Zombie Shooter Project 3d

Task 1. Setup Mecanim for player firing

Explanation

- Now we have both our **Player firing** and **Player firing Idle** animations, we shall setup the **Animator Controller**
- The **Animator Controller** will allow us to **swap** between animations while the game is running

- In our Zombie Shooter Game, our **fire button** is the **left mouse button**
- We want to **play** the **Player firing** animation while the **fire button is down**
- When the **fire button is not down**, we want to play the **Player firing Idle** animation

- The **Animator Controller** will allow is to do this, with a little bit of scripting!

Useful links

- More information about **Animator Controller** Animator Controller

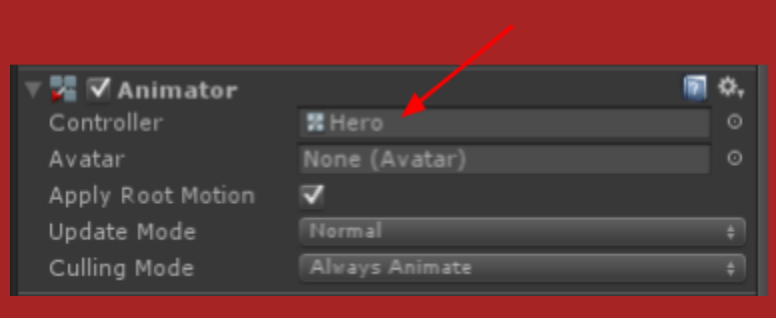
Explanation - Animator Controller

- We already have our **Animator Controller** created and attached to our **Hero** GameObject!
- This was made for us by **Unity** when we created our **Player firing Idle** animation

Check this

- Select the **Hero** GameObject in the **Hierarchy**
- Check the **Animator Component** has our **Hero Animator Controller** in the **Controller** inlet

Check our Hero Animator Controller is here



Useful links

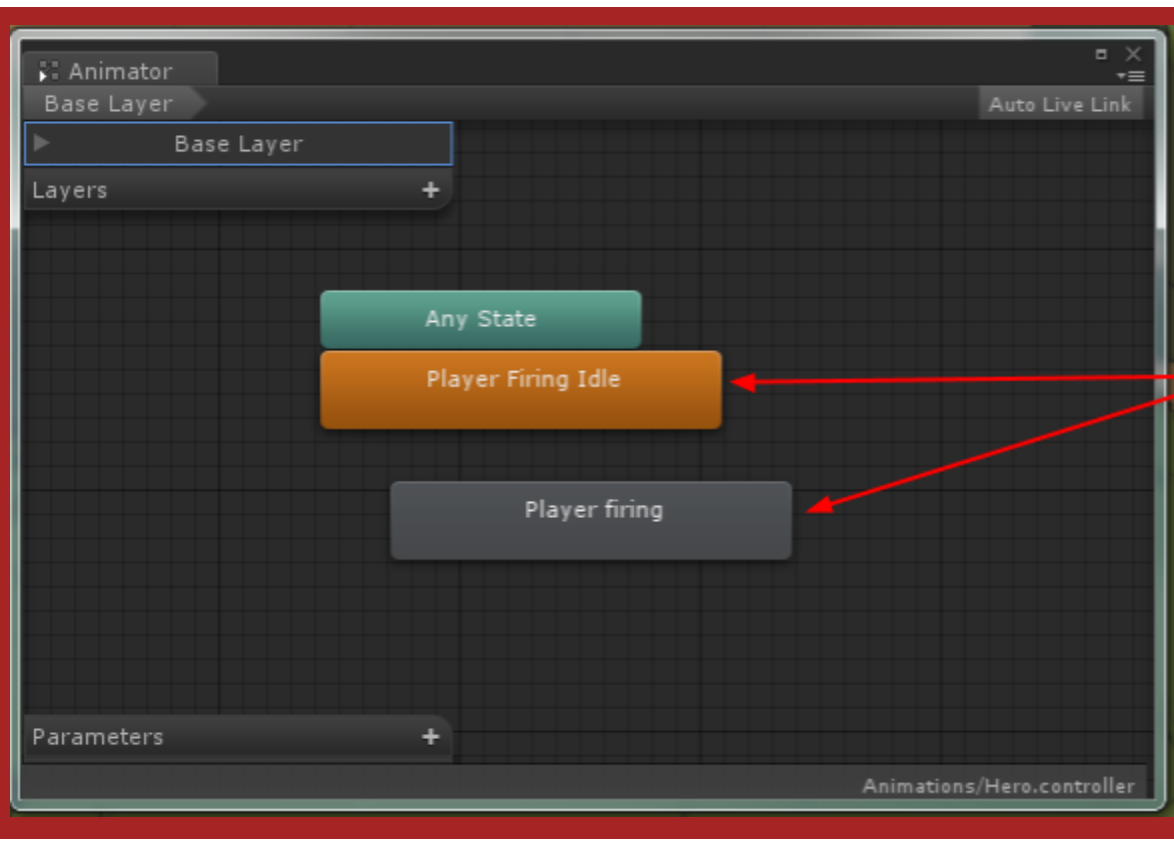
- More information about **Animator Component** Animator Component

Do this

- In the **Hierarchy**, select the **Hero** GameObject
- Open the **Animator view**
  - **Top menu**: **Window** > **Animator**

