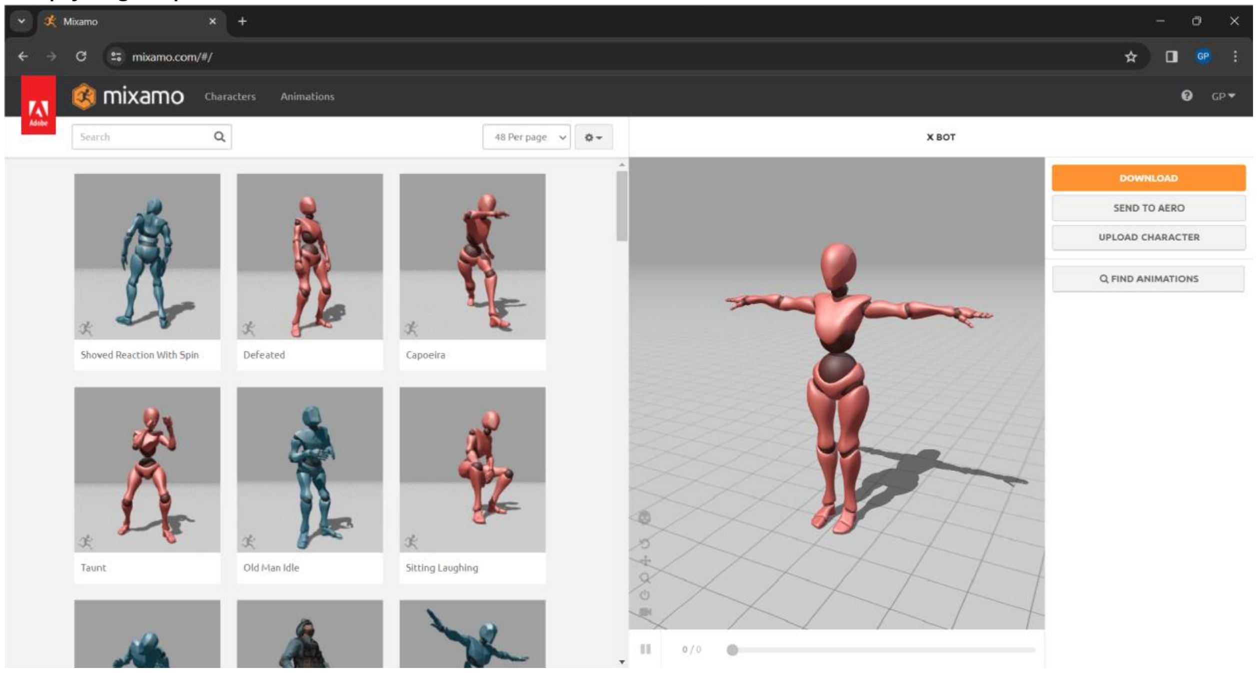
Practical No. 3

Aim: How to animate character in Unity, import your character and an add animator

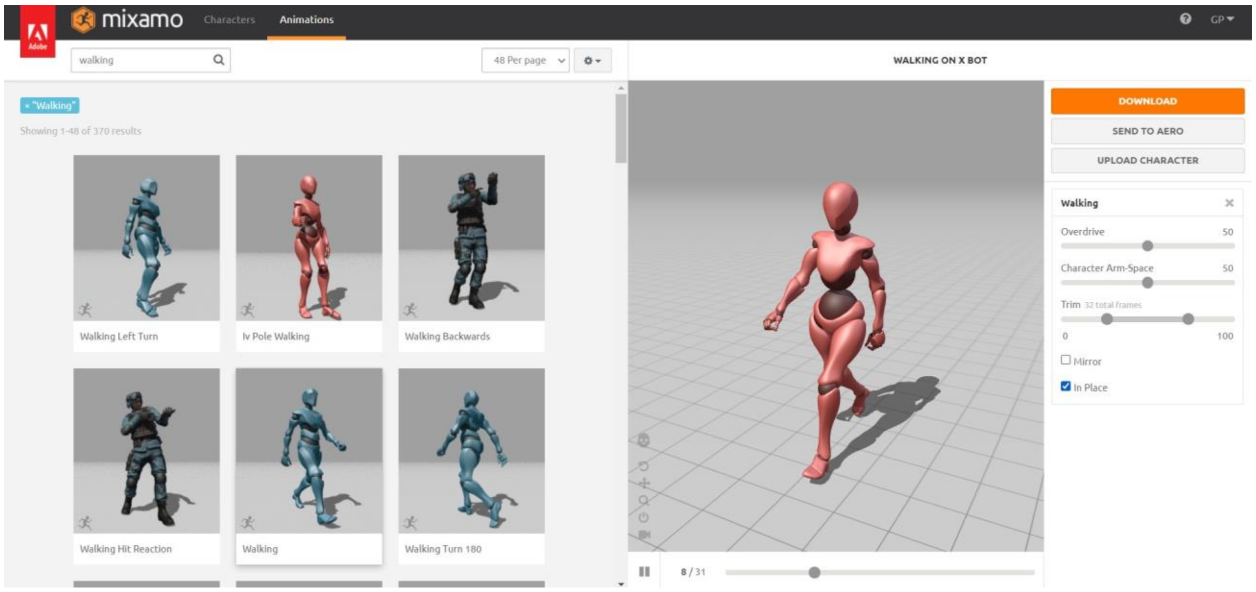
For Character Model creation here we are using Mixamo (here we can also use blender)
https://www.mixamo.com/#/

What is Mixamo?
Mixamo is a web-based service provided by Adobe that offers a variety of tools and resources for creating and animating 3D characters. It is particularly known for its vast library of pre-made 3D character models and animations, which can be easily customized and applied to various projects.

Simply sign up on Mixamo



You can explore more animation characters
For our Practical we will go for 3 animations i.e. Idle animation, Walking Animation, and Running Animation



Walking

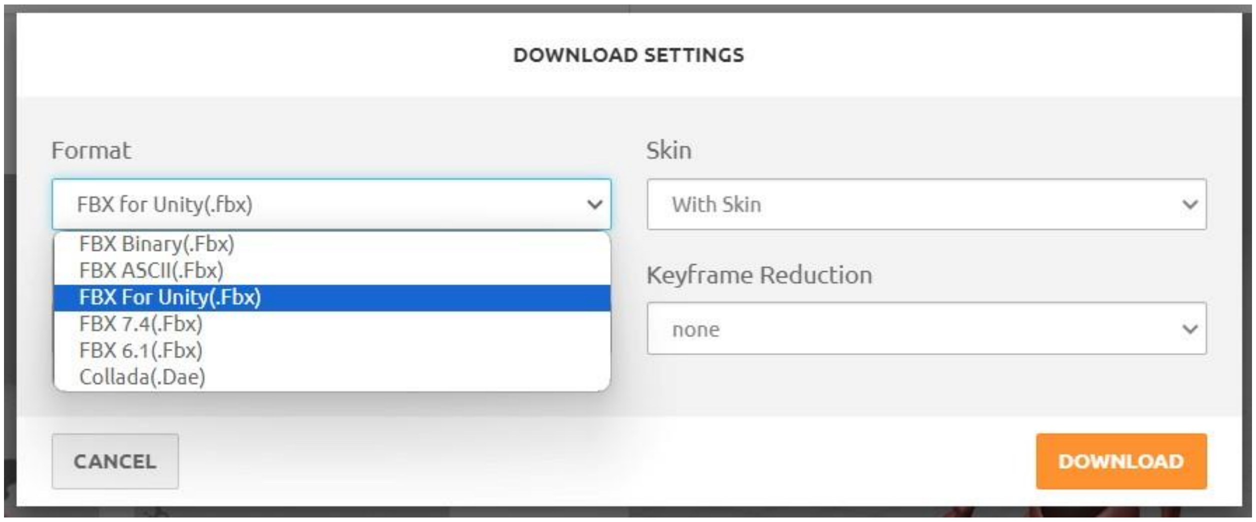
Overdrive 50

Character Arm-Space 50

Trim 32 total frames 0 100

☐ Mirror

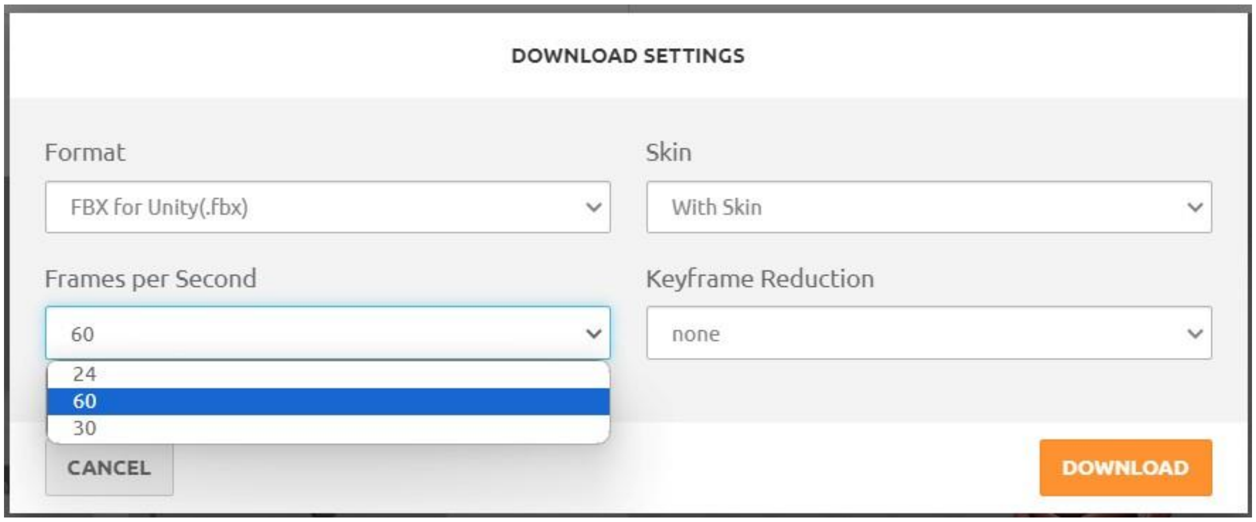
☑ In Place


DOWNLOAD SETTINGS

Format: FBX for Unity(.fbx)

Skin: With Skin

Keyframe Reduction: none

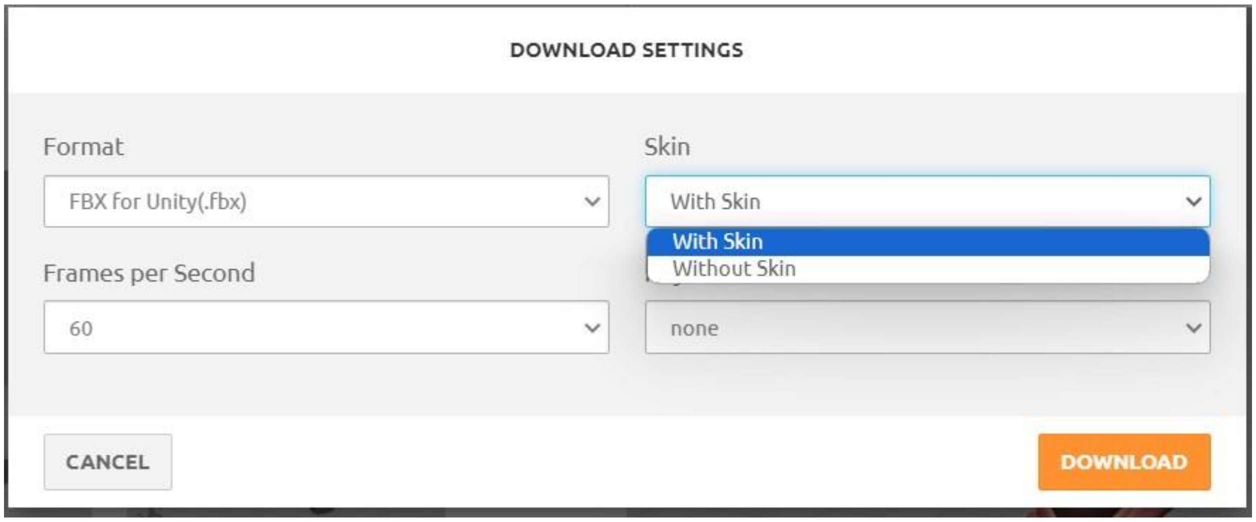

DOWNLOAD SETTINGS

Format: FBX for Unity(.fbx)

Skin: With Skin

Frames per Second: 60

Keyframe Reduction: none

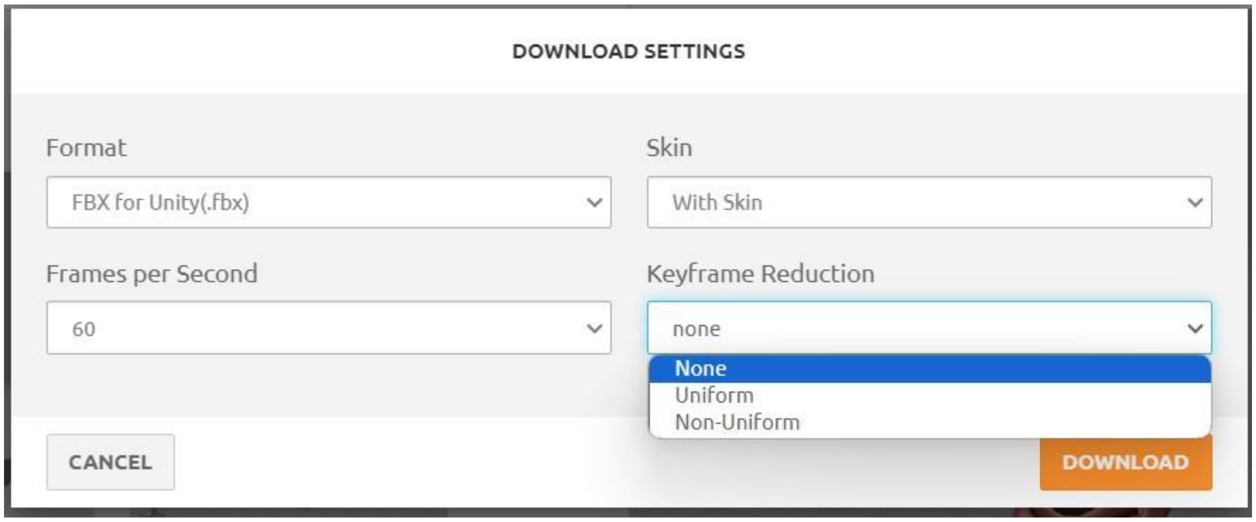

DOWNLOAD SETTINGS

Format: FBX for Unity(.fbx)

Skin: With Skin

Frames per Second: 60

none


DOWNLOAD SETTINGS

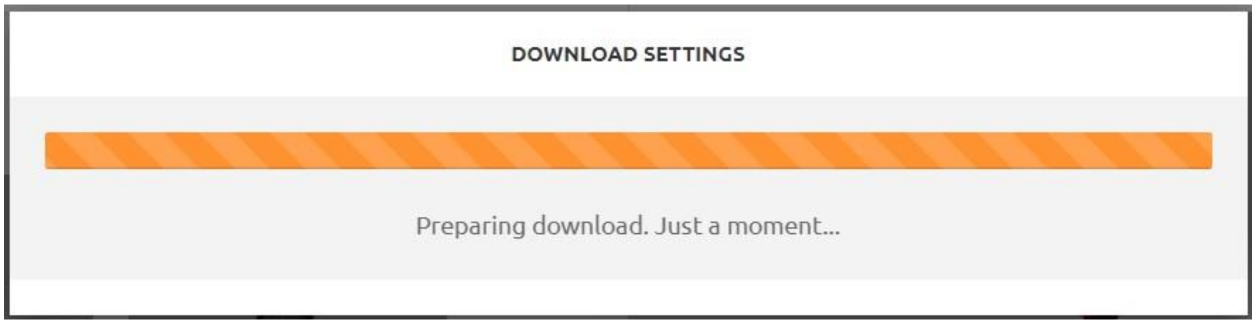
Format: FBX for Unity(.fbx)

