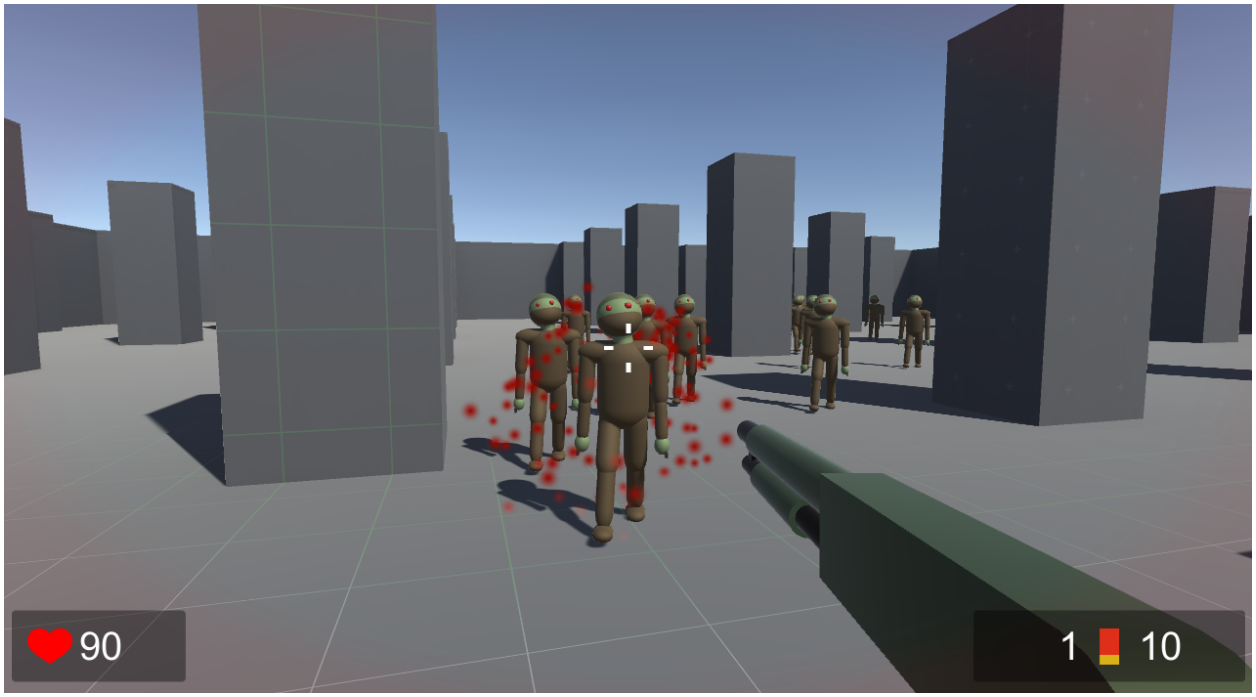
FPS Integration Tool (Version 1.1)

User Documentation

Thanks for downloading the Grey Gear FPS Integration Tool, this allows you to implement First Person Shooter (FPS) game mechanics into your Unity project by supporting the systems and functionality automatically. Therefore allowing you to focus on developing the assets to then apply to the tool.



Carefully read this document as it is vital for using this tool properly and without issues. For information on setting-up the project to play the provided demo scenes, go to the Set-Up the Project to Play the Demo Scenes section.

This product is designed for users who are familiar with using Unity, as a fundamental understanding is strongly recommended to avoid confusion.

If you are new to Unity, consider revising the tutorials from learn.unity.com.

