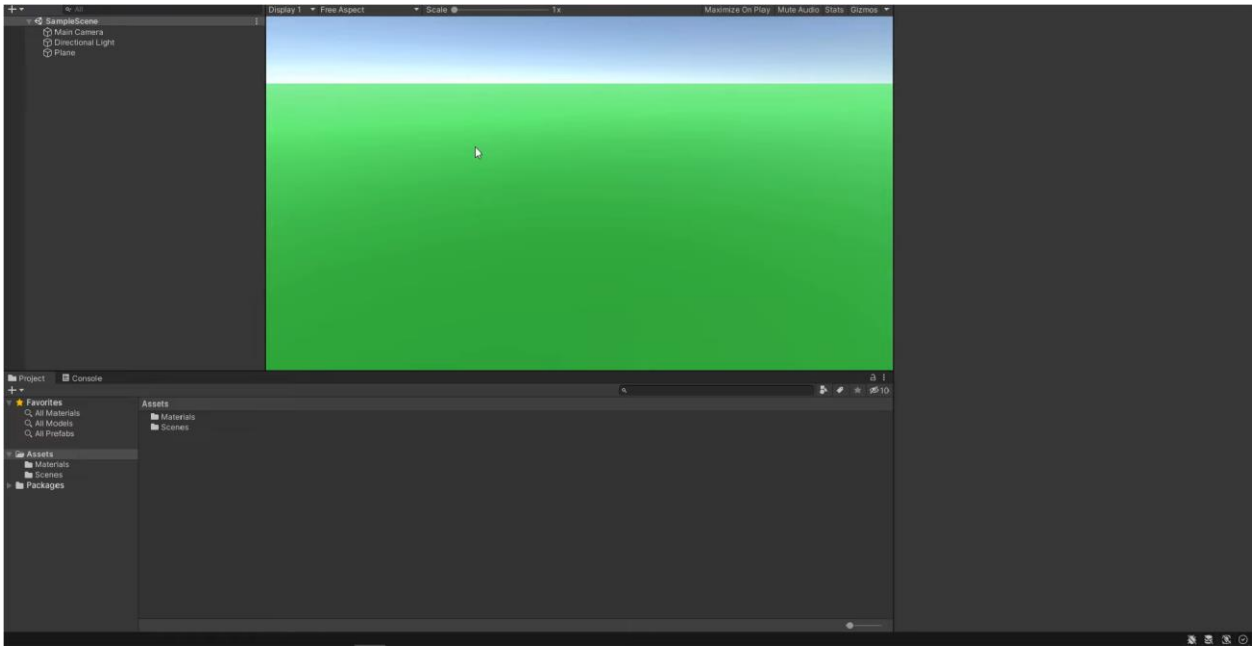
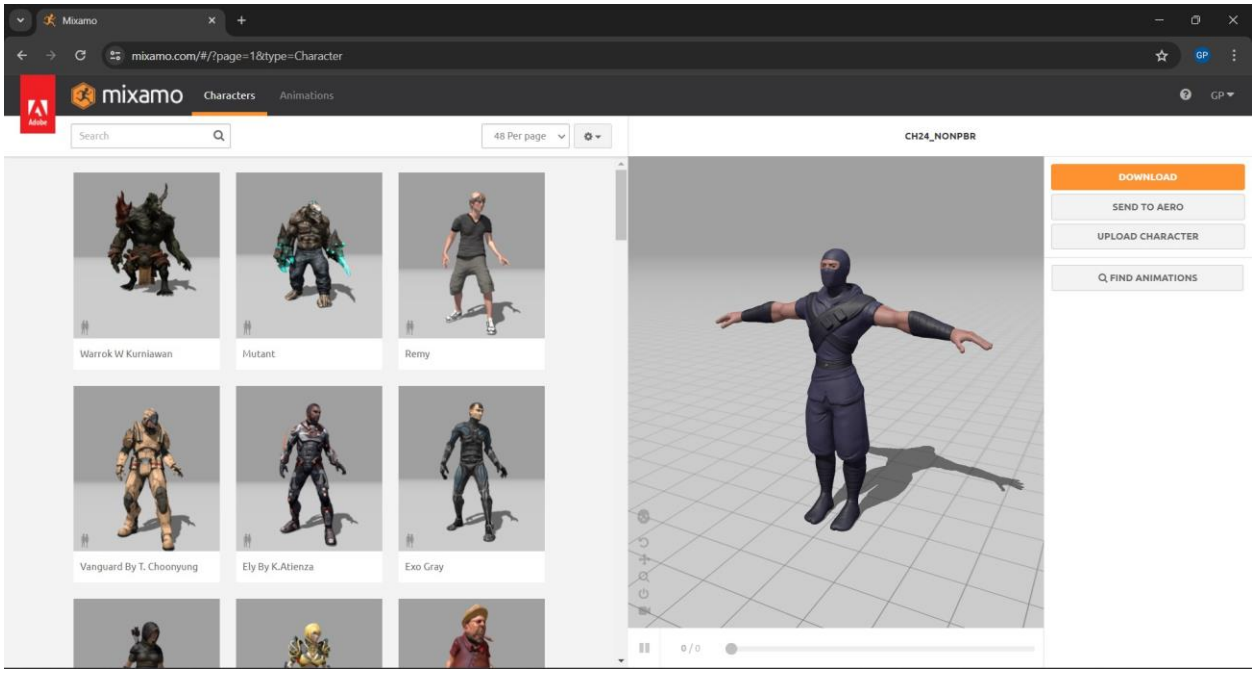
Practical No. 07

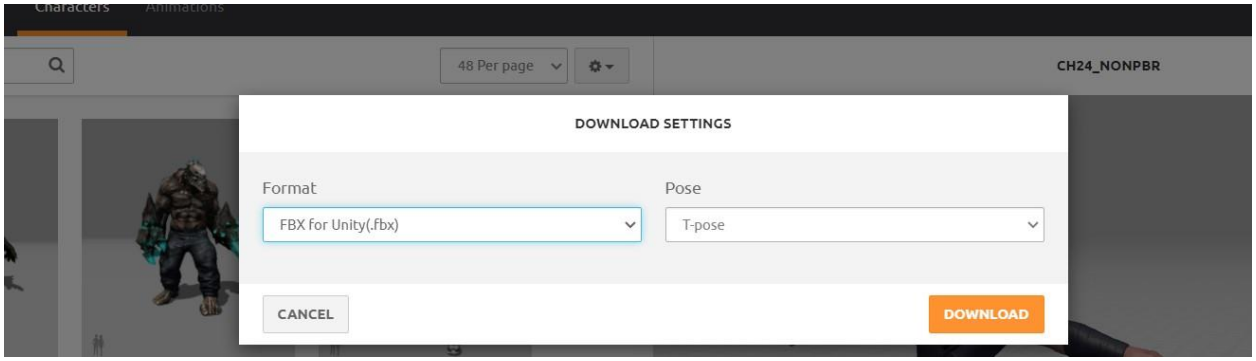
Aim: Implement animation layers in Unity