Check this

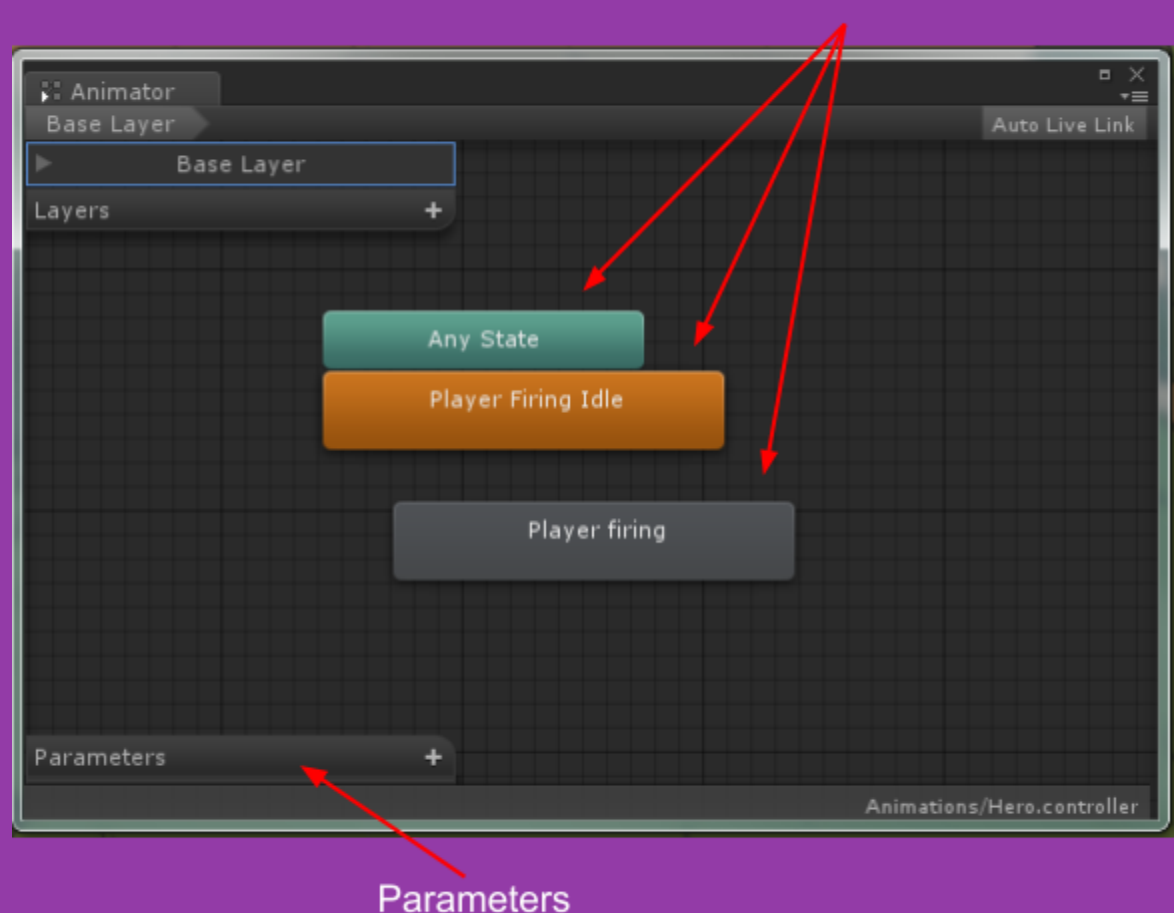
- The Animator view looks like this:



You will see your both the Player firing animations here

Explanation - Animator view

- There are 2 things we will focus on in the **Animator view**
- The **Animation States**
- The **Parameters**



Explanation - Animation States

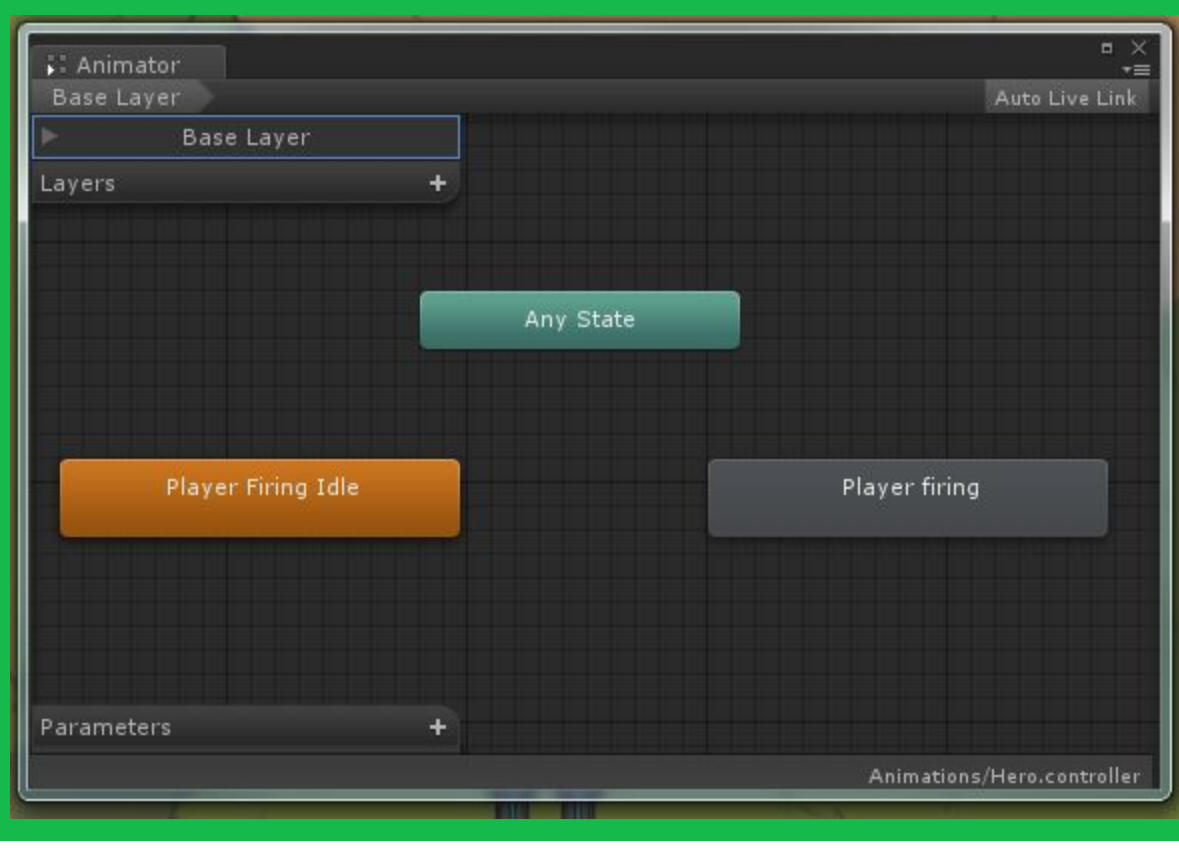
- The Animations we created earlier are called **States** in the **Animator view**
- This is because the Animator view works like a State machine
- Our simple State Machine is like a light switch
  - If the fire button is down - go to the **Player firing State**
  - If the fire button is NOT down - go to the **Player firing Idle State**

