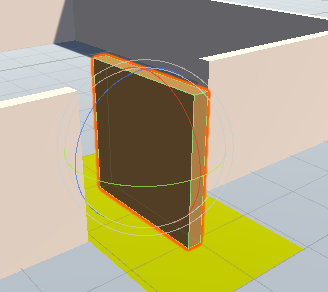
Animation and Animators

Animating object in Unity typically requires two components: an **Animation**, and an **Animator**. Animations represent the changes in properties (material, position, orientation, …) over time. Animators are used to controlling the logic behind animations, specifically how and when they are triggered.

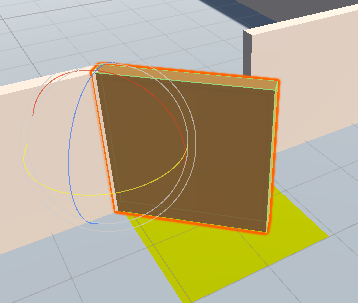
# The Door Pivot

Select the GameObject you want to animate. In this case, we will use the door from the last session. The door was created from a standard Unity cube. By default, their pivot (the point around which they rotate) is in their centre. If we manually rotate the door on the Y axis, it will not open correctly.



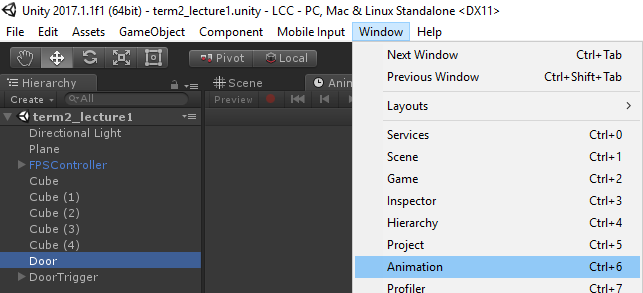
To fix this, we need to make sure that the door pivot is on the side, not on the centre. If you have created the door model with an external 3D application, you can change this by hand. However, Unity does not allow to do edit 3D models.

A simple solution is to place the door inside an empty gameobject, with a slight offset. You can rotate the pivot so that it causes the door to open and close as expected.

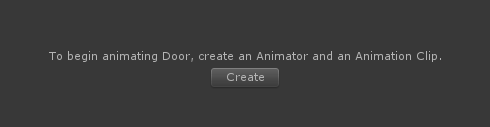


# Creating an Animation

Once the Door pivot is selected, go to the Animation window. If that is not present in your space, click on **Windows > Animation**.



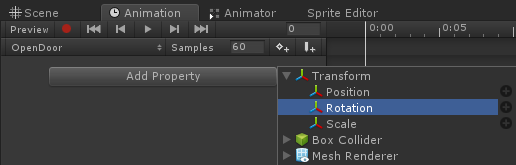
When the door is selected, you can now create an Animation using the **Create** button.



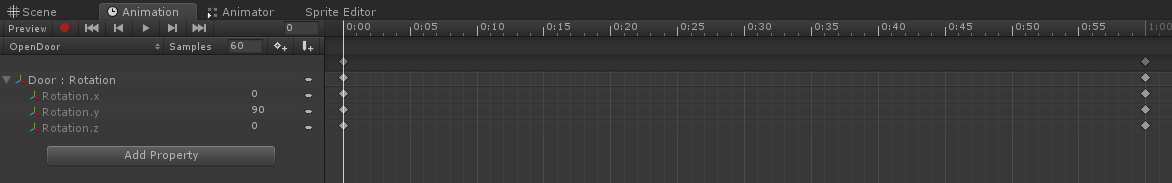
This will add an **Animator component** to the selected GameObject, and create an **Animation Clip asset**. The animation clip is a file that needs to be saved in your project; let’s call it **OpenDoor**. It is good practice to create a special folder for all the animations.

# Animating the Door

From the Animation window, you can decide which properties of the GameObject need to be animated. Since the door is open by a rotation, we can add a **Rotation property**.



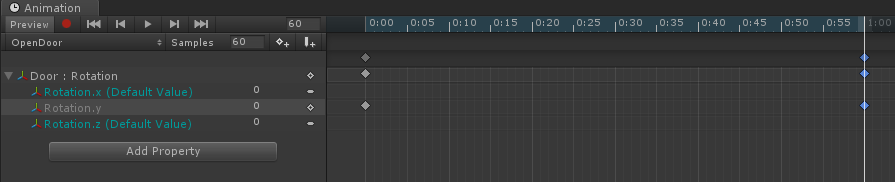
This creates a timeline, with two **key frames** (white diamonds). We can specify the rotation of the door at those two specific points in time; Unity will interpolate between them.



