Practical 7 Animations

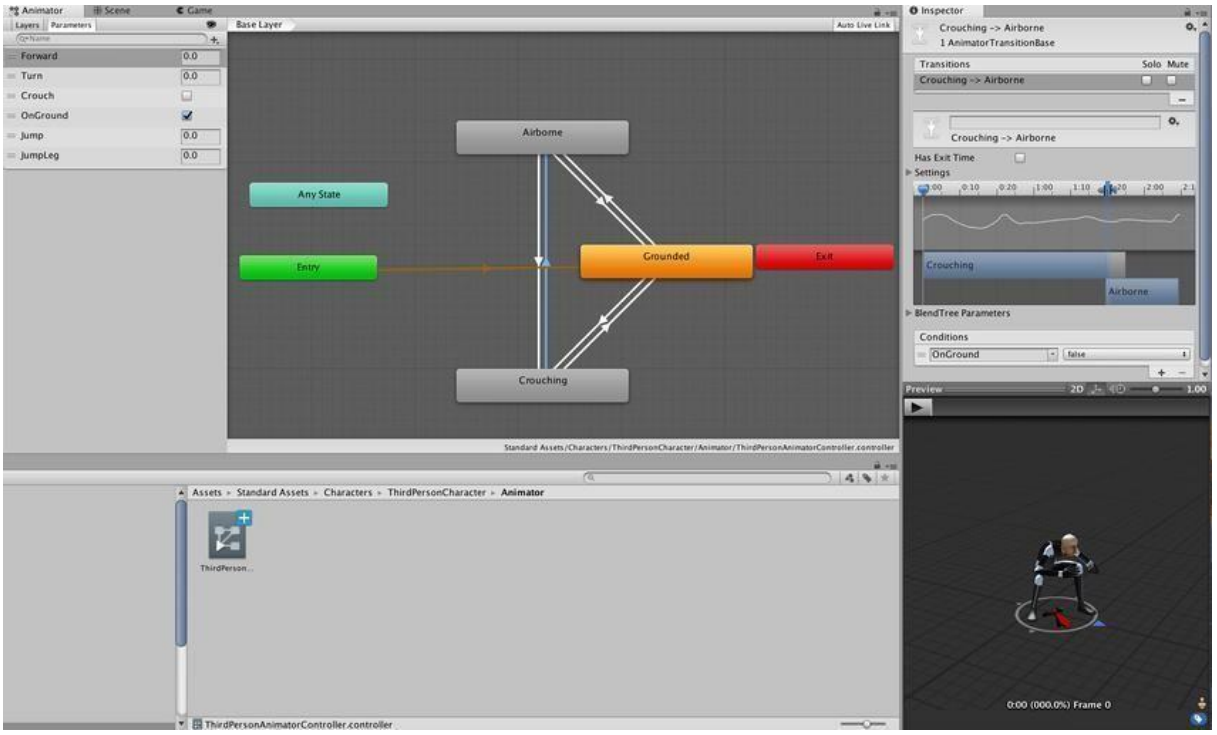
Aim : implement animation effect on game object

Animation System Overview

Unity has a rich and sophisticated animation system (sometimes referred to as '**Mecanim**'). It provides:

Easy workflow and setup of animations for all elements of Unity including objects, characters, and properties.
Support for imported animation clips and animation created within Unity

retargeting
The ability to apply animations from one character model onto another. Simplified workflow for aligning animation clips.
Convenient preview of animation clips, transitions and interactions between them. This allows animators to work more independently of programmers, prototype and preview their animations before gameplay code is hooked in.
Management of complex interactions between animations with a visual programming tool.
Animating different body parts with different logic. Layering and masking
features



Typical view of an Animation State Machine in the Animator window

Animation workflow

Unity's animation system is based on the concept of Animation Clips, which contain information about how certain objects should change their position, rotation, or other properties over time. Each clip can be thought of as a single linear recording. Animation clips from external sources are created by artists or animators with 3rd party tools such as Autodesk® 3ds Max® or Autodesk® Maya®, or come from motion capture studios or other sources.

Animation Clips are then organised into a structured flowchart-like system called an Animator Controller. The Animator Controller acts as a "State Machine" which keeps track of which clip should currently be playing, and when the animations should change or blend together.