Firstly add a Plane for our character to stand on



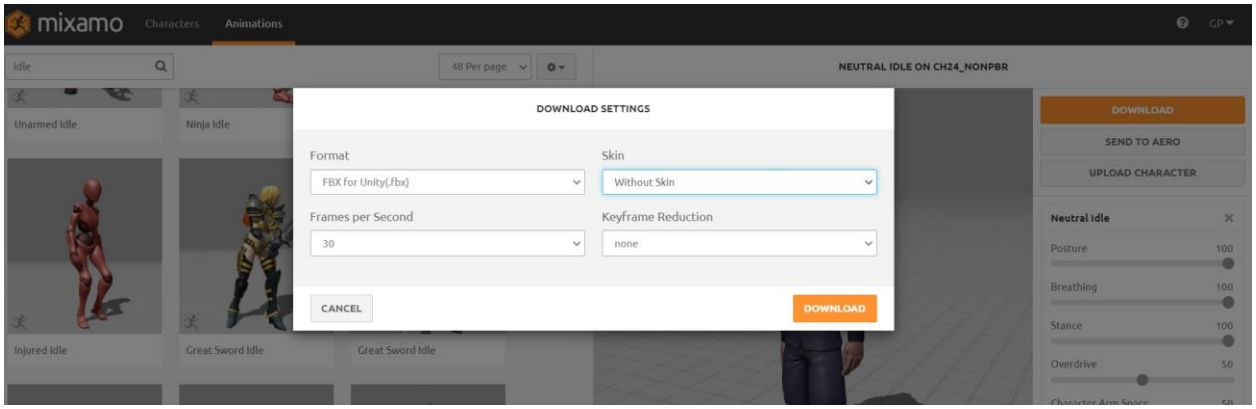
Download character of your choice from mixamo



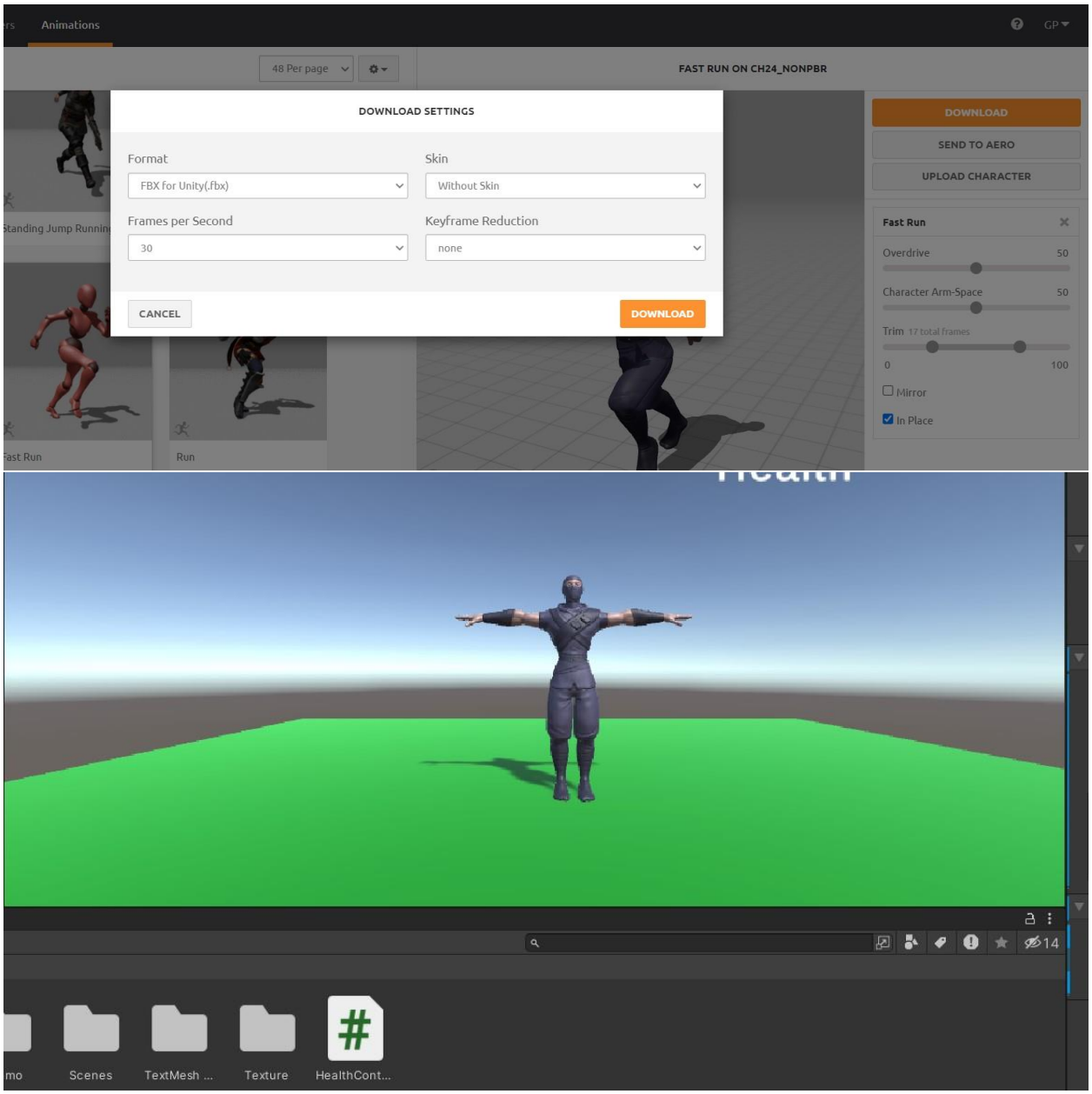


Download Animations Idle, Fast run, Injured Idle, Injured Run without skin and add it in asset folder create a folder Mixamo and import the character and animations in your mixamo folder