Contents

Package Layout

Step-By-Step Guides

- Set-Up the Project to Play the Demo Scenes
  - Step 1
  - Step 2
  - Step 3
  - Demo Scene Gameplay
    - Controls
    - Shooting Range Demo
    - Zombie Room Demo
- Create A New Scene
  - Step 1
  - Step 2
  - Step 3
  - Step 4
- Create a Weapon from Weapon Files
  - Step 1
  - Step 2
  - Step 3
- Edit Weapon Attributes
  - Step 1
  - Step 2
- Edit Weapon Prefab
  - Step 1
  - Step 2

- Step 3
- Edit Weapon Animations
  - Step 1
  - Step 2
  - Step 3
  - Step 4
- Creating Weapon Collectables
  - Step 1
  - Step 2
  - Step 3
- Introducing Zombies to your Scene
  - Step 1
  - Step 2
  - Step 3
  - Step 4
  - Step 5
- Adjusting the Zombies
  - Speed
    - Step 1
    - Step 2
  - Damage To Player
  - Damage From Player
  - Health

WeaponSpace Animation Guidelines

- Terminology
  - Animation Tab
  - Animator Tab

        - Layers (Animation)
        - Hierarchy
    - General
    - Idle Animations
    - Fire Animations
        - Inconsistent Animation
        - Revolving Chambers
    - Reload Animations
    - Hip Animations
    - Aim Animations
    - Switch Animations
    - Run Animations
- ScriptableObjects
    - Weapon
    - WeaponCollection
    - Ammo
- Components
    - FPSPlayerController
    - WeaponSpace
    - CollectableObject
    - Grenade
    - Explosion
    - PlayerHealth
    - NonPlayerHealth
    - NonPlayerController
    - NonPlayerAudio
    - NonPlayerDamageInfliction

Create Options

Weapon Files

Layers (GameObject)

Intended use of the FPS Integration Tool

Package Layout

After importing the package into your Unity project, a folder named GG FPS Integration Tool should exist under the Assets folder.

The GG FPS Integration Tool folder contains the files used by the tool's systems.

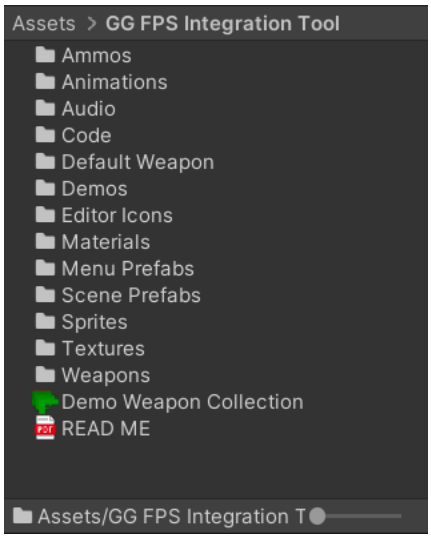
Beware that editing the folder or file names, directories or content may impair the functionality of the tool.

Avoid saving your files inside this folder, even if produced when using this tool. This allows you to delete the tool for reimporting if it malfunctions.

The contents of the GG FPS Integration Tool folder and subfolders involve:

- **Ammos** - Ammo ScriptableObjects involved in weapons.
- **Animations** - Animations used in Mouse, Movement Influence Layers and NonPlayer Zombies.
- **Audio** - Audio files used in weapons, movement and NonPlayer Zombies.
- **Code** - Script files used in the tool.
- **Default Weapon** - Used by the tool to generate template weapons which can be customised.
- **Demo Weapon Collection** - The WeaponCollection ScriptableObject utilised in the included scenes.
- **Demos** - Contains the demonstration scenes and their dependencies folders. They must be properly configured before running. For more information review the Set-Up the Project to Play the Demo Scenes section.
- **Editor Icons** - Icons applied to ScriptableObjects.
- **Materials** - Materials for the included weapons and other assets.
- **Menu Prefabs** - Prefabs often used within the Inspector tab fields.

- **READ ME** - This document.
- **Scene Prefabs** - Prefabs usually used as a feature of a scene.
- **Sprites** - Sprite images involved in the Heads-Up Display (HUD) within the Canvas GameObject.
- **Textures** - Holds texture files for Materials.
- **Weapons** - The included weapons used in the demonstration scenes which can also be used in your own scenes.