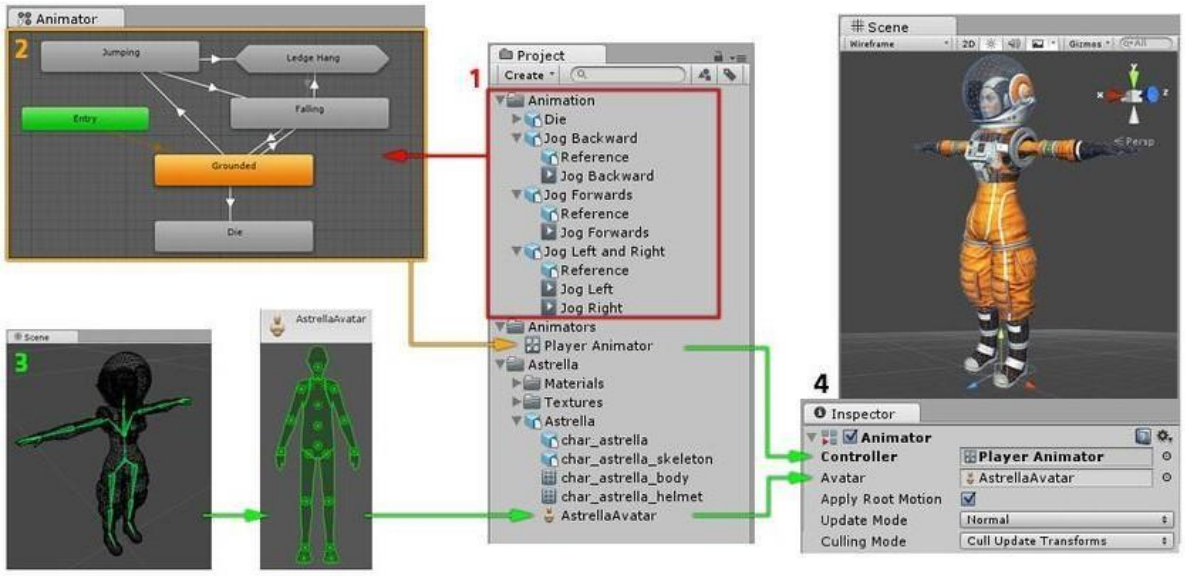
A very simple Animator Controller might only contain one or two clips, for example to control a powerup spinning and bouncing, or to animate a door opening and closing at the correct time. A more advanced Animator Controller might contain dozens of humanoid animations for all the main character's actions, and might blend between multiple clips at the same time to provide a fluid motion as the player moves around the **scene.**

Unity's Animation system also has numerous special features for handling humanoid characters which give you the ability to retarget humanoid animation from any source (for example: motion capture; the Asset Store; or some other third-party animation library) to your own character model, as well as adjusting muscle definitions

These special features are enabled by Unity's Avatar system, where humanoid characters are mapped to a common internal format.

Each of these pieces - the Animation Clips, the Animator Controller, and the Avatar, are brought together on a **GameObject** via the Animator Component

. This component has a reference to an Animator Controller, and (if required) the Avatar for this model. The Animator Controller, in turn, contains the references to the Animation Clips it uses.



How the various parts of the animation system connect together

The above diagram shows the following:

1. Animation clips are imported from an external source or created within Unity. In this example, they are imported motion captured humanoid animations.
2. The animation clips are placed and arranged in an Animator Controller. This shows a view of an Animator Controller in the **Animator window**

4. . The States (which may represent animations or nested sub-state machines) appear as nodes connected by lines. This Animator Controller exists as an asset in the **Project window**

6. .
7. The rigged character model (in this case, the astronaut "Astrella") has a specific configuration of bones which are mapped to Unity's common Avatar format. This mapping is stored as an Avatar asset as part of the imported character model, and also appears in the Project window as shown.
8. When animating the character model, it has an Animator component attached. In the **Inspector**

10. view shown above, you can see the Animator Component which has both the Animator Controller and the Avatar assigned. The animator uses these together to animate the model. The Avatar reference is only necessary when animating a humanoid character. For other types of animation, only an Animator Controller is required.