Skin: With Skin

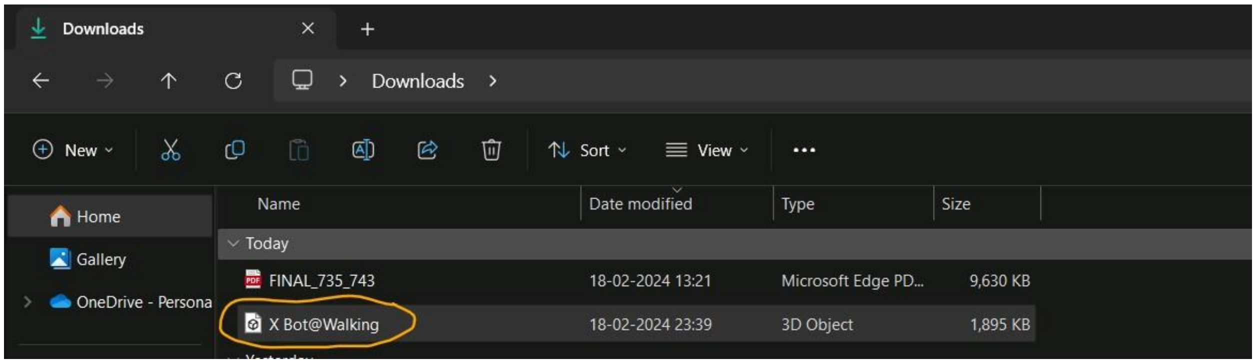
Frames per Second: 60

Keyframe Reduction: none

Click on Download


DOWNLOAD SETTINGS

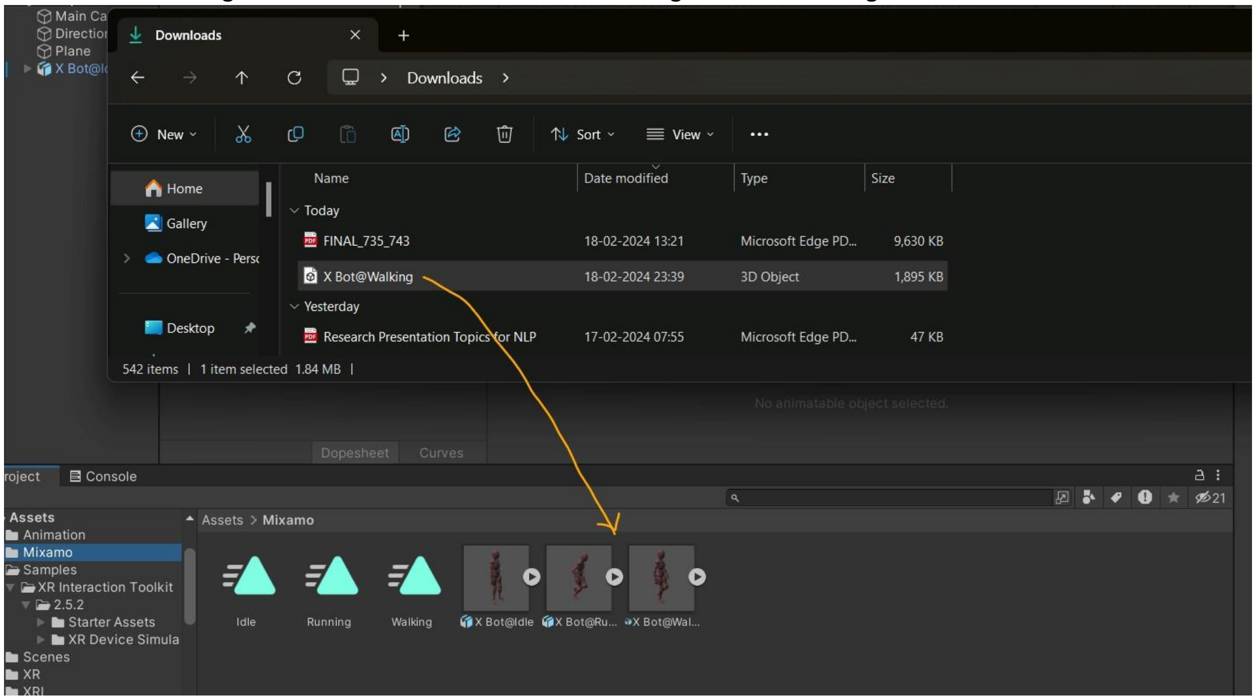
Preparing download. Just a moment...



Name	Date modified	Type	Size
Today			
FINAL_735_743	18-02-2024 13:21	Microsoft Edge PD...	9,630 KB
X Bot@Walking	18-02-2024 23:39	3D Object	1,895 KB

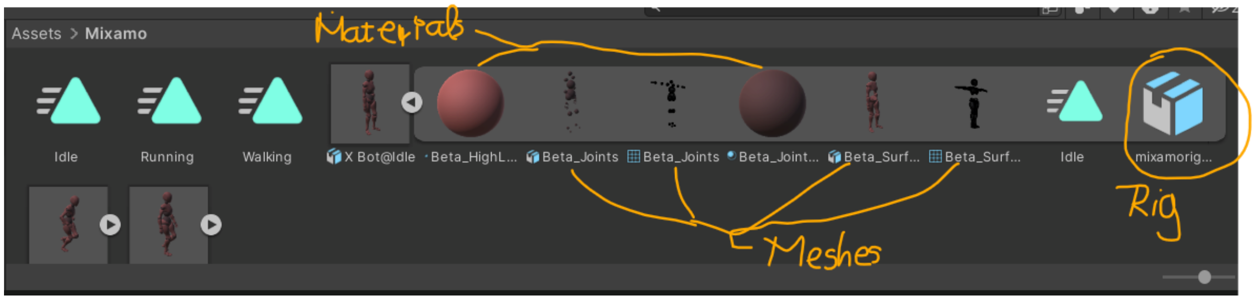
Create your Unity 3D Project ➞Create 2 folders named Animation for storing Animation Controller and second folder Mixamo for storing the 3D characters we downloaded ➞Now drag and drop your downloaded file from folder to Unity [This will basically Import your animation character on Unity, once added it won't start moving when we click on play in game mode, there are some settings to be done which we see..]

Do the same thing for other Animation Characters eg. idle & running animation

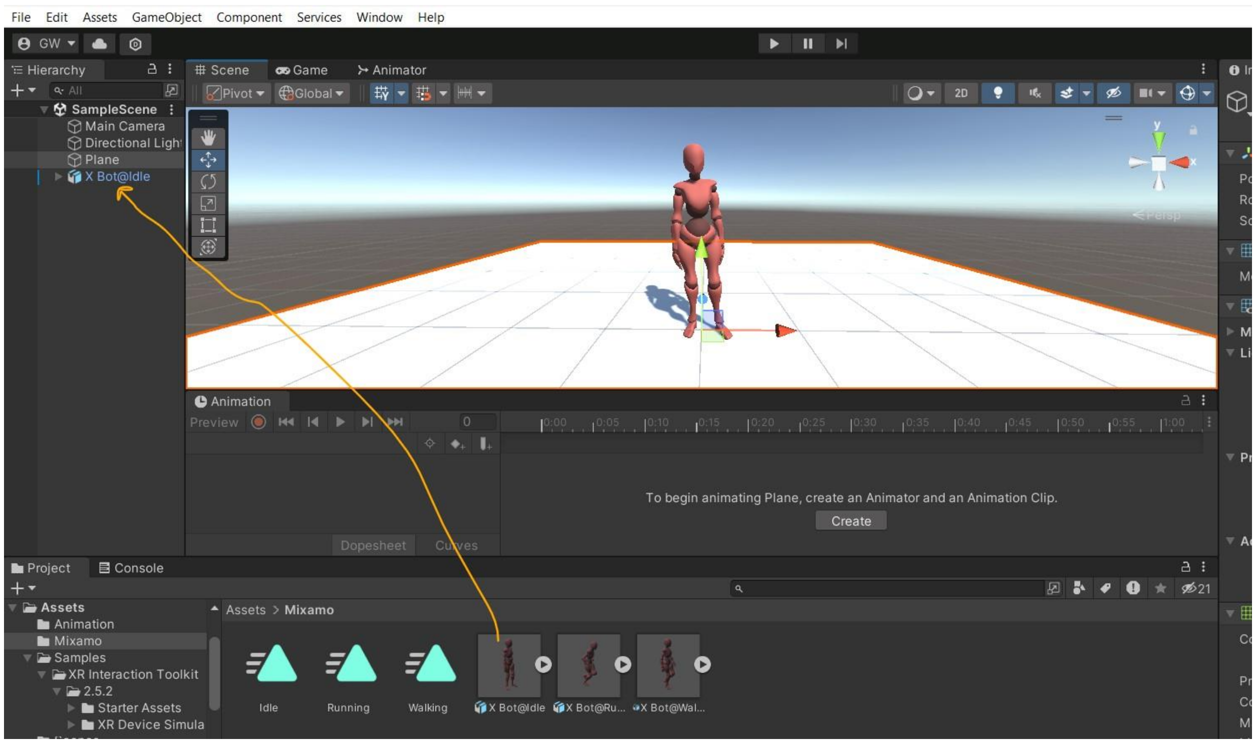


Name	Date modified	Type	Size
Today			
FINAL_735_743	18-02-2024 13:21	Microsoft Edge PD...	9,630 KB
X Bot@Walking	18-02-2024 23:39	3D Object	1,895 KB
Yesterday			
Research Presentation Topics for NLP	17-02-2024 07:55	Microsoft Edge PD...	47 KB

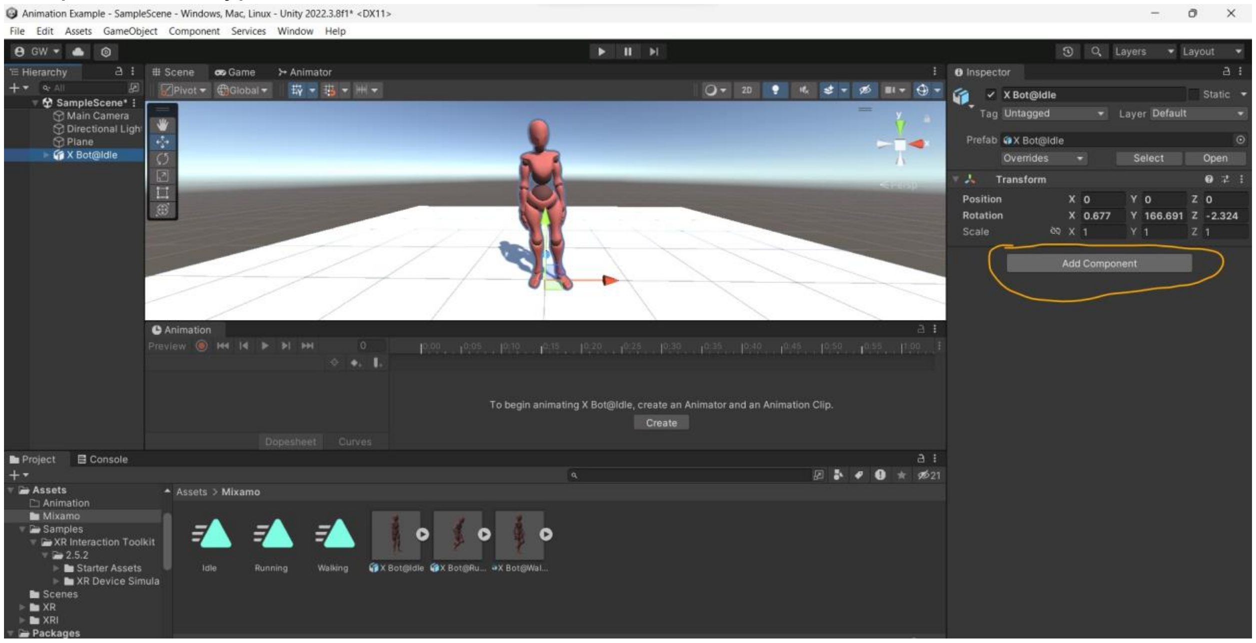
542 items | 1 item selected 1.84 MB |

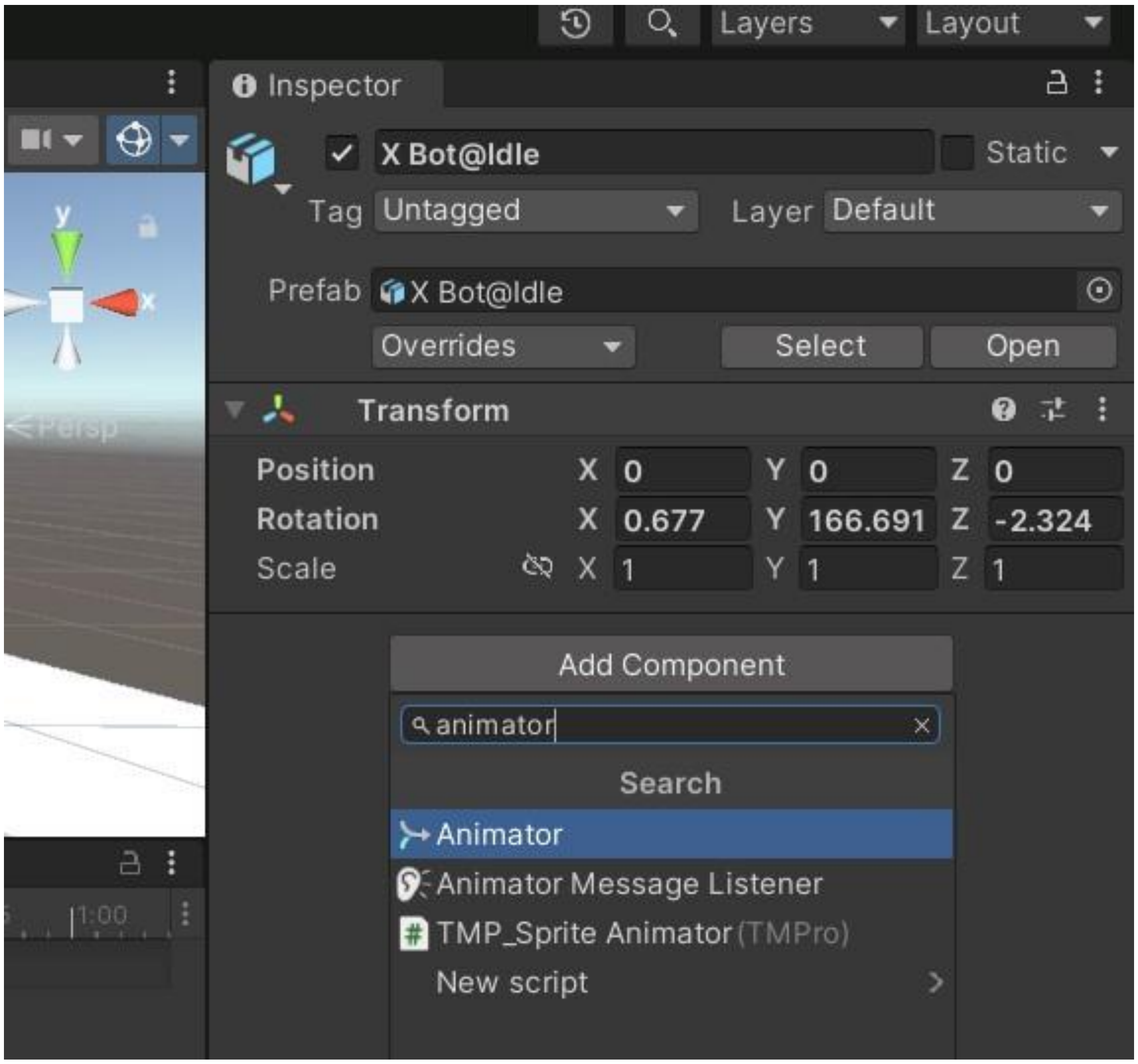


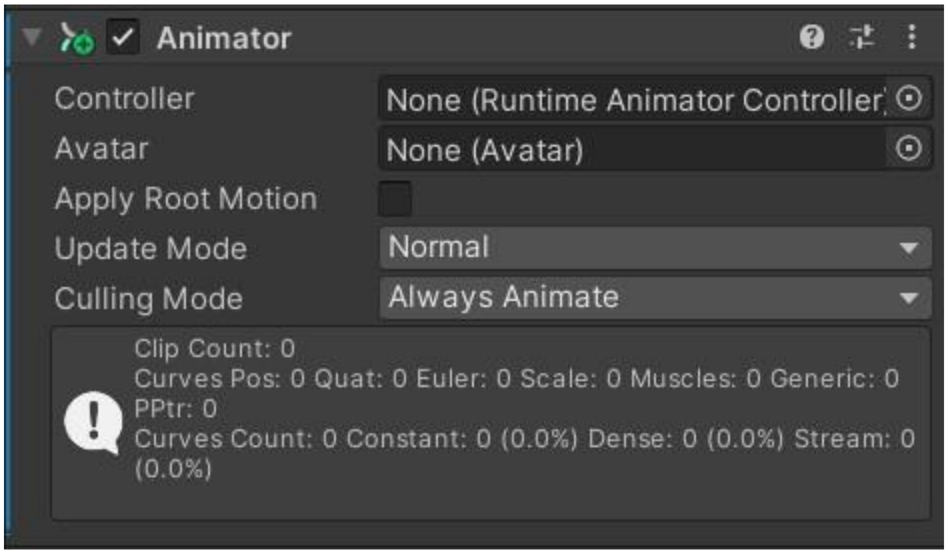
Add a Plane gameObject in Scene and add idle bot on Scene Hierarchy panel



Select the x-bot gameObject so that it's highlighted in the inspector → Let's navigate to Add Component and type 'Animator' and select it