Step-By-Step Guides

Although these guides are sectioned, you should follow each of them in their current order if you are starting from scratch.

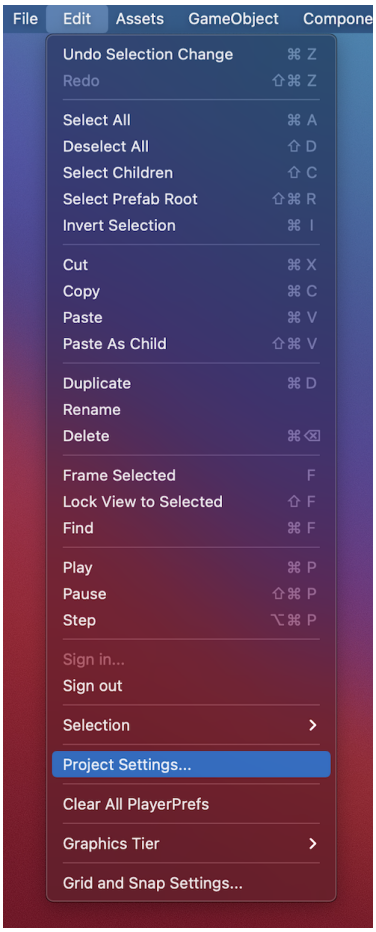
Set-Up the Project to Play the Demo Scenes

This procedure is essential for your project and the included scenes to work correctly. As they require amendments within the Project Settings, the tool cannot work ‘out of the box’ and ignoring this will cause errors.

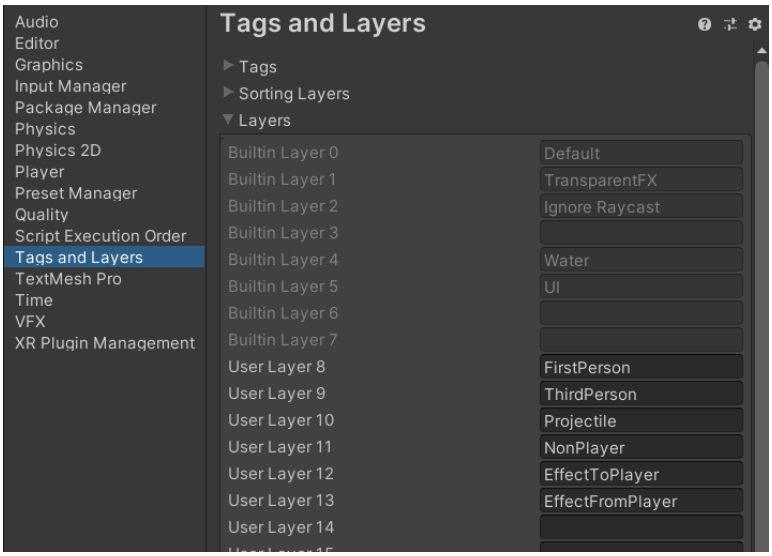
Step 1

Add the required User Layer names via the **Tags and Layers** option in the Project Settings tab:

1. Access the Project Settings tab via the Edit menu on the toolbar.



2. Within the Project Settings tab, select **Tags and Layers** from the left column.
3. Under the Layers list, enter the six layer names into the correct User Layer field and ensure their spelling is correct:
    a. User Layer 8 - ‘FirstPerson’
    b. User Layer 9 - ‘ThirdPerson’
    c. User Layer 10 - ‘Projectile’
    d. User Layer 11 - ‘NonPlayer’
    e. User Layer 12 - ‘EffectToPlayer’
    f. User Layer 13 - ‘EffectFromPlayer’