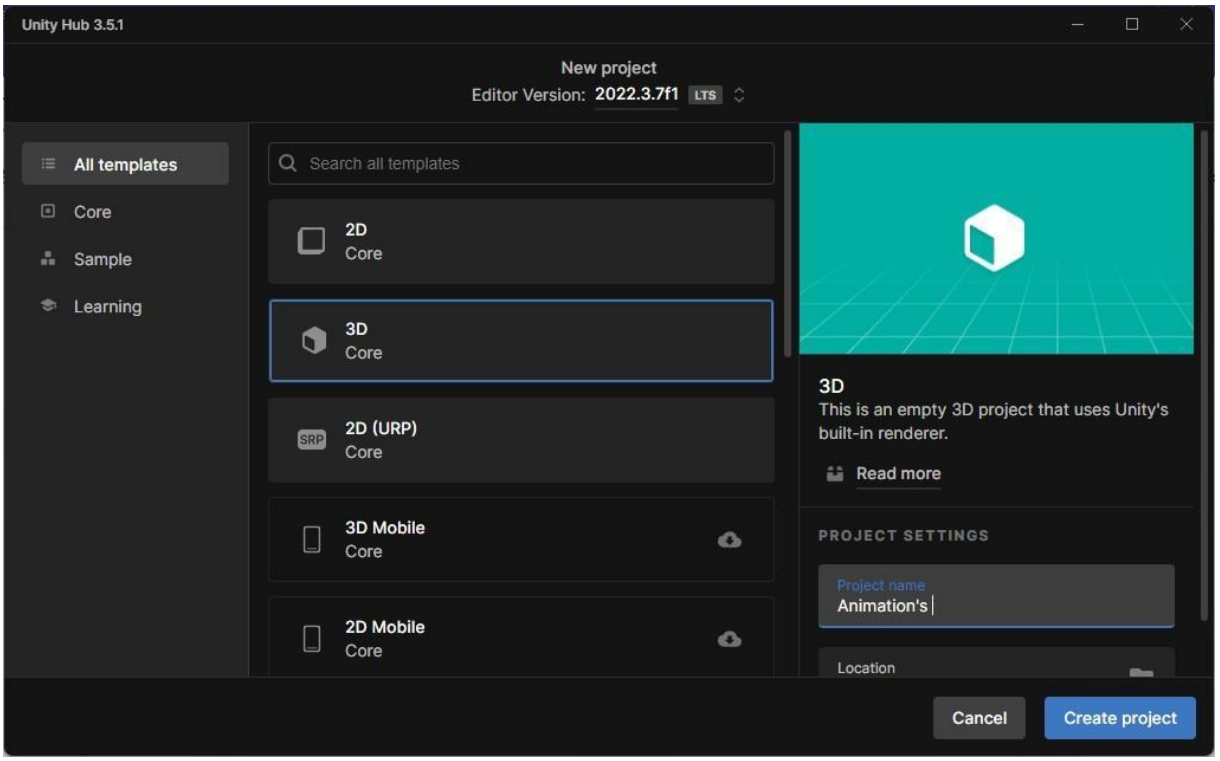
Unity's animation system comes with a lot of concepts and terminology. If at any point, you need to find out what something means, go to our Animation Glossary.

Refrences https://docs.unity3d.com/Manual/AnimationOverview.html

Implementation :