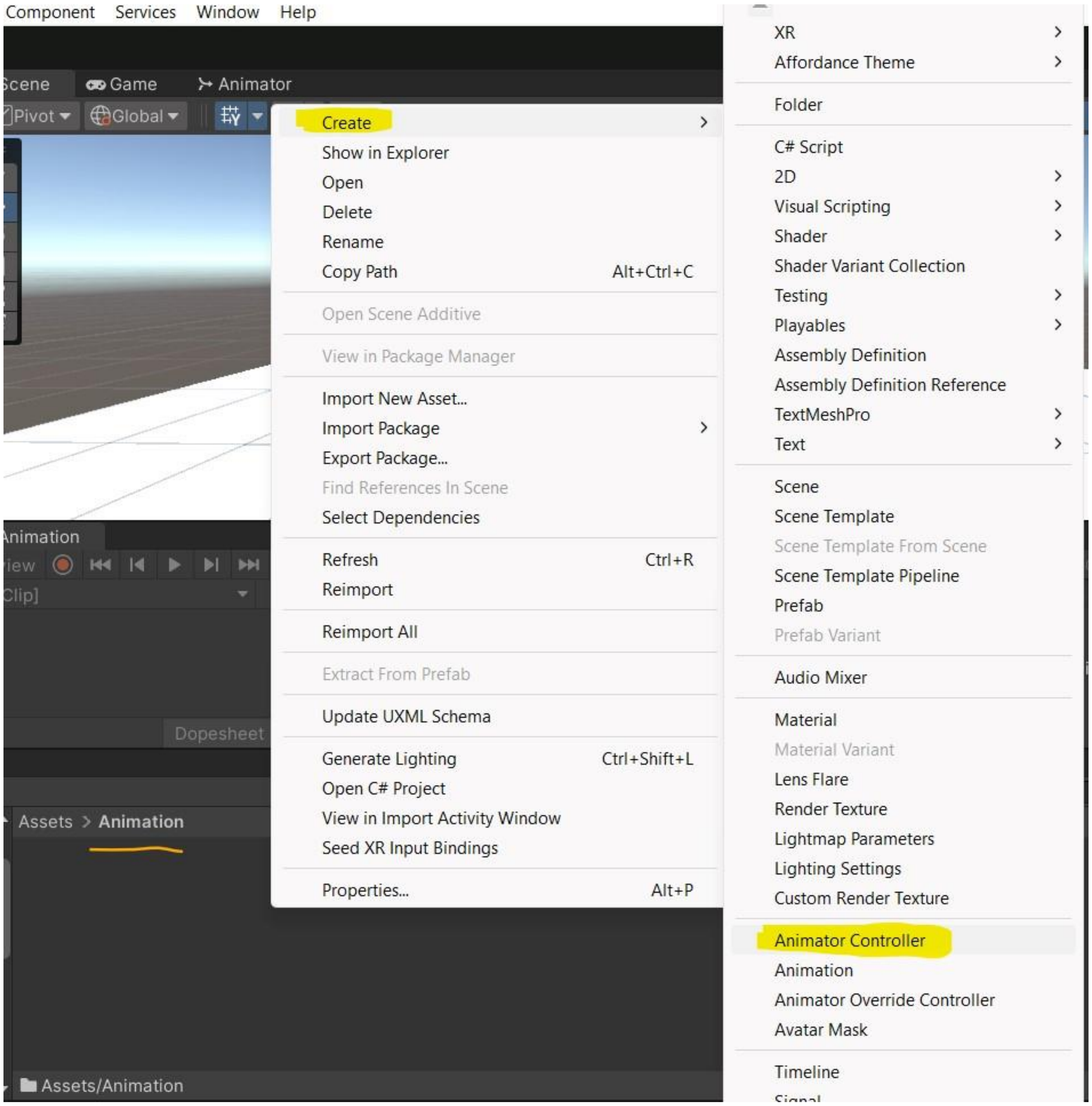




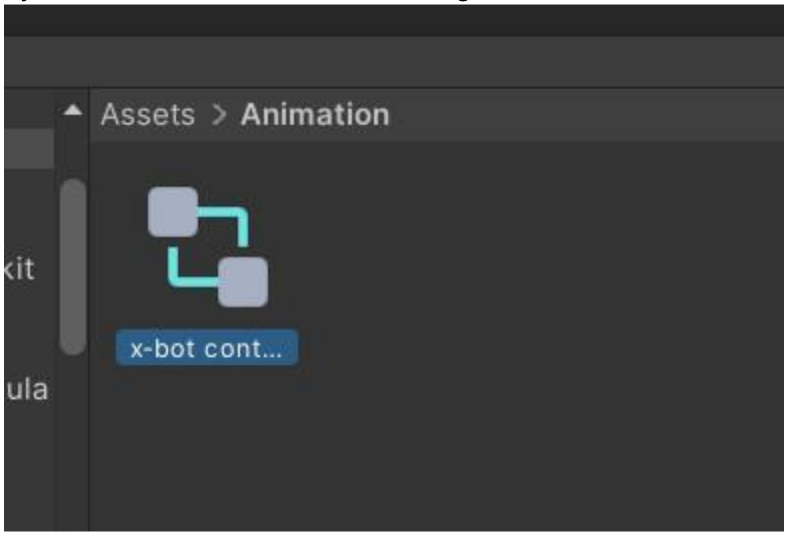
Controller: None (Runtime Animator Controller)
Avatar: None (Avatar)
Apply Root Motion:
Update Mode: Normal
Culling Mode: Always Animate

```
Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0
PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 0
(0.0%)
```

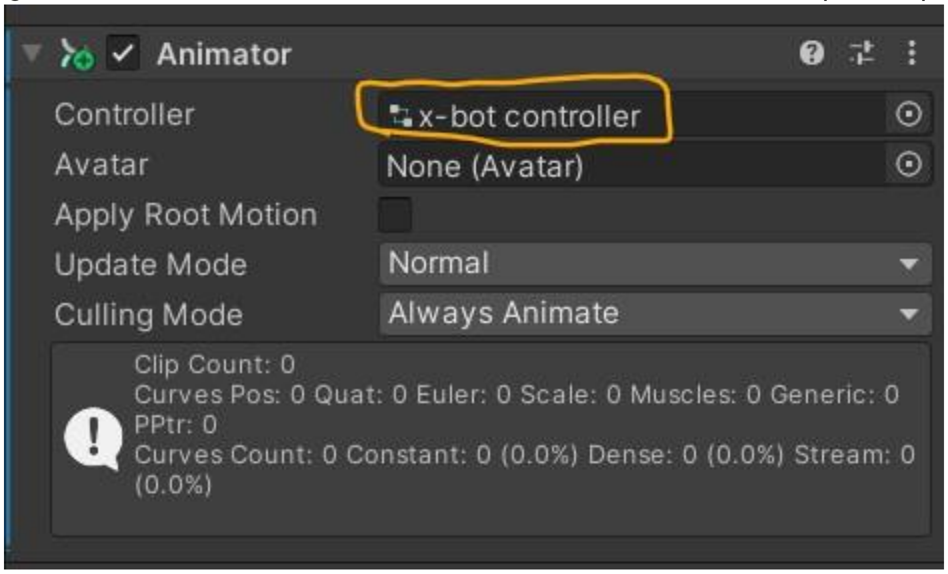
- First property **Controller** property is used to assign an Animator Controller from our Assets
  ➞For Animator Controller ➡ let's create it by right-clicking inside our Animator folder of the Assets tab ➡ Create ➡ Click on Animator Controller.



Rename it with your desired name here we have given ‘x-bot controller’.



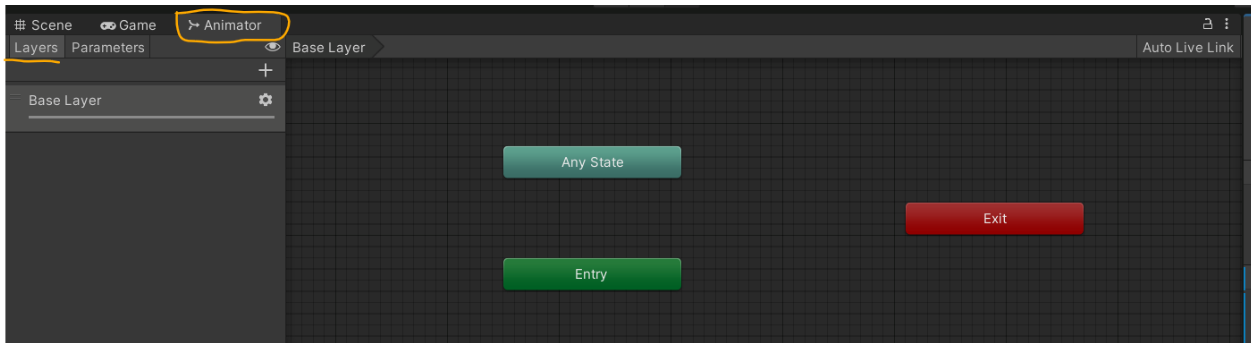
Now drag this asset into the slot next to **controller** in the Animator component properties

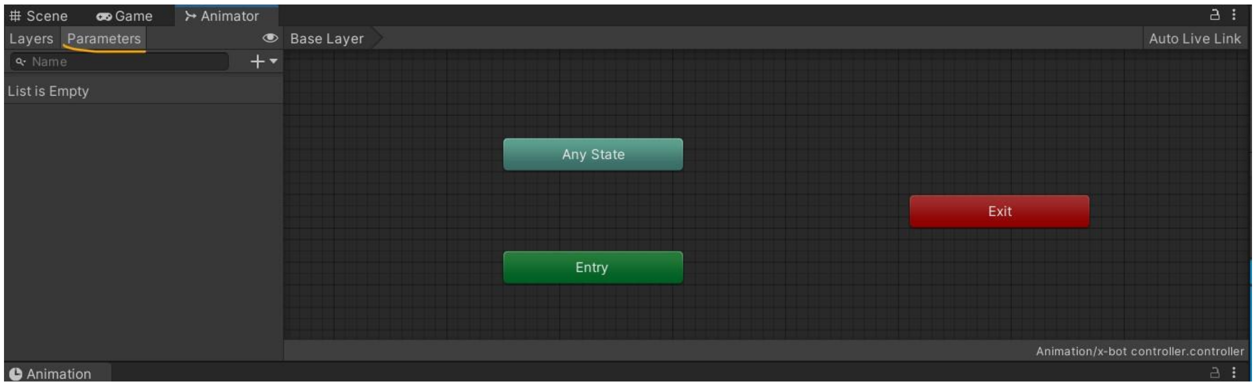

Animator

Controller	x-bot controller
Avatar	None (Avatar)
Apply Root Motion	
Update Mode	Normal
Culling Mode	Always Animate

Clip Count: 0
Curves Pos: 0 Quat: 0 Euler: 0 Scale: 0 Muscles: 0 Generic: 0 PPtr: 0
Curves Count: 0 Constant: 0 (0.0%) Dense: 0 (0.0%) Stream: 0 (0.0%)

Now double click on “x-bot controller”, Unity will open window titled **Animator** which consists of two panes: the grid pane on the right and the layers and parameters on the left





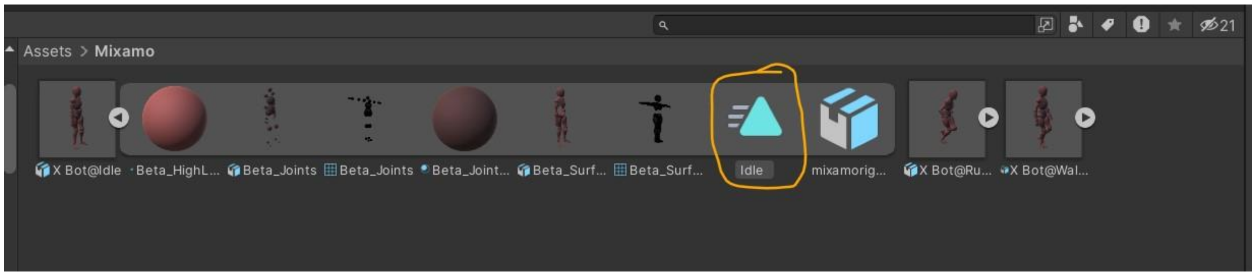
The grid pane serves as a visual interface for us to create, modify and connect animation, but first we need to know what is **Animation State?**

1. For this let us take an example of our three downloaded Animations: Idle, Walk and Run, Each of these represents own animation state in our project, so for Instance, if a player is not moving, then the character will be in **Idle state** (Current State →Idle)

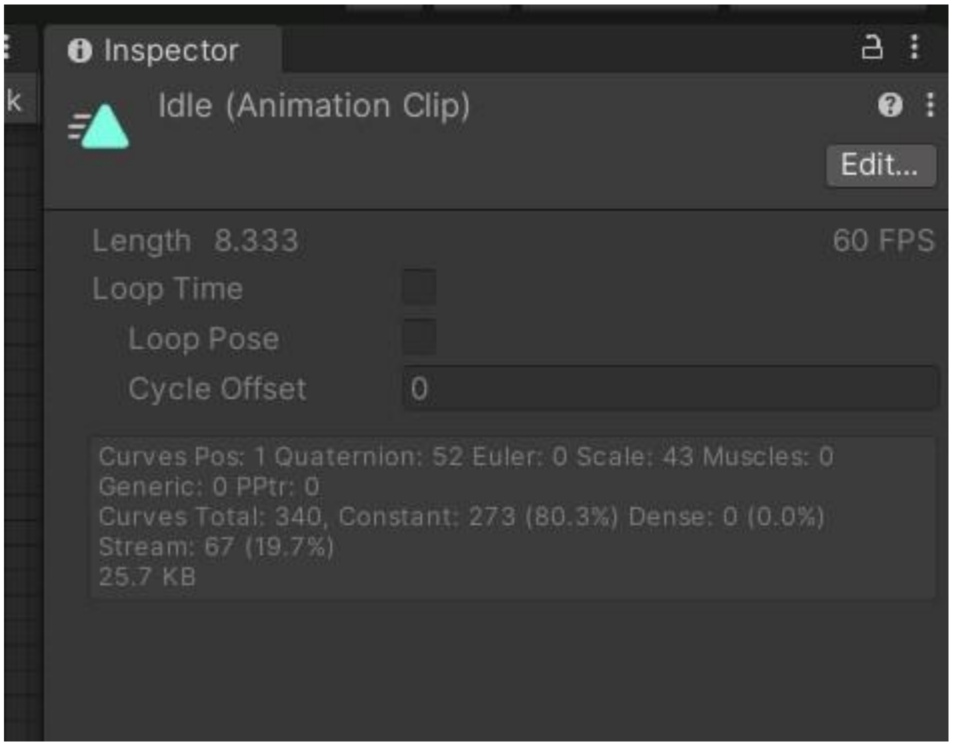
2. If the player presses forward, the current animation state should logically be switched to the **Walking State (now Current State: Walking)** as the model should move.
3. And if we stop, then it should move back to **Idle state.** We will work on this little later before that lets complete rest of the settings.

Its time to grab animations from our imported mixamo assets

Open up Mixamo folder in the assets ➞we will find our three models, let's select one of the animations out of the expanded model

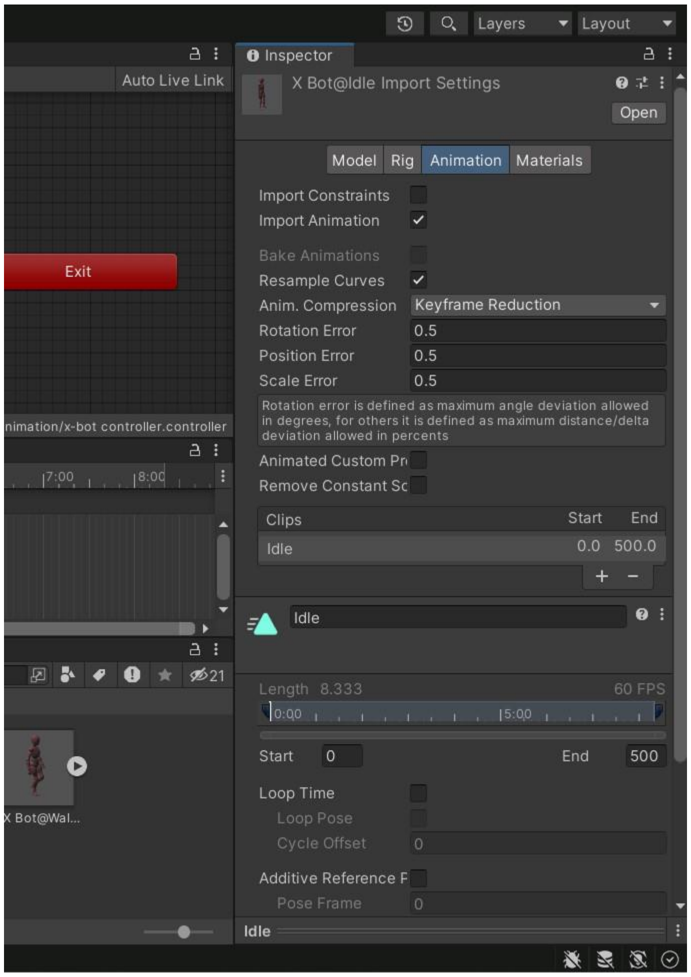


Take a look at the Inspector and you'll notice that all the options are grayed out, This is because the animation is child of the Unity parent model asset


Inspector
Idle (Animation Clip)
Edit...
Length 8.333 60 FPS
Loop Time
Loop Pose
Cycle Offset 0

```
Curves Pos: 1 Quaternion: 52 Euler: 0 Scale: 43 Muscles: 0
Generic: 0 PPtr: 0
Curves Total: 340, Constant: 273 (80.3%) Dense: 0 (0.0%)
Stream: 67 (19.7%)
25.7 KB
```

By clicking on Edit this is how the window will appear.