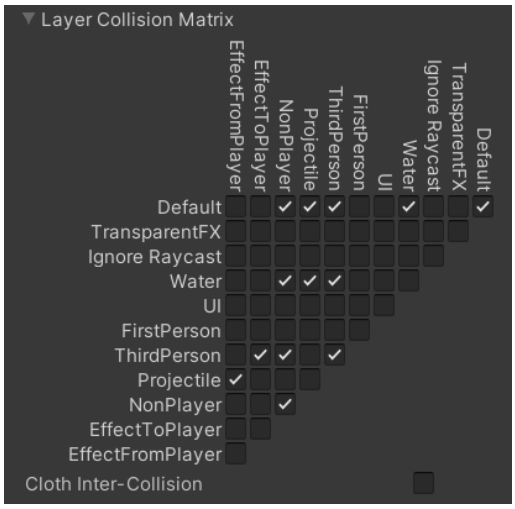


For more information on Layers, review the Layers (GameObject) section.

Step 2

Within the Project Settings in the Physics option, adjust the Layer Collision Matrix to control which kind of GameObject Colliders can collide with each other:

1. Within the Project Settings tab, select Physics from the left column.
2. Configure the Layer Collision Matrix as shown in the image below:



How this matrix works is each tickbox, which can be ticked (true) or unticked (false), has two layer names aligned from the left and top axes. For example, if the tickbox of

Projectile and EffectFromPlayer is ticked, collisions between such types of GameObjects will be allowed. If unticked, they will intersect and not influence each other.

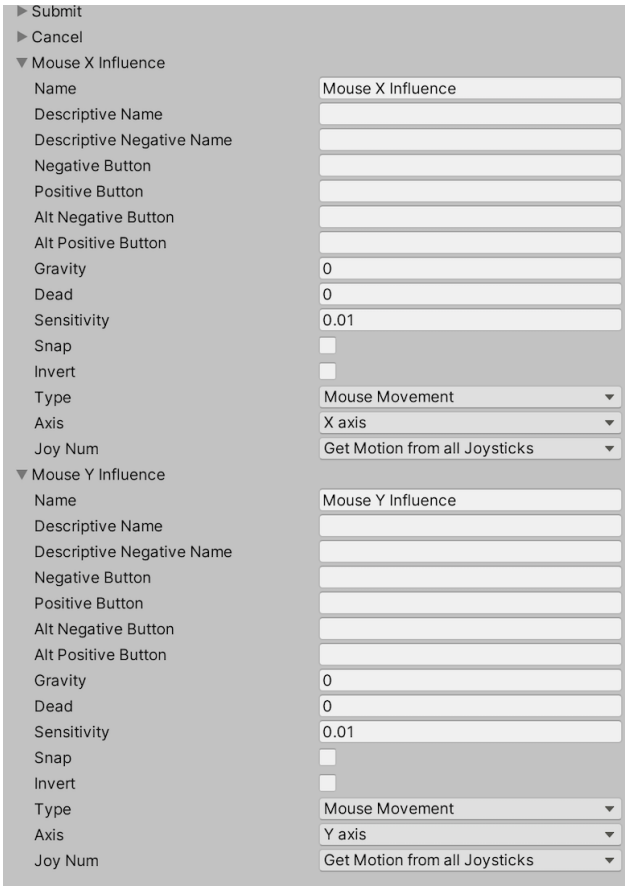
Please note that the arrangement of the Layer Collision Matrix displayed in the above image can tolerate a degree of variability and such alterations may be required for certain scenarios. Testing and iterating could also be the most necessary approach to tuning these settings.

Step 3

From the Input Manager in the Project Settings tab, add two new controls which are similar to the ‘Mouse X’ and ‘Mouse Y’ entries but will involve lower Sensitivity values:

1. In the Project Settings tab, select Input Manager from the left column.
2. Under Axes, increase the value of the Size field by two.
3. Expand the two new controls (both probably named ‘Cancel’) and implement these adjustments in their fields:
    a. Name - ‘Mouse X Influence’ and ‘Mouse Y Influence’ respectively
    b. Descriptive Name - EMPTY
    c. Descriptive Negative Name - EMPTY
    d. Negative Button - EMPTY
    e. Positive Button - EMPTY
    f. Alt Negative Button - EMPTY
    g. Alt Positive Button - EMPTY
    h. Gravity - 0
    i. Dead - 0
    j. Sensitivity - 0.01
    k. Snap - Unticked
    l. Invert - Unticked
    m. Type - Mouse Movement
    n. Axis - **X axis** and **Y axis** respectively

o. Joy Num - **Get Motion from all Joysticks**



The values of these fields can be adjusted to accommodate specific characteristics.