Step 1

Create a new 3D unity project

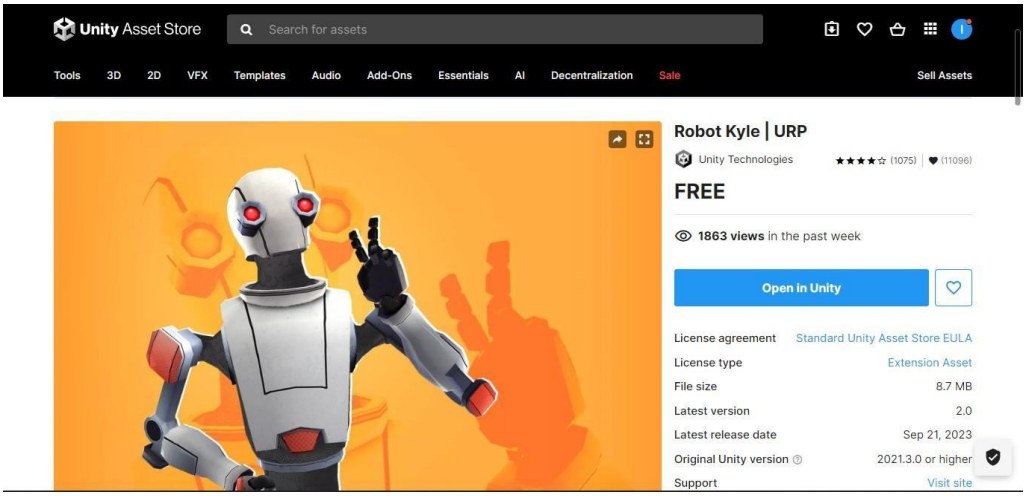


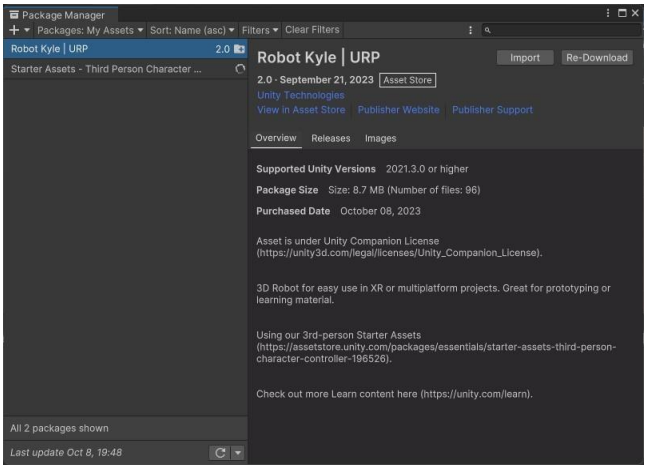
Step 2

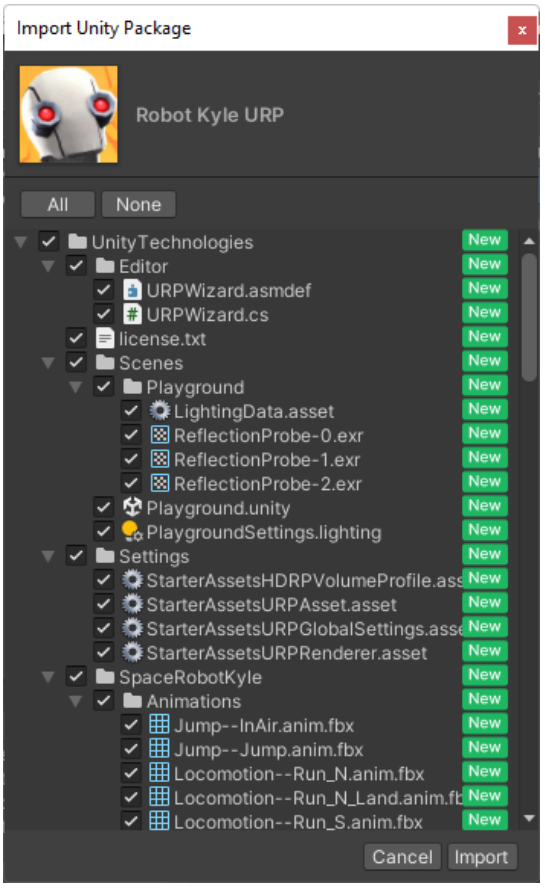
Go to unity asset store and search for Humanoids

https://assetstore.unity.com/packages/3d/characters/robots/robot-kyle-urp-4696

Import the asset to the unity project
OR you can import whatever model from internet i.e mixamo is good place