X Bot@Idle Import Settings

Model | Rig | Animation | Materials

Import Constraints ☐
Import Animation ✓
Bake Animations ☐
Resample Curves ✓
Anim. Compression: Keyframe Reduction
Rotation Error: 0.5
Position Error: 0.5
Scale Error: 0.5

Rotation error is defined as maximum angle deviation allowed in degrees, for others it is defined as maximum distance/delta deviation allowed in percents

Animated Custom Pr ☐
Remove Constant Sc ☐

Clips	Start	End
Idle	0.0	500.0

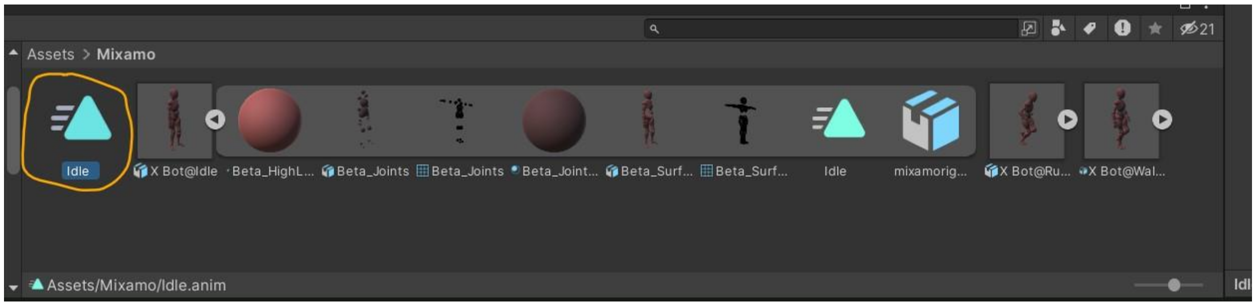
Idle

Length 8.333 60 FPS

Start 0 End 500

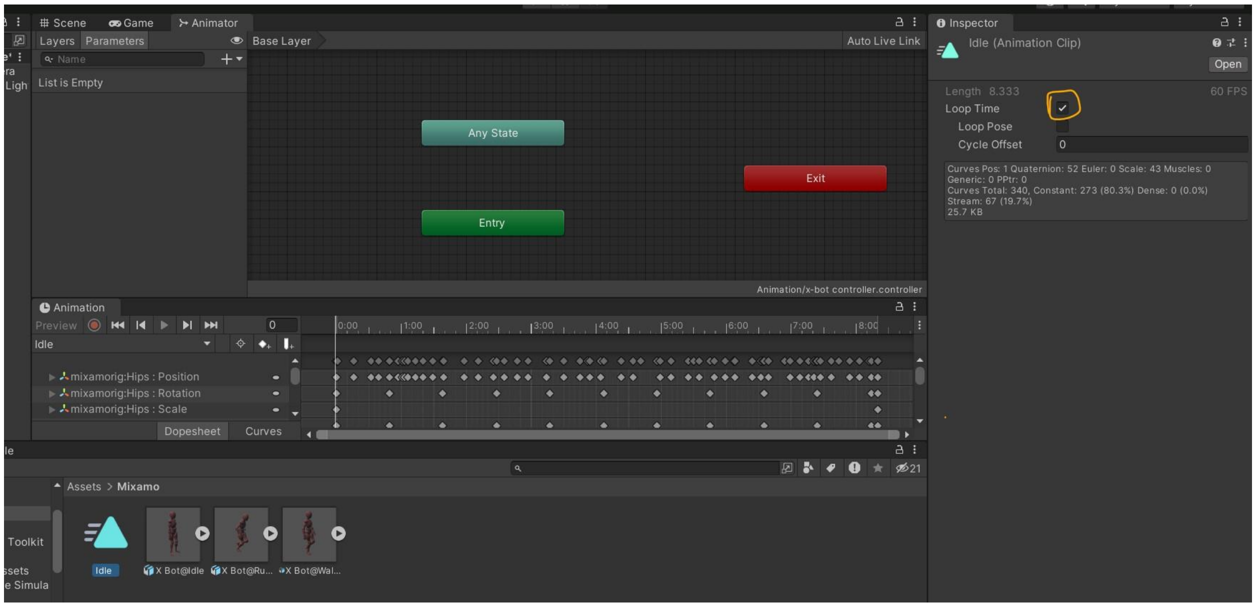
Loop Time ☐
Loop Pose ☐
Cycle Offset 0
Additive Reference F ☐
Pose Frame 0

Instead, we are going to separate the animation from the model. To do so we will duplicate the animation (This is done by selecting the animation and press command and Ctrl + D on the keyboard)



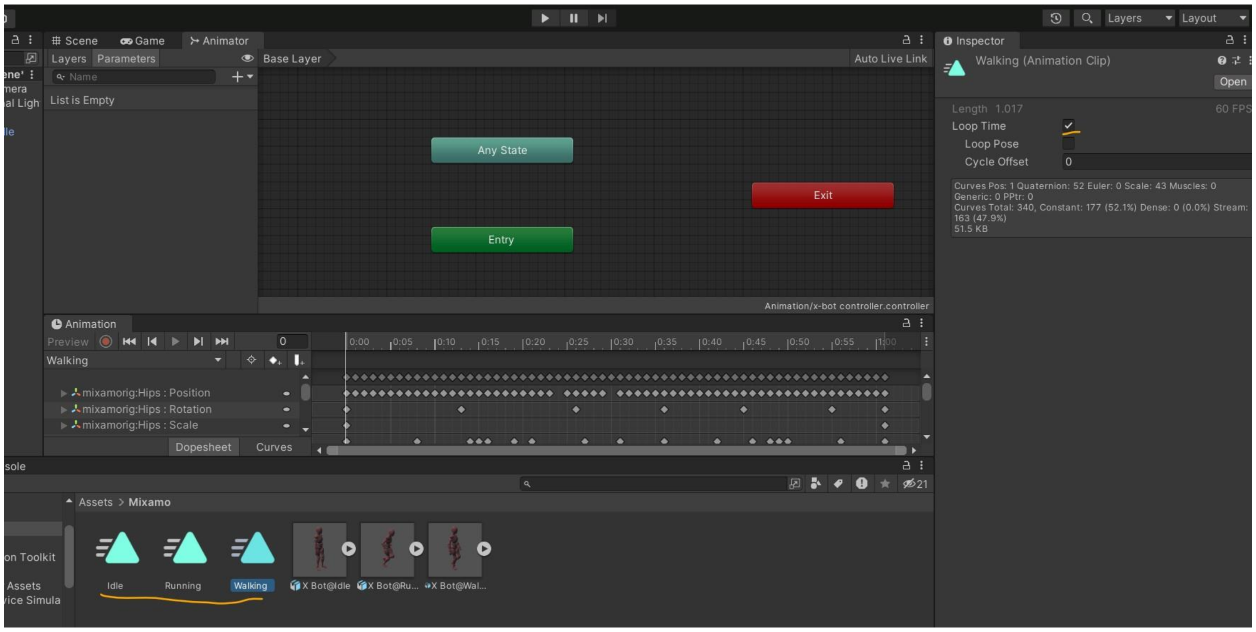
Now when we select our duplicated asset, we can modify its properties in the inspector window, ➞Let's duplicate all three, check the **loop** property from each animation and move them into animation folder

[Looping the animation will continuously repeat the animation if thats our character's current animation state]



Length 8.333 60 FPS
Loop Time ✓
Loop Pose
Cycle Offset 0

Curves Pos: 1 Quaternion: 52 Euler: 0 Scale: 43 Muscles: 0
Generic: 0 PPtr: 0
Curves Total: 340, Constant: 273 (80.3%) Dense: 0 (0.0%)
Stream: 67 (19.7%)
25.7 KB

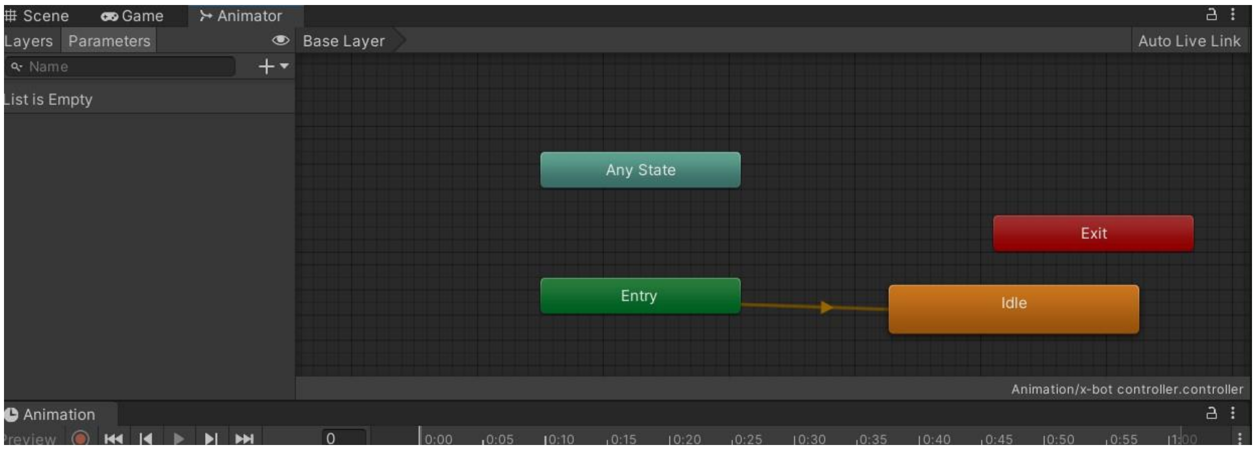


Length 1.017 60 FPS
Loop Time ✓
Loop Pose
Cycle Offset 0

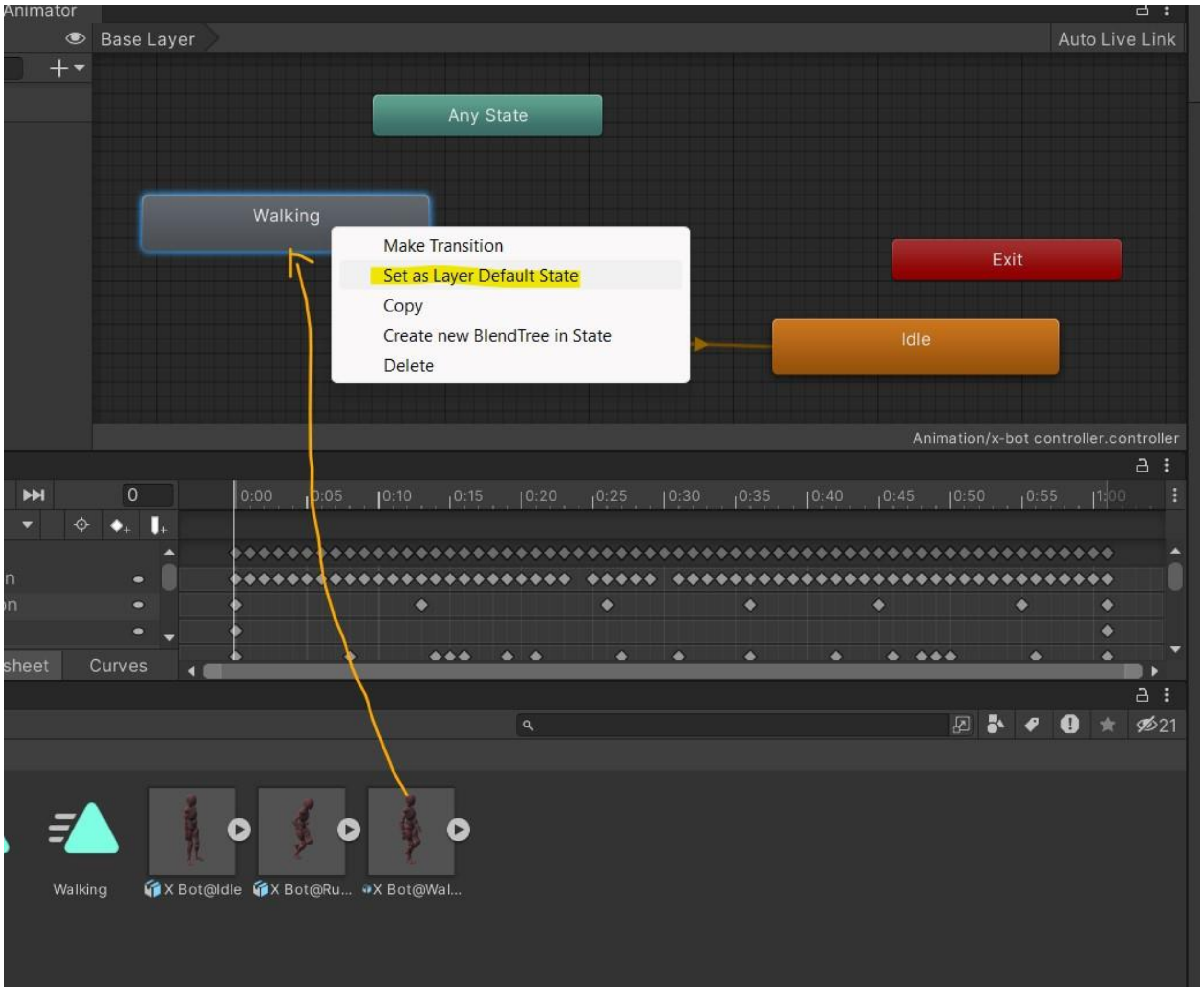
Curves Pos: 1 Quaternion: 52 Euler: 0 Scale: 43 Muscles: 0
Generic: 0 PPtr: 0
Curves Total: 340, Constant: 177 (52.1%) Dense: 0 (0.0%) Stream: 163 (47.9%)
51.5 KB

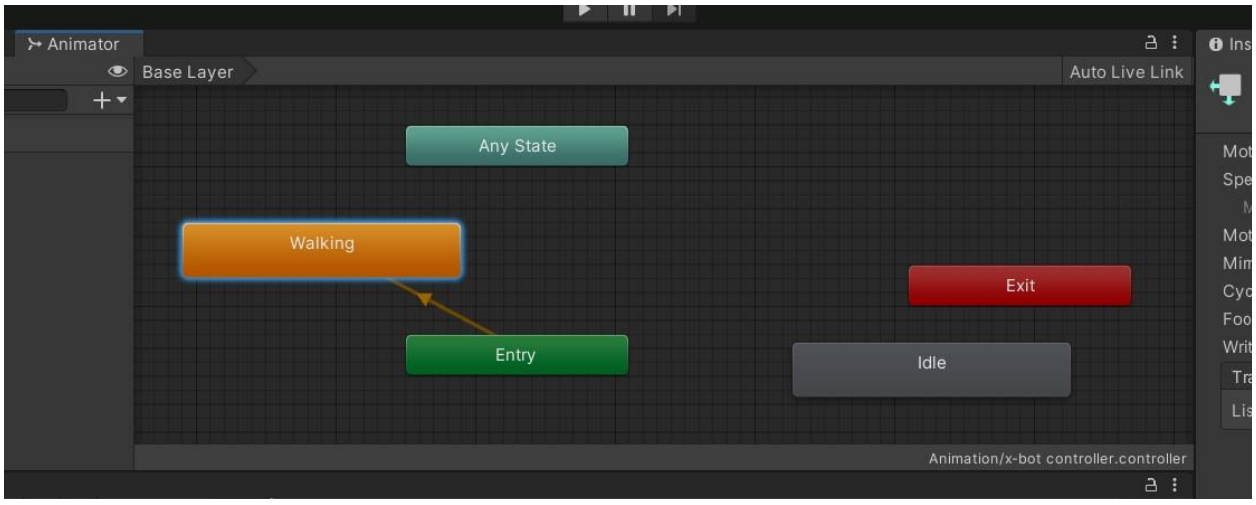
Lets get back to **xbot controller ➞**with our Idle animation selected, lets drag it into the grid layer and we'll immediately see the Idle animation state created!

You'll notice that the Entry is automatically connected to the Idle animation state.. This is because the Idle animation state was automatically set as the default animation state.