Useful links

- More information about **Animator States** — Animator States
- More information about **Finite State Machine** — Finite State Machine

Do this

- In the **Animator view**, drag the **States** so the **Player firing Idle** is on the **LEFT** and the **Player firing** is on the **RIGHT**
- This will make it easier to see our State Machine flow

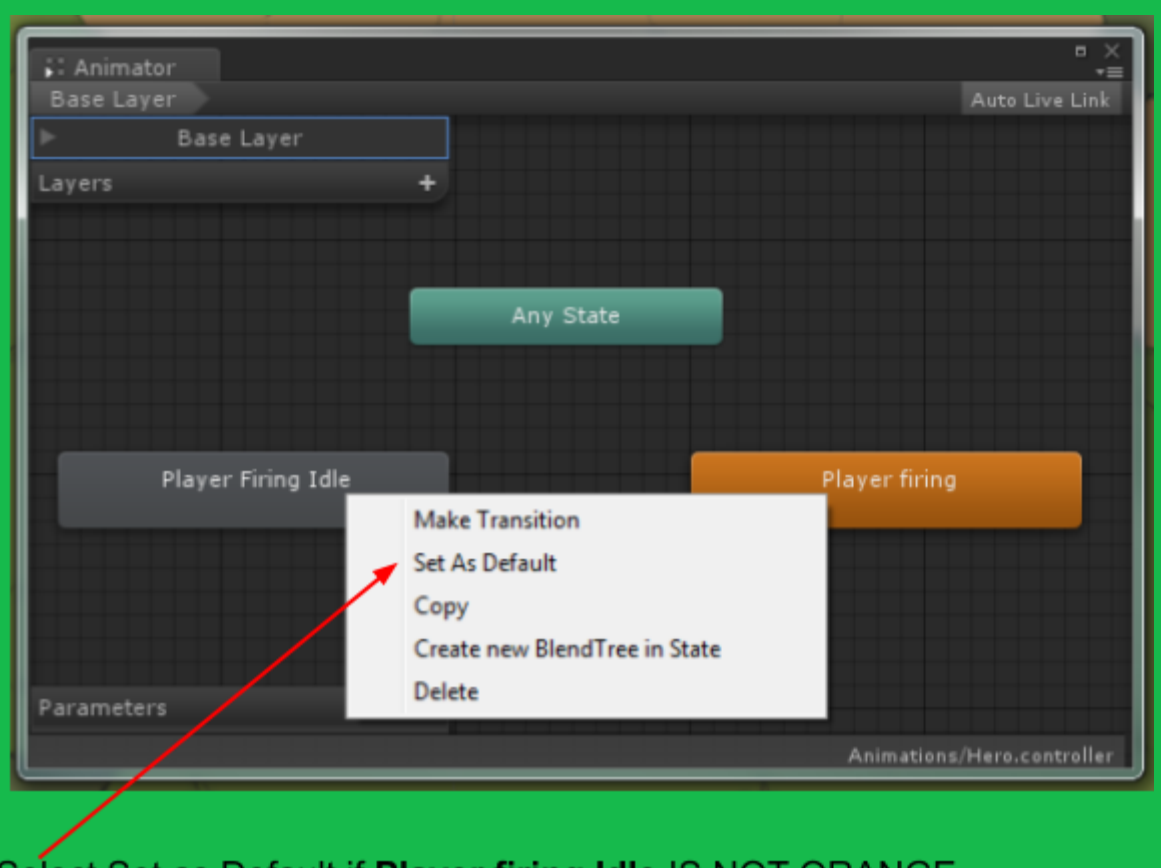


Explanation - Default Animation State

- Note in the picture above, the **Player firing Idle State** is coloured **orange**, while the **Player firing State** is **grey**
- This means the **DEFAULT State** when the game starts will be the **Player firing Idle State**
- We want to keep it like this!
- If yours is different, do the following:

Do this

- If your **Player firing Idle State** is **not orange**, right click on it
- Select **Set as Default**



Select Set as Default if **Player firing Idle** IS NOT ORANGE

Explanation - Parameters

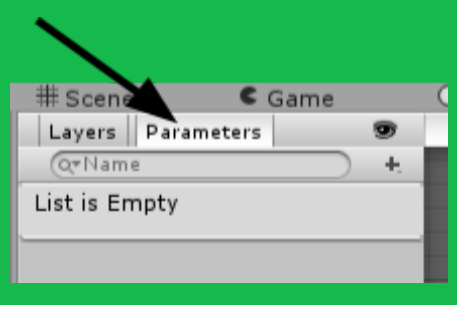
- In the **Bottom Left** of the **Animator view** are the **Parameters**
- These are values we can use in scripts to control which State we are in
- The type of values we can use are
  - **Float** (decimal number)
  - **Int** (whole number)
  - **Bool** (true/false)
  - **Trigger** (a special custom type)
- We will only need a **Bool** for our Player firing States