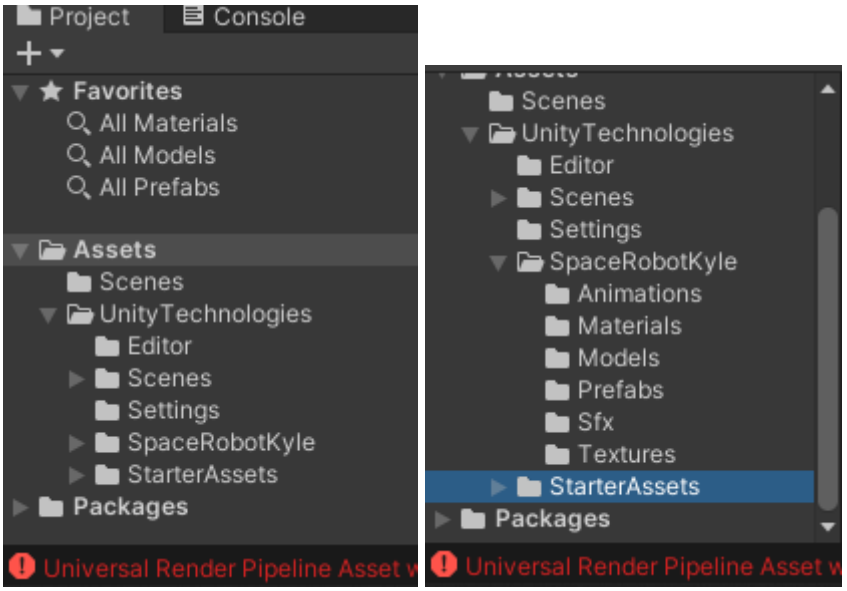






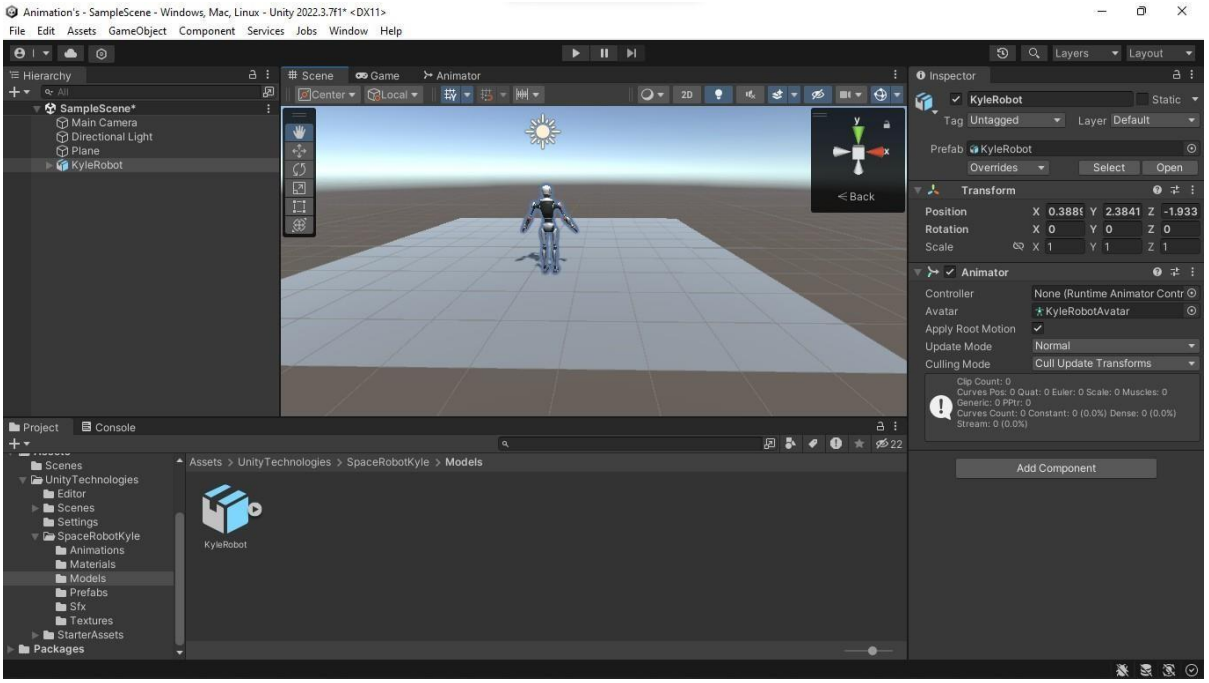
Step 3

The imported model has SpaceRobotKyle folder inside it we have model textures and animations