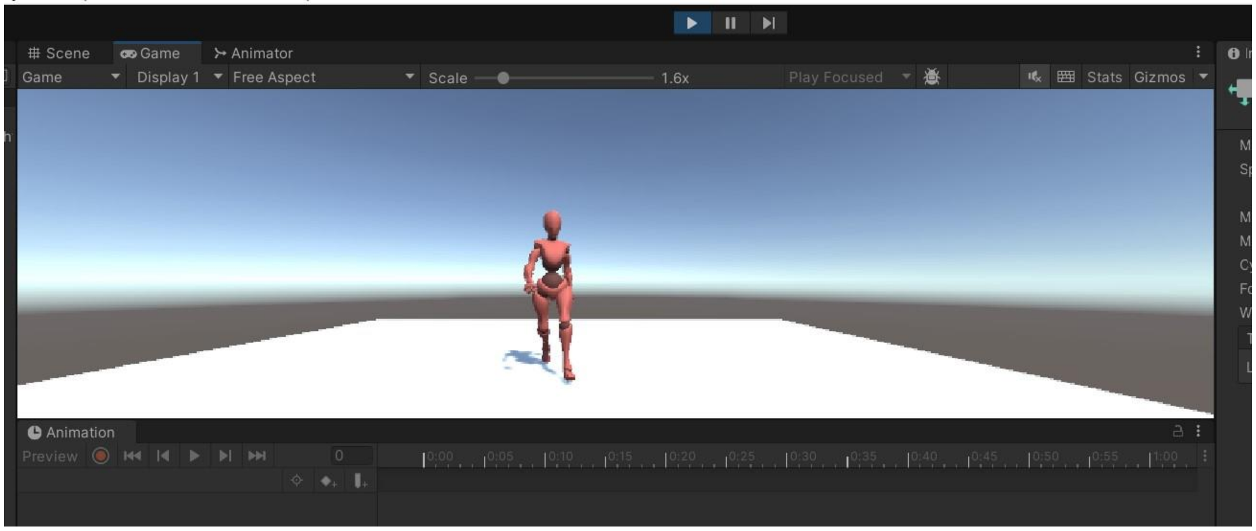


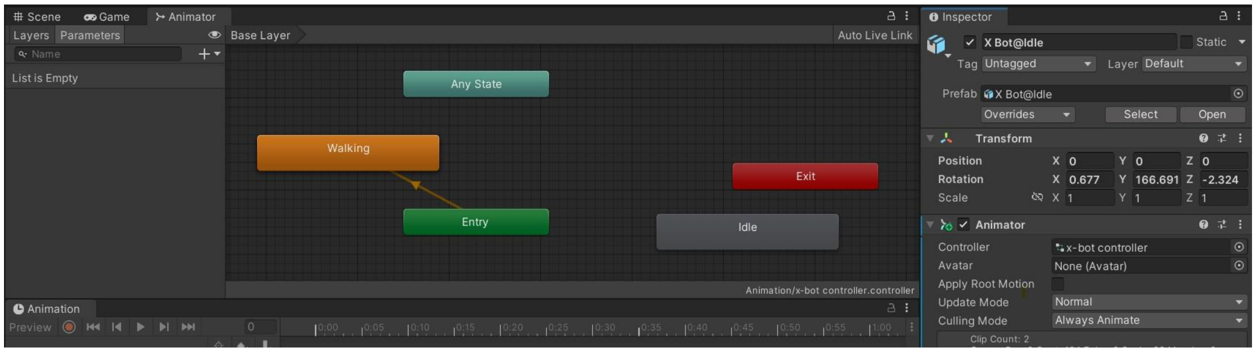
If we want to switch default states, we right-click on a different animation state and select “**Set as layer default state**” (this means that this animator controller will automatically use this animation when we enter play mode / game mode)





Now if we press play, we will see our character performing





X Bot@Idle Static
Tag Untagged Layer Default
Prefab X Bot@Idle
Overrides Select Open

Transform

Position	X	0	Y	0	Z	0
Rotation	X	0.677	Y	166.691	Z	-2.324
Scale	X	1	Y	1	Z	1

Animator

Controller	x-bot controller
Avatar	None (Avatar)
Apply Root Motion	
Update Mode	Normal
Culling Mode	Always Animate

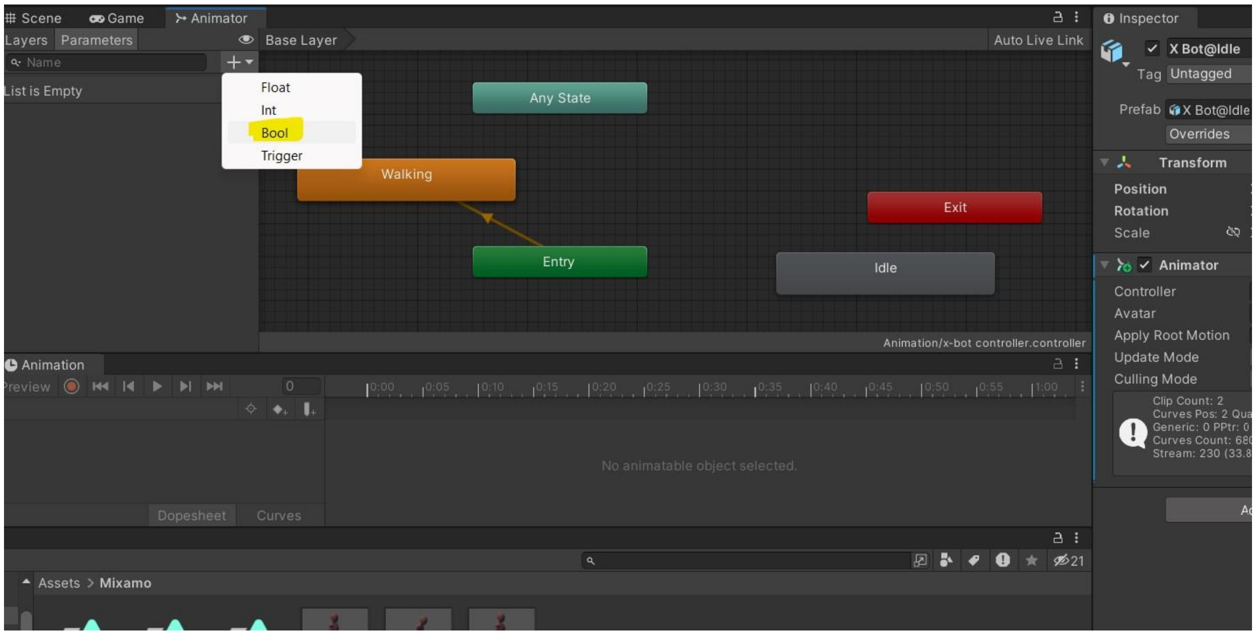
Clip Count: 2

Practical No. 4

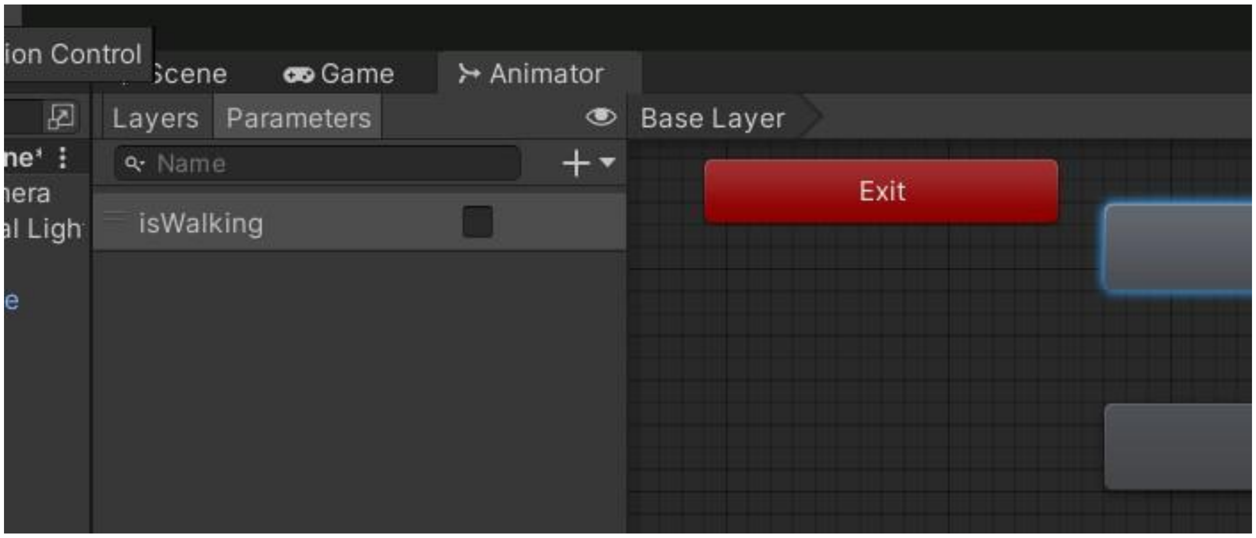
Aim: How to do Animation transition with Boolean in Unity

In the previous project

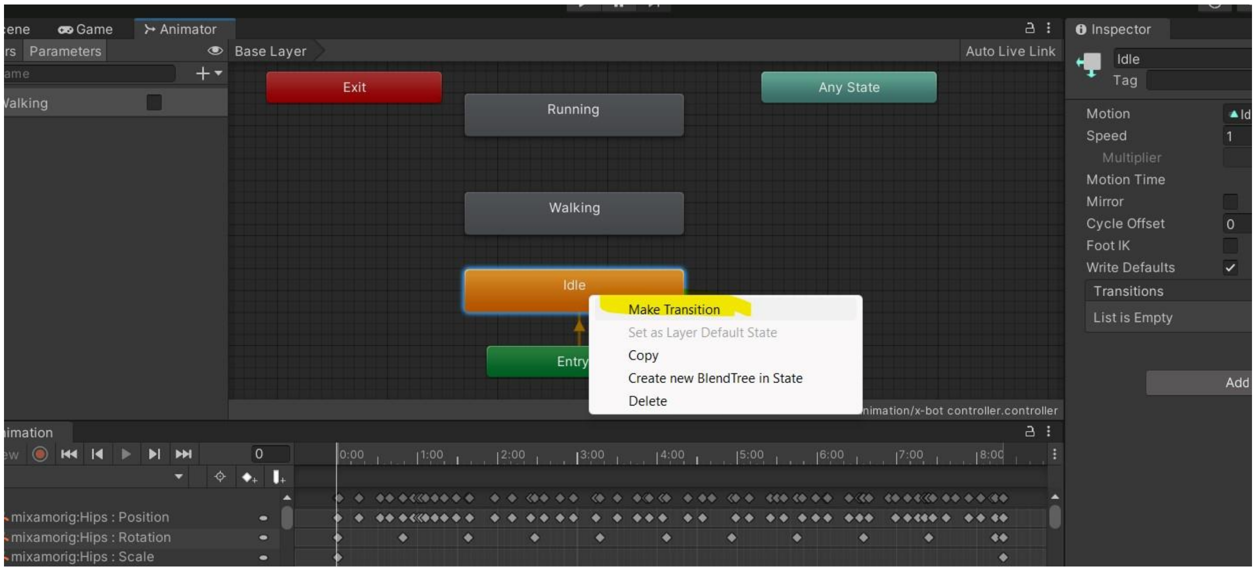
Now go to Animator →Parameters →we are working with boolean


Float
Int
Bool
Trigger
Any State
Walking
Exit
Entry
Idle
List is Empty
Animation/x-bot controller.controller
No animatable object selected.

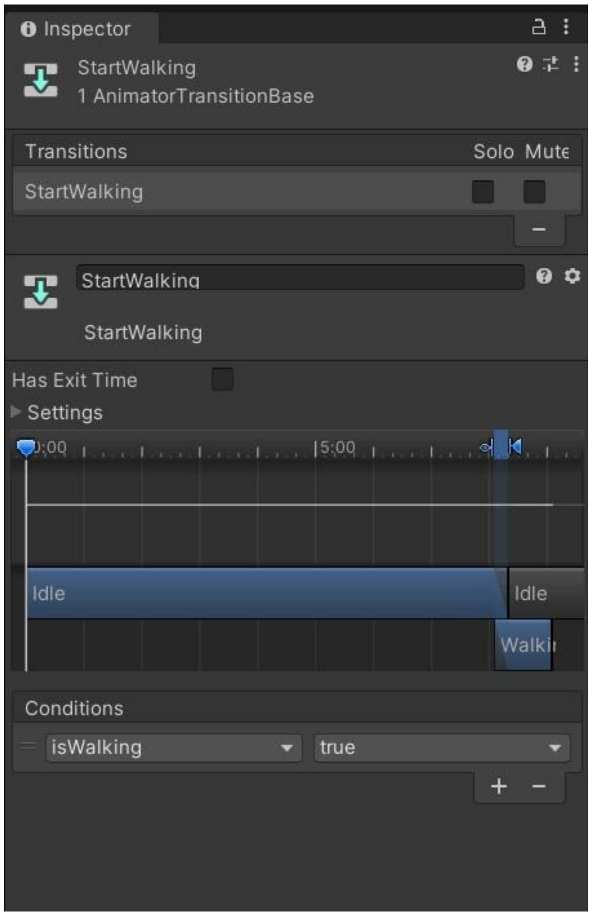
Click on ✚icon to add parameter ➞rename the parameter to isWalking



Now we need to go from simply Idle to Walking



Right click on Idle animation ➝and click on Make Transition ➝select transition to Walking animation ➝Now click on this transition go to inspector panel type StartWalking ➝uncheck Has Exit Time ➝Go to Settings ➝on Conditions section click on add ➕select parameter isWalking and make boolean value to true



Inspector

StartWalking
1 AnimatorTransitionBase

Transitions	Solo	Mute
StartWalking		

StartWalking

StartWalking

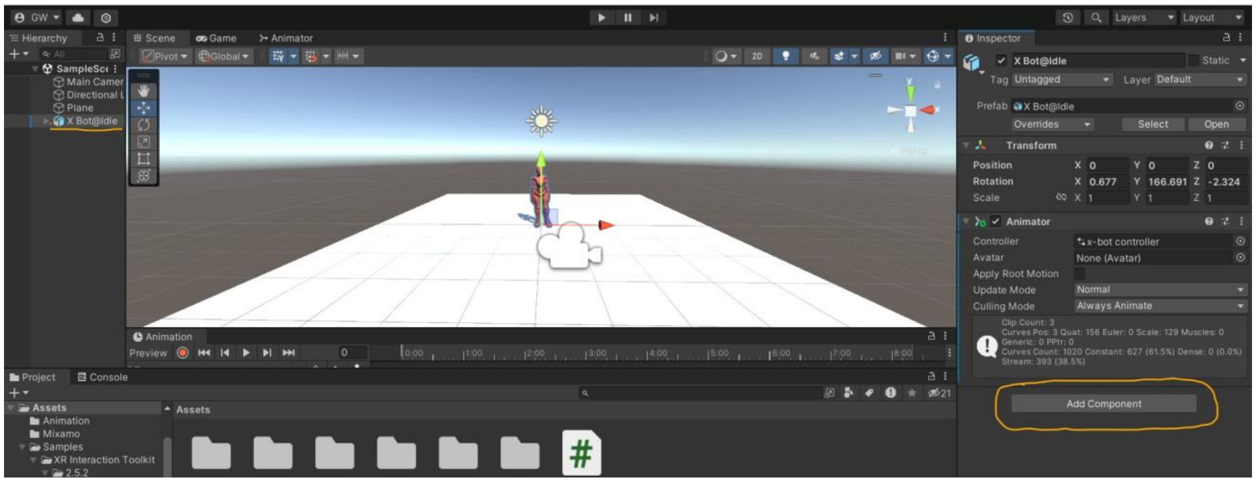
Has Exit Time: unchecked

Settings

Conditions

isWalking	true

Go to Hierarchy click on X Bot game object ➝click on Add Component



Inspector

X Bot@Idle — Static

Tag: Untagged — Layer: Default

Prefab: X Bot@Idle — Overrides — Select — Open

Transform

	X	Y	Z
Position	0	0	0
Rotation	0.677	166.691	-2.324
Scale	1	1	1

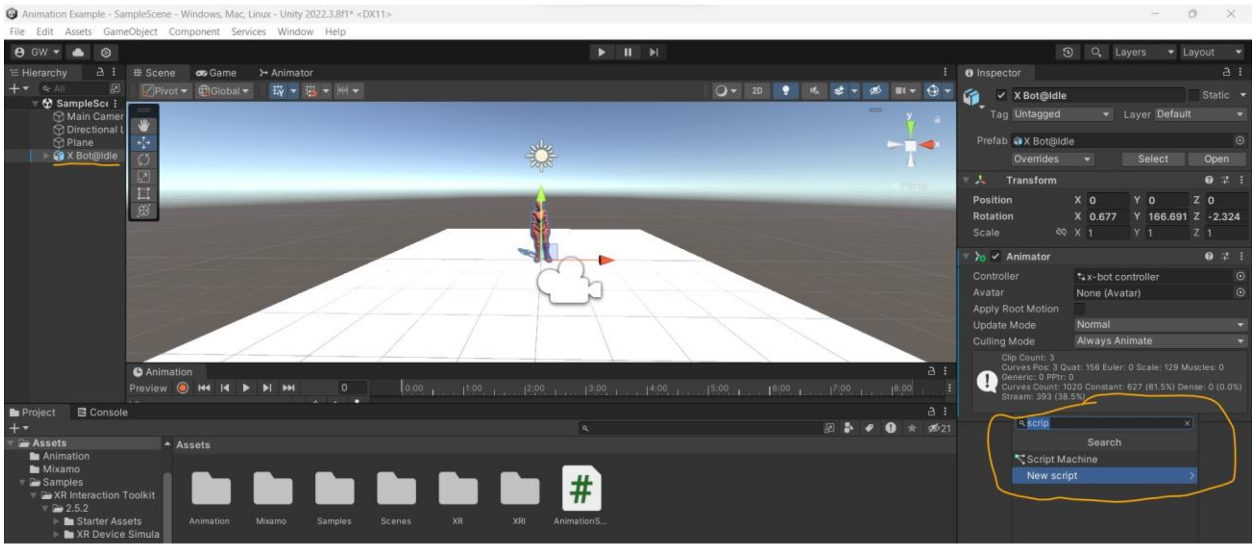
Animator

Controller: x-bot controller
Avatar: None (Avatar)
Apply Root Motion: unchecked
Update Mode: Normal
Culling Mode: Always Animate

