

# **Table of Contents**

## **Setting up the Player Controller**

[Player Controller](#)

[Player Noise](#)

[Creating the Bullet](#)

[Entity](#)

[Controls](#)

## **Setting up the Main Menu and Camera**

[Main Menu](#)

[Game Camera](#)

[Customizing the HUD](#)

## **Setting up the AI**

[AI](#)

[Waypoints](#)

## **Setting up Spawners**

## **Setting up Level Managers**

## **Setting up Animations**

[Movement \(idle, walk, run\)](#)

[Melee Attack](#)




[Shooting](#)

[Reloading](#)

# Setting up the Player Controller and Camera

## Player Controller

The player controller is a lot simpler than it seems, as it automatically adds most of the required values. Simply dragging the script onto any game object will add a character controller and an Entity script explained below.

▼  <b>Character Controller</b>	
Slope Limit	45
Step Offset	0.3
Skin Width	0.08
Min Move Distance	0.001
Center	X 0 Y 0
Radius	0.5
Height	1
▼  <b>Entity (Script)</b>	
Script	Entity
Max Health	10
Death Prefab	None (Game Object)
▼  <b>Player Controller (Script)</b>	
Script	PlayerController
<b>Animation</b>	
Anim	None (Animator)
Moving Anim	<input type="checkbox"/>
Shooting Anim	<input type="checkbox"/>
Reloading Anim	<input type="checkbox"/>
<b>Movement Variables</b>	
Rotation Speed	450
Walk Speed	4
Run Speed	8
Acceleration	5
<b>Stealth Variables</b>	
Noise Object	None (Transform)
Walking Noise Size	2.5
Running Noise Size	5
Noise Change Speed	6
<b>Combat Variables</b>	
Gun Type	Auto
► Fire Points	
Bullet	None (Rigidbody)
Bullet Speed	1000
Bullet Damage	2
Shots Per Minute	60
Max Ammo	300
Ammo Per Mag	30

### Animation variables:

If you have no animations, ignore these variables.

**Anim:** the animator containing the animations for the player.

**Anim bools:** ticking these will tell the code to set the animation parameters covered later.

### Movement variables:

**Rotation speed:** the speed at which the player can rotate.

**Walk/Run speed:** the walking and running speed of the character.

**Acceleration:** how quickly the object speeds up to the walk or run speed.

### Stealth variables:

If you do not want to use stealth, you can ignore these variables, leaving the noise object blank.

**Noise Object:** the object that will scale out to form a “sound area”.

**Walking/Running Noise Size:** the radius the noise object will grow to when walking or running (note that these values must be lower than the respective speeds, and has been limited).

### Combat variables:

**Gun Type:** setting whether the gun fires automatically or on a single shot basis.

**Fire Points:** the transforms that a bullet will spawn at every time the player shoots, there can be as many fire points as you like.

**Bullet:** the Rigidbody that holds the bullet script, this will be fired from each fire point.

The remaining combat variables are fairly self-explanatory.

## Player Noise

This game object will represent how far sound travels from your player while they move, and it is simple to create:

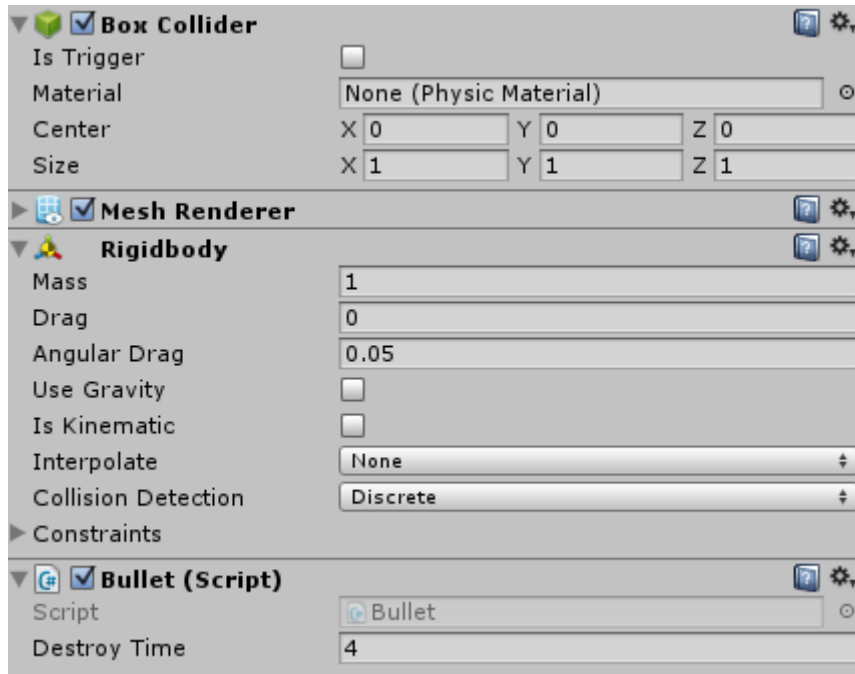
- Create a cylinder as a child of the player.
- Add the Player Noise script, which will then add a Rigidbody.
- Untick the “Use Gravity” bool on the Rigidbody.
- Remove the Capsule Collider and replace it with a Mesh Collider.
- Tick the “Convex” and “Is Trigger” bools on the Mesh Collider.
- Scale the Noise Object to 0 on the X axis, 0.01 on the Y, and 0 on the Z.
- Drag this object into the “Noise Object” variable of the Player Controller.