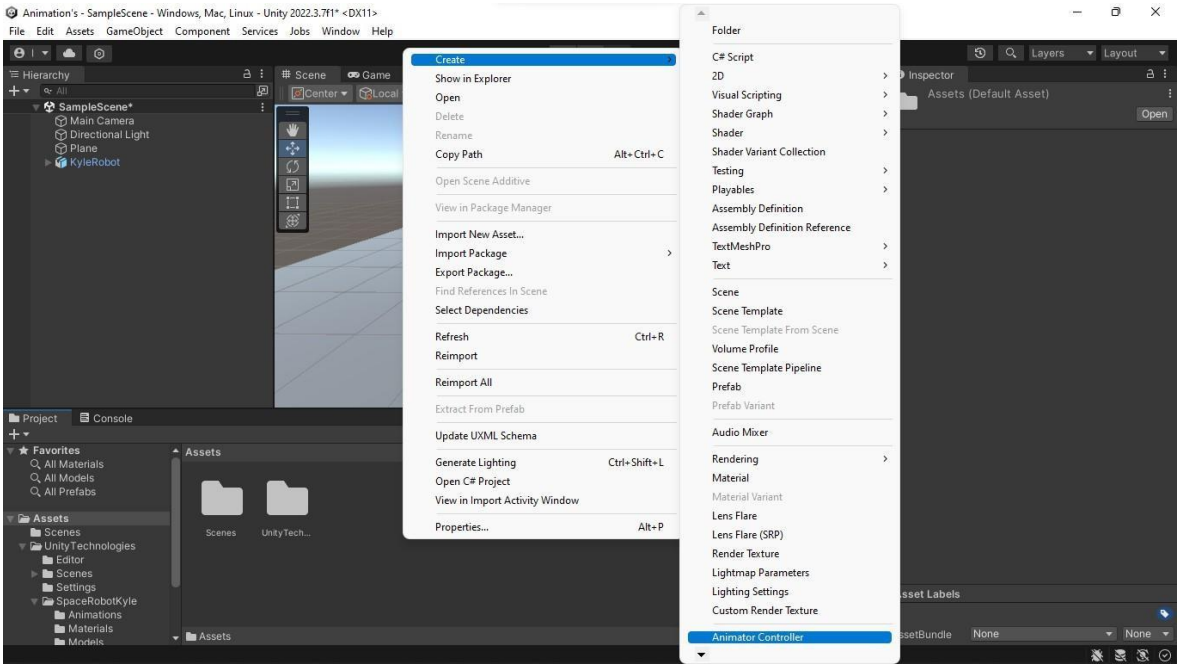
Step 4

Create and 3d plane from game objects and drag the model from models folder

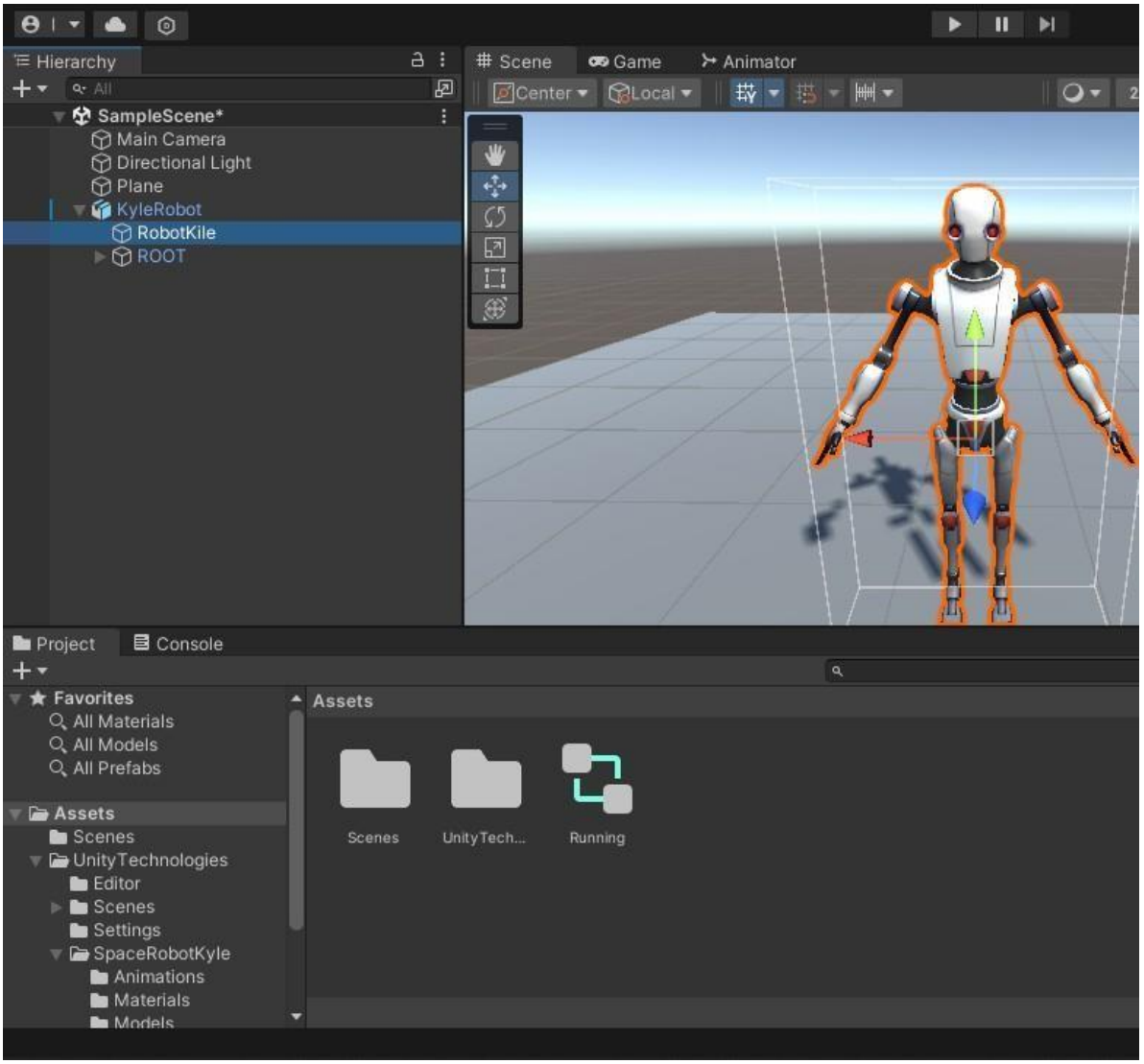