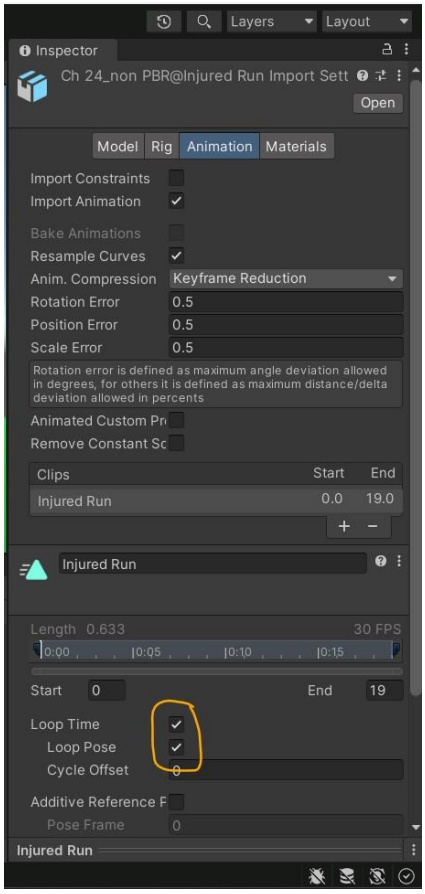
Idle Animation



Fast Run Animation

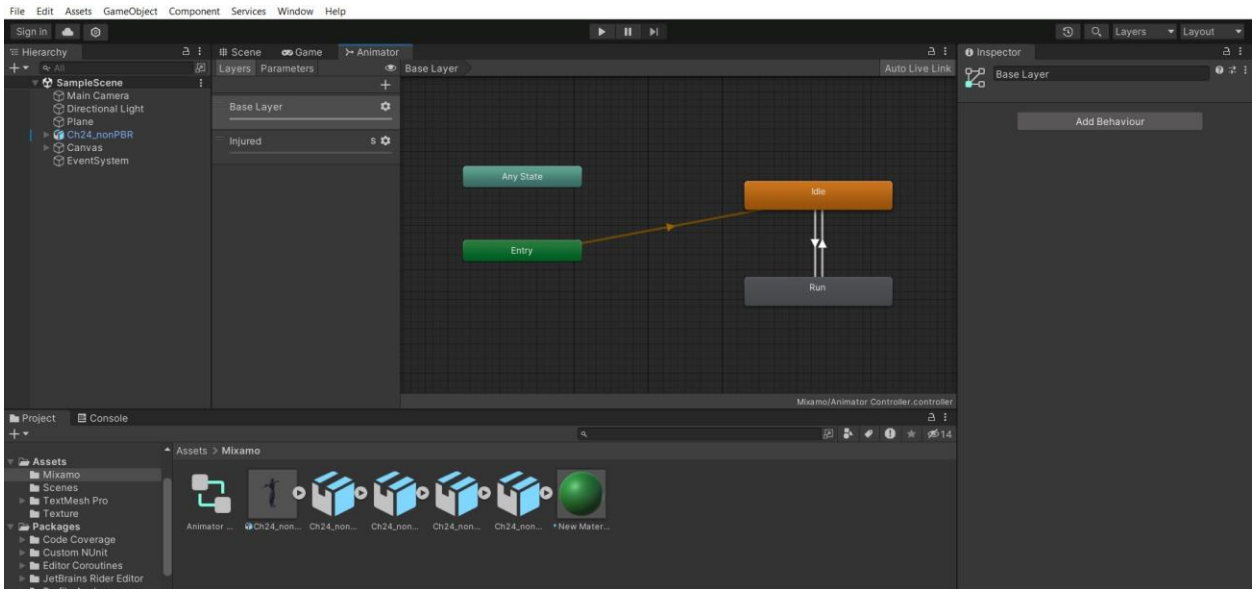


Select Animation from the asset and Enable loop time & loop pose for all the animations.

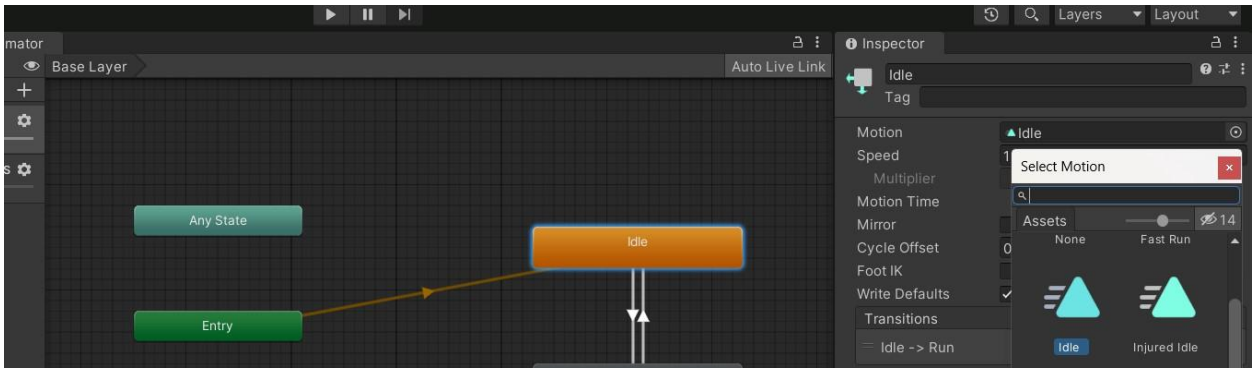


Create and add an Animator Controller to your character [Right-click on Asset folder ➞click on Create ➞Animator Controller ➞Rename it accordingly and add it to your character]

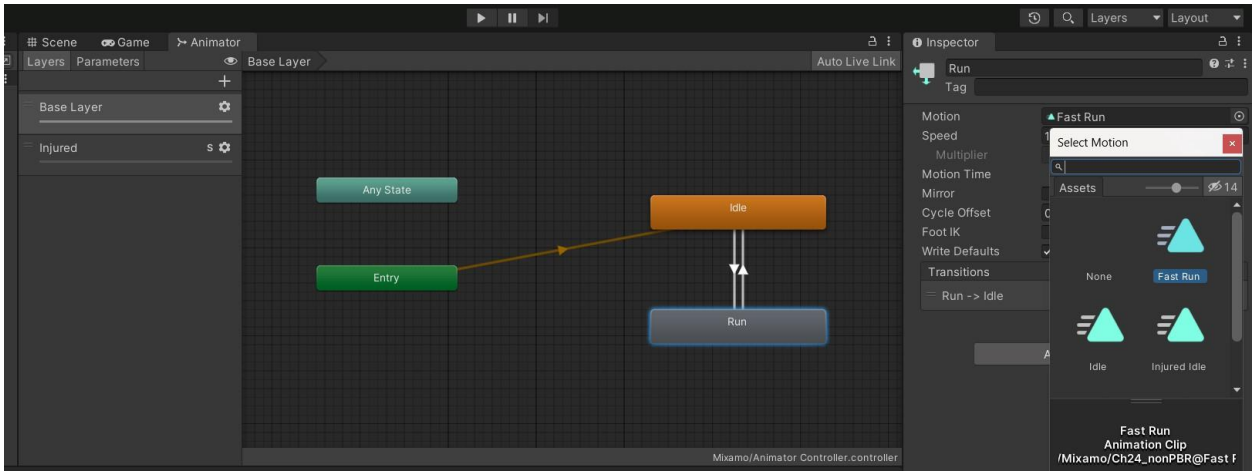
Click on Animator Window ➞Right click and Click on New State