Useful links

- More information about **Animation Parameters** Animation Parameters

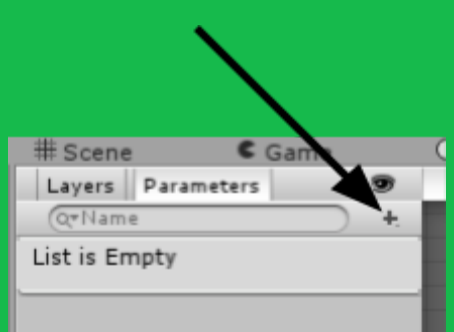
Do this

- In the **Animator view**, select the **Parameters** tab



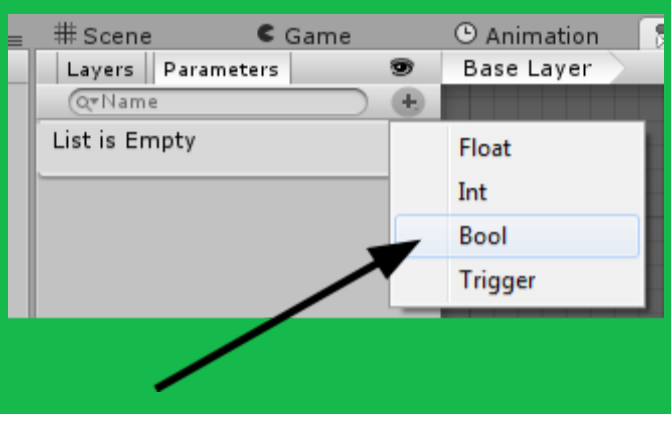
Do this

- Click the **plus button** (+) in the **Parameters** section



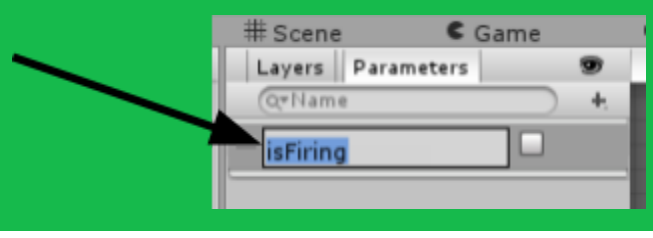
Do this

- Select **Bool**



Do this

- Name the new Parameter **isFiring**
- Leave it **unticked**



Explanation - Transitions

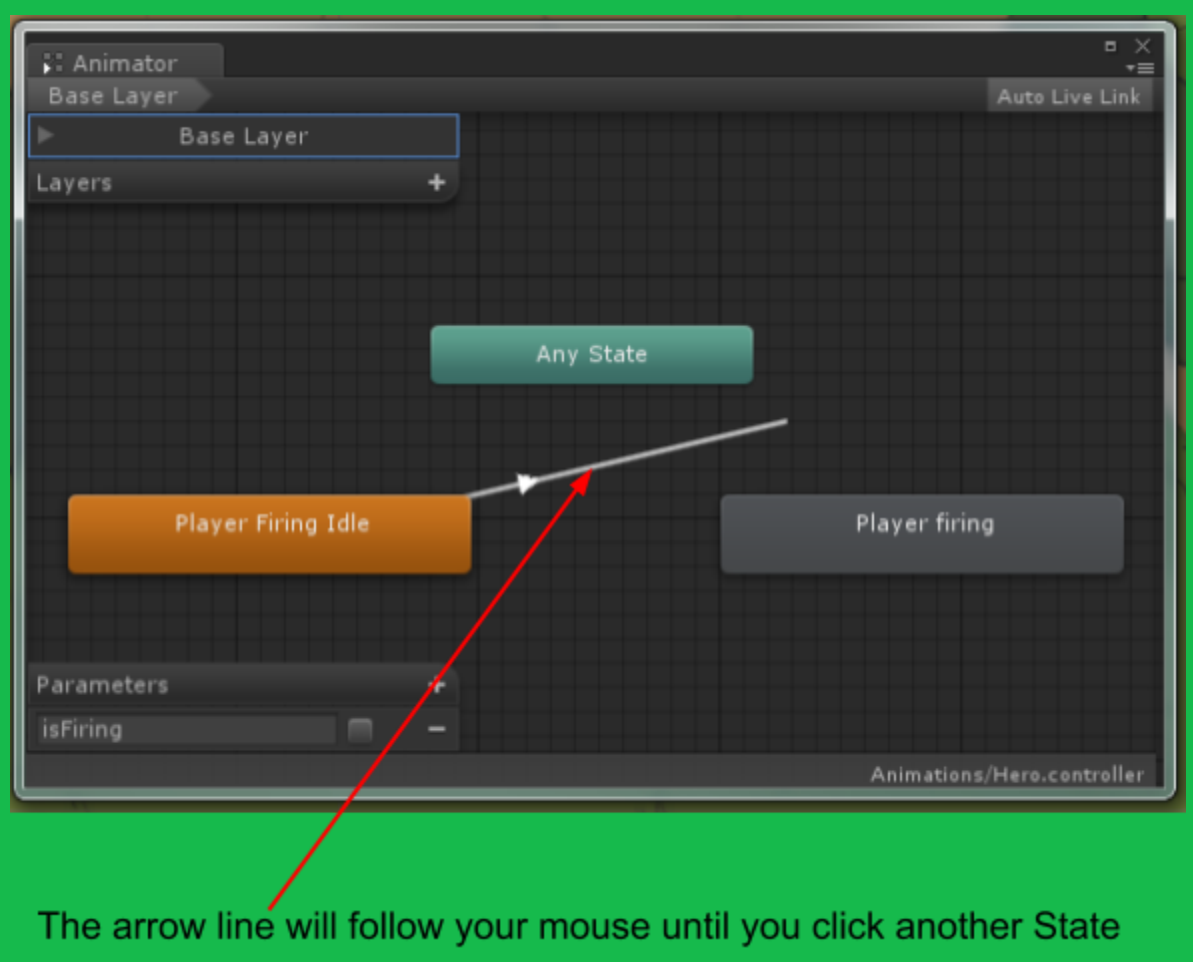
- A Transition is what happens when you change from one state to another
- In our case, we want to **Transition** FROM the **Player firing Idle State** TO the **Player firing State**
- The **Transition** will allow this to happen, based on a **Condition**
    - Hint: This is where we get to use our **isFiring** Parameter!

