# Unit 2 – Lesson 2. Introduction to Humanoid Animation

**Aim:**

* How does Unity handle animation?
* What is Legacy Animation and how do we use Legacy Animation in game programming?

**Objectives:** After the lesson, students will be able to:

* Obtain understanding of animated assets in Unity game development
* Import animated assets into Unity project
* Set up Legacy animation in Unity and write C# scripts to trigger and stop animation

**CLASS PROCEDURE:**

***Do Now:***

Open the Garden Defender in Unity.

1. How do we make the bench move towards the garden gate?
2. How do we make the garden gate automatically open when the bench approaching the gate?
3. How do we make the garden gate automatically close after the bench pass through?

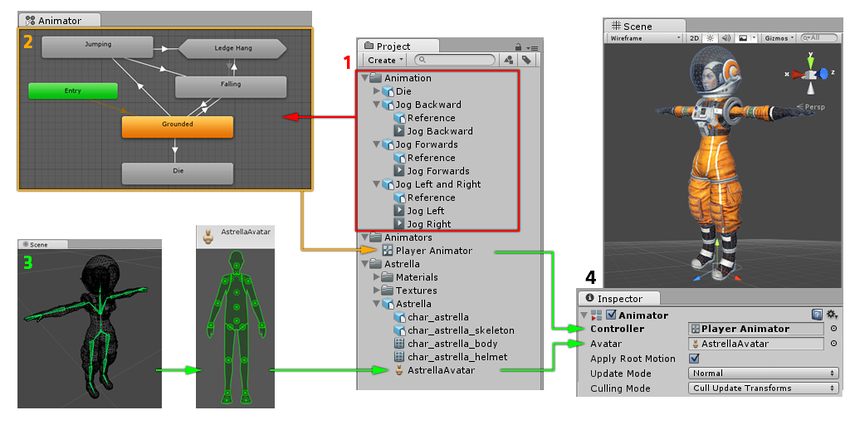
***Pair – sharing Activity #1:***

Work with your partner, use yesterday’s lesson note as reference,

1. When the game starts, make the bench moving towards the garden gate.
2. Then make the garden gate open automatically to let the bench passing through.
3. Make the garden gate automatically close after the bench exits the gates.

***Class Discussion / Presentation – How does the Unity Animation System work?***

1. Unity’s animation system is based on the concept of [Animation Clips](https://docs.unity3d.com/Manual/class-AnimationClip.html), 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 Max or Maya, or come from motion capture studios or other sources.
2. Animation Clips are then organized into a structured flowchart-like system called an Animator Controller. The Animator Controller acts as a “[State Machine](https://docs.unity3d.com/Manual/AnimationStateMachines.html)” which keeps track of which clip should currently be playing, and when the animations should change or blend together.
3. A very simple Animator Controller might only contain one or two clips, for example to control a power up 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.
4. Unity’s Animation system also has numerous special features for handling humanoid characters which give you the ability to [retarget](https://docs.unity3d.com/Manual/Retargeting.html) humanoid animation from any source (Eg. motion capture, the asset store, or some other third-party animation library) to your own character model, as well as adjusting [muscle definitions](https://docs.unity3d.com/Manual/MuscleDefinitions.html). These special features are enabled by Unity’s [Avatar](https://docs.unity3d.com/Manual/class-Avatar.html) system, where humanoid characters are mapped to a common internal format.
5. Each of these pieces - the [Animation Clips](https://docs.unity3d.com/Manual/class-AnimationClip.html), the [Animator Controller](https://docs.unity3d.com/Manual/class-AnimatorController.html), and the [Avatar](https://docs.unity3d.com/Manual/class-Avatar.html), are brought together on a GameObject via the [Animator Component](https://docs.unity3d.com/Manual/class-Animator.html). 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](https://docs.unity3d.com/Manual/class-AnimationClip.html) it uses.

Diagram showing 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. 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.
3. The rigged character model (in this case, the astronaut “Astrella”) has a specific configuration of bones which are mapped to Unity’s common [Avatar](https://docs.unity3d.com/Manual/class-Avatar.html) 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.
4. When animating the character model, it has an Animator component attached. In the Inspector view shown above, you can see the [Animator Component](https://docs.unity3d.com/Manual/class-Animator.html) which has both the [Animator Controller](https://docs.unity3d.com/Manual/class-AnimatorController.html) and the [Avatar](https://docs.unity3d.com/Manual/class-Avatar.html) 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.

***Pair – sharing Activity #2:*** Open the Garden Defender,

1. From the Animated Assets folder, select the Scarecrow and check it out in the Inspector.
2. In the Rig section, select Humanoid for the Animation Type.
3. Click on the Configure button.
4. Save the scene and also to Applying the settings.
5. The Inspector now should show the character and the bones that mapped as green.
6. Click the Muscles tab (next to the Mapping section).
7. Try adjusting the sliders in the Muscle Group Preview to see how the character’s rigging holds up by making the Scarecrow make different poses.
8. Click the Reset All button at the top of the sliders.

***HW:***

Read <https://docs.unity3d.com/Manual/MecanimFAQ.html>