Idle State

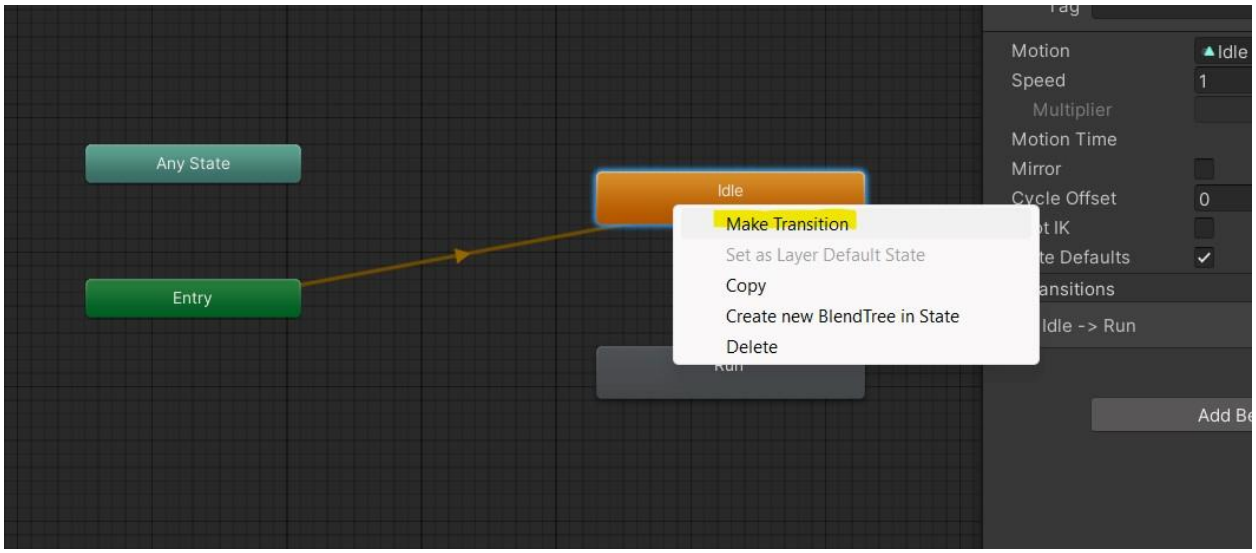


Run State

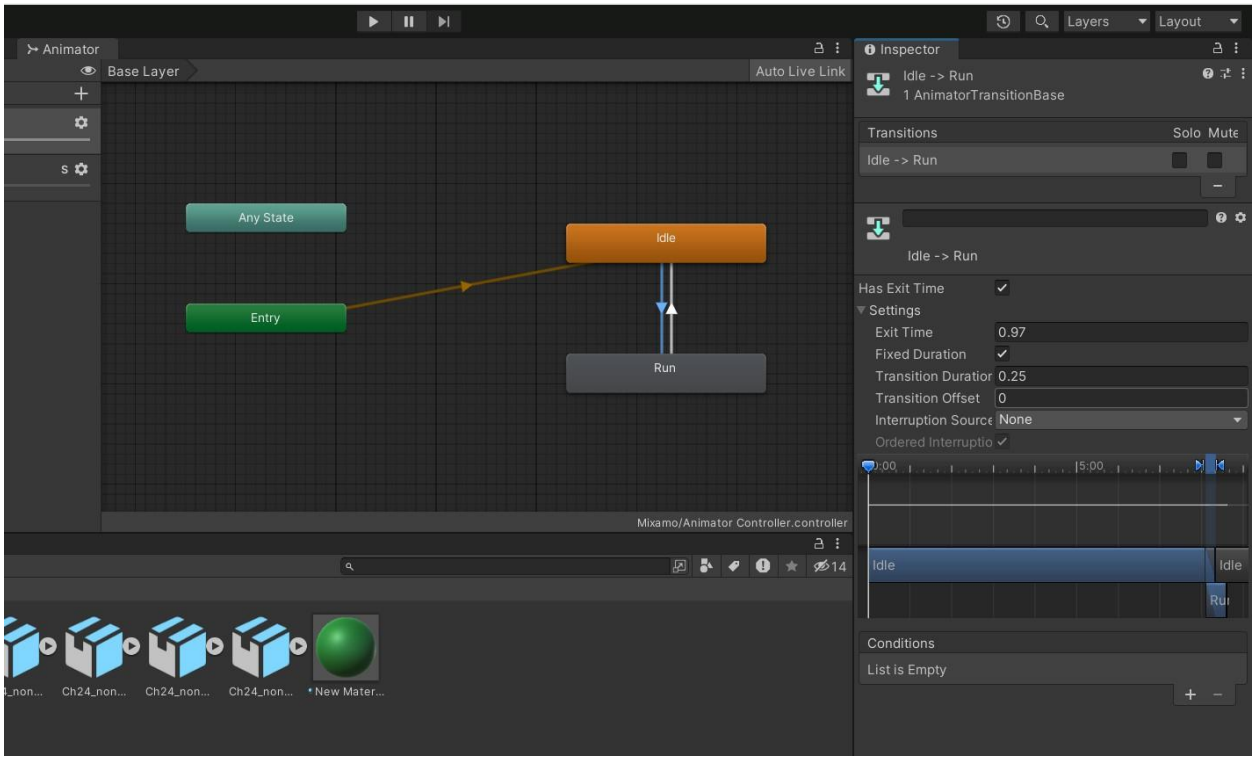


Now add a transition from Idle state to Run state

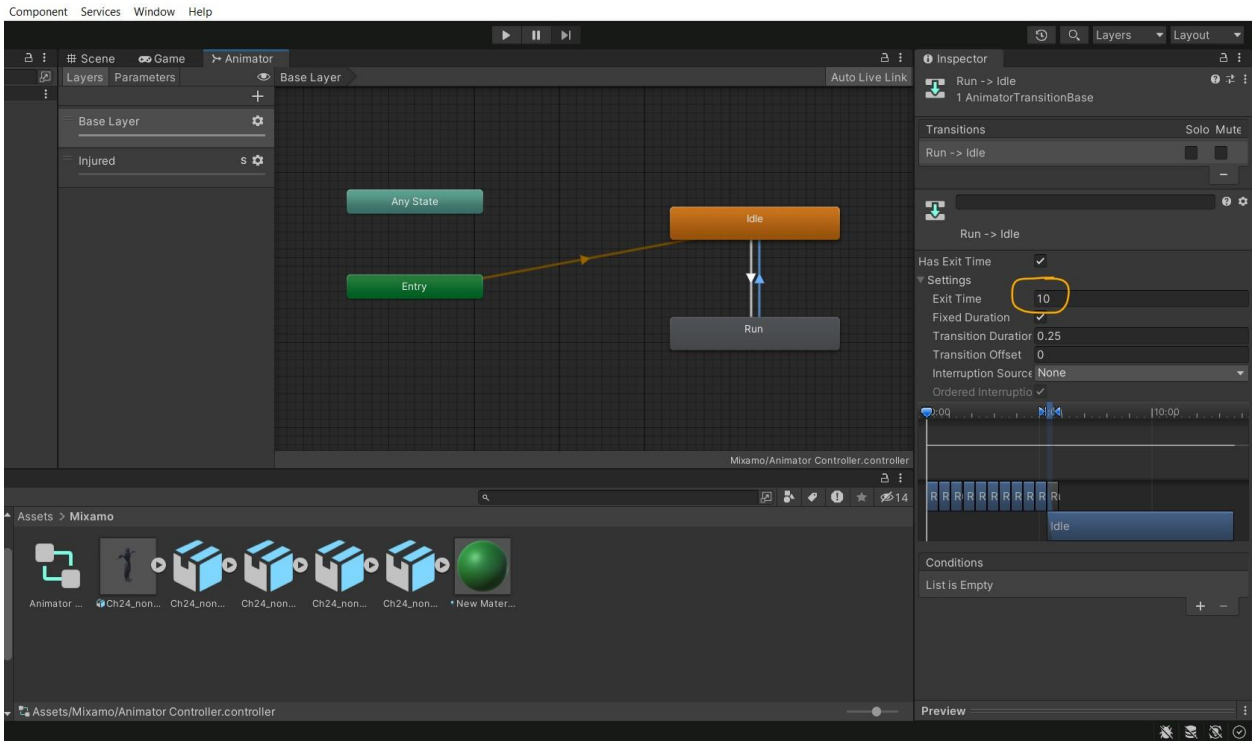
Right-click on Idle state ➞Click on make Transition and connect it to Run state and similarly Make Transition from Run State to Idle State