Useful links

- Learn more about **Animation transitions** [Animation transitions]

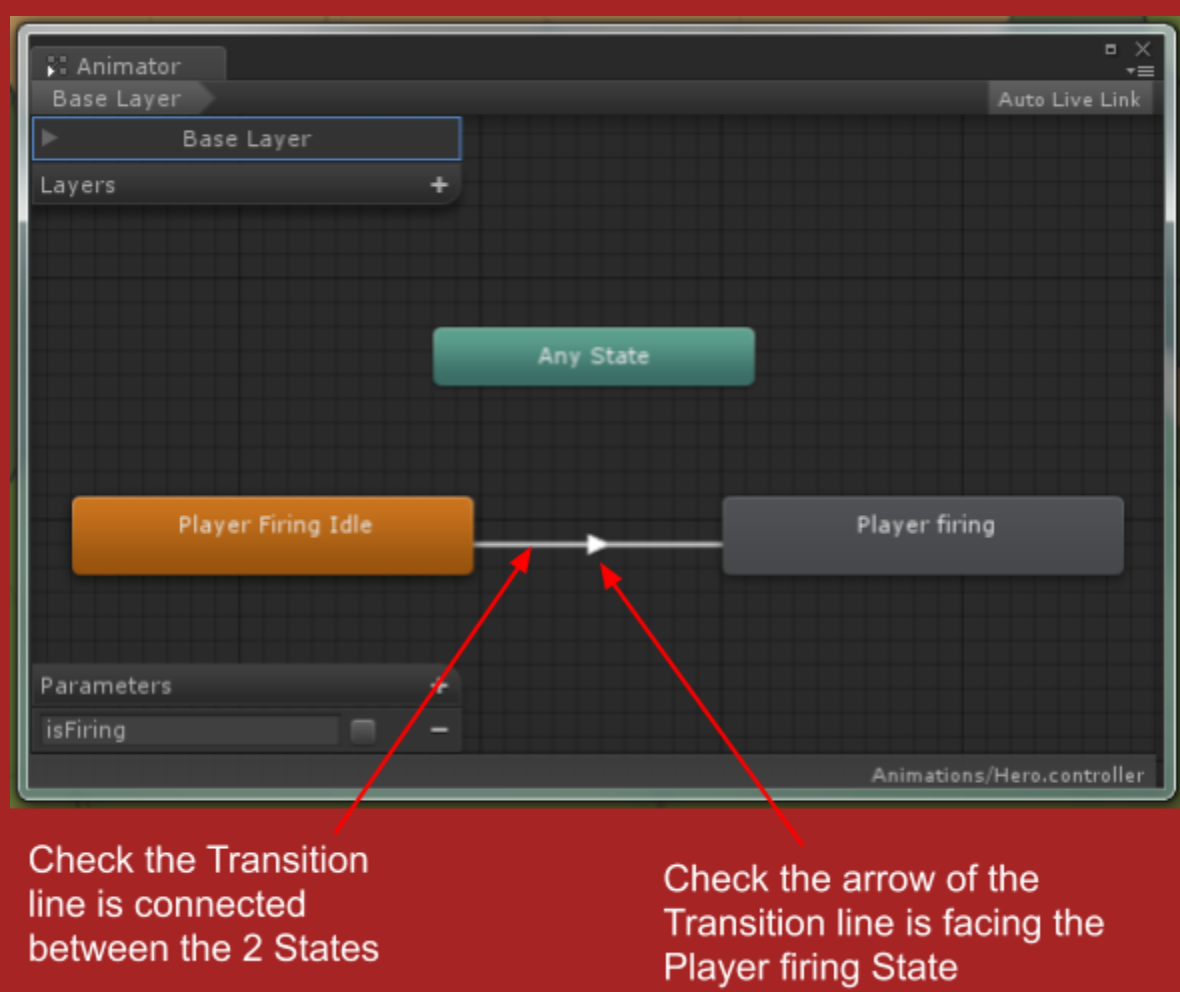
Do this

- Right click the **Player firing Idle State**
- Select **Make Transition**
- The Arrow line will follow your mouse pointer until you click!
- Click on the **Player firing State** to connect the **Transition**



Check this

- The **Transition** is **connected** between the **Player firing Idle State** and the **Player firing State**
- The **arrow** in the middle of the **Transition** is facing towards the **Player firing State**

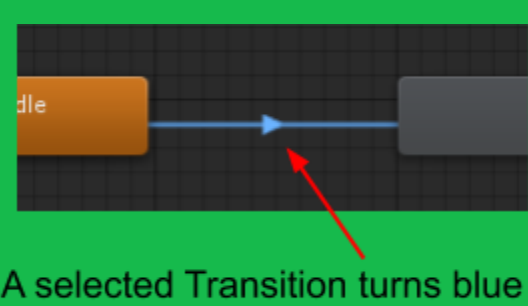


Explanation - Deleting Transitions

- If you connect the Transition up the wrong way, simply do the following
- click on the Transition to select it
    - It will turn blue
- Press delete!