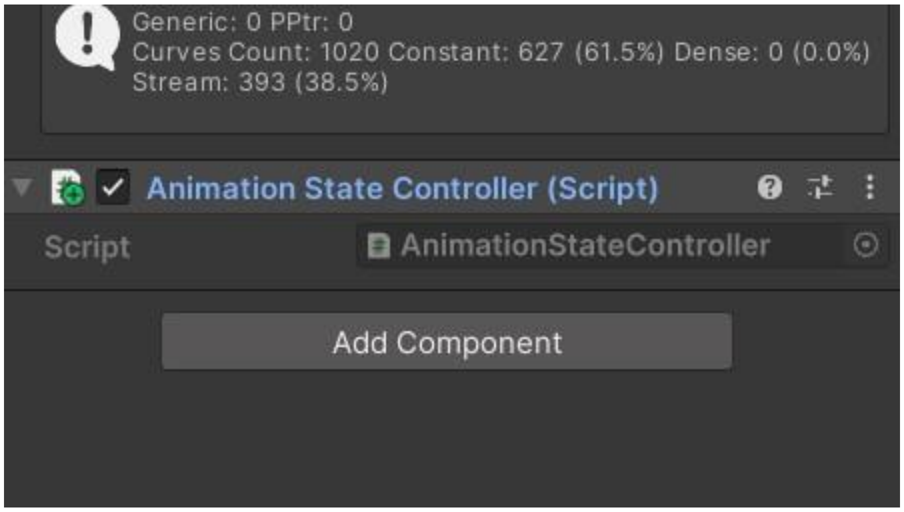
```
Clip Count: 3
Curves Pos: 3 Quat: 156 Euler: 0 Scale: 129 Muscles: 0
Generic: 0 PPtr: 0
Curves Count: 1020 Constant: 627 (61.5%) Dense: 0 (0.0%)
Stream: 393 (38.5%)
```

Add Component

Add script component



X Bot@Idle
Tag Untagged Layer Default
Prefab X Bot@Idle
Transform
Position X 0 Y 0 Z 0
Rotation X 0.677 Y 166.691 Z -2.324
Scale X 1 Y 1 Z 1
Animator
Controller x-bot controller
Avatar None (Avatar)
Apply Root Motion
Update Mode Normal
Culling Mode Always Animate
Clip Count: 3
Curves Pos: 3 Quat: 156 Euler: 0 Scale: 129 Muscles: 0
Generic: 0 PPtr: 0
Curves Count: 1020 Constant: 627 (61.5%) Dense: 0 (0.0%)
Stream: 393 (38.5%)
scrip



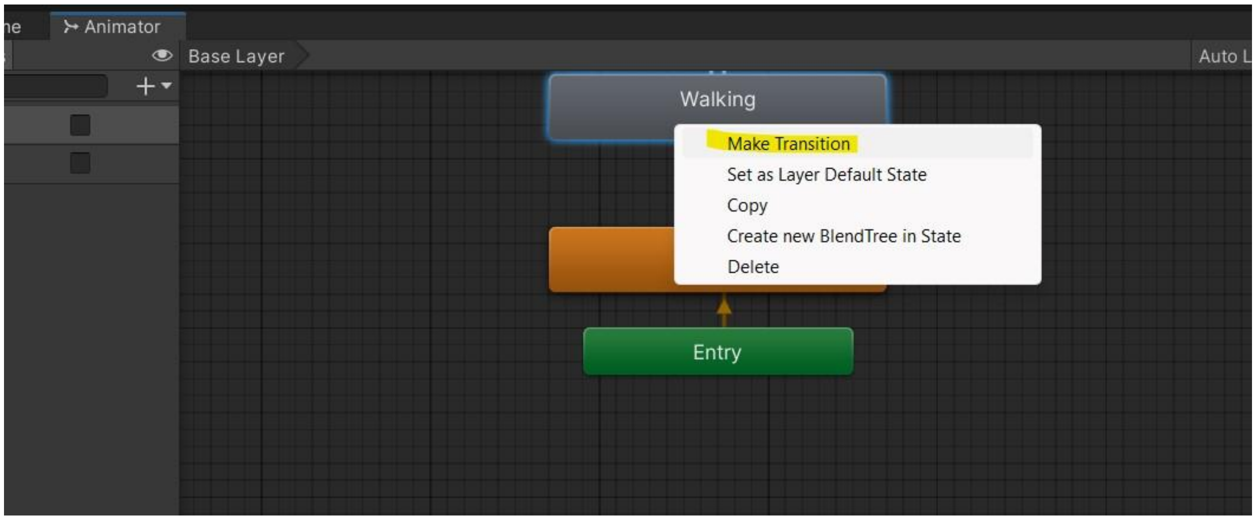
Generic: 0 PPtr: 0
Curves Count: 1020 Constant: 627 (61.5%) Dense: 0 (0.0%)
Stream: 393 (38.5%)

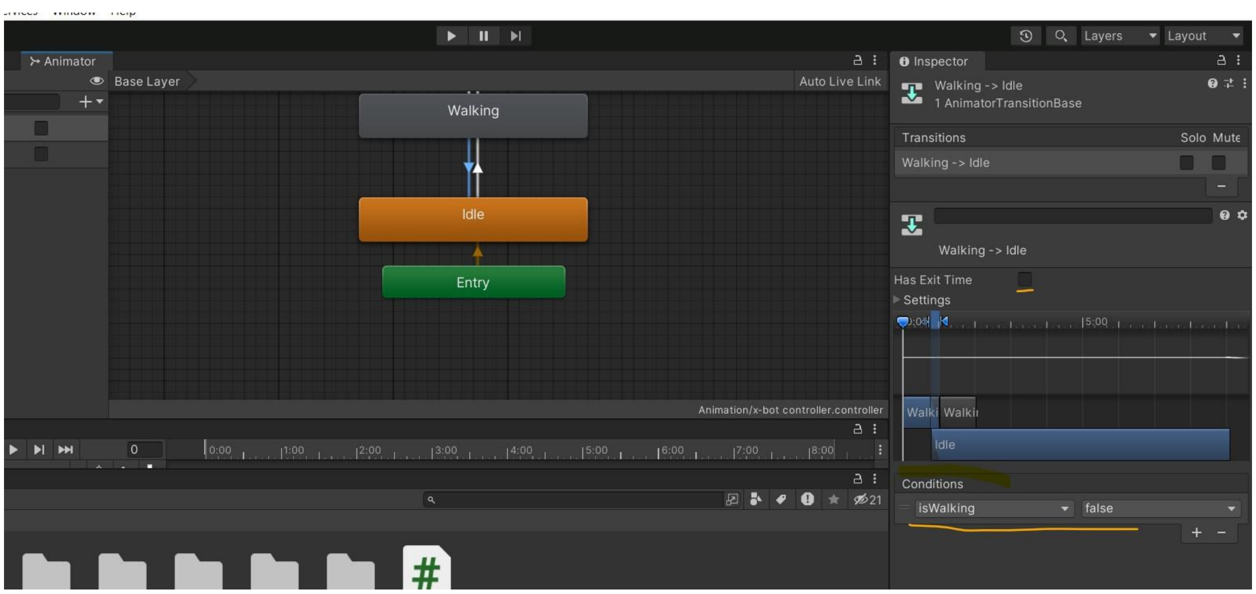
Animation State Controller (Script)
Script AnimationStateController

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

public class AnimationStateController : MonoBehaviour
{
    Animator     animator;
    int isWalkingHash;

    // Start is called before the first frame update void
    Start()
    {
        animator             =            GetComponent<Animator>();
        isWalkingHash = Animator.StringToHash("isWalking");

    }

    // Update is called once per frame void
    Update()
    {
```

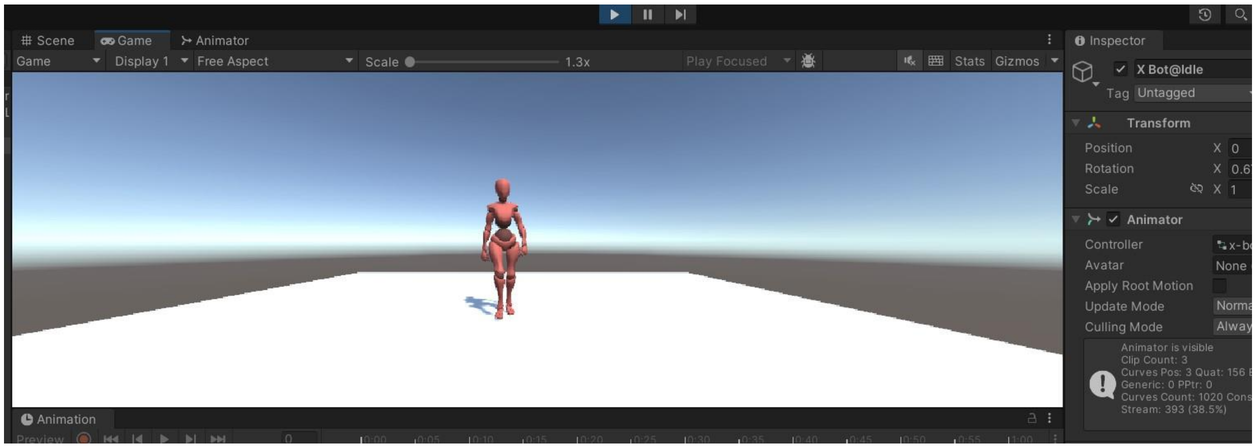
```
bool   isWalking   =   animator.GetBool(isWalkingHash);
bool forwardPressed = Input.GetKey("w");

//if   player   presses   w   key   if
(!isWalking && forwardPressed)
{
    //then set the isWalking boolean to be true animator.SetBool(isWalkingHash,
    true);
}