## Creating the Bullet

Creating the bullet is incredibly simple:

- Create a bullet object (a cube, a sphere, anything).
- Add the bullet script, this will add a Rigidbody.
- Untick the “Use Gravity” box.
- Make this object a prefab.
- Drag the prefab into the “Bullet” box.

Your inspector should look like this for the bullet object:



The “**Destroy Time**” variable is the length of time the bullet travels for before destroying itself if it does not hit anything.

## Entity

To set the max health of this object simply type in the desired value into the “**Max Health**” variable. The “**Death Prefab**” variable can be left blank if you just want the object to delete itself upon death, however placing a prefab into the box will spawn that prefab upon death.

## Controls

The controls for the player are as follows:

### **Movement:**

W = up

A = left

S = down

D = right

Left Shift = sprint

### **Combat:**

Left Click = fire

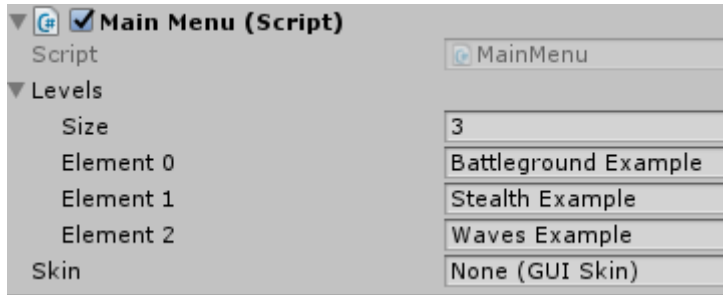
R = reload

### **Misc:**

Esc = pause

## Setting up the Main Menu

The main Menu will by default create the Start, Level Select, and Quit buttons. The start function will play the first level in the “Levels” list, and the level select will bring up every level in the list.



**Levels:** Simply typing the names of your levels and adding them to the build settings will allow the level select buttons to load them.

**Skin:** the GUI skin to use when showing the HUD. Note that the pre-set HUD uses GUI

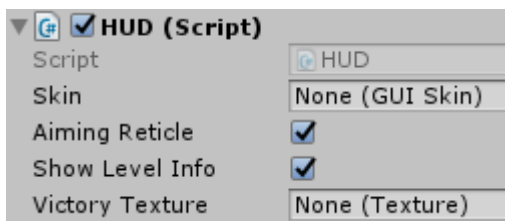
cubes, and therefore you must open the HUD script and specify your custom style. How to implement this is explained in the [Unity API](#) under the “Working with Skins” section.

## Game Camera

Setting up the camera can be done in three easy steps:

- Rotate the camera to 90 degrees on the X axis so it is above the player looking down.
- Drag the Camera Follow script onto the camera.
- If you want a HUD, drag the HUD script on as well.