Step 5

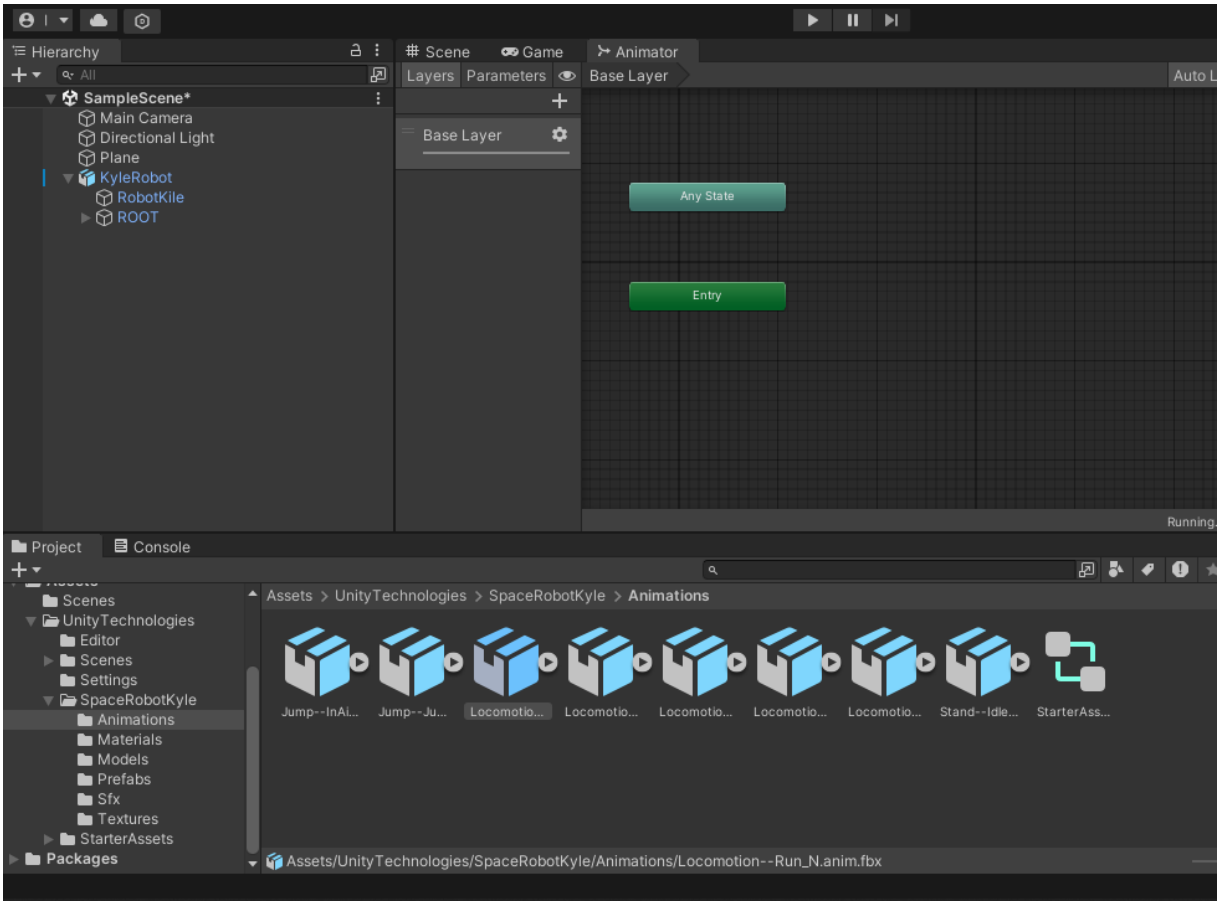
Go to asset folder right click and create new animation controller name it running