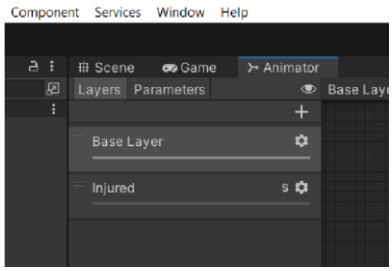
Now click on Transition from Idle state to Run state, and let the checkbox be enabled for Has Exit Time [Exit time 0.97 indicates that the transition to the next state will occur when the current animation has played up to 97% of its duration]



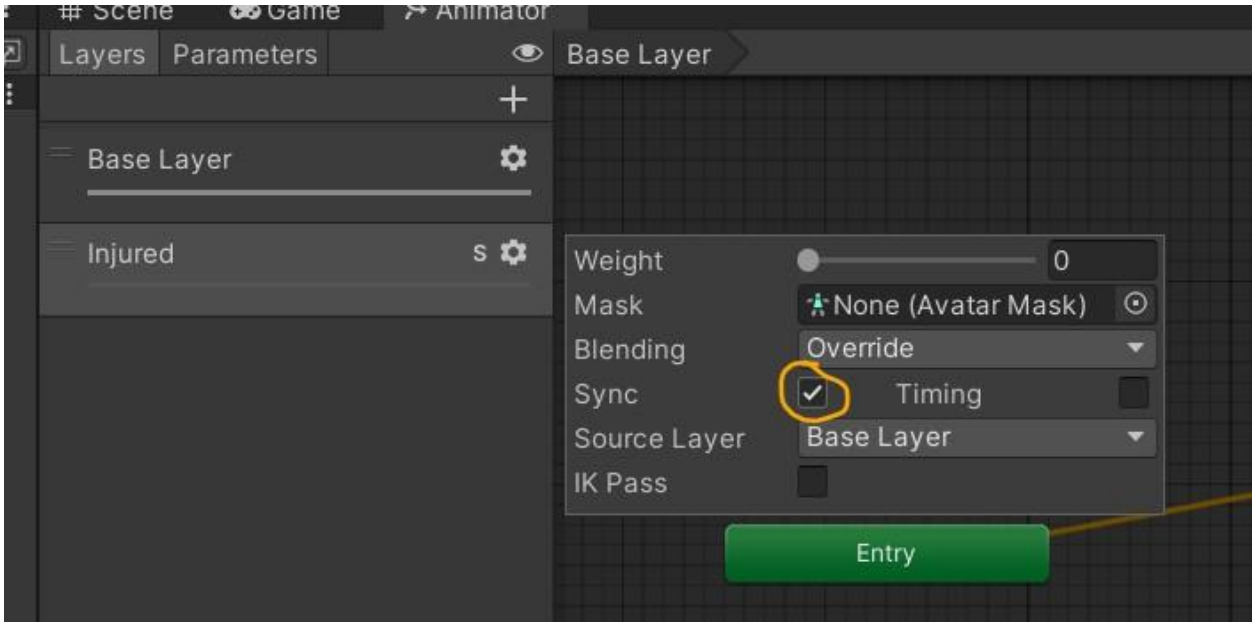
Adding animation layer for Injured animations