Click on the last key frame and make sure that the vertical white line is on it.

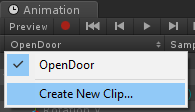
The rotation fields in the Inspector are now blue (or red), meaning that any change done do it will be saved in the animation. Set the rotation on the so that your door is now open.

You can also select individual key frames for individual properties (such as **Rotation.x** and **Rotation.z)** and delete them. This means that the animation will leave **Rotation.x** and **Rotation.z** unchanged, and will only affect **Rotation.y**.



You can test the animation by pressing the play button in the Animation window.

You can create a second animation to close the door:

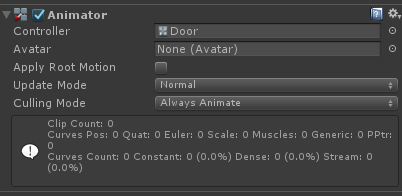


If you run the game now, there will be an obvious issue: the first transition you have created plays endlessly. To make it happen only once, find its Animation Clip asset and from the Inspector and uncheck **Loop Time**. Now it will only play once.

# Configuring the Animator

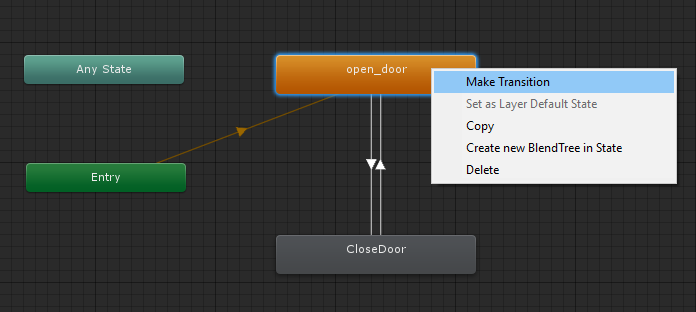
Now that the two animations are done, we need to configure the Animator to control when they have to be played.

You can open the **Animator** window by double-clicking on the **Controller** property of the **Animator** component.

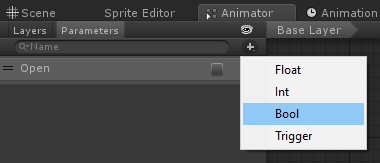


The animator is a **Finite State Machine** which has a state for each animation.

You can connect the two animations using transitions. This indicates that we can move from **OpenDoor** to **CloseDoor**, and from **CloseDoor** to **OpenDoor**.



Transitions are triggered by **parameters**. They are properties of the Animator that scripts can change. The state of the door (open or close) can be represented with a boolean parameter. Create one, and call it **Open**.

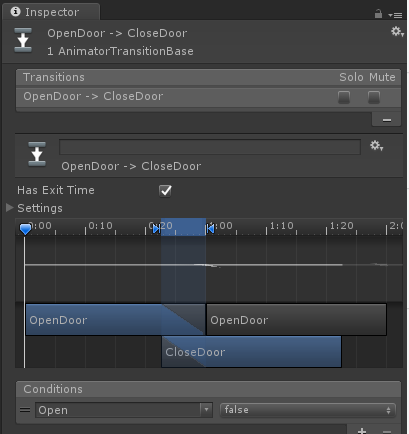


We will use it to control the door.

# Using Transitions

Select each transition, and change its **Conditions**.

For instance, the transitions from **OpenDoor** to **CloseDoor** should happen the parameter **Open** is set to **false** (image below).



Since we have not used any script to change the value of Open, the door will not react yet to our inputs.

Create a new script and attach it to the Trigger Collider that will open the door, as we did in the previous lecture.

|  |
| --- |
| public class DoorTrigger : MonoBehaviour  {  **public Animator Animator;**  void OnTriggerStay(Collider other)  {  if (other.CompareTag("Player") && Input.GetKeyDown(KeyCode.LeftControl))  **Animator.SetBool("Open", true);**  }  void OnTriggerExit(Collider other)  {  if (other.CompareTag("Player"))  **Animator.SetBool("Open", false);**  }  } |

After linking the Animator from the inspector, the script will change its parameter **Open**, causing the animation to change.