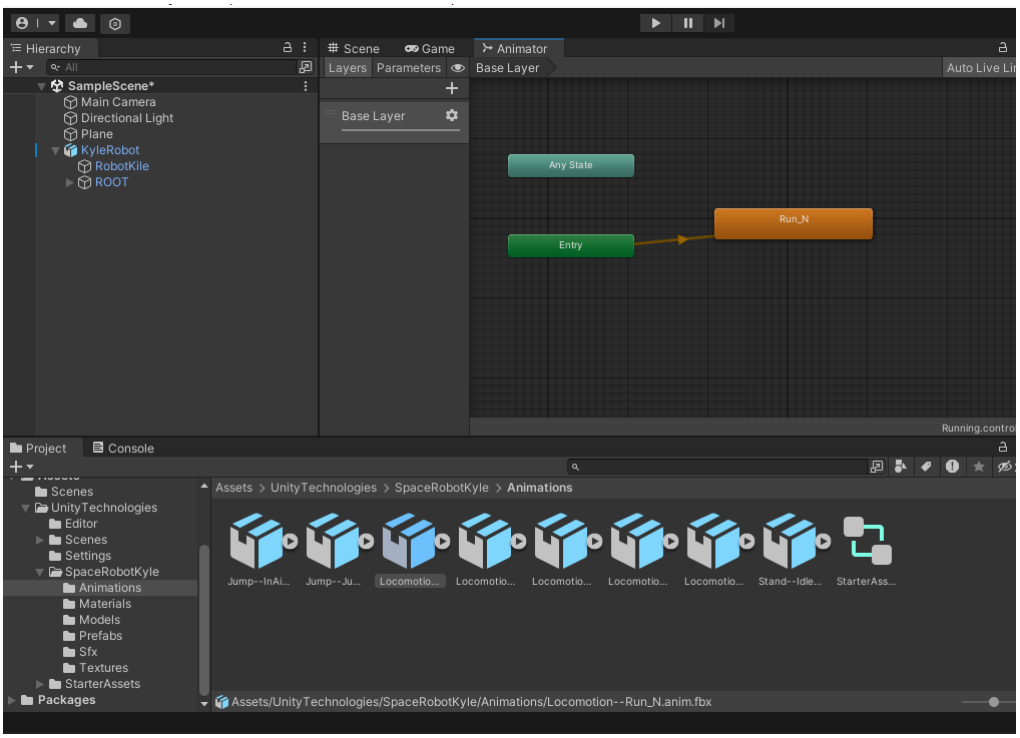
Drag running animation component to our model



Double click the Running component

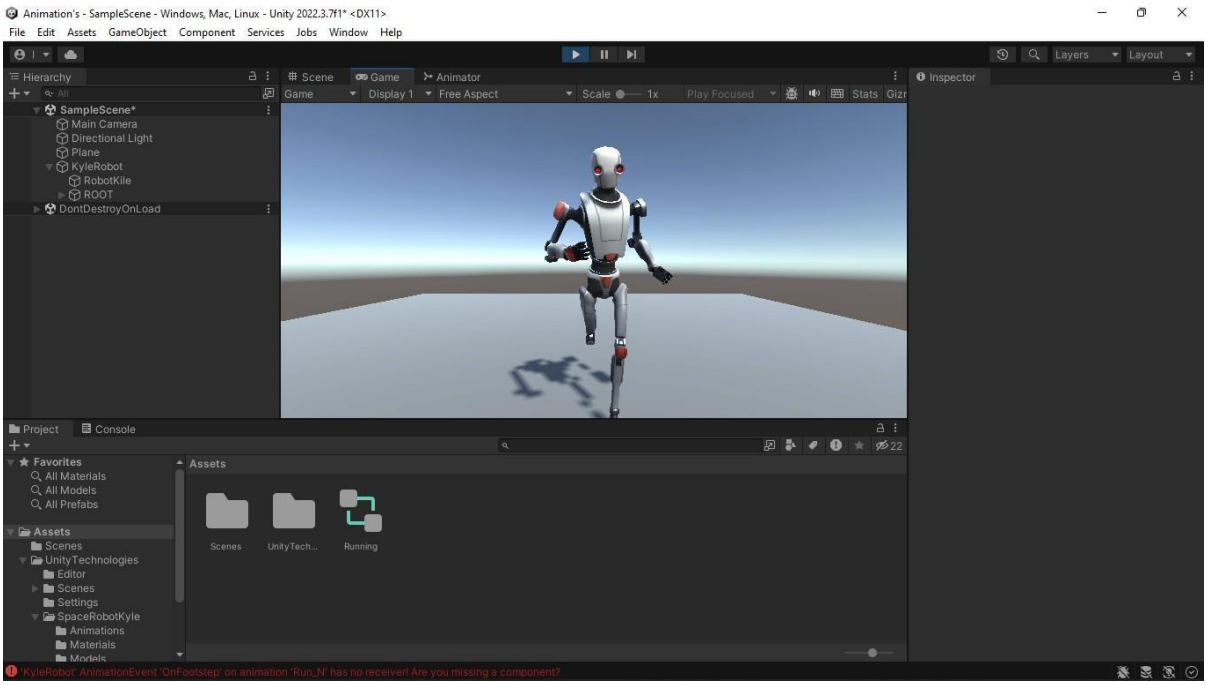


Go to Animation folder of the model and select any animation and drag it to the Animations



Step 7

Click on run try tweaking settings



Conclusion

Successfully implemented animations on 3d models