Do this

- Click on the **Transition** to select it
- It will turn **blue** when **selected**

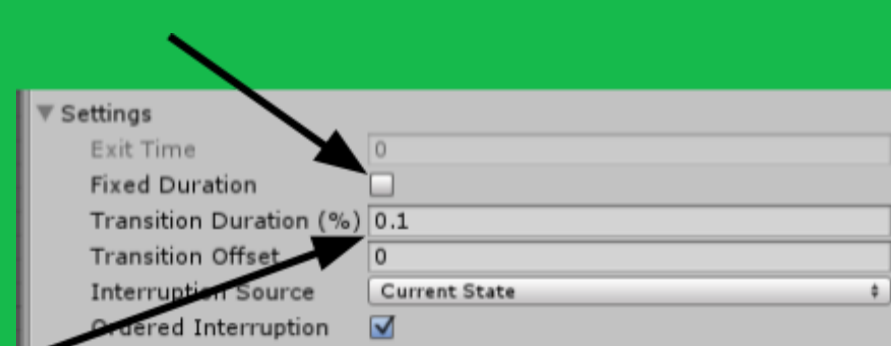


Do this

- In the **Inspector**, untick the **Has Exit Time** box

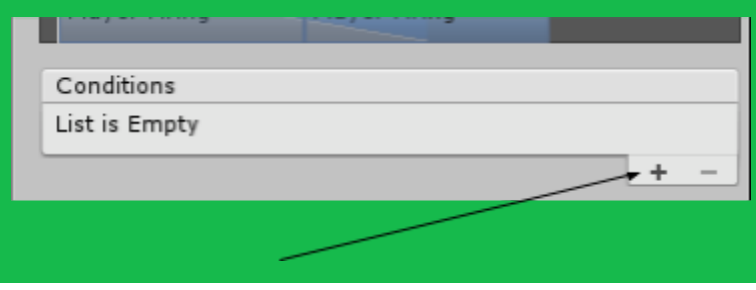
Do this

- In the **Inspector**, open the **Settings** section
- Untick the **Fixed Duration** box
- Set **Transition Duration (%)** to **0.1**



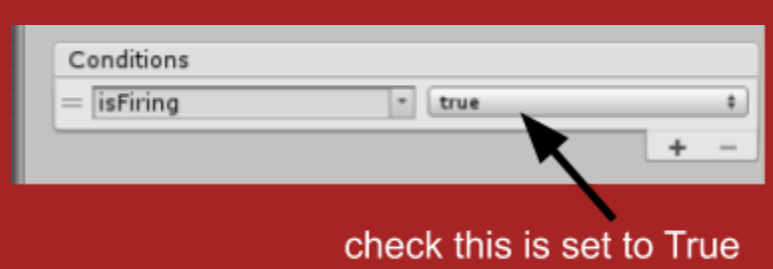
Do this

- In the **Inspector**, find the **Conditions** section at the bottom
- Press the Plus button (+) to create a new condition



Check this

- Check the value for **isFiring** is **True**



Explanation - A Transition back

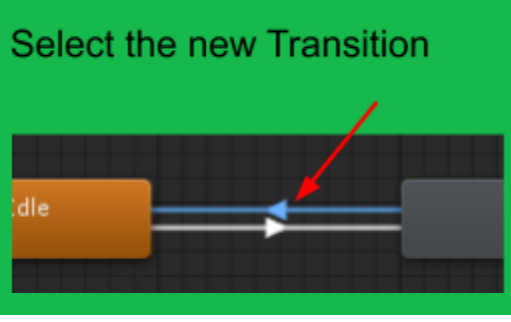
- Now we have a **Transition** going TO the **Player firing State**, we need a **Transition** BACK to the **Player firing Idle State**
- We will go over the same steps as before to create the **Transition**
- The isFiring value will be flipped to False again

Do this

- Right click the **Player firing State**
- Select **Make Transition**
- The Arrow line will follow your mouse pointer until you click!
- Click on the **Player firing Idle State** to connect the **Transition**

Do this

- Click on the new **Transition** to select it
- It will turn **blue** when **selected**

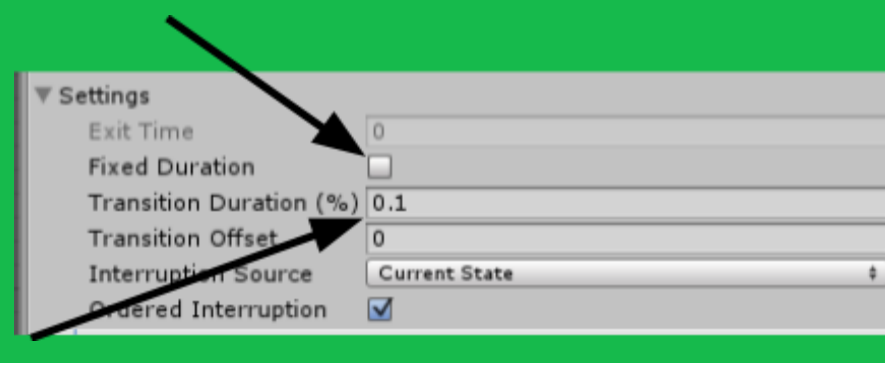


Do this

- In the **Inspector**, untick the **Has Exit Time** box

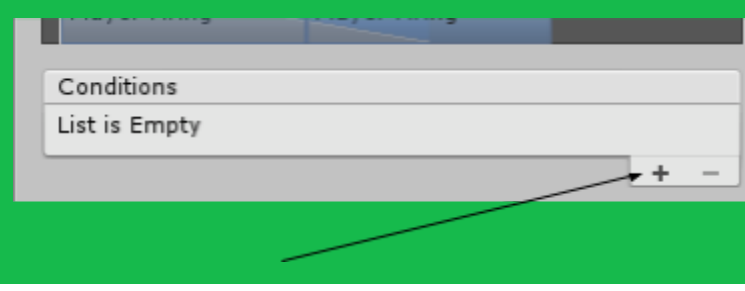
Do this

- In the **Inspector**, open the **Settings** section
- Untick the **Fixed Duration** box
- Set **Transition Duration (%)** to **0.1**