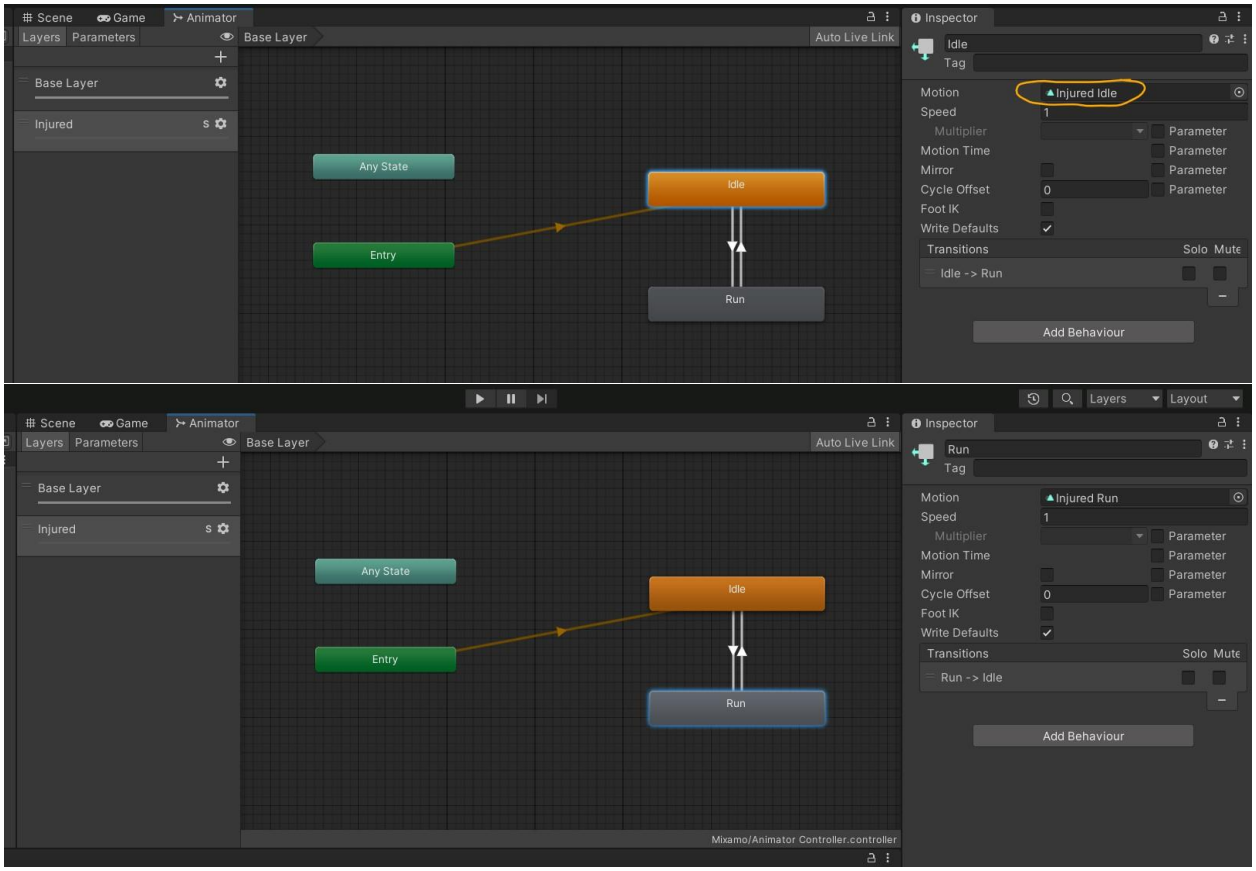
In layers tab below base layer add a layer, rename this layer as Injured



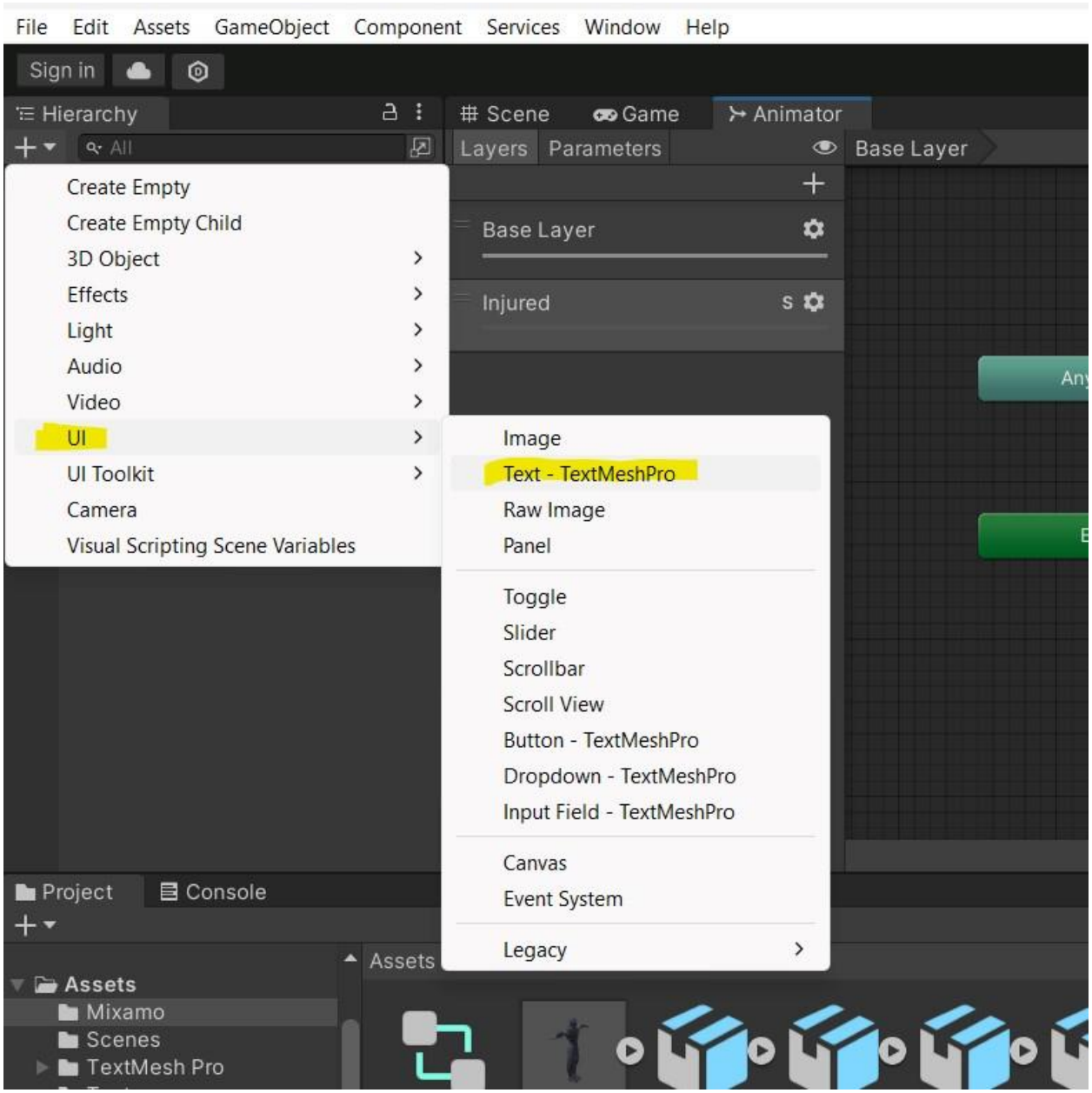
Click on the setting icon and enable sync, this will sync the base layer states with the newly created layer Injured