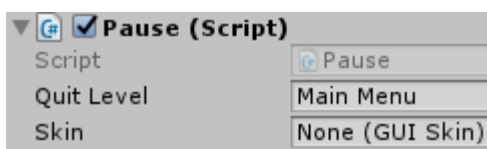
## Customizing the HUD and Pause Menu



**Show Reticle:** ticking this bool will show the reticle as opposed to the mouse in the scene.

**Show Level Info:** ticking this bool will show the level information at the top of the screen.

**Victory Texture:** This is the texture that will be stretched across the screen upon completion of the level.



**Quit Level:** this string will tell the script which level to load when the “Quit” button is pressed.

**Skin:** see the “Skin” variable in the “[Setting up the Main](#)

[Menu](#)” section

# Setting up the AI

## AI

The Generic AI script could seem daunting due to the sheer number of variables; however, it is far simpler than it appears. Dragging the script onto a game object will add the Entity script, a box collider (if there is none present) and the Nav Mesh Agent component. Note that

The screenshot shows the 'AI (Script)' inspector with the following settings:

- Setup variables**
  - Combat Mode: Seek And Destroy
  - Target Tags: (empty)
  - Current Target: None (Transform)
  - Troop Type: Ranged
  - Mask: Mixed ...
  - Kill Cone: -5
- Debugs**
  - Show Range Sphere: ☐
  - Show Vision Cone: ☐
  - Show Kill Cone: ☐
- Animation variables**
  - Has Animations: ☒
  - Anim: RedAnimRanged ( )
  - Moving Anim: ☒
  - Shooting Anim: ☒
  - Reloading Anim: ☒
- Movement variables**
  - Walk Speed: 3
  - Run Speed: 6
  - Look At Target Distance: 12
  - Stopping Distance: 10
- Is A Guard**: ☒
- Patrolling variables**
  - Wait At Point Time: 2
- Patrol Targets**
  - Size: 0
- Sight variables**
  - Vision Cone: 5
  - Detect Distance: 20
  - Chasing: ☐
- Ranged Variables**
  - Fire Points: (empty)
  - Bullet: Bullet (Rigidbody)
  - Bullet Speed: 1000
  - Bullet Spread: 10
  - Bullet Damage: 2
  - Shots Per Minute: 240
  - Ammo Per Mag: 10

### Setup variables:

**Combat Mode:** chase on sight means the AI will only attack enemies it can see, whereas seek and destroy will cause the AI to hunt any enemy within its "Detect Distance", ignoring the "Vision Cone".

**Target Tags:** the tags that this AI considers enemies, for example anything tagged as "Red".

**Current Target:** the current closest target of this AI. Note that the AI will only attack this target if it meets the requirements stated by "Combat Mode", "Detect Distance" and "Vision Cone".

**Troop Type:** setting whether it is a melee or ranged troop.

**Mask:** sets the layers that the AI can see, it is advisable to tick everything aside from "Ignore Raycast".

**Kill Cone:** the angle that a bullet must hit this object to cause an instant kill, -5 is a 45-degree angle behind the AI, 10 being directly behind (if set to -10 there will be no instant kill). This works best if the box collider is the same width and length as the nav mesh agent.

### Debugs:

**Show Range Sphere:** this will create a sphere around the object in editor to show you their range in game.

**Show Vision Cone:** this will draw two lines creating a cone to represent the angle at which the AI will see its target.

**Show Kill Cone:** this will draw a similar cone representing the angle at which a bullet must hit this object to cause an

instant kill.

### **Animation variables:**

If you have no animations, ignore these variables.

**Has Animations:** ticking or unticking this bool will show the animation variables.

**Anim:** the animator containing the appropriate animations for this object.

**Anim bools:** ticking these will tell the code to set the animation parameters covered [later](#).

### **Movement variables:**

**Walk/Run Speed:** the walking and running speed of the AI, it will only walk if it is a guard, if it is fighting it will run.

**Look at Target Distance:** the distance from the AI's target at which the AI will turn to look at their target.

**Stopping Distance:** the distance from the AI's target at which the AI will stop and attack.

**Is a Guard:** ticking this will make the AI follow a path of waypoints (set in the Patrol targets array). This will also show or hide the following movement variables.

**Wait at Point Time:** the number of seconds the AI stops at each waypoint.

**Patrol Targets:** the AI will cycle through these positions and once it reached the final waypoint, it will target the first waypoint again, perfect for a circular route or simply patrolling a corridor. The creation of these points is [covered later](#).

### **Sight variables:**

**Vision Cone:** the angle that the enemy can see a target, 10 is directly ahead, -10 is directly behind, 0 is to either side.

**Detect Distance:** the range at which the AI will begin chasing/looking for their target.

### **Melee variables:**

If troop type is ranged, these variables will be hidden.