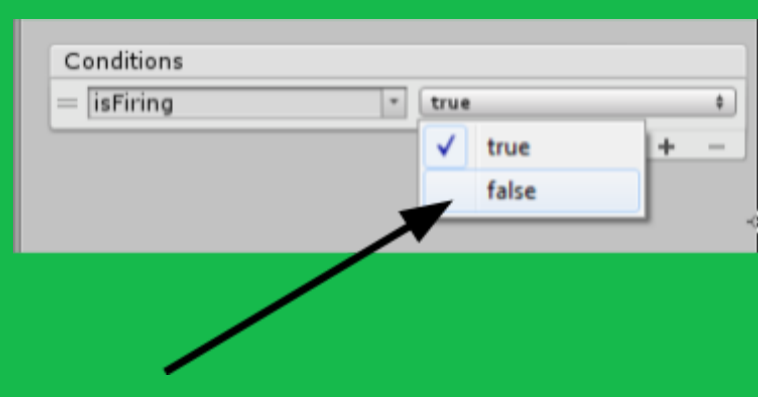
Do this

- In the **Inspector**, find the **Conditions** section at the bottom
- Press the Plus button (+) to create a new condition



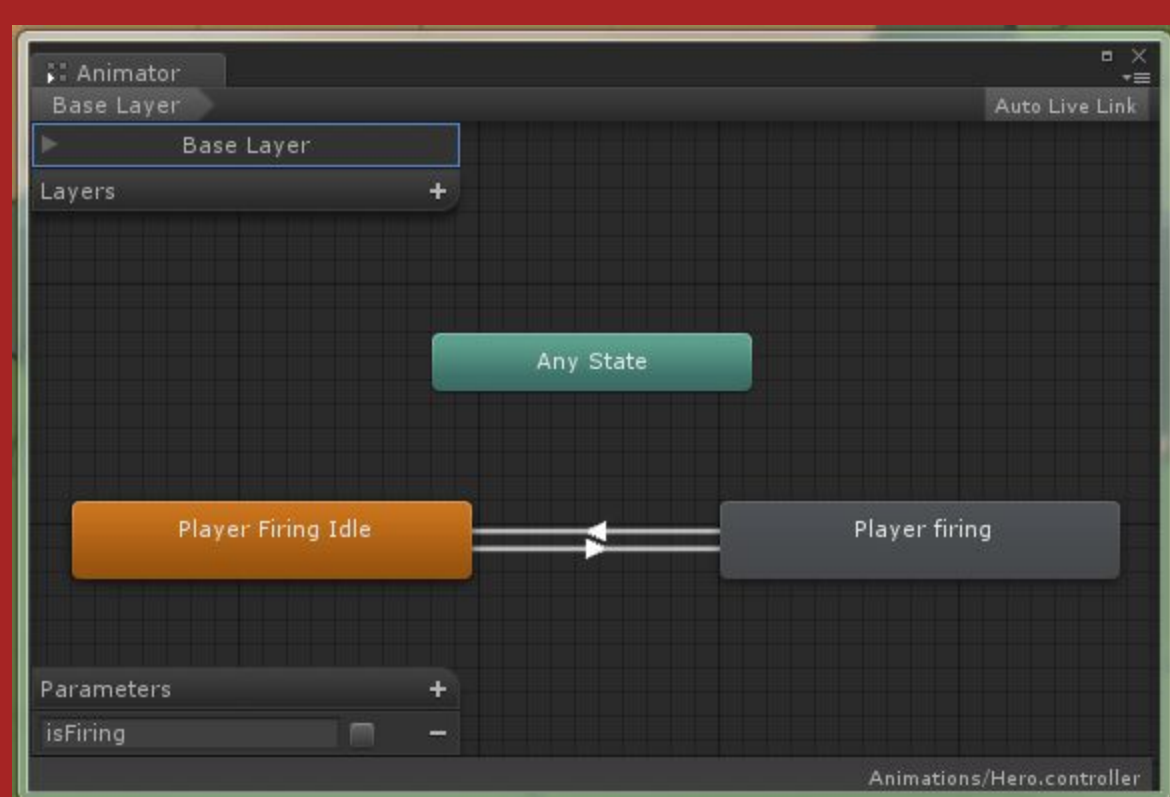
Do this

- Set the **value** for **isFiring** to **False**



Check this

- Our Animator Controller should now be set up for the shooting animations!
- Check your Animator view looks like this:



Task 2. Player script

Explanation

- We will create a script for the gun firing animation to play when the mouse is pressed

Do this

- In the **Scripts** folder of the **Project view**, create a new script
- Name the script **Player**
- Drag the **Player** script onto the **Hero** Gameobject in the **Hierarchy**

Do this

- Type out this code into your script file
- Make sure your code is ***EXACTLY*** the same!

```
using UnityEngine;

public class Player : MonoBehaviour {

    private Animator gunAnim;

    private void Start () {
        gunAnim = GetComponent<Animator>();
    }

    private void Update(){
        if(Input.GetMouseButton(0))
        {
            gunAnim.SetBool("isFiring", true);
        }
        else
        {
            gunAnim.SetBool("isFiring", false);
        }
    }
}
```

Explanation - gunAnim

- We will control the **gun animations** using the **gunAnim** property
- **gunAnim** is a **reference** to the **Animator** Component attached to the **Hero** GameObject
- **gunAnim** is a type of **Animator**
- **gunAnim** is a **private property**, so it is not **editable** in the **Unity Editor**

```
private Animator gunAnim;
```