//if player is not pressing w key if
(isWalking && !forwardPressed)
{
    //then set the isWalking boolean to be false animator.SetBool(isWalkingHash,
    false);
}
```

Before running on game mode go to Animator panel and click on Walking animation ➞Click on Make Transition and select back to idle ➞select the create transition from walking to idle ➞go to inspector panel ➞uncheck Has Exit Time ➞In condition section click on ✚list and add parameter isWalking and value as false



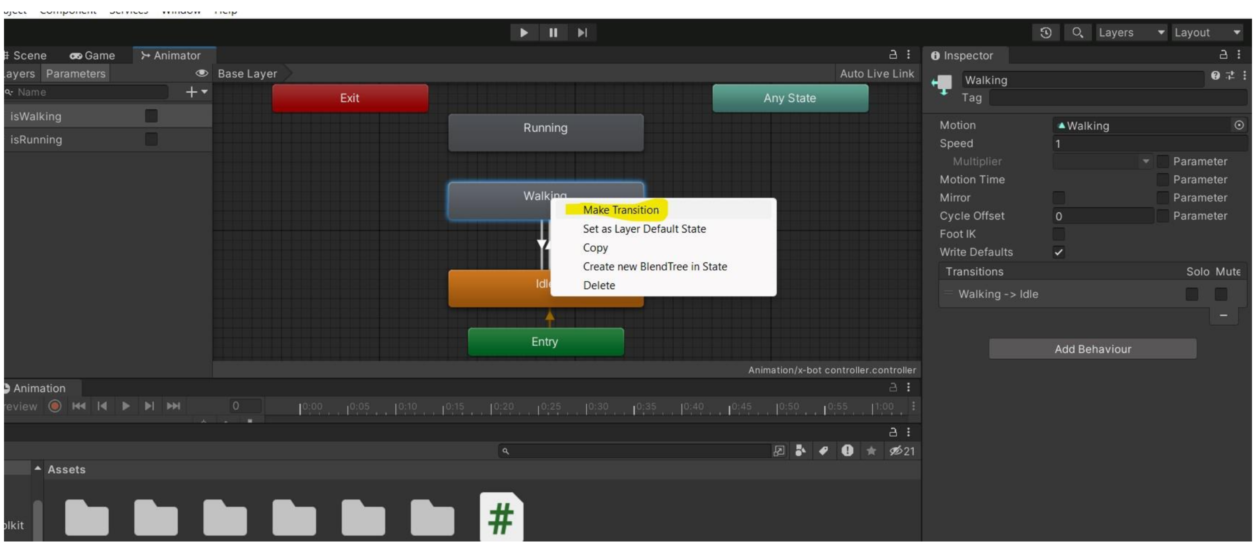


On key w pressed the character starts to walk from idle state and once removed from w key goes back to the idle state



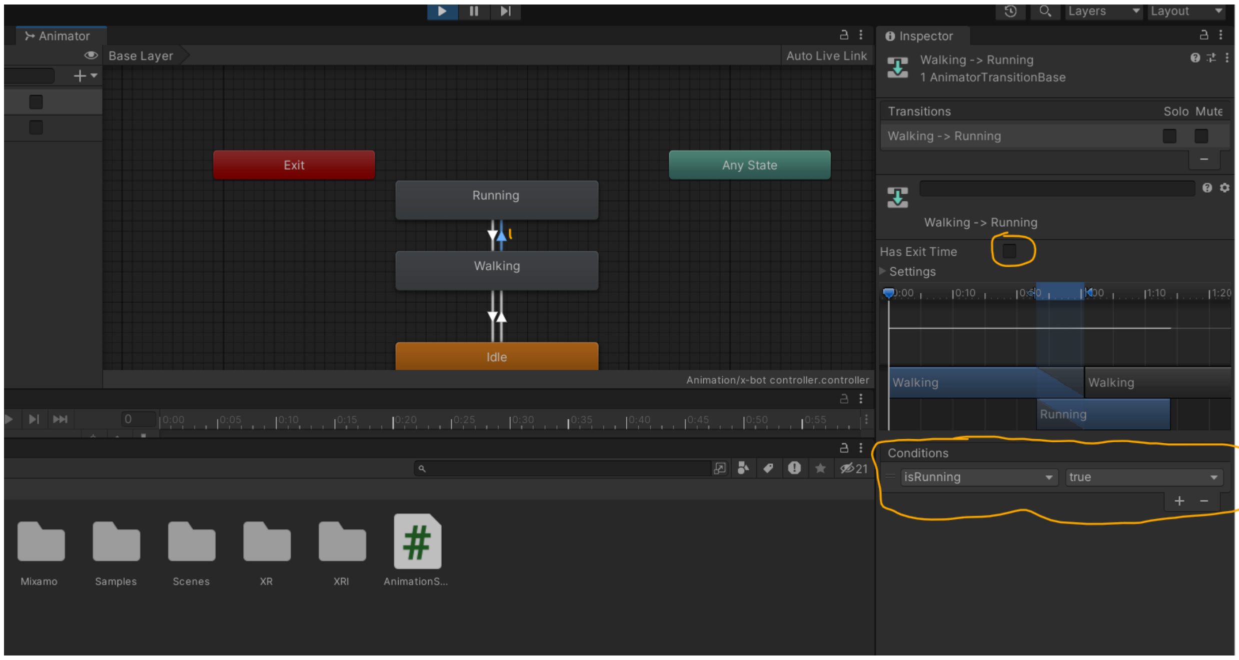
We will do the same thing for running animation as well and add code for the same as well and check the output

Right click on Walking animation →select Make Transition →go to Inspector Panel and



Inspector
Walking
Tag
Motion: Walking
Speed: 1
Multiplier: Parameter
Motion Time: Parameter
Mirror: Parameter
Cycle Offset: 0 Parameter
Foot IK
Write Defaults: ✓

Transitions	Solo	Mute
Walking -> Idle		



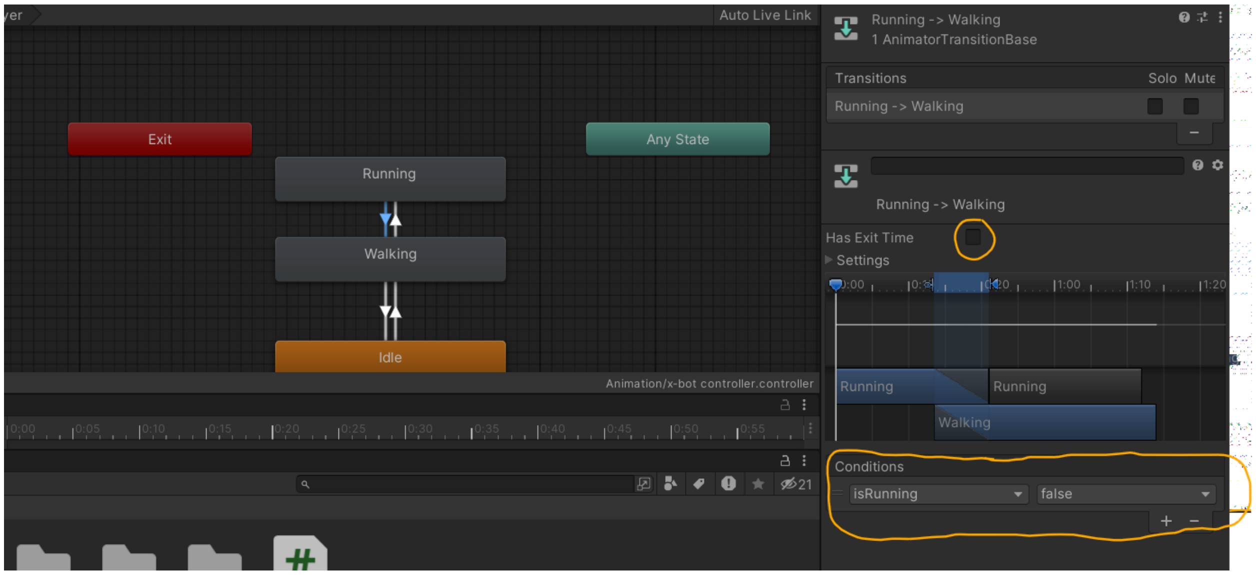
Inspector
Walking -> Running
1 AnimatorTransitionBase

Transitions	Solo	Mute
Walking -> Running		

Walking -> Running
Has Exit Time: ☐
Settings

Conditions

isRunning	true



Running -> Walking
1 AnimatorTransitionBase

Transitions	Solo	Mute
Running -> Walking		

Running -> Walking

Has Exit Time

Settings

Conditions

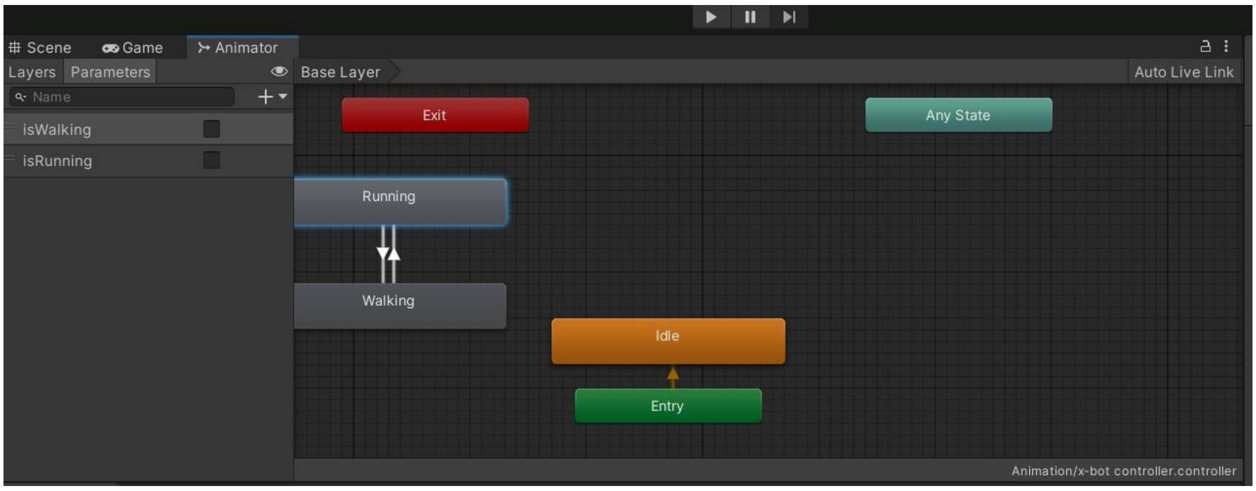
isRunning false

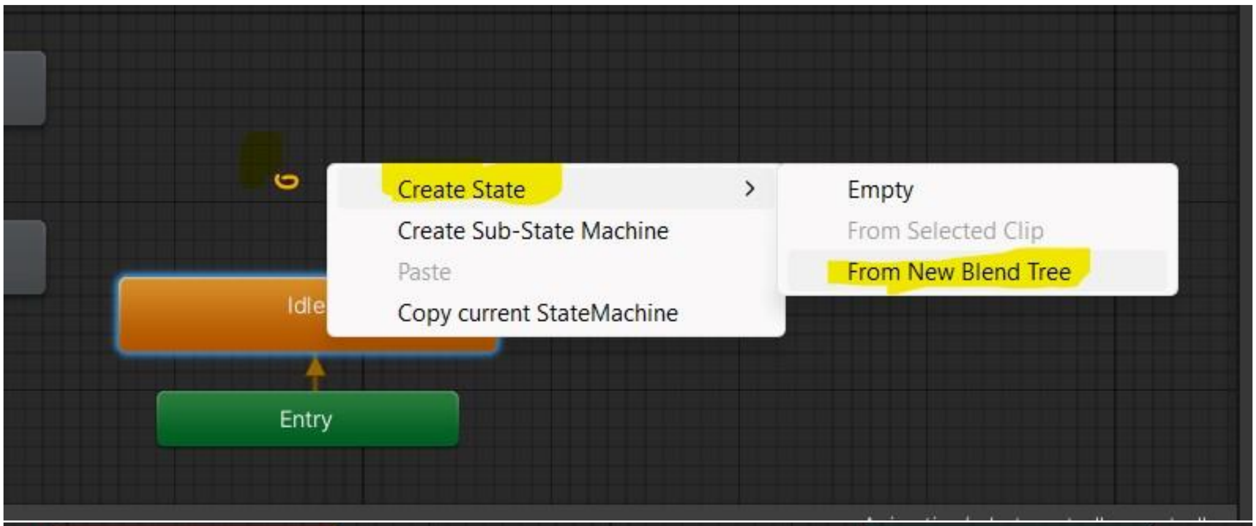
Addition in code for running

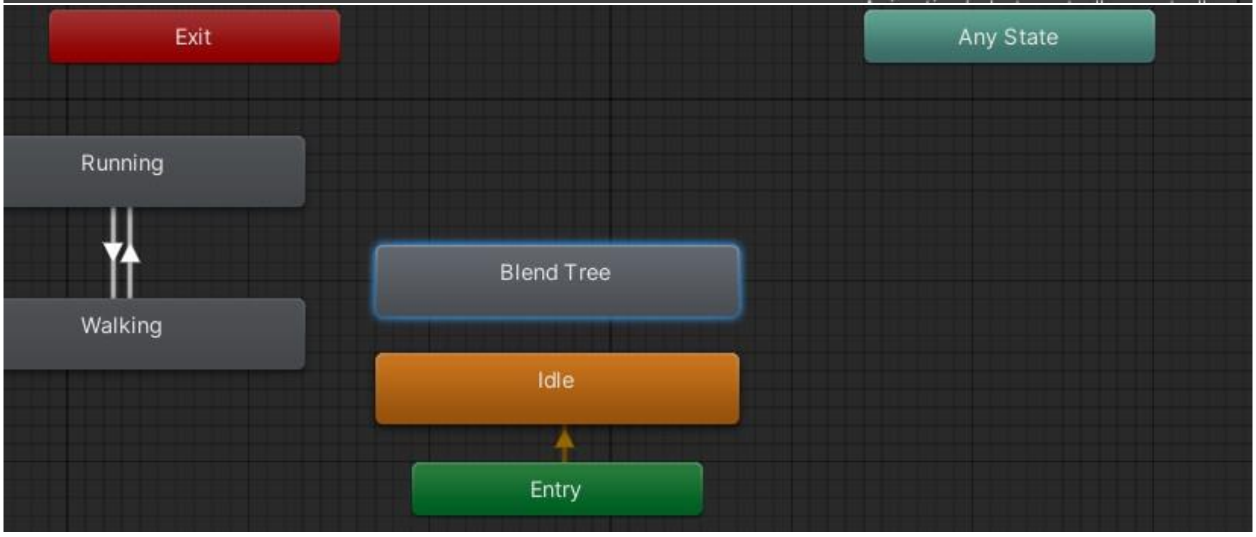
```
animation using
System.Collections; using
System.Collections.Generic; using
UnityEngine;

public class AnimationStateController : MonoBehaviour
{
    Animator
    animator; int
    isWalkingHash;
    int
    isRunningHash;

    // Start is called before the first frame update
    void Start()
    {
        animator = GetComponent<Animator>();
        isWalkingHash =
        Animator.StringToHash("isWalking");
        isRunningHash =
        Animator.StringToHash("isRunning");
    }

    // Update is called once per frame
    void Update()
    {
        bool isrunning =
        animator.GetBool(isRunningHash); bool
        isWalking = animator.GetBool(isWalkingHash);
        bool forwardPressed = Input.GetKey("w"); bool
        runPressed = Input.GetKey("left shift");
        //if   player   presses   w   key   if
        (!isWalking                     &&
        forwardPressed)
        {
            //then      set      the      isWalking      boolean      to      be      true
            animator.SetBool(isWalkingHash, true);
        }

        //if player is not pressing w key
        if          (isWalking          &&
        !forwardPressed)
        {
            //then set the isWalking boolean to be false
            animator.SetBool(isWalkingHash, false);
        }

        //if player is walking and not running and presses left
        shift if (!isrunning && (forwardPressed && runPressed))
        {
            //the set the isRunning boolean to be true
            animator.SetBool(isRunningHash, true);
        }