**Melee Cooldown:** how long the AI must wait between attacking again.

**Melee Damage:** how much damage the melee attacks deal to enemies.

## Ranged variables:

If troop type is melee, these variables will be hidden.

Most of these are covered in the Player Controller section, the only new one being the “**Bullet Spread**”. This is how many degrees of bullet spread there is a possibility for, making the AI have varying accuracy.

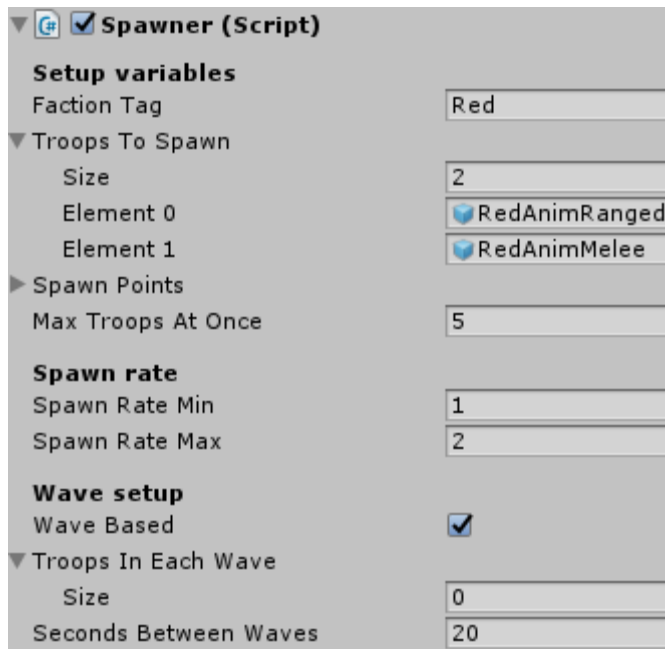
## Waypoints

Adding waypoints is a fairly simple process:

- Create a cube and move it to the desired patrol point.
- Drag the Waypoint script onto the cube.
- Again a Rigidbody will be added, again untick the “Use Gravity” box.
- Tick the “Is Trigger” on the collider.
- Drag the waypoint into the Patrol Targets variable of the AI you want to follow the path.

## Setting up Spawners

The Spawner script works by storing the different enemy prefabs and the different spawn points, and spawning a random prefab at a random spawn point within their respective arrays. The spawners will spawn AI with the “Seek and Destroy” combat mode and an infinite “Detect Distance”.



### Setup variables:

**Faction Tag:** the tag of the troops that will be spawned (note that this does not set their tags, but will use this to determine how many troops are active in the scene).

**Troops to Spawn:** an array of the different troops this spawner will spawn (these will be randomly chosen).

**Spawn Points:** the possible transforms at which a troop may spawn (these will be randomly chosen).

**Max Troops at Once:** the number of troops can be in the level at any one time (the troops' tag must be the same as the faction tag).



## Spawn rate:

These variables determine the number of seconds between spawning an enemy, **min** is the smallest length of time it will wait and the **max** is the greatest length of time it could wait (note the spawner could activate at any point between these two values).

## Wave Setup:

**Wave Based:** ticking this bool will show the wave setup variables.

**Troops in Each Wave:** this array contains the number of troops the spawner will spawn in each wave before stopping and waiting for the wave to be over.

**Seconds Between Waves:** the number of seconds the spawner will wait once there are no more troops remaining before starting the next wave.

## Setting up the Level Manager

The level manager is designed to make it as easy as possible for there to be a final goal (and an end) to the level.



**Type of Level:** this will decide what the final goal of the level is. If set to “Battleground”, the goal will be to kill the number specified in “Target kills”; if set to “Stealth” the goal will be to dispatch every AI tagged as “Red”, and if set to “Waves” it will simply wait until all the waves are completed, and then complete the mission.

**Checkpoint:** the respawn transform of the player.

**Target enemies / Enemies Killed:** these are the number of kills the player’s team must get, and how close they are to this goal.