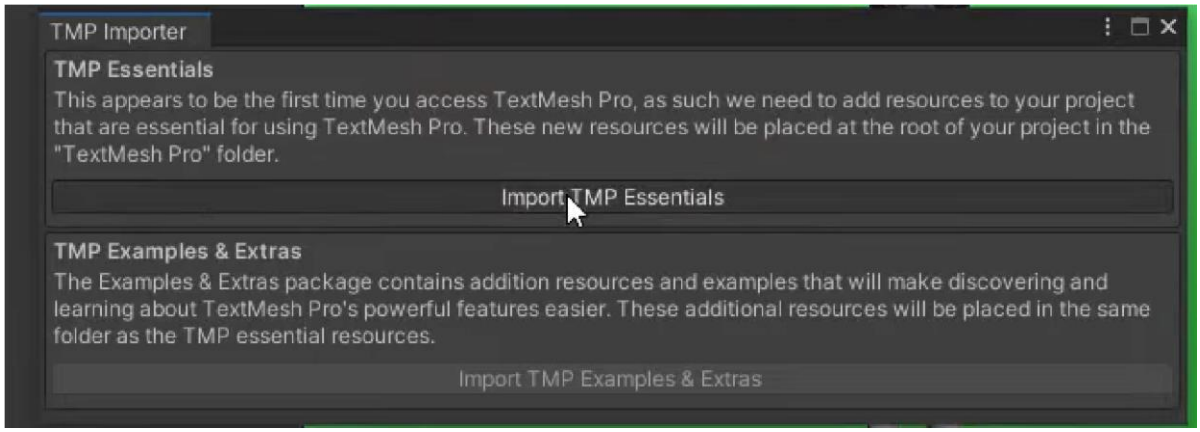


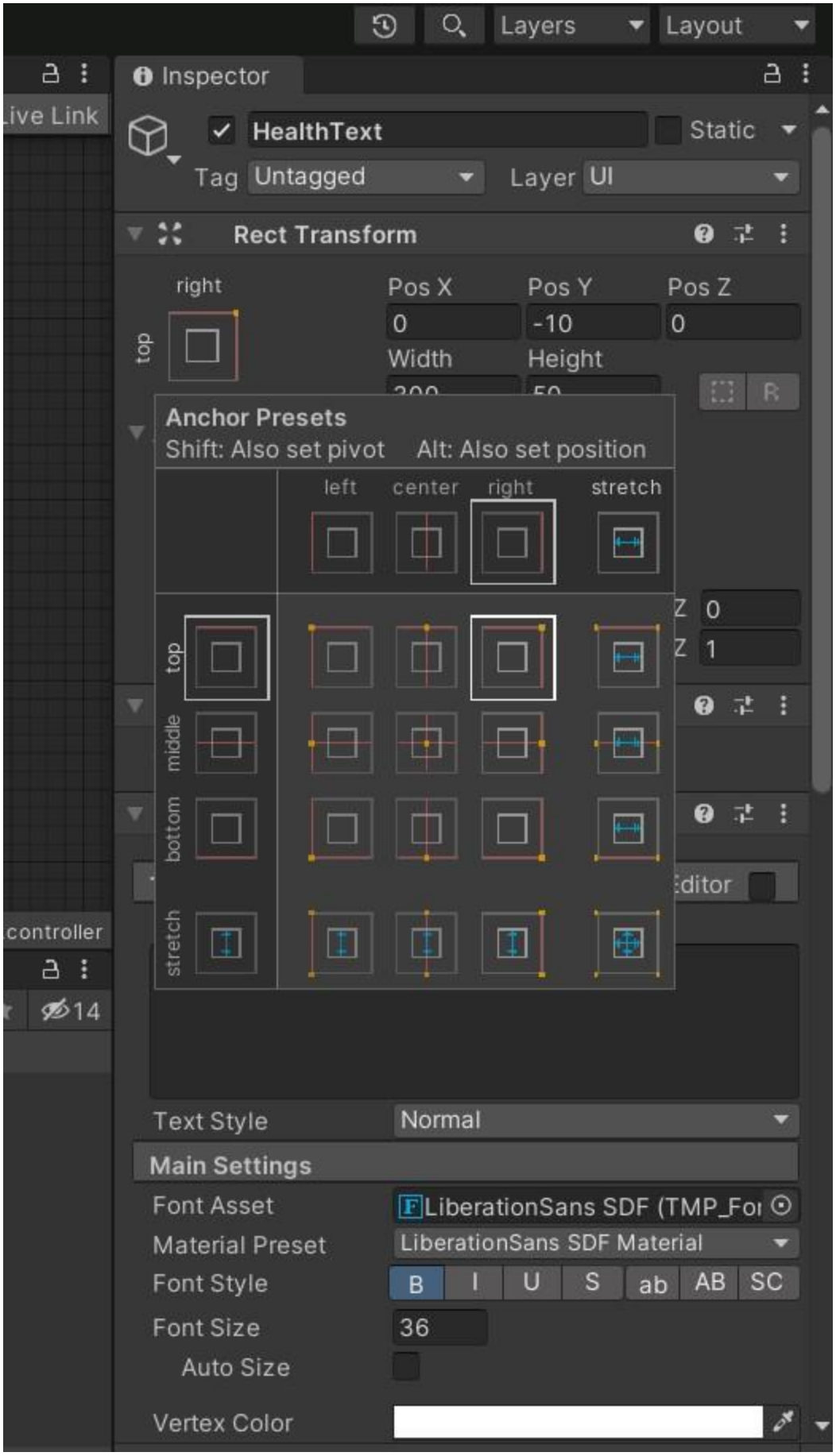
Select Idle state from Injured Layer and in motion add Injured Idle animation