Explanation - Our Start method

```
private void Start ()
{
    gunAnim = GetComponent<Animator>();
}
```

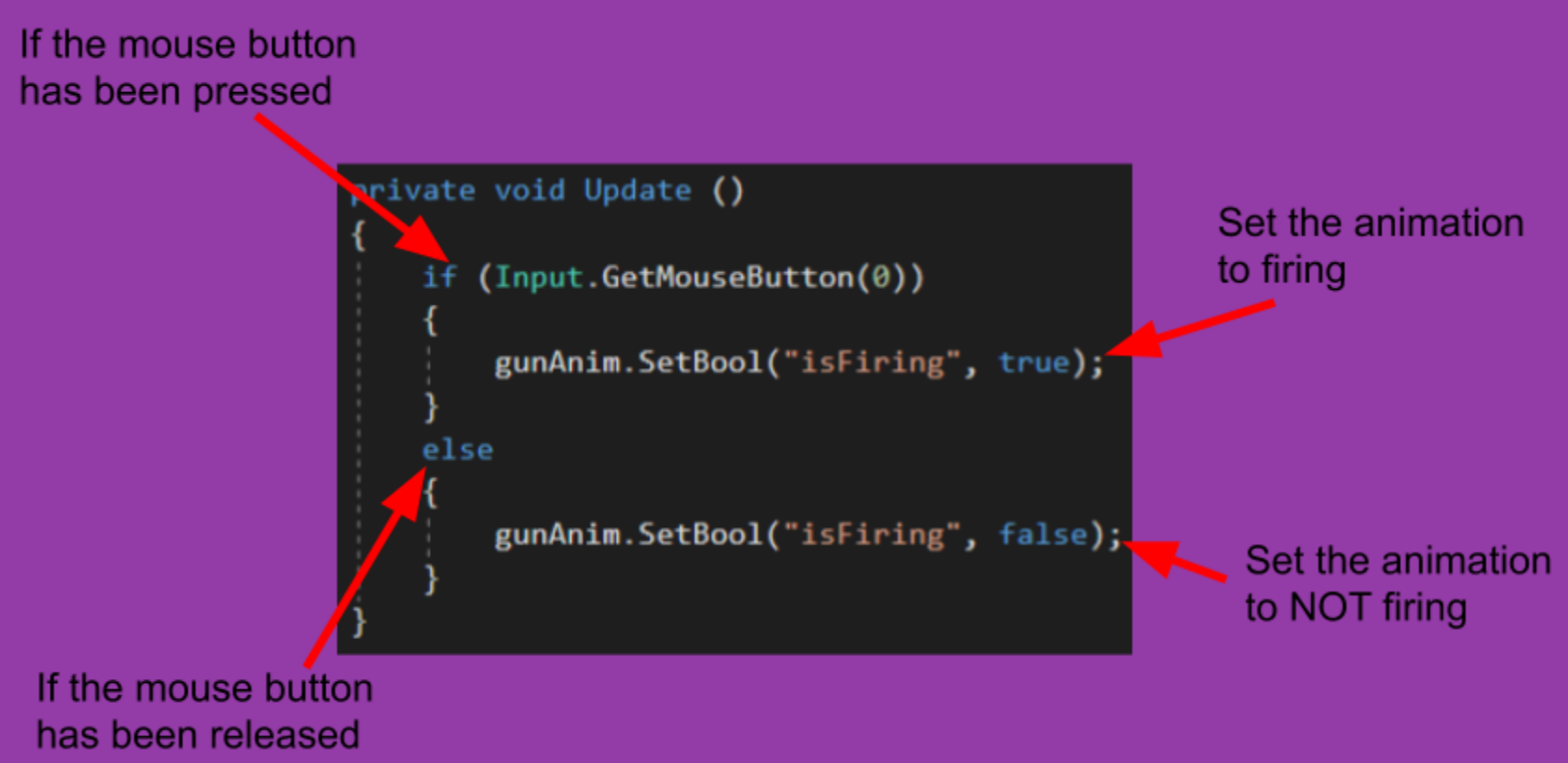
Store the gun animator component to use later

Explanation - Our Update method

- Our custom Update method code looks like this

```
private void Update(){
    if(Input.GetMouseButton(0))
    {
        gunAnim.SetBool("isFiring", true);
    }
    else
    {
        gunAnim.SetBool("isFiring", false);
    }
}
```

Explanation - Our custom Update code



Explanation - Line 1

- Check the left mouse button is down

```
private void Update(){
    if(Input.GetMouseButton(0))
    {
        gunAnim.SetBool("isFiring", true);
    }
    else
    {
        gunAnim.SetBool("isFiring", false);
    }
}
```

Useful links

- More information about **GetMouseButtonDown** GetMouseButtonDown - Scripting

Explanation - Line 2

- Set the **isFiring** bool parameter in our Animator to **true**
- This will start the **Player firing** animation in our Animator, as we set up earlier

```
private void Update(){
    if(Input.GetMouseButton(0))
    {
        gunAnim.SetBool("isFiring", true);
    }
    else
    {
        gunAnim.SetBool("isFiring", false);
    }
}
```

Useful links

- More information about **Animator** — Animator - Scripting reference
- More information about **Animator SetBool** — Animator SetBool - Scripting reference

Explanation - Line 3

- Set the **isFiring** bool parameter in our Animator to **false**
- This will start the **Player firing Idle** animation in our Animator, as we set up earlier

```
private void Update(){
    if(Input.GetMouseButton(0))
    {
        gunAnim.SetBool("isFiring", true);
    }
    else
    {
        gunAnim.SetBool("isFiring", false);
    }
}
```

Useful links

- More information about **Animator SetBool** — Animator SetBool - Scripting