**Victory Texture:** this texture that will be stretched across the screen upon completion of the level.

**Level to Load:** the name of the level to load after the current level is completed.

## Changing the target tag of the stealth level

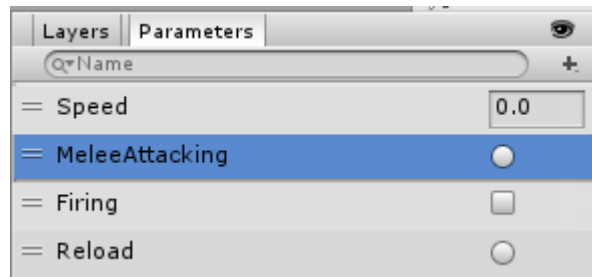
To change the target tag of the stealth level, or to add more tags, you must open the **Level Manager** script located in Assets>Top Down Shooter Pack>Scripts>Managers. Once opened, you must find the following line of code (Line 36):

```
else
{
    if(typeOfLevel == LevelType.Stealth && GameObject.FindGameObjectsWithTag("Red").Length == 0 && !won){
        StartCoroutine (Winning ());
    }
}
```

Once found, you must either Change “Red” with your tag to change the tag looked for, or simply copy and paste “ && GameObject.FindGameObjectsWithTag(“Red”).Length == 0 ” directly next to itself and then replace the tag to add your own tags.

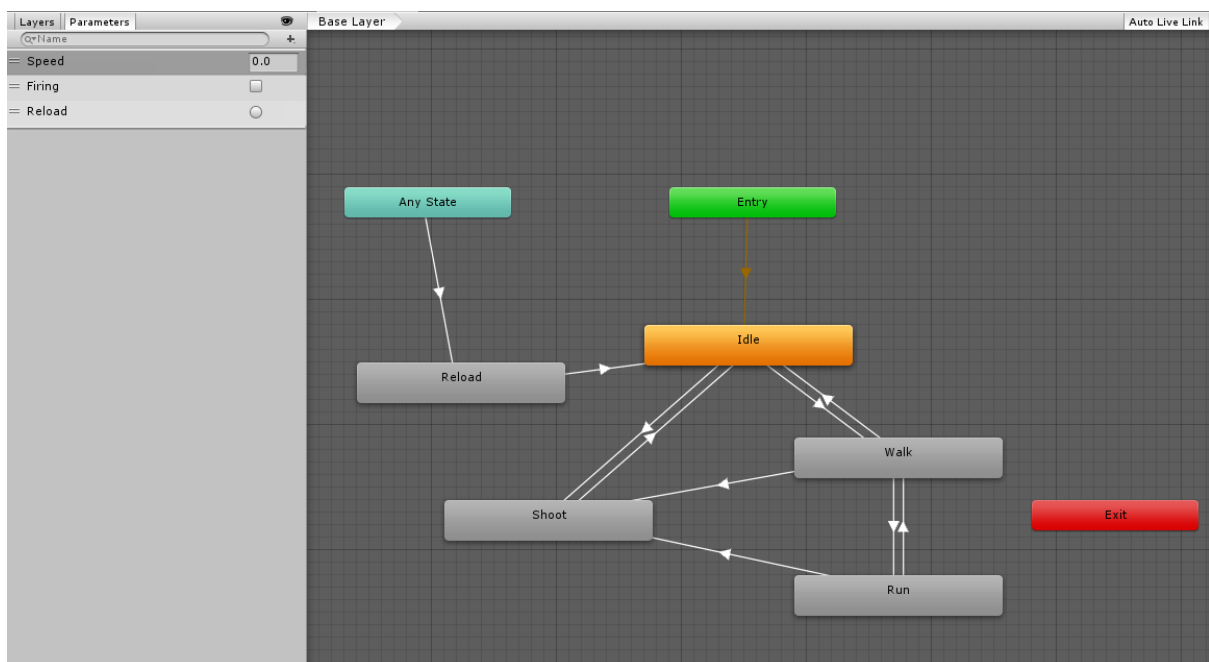
## Setting up Animations

Adding animations requires at least a basic understanding of how animators work in Unity. Beyond that, simply filling the “Anim” variable and ticking the animations will cause the code to search for a set of parameters of the names shown below:

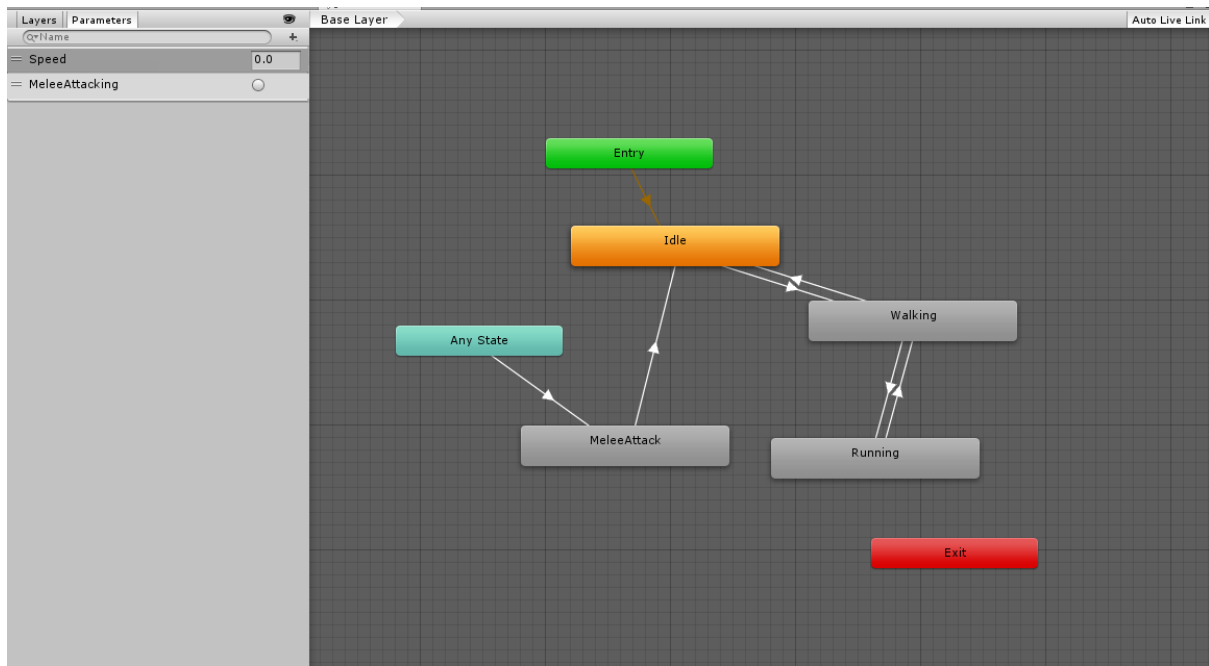


As you can see, the “**Speed**” parameter is a **float**, the “**MeleeAttacking**” and “**Reload**” parameters are **triggers**, and the “**Firing**” parameter is a **bool**.

### The Ranged AI animator example used in this pack:



## The Melee AI animator example used in this pack:



### **Movement animations (idle, walk, run)**

Ticking the Moving Anim bool will cause the Generic AI script to search for and set the “Speed” parameter.

#### **Idle**

- Set this animation as the “Layer default state” (done by simply right clicking on it).
- Make transitions from any animations that could lead into idle.
- Set the conditions to speed being less than 0.4 (this is a recommendation, but you may set it to any number).

#### **Walk**

- Once added into the animator, make transitions from any animation that could lead into this one.
- Set the conditions to the speed being greater than 0.4 but less than the running speed. Note that this animation will only play on guards following a waypoint path.

#### **Run**

- Again make the necessary transitions.
- Set the entry conditions to the speed being greater than the AI’s walk speed.

## **Melee attack animation**

Add your animation, and within the animator set the transition condition to “MeleeAttacking”. Note that if you tick the Melee Attack Anim bool you must do one of the following methods in order for the melee damage to be dealt:

### **Method 1 (guaranteed hit)**

- Create an animation event that calls the **MeleeHit()** function at the point in your animation the damage is dealt, this will mean the damage will be dealt no matter what.

### **Method 2 (can be avoided)**

- Add a collider (be it a mesh or box) to the melee weapon in the animation, it is advised to make this melee weapon a child of the AI object.
- Make this collider a trigger.
- Add the “DamageOther” script to the melee weapon.

## **Shooting animation**

As with the previous animation, add the animation and set the transition condition to “Firing”. Note that if you tick the Shooting Anim bool you must do the following for a bullet to spawn:

- Create an animation event within the animation that calls the **Shoot()** function at the frame intended for the bullet to spawn. Note that if you do not do this the shooting animation will play without spawning any bullets.

## **Reloading animation**

The transition condition for this animation would be “Reload”. If you tick the Reloading Anim bool you must do the following:

- Create an animation event that called the **FinishReload()** function at the end of the reload animation. Note that if you do not do this the reloading animation will play continuously without actually reloading.