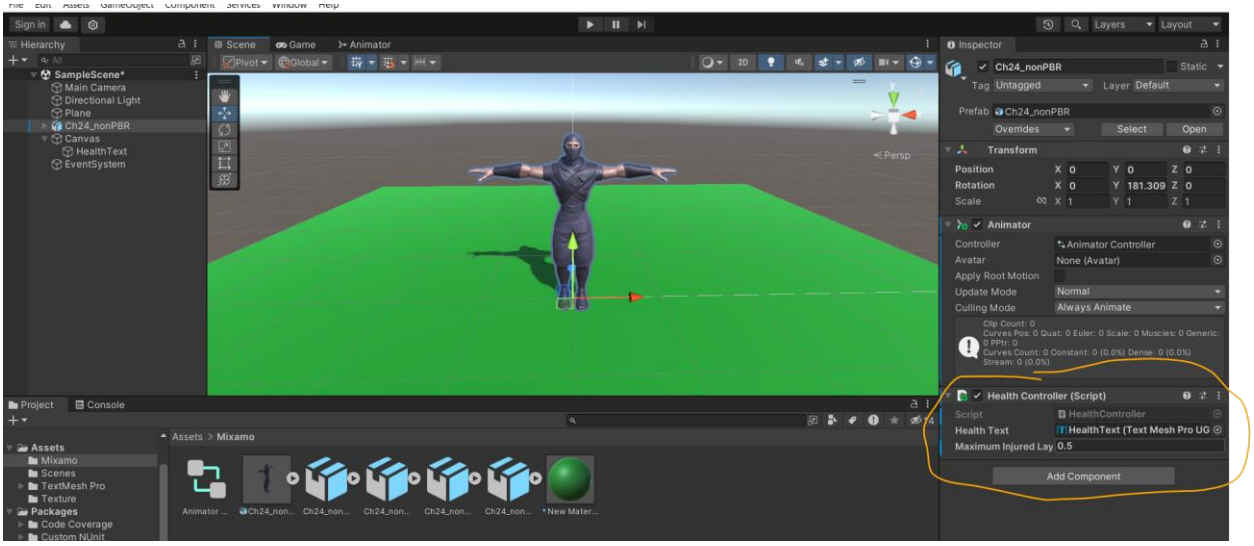
Adding UI to display the health of the player


File Edit Assets GameObject Component Services Window Help
Sign in
Hierarchy
All
Scene
Game
Animator
Layers
Parameters
Base Layer
Create Empty
Create Empty Child
3D Object
Effects
Light
Audio
Video
UI
UI Toolkit
Camera
Visual Scripting Scene Variables
Image
Text - TextMeshPro
Raw Image
Panel
Toggle
Slider
Scrollbar
Scroll View
Button - TextMeshPro
Dropdown - TextMeshPro
Input Field - TextMeshPro
Canvas
Event System
Legacy
Injured
Project
Console
Assets
Mixamo
Scenes
TextMesh Pro


TMP Importer
TMP Essentials
This appears to be the first time you access TextMesh Pro, as such we need to add resources to your project that are essential for using TextMesh Pro. These new resources will be placed at the root of your project in the "TextMesh Pro" folder.
Import TMP Essentials
TMP Examples & Extras
The Examples & Extras package contains addition resources and examples that will make discovering and learning about TextMesh Pro's powerful features easier. These additional resources will be placed in the same folder as the TMP essential resources.
Import TMP Examples & Extras


Layers
Layout
Inspector
HealthText
Static
Tag
Untagged
Layer
UI
Rect Transform
right
top
Pos X
Pos Y
Pos Z
0
-10
0
Width
Height
Anchor Presets
Shift: Also set pivot
Alt: Also set position
left
center
right
stretch
top
middle
bottom
stretch
Z 0
Z 1
Text Style
Normal
Main Settings
Font Asset
LiberationSans SDF (TMP_For
Material Preset
LiberationSans SDF Material
Font Style
B
I
U
S
ab
AB
SC
Font Size
36
Auto Size
Vertex Color

Select your character and click on Add component →add script →remane the script to Health Controller



Double click on script and add the below code to drop health of character once space bar is clicked

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

public class HealthController : MonoBehaviour
{
    // Serialized field allows private variables to be exposed in the Unity Editor for easy tweaking. [SerializeField] private TextMeshProUGUI healthText; // Reference to UI text for displaying
health information

    [SerializeField] private float maximumInjuredLayerWeight; // Maximum
    weight for the "Injured"
animation layer

    private float maximumHealth = 100; // Maximum health value
    private float currentHealth; // Current health value

    private Animator animator; // Reference to the Animator component for
```

```
animation control private int injuredLayerIndex; // Index of the
   "Injured" animation layer private float layerWeightVelocity; //
   Velocity for smooth animation transitions

   // Start is called before the first frame update
   void Start()
   {
      // Initialize current health to maximum health
      currentHealth = maximumHealth;

      // Get reference to the Animator component attached to this game object
      animator = GetComponent<Animator>();

      // Get the index of the "Injured" animation layer in the Animator
      injuredLayerIndex = animator.GetLayerIndex("Injured");
   }

   // Update is called once per frame
   void Update()
   {
      // Check if the spacebar is pressed if
      (Input.GetKeyDown(KeyCode.Space)
      )
      {
         // Reduce current health by 10% of maximum health
         currentHealth -= maximumHealth / 10;
      }

      // Ensure current health doesn't go below
      0 if (currentHealth < 0)
      {
         currentHealth = maximumHealth; // Reset health to maximum if it's
below 0
      }

      // Calculate health percentage float healthPercentage =
      currentHealth / maximumHealth;
```

```
        // Update UI text to display health percentage
        healthText.text = $"Health: {healthPercentage * 100}%";

        // Get the current weight of the "Injured" animation layer
        float currentInjuredLayerWeight =
animator.GetLayerWeight(injuredLayerIndex);
        // Calculate the target weight based on health percentage and
maximum injured layer weight float targetInjuredLayerWeight = (1 -
healthPercentage) *
maximumInjuredLayerWeight;

        // Smoothly transition the weight of the "Injured" animation
        layer animator.SetLayerWeight( injuredLayerIndex,
            Mathf.SmoothDamp(
                currentInjuredLayerWeigh
                t,
                targetInjuredLayerWeight,
                ref layerWeightVelocity,
                0.2f // Smoothing duration
            )
        );
    }
}
```