        //if player stops running and stops running or stops
        walking if (isrunning && (!forwardPressed ||
        !runPressed))
        {
            //the set the isRunning boolean to be false
            animator.SetBool(isRunningHash, false);
        }
    }
}
```

Practical No. 05

Aim: How to do Animation transition with Blend Trees in Unity





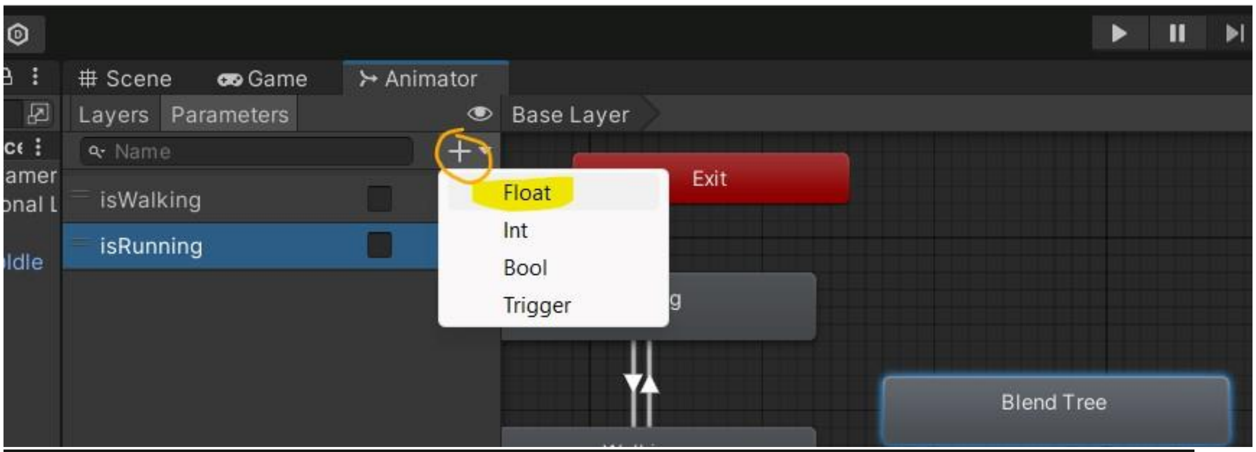


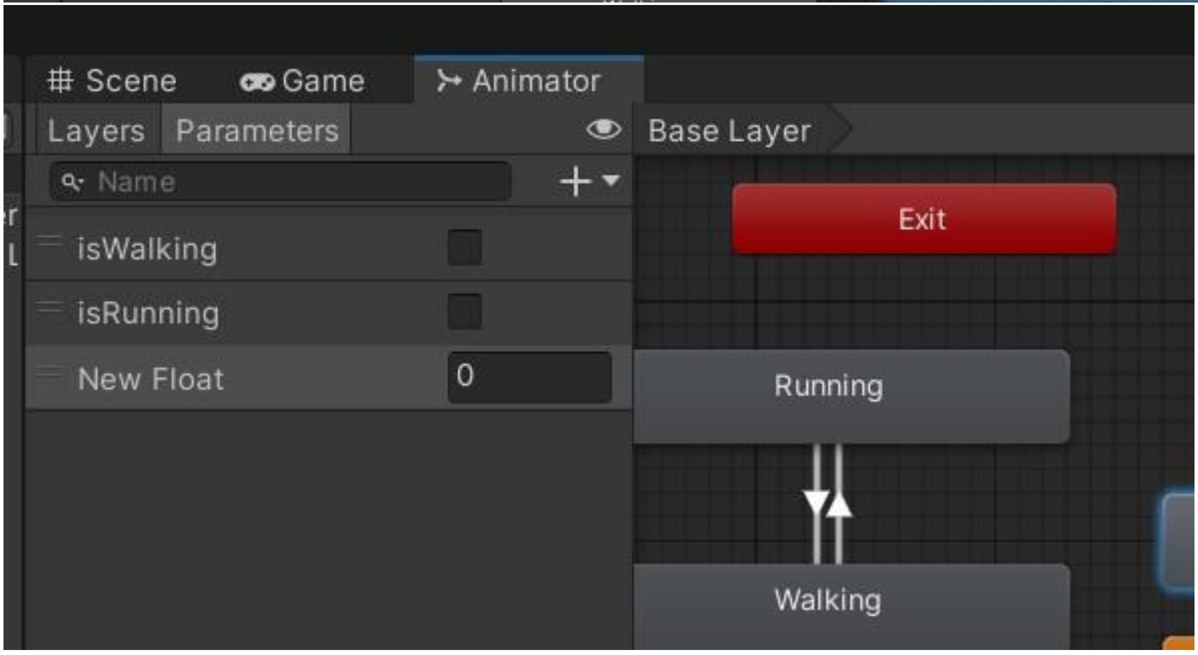
What is a blend tree?

A blend tree is used to dynamically "blend" multiple animations together

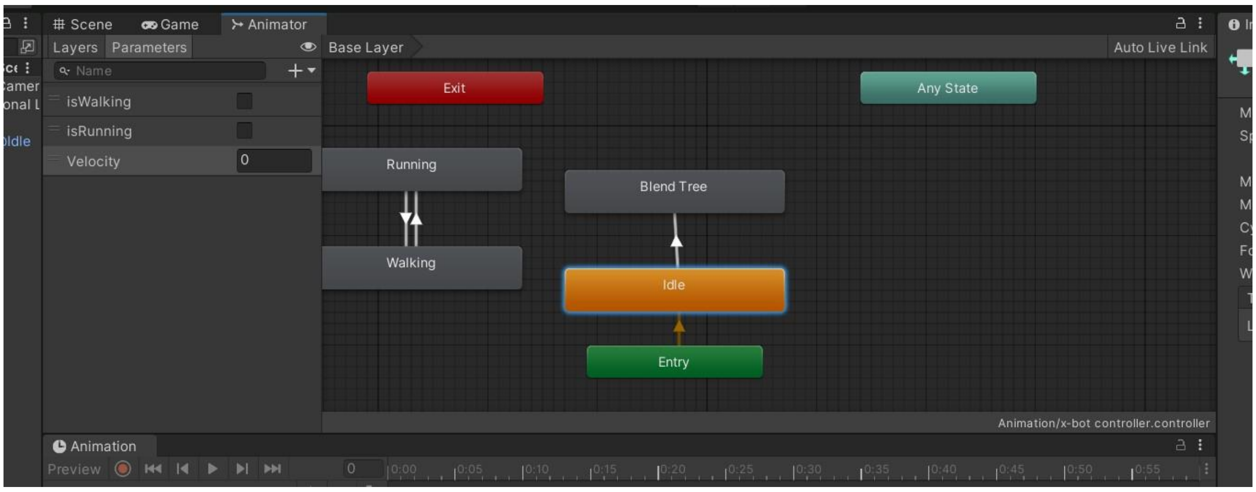
We will work with floats, We will create new floats parameter

In Parameters tab of Animator Window, click on the ➕plus sign and select Float

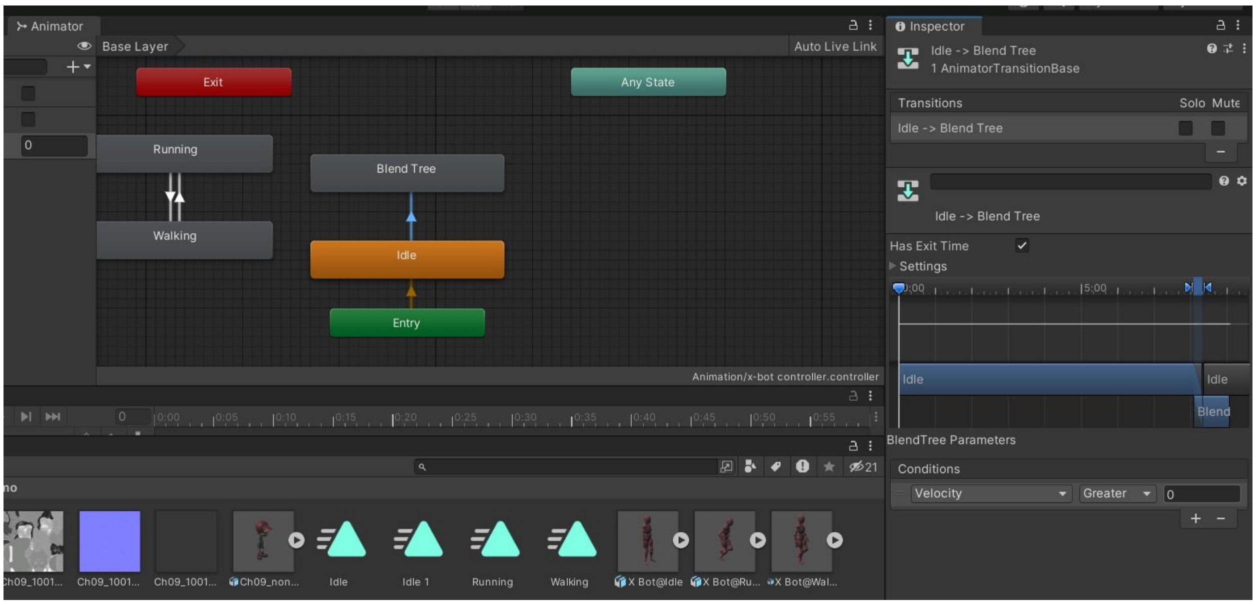




Rename it to velocity and make transition from Idle state to Blend trees

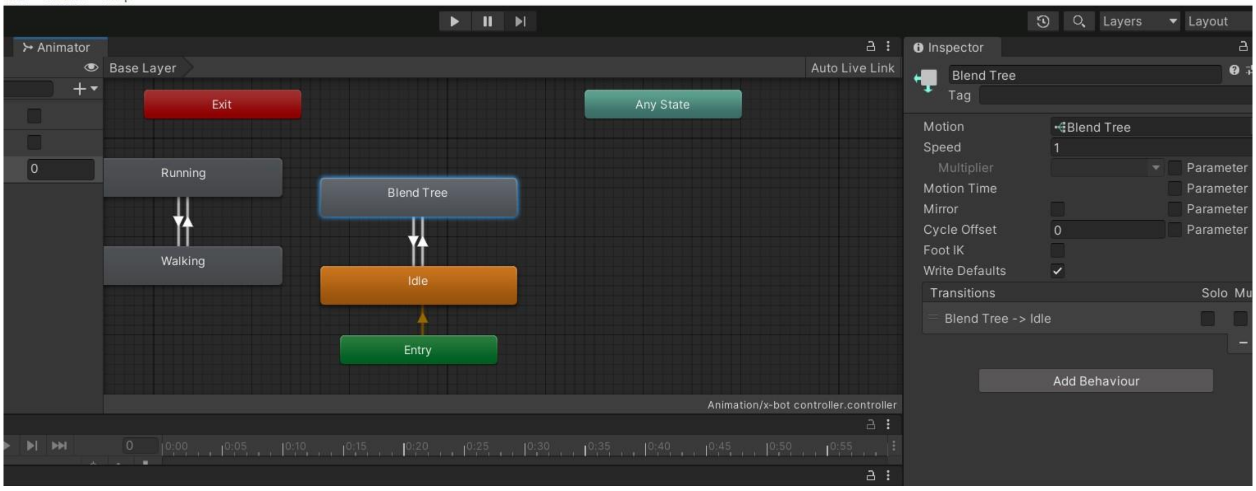


Select transition and on Inspector panel, under Blend Trees parameter add condition and select parameter velocity

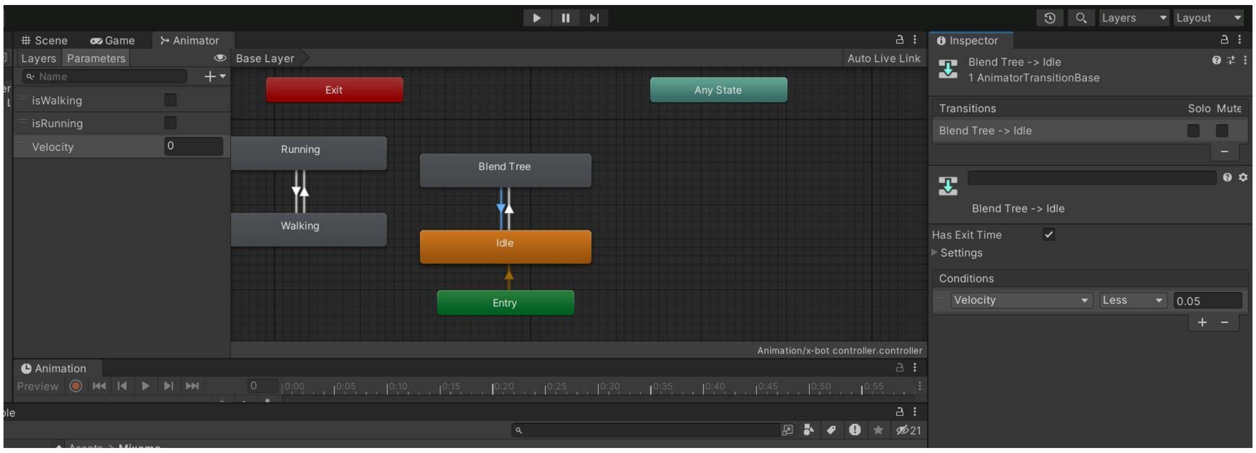


Has Exit Time ✓
Settings
Conditions
Velocity Greater 0

Make transition from Blend Tree to idle



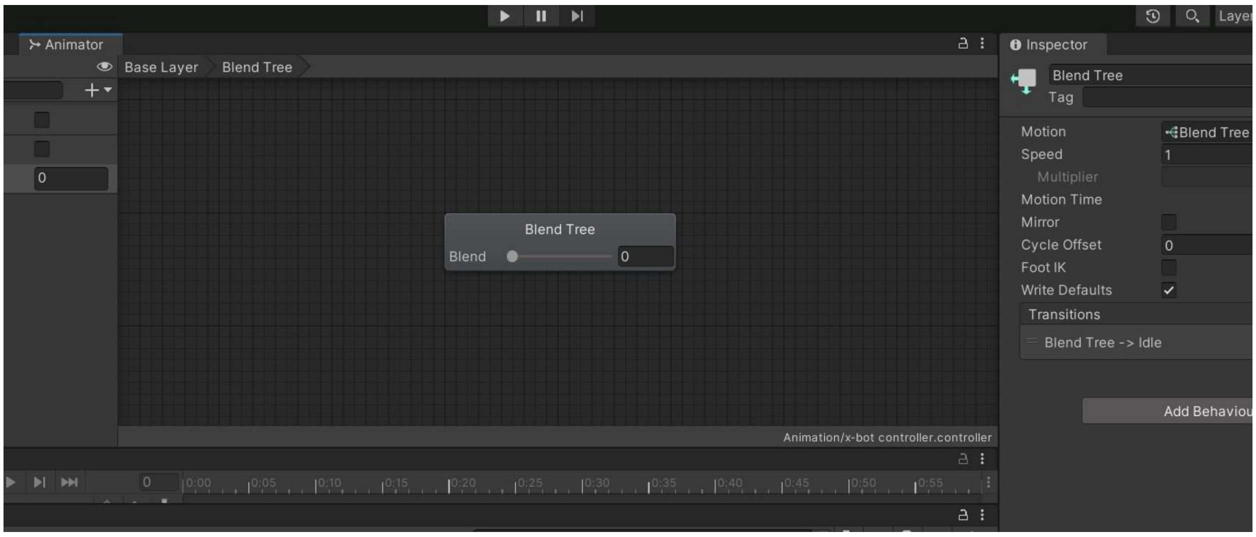
Blend Tree
Tag
Motion Blend Tree
Speed 1
Multiplier Parameter
Motion Time Parameter
Mirror Parameter
Cycle Offset 0 Parameter
Foot IK
Write Defaults ✓
Transitions Solo Mu
Blend Tree -> Idle



isWalking
isRunning
Velocity 0

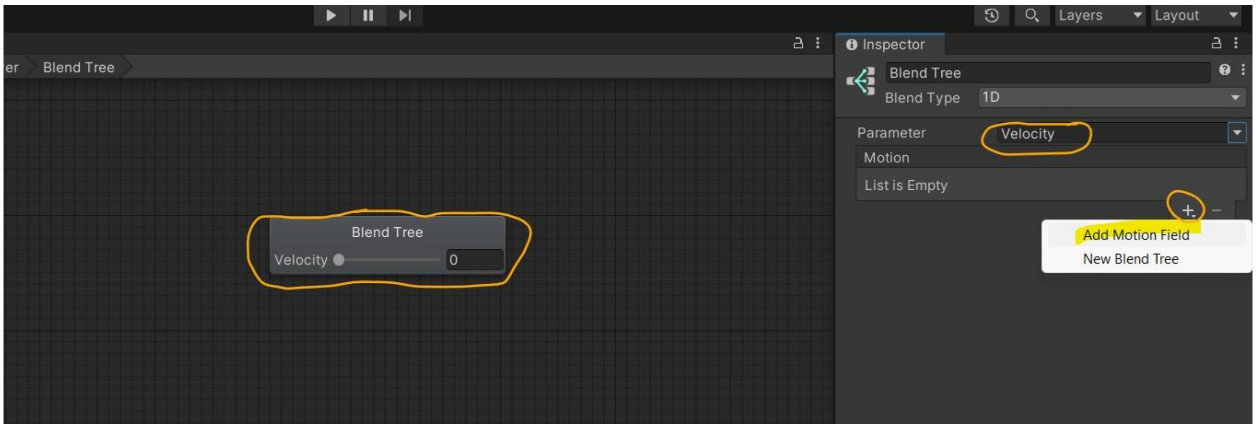
Has Exit Time ✓
Settings
Conditions
Velocity Less 0.05

Then double click on Blend tree node



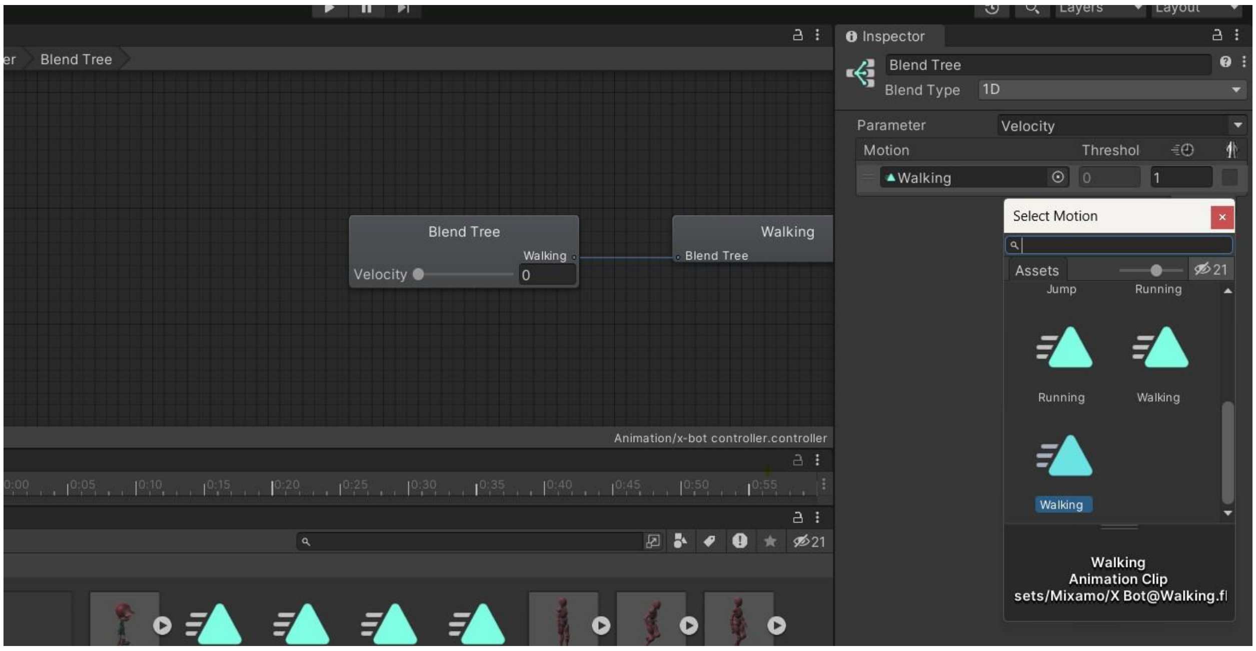
Blend Tree
Tag
Motion: Blend Tree
Speed: 1
Multiplier
Motion Time
Mirror
Cycle Offset: 0
Foot IK
Write Defaults: ✓
Transitions
Blend Tree -> Idle

Select blend tree and check on inspector panel to do following



Blend Tree
Blend Type: 1D
Parameter: Velocity
Motion
List is Empty

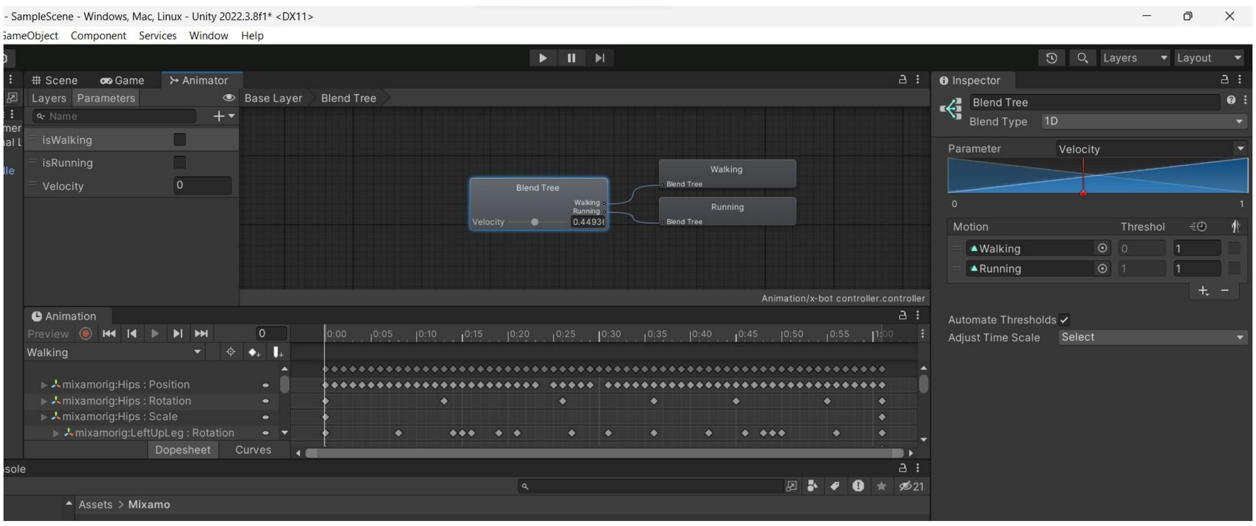
Add walking animation in the field and you will see blend tree named walking already connected with 1st blend tree



Blend Tree
Blend Type: 1D
Parameter: Velocity
Motion | Threshol
Walking | 0 | 1

Walking
Animation Clip
sets/Mixamo/X Bot@Walking.fl

Similarly add run animation



isWalking	☐
isRunning	☐
Velocity	0

Blend Tree
Blend Type 1D
Parameter Velocity

Motion	Threshol	
Walking	0	1
Running	1	1

Automate Thresholds ✓
Adjust Time Scale Select

Now let’s work in our code:

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

public class AnimationStateController : MonoBehaviour
{
    Animator animator; float
    velocity = 0.0f; public float
    accelaration = 0.1f; public
    float deccelaration = 0.5f; int
    velocityHash;

    // Start is called before the first frame update
    void Start()
    {
        //set reference for animator animator
        = GetComponent<Animator>();

        //increase performance velocityHash =
        Animator.StringToHash("Velocity");
    }

    // Update is called once per frame
    void Update()
    {
        //get key input from players bool
        forwardPressed = Input.GetKey("w");
        bool runPressed = Input.GetKey("left
        shift"); if (forwardPressed && velocity
        < 1.0f)
        {
            velocity += Time.deltaTime * accelaration;
        }
        if (!forwardPressed && velocity > 0.0f)
        {
            velocity -= Time.deltaTime * deccelaration;
        }
        if(!forwardPressed && velocity < 0.0f)
        {
            velocity = 0.0f;
        }

        animator.SetFloat(velocityHash, velocity);
    }
}
```