The Shooting Range Demo and Zombie Room Demo should now be fully operational, open these scenes from the Demos folder and run them.

Demo Scene Gameplay

The scenes included in the FPS Integration Tool are designed to demonstrate the results of using this system. They can also be used as templates, but if doing so, consider duplicating the scenes in a location outside the GG FPS Integration Tool folder.

Upon running the scenes prior to additional customisations, the player is spawned in the environment and the HUD featuring health and ammo information is displayed. The Pistol is equipt from the start and can be swapped with other weapons once their collectable has been obtained.

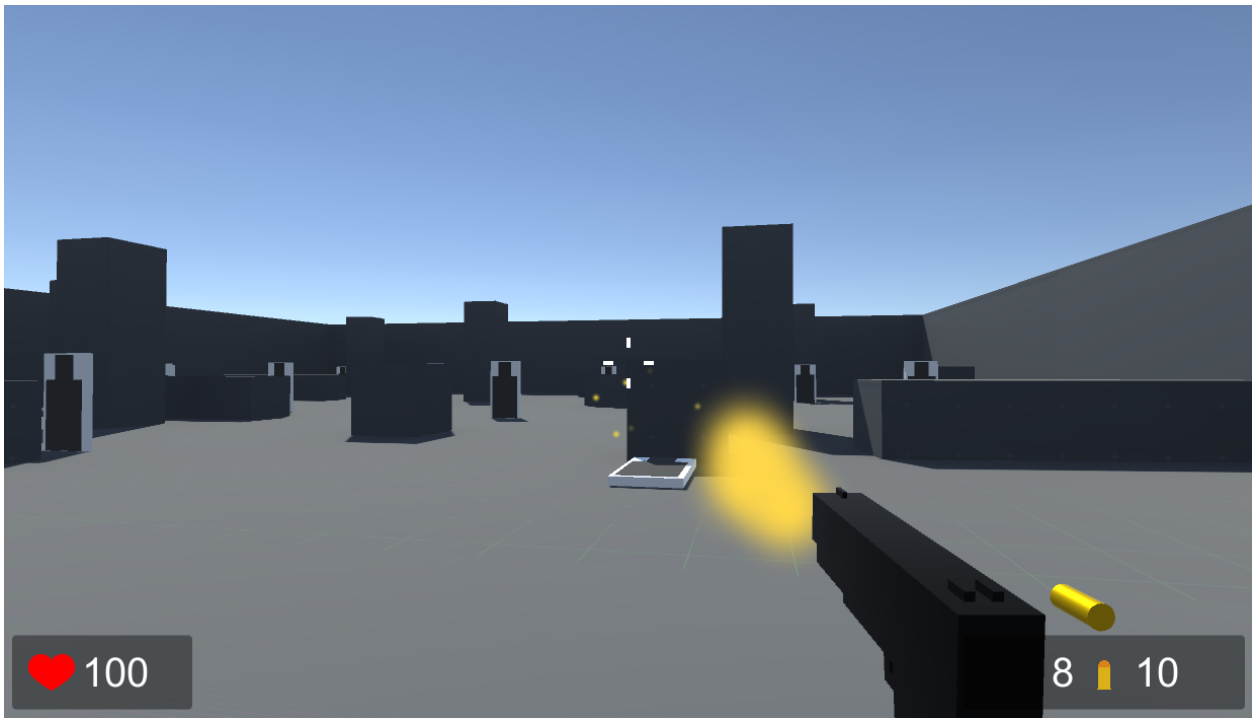
Controls

The player controls set by default involve:

- Move Around - W, A, S, D or Arrow Keys
- Jump - Space Bar
- Run - Hold Left Shift with W, A, S, D or Arrow Keys
- Rotate Horizontally - Left & Right Mouse Movements
- Rotate Vertically - Forward & Back Mouse Movements
- Fire Weapon - Left Mouse Button
- Aim Weapon - Right Mouse Button
- Reload Weapon - R Key
- Switch Weapon - Q Key
- Unlock Cursor from Application - Escape Key

Shooting Range Demo

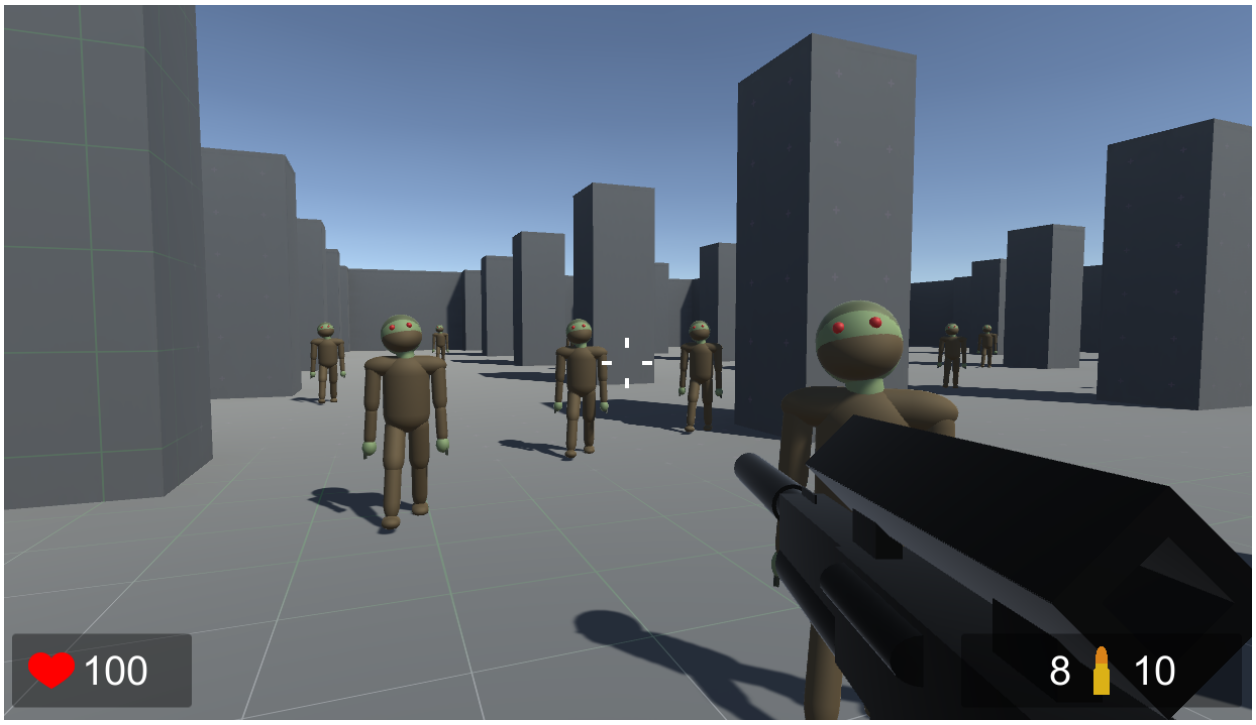
This scene features a shooting range where weapons can be easily collected from the weapon racks to be trialed on the targets, positioned at varying distances and exposure within the range itself. The targets utilise Rigidbody physics, allowing them to be affected by the weapon's Raycast firing and projectiles.



Zombie Room Demo

The scene contains 16 zombie entities that are positioned across a uniform environment separated by pillars, where the weapons featured in the Shooting Range Demo can be found lying around.

The zombies will slowly navigate to the player and will inflict damage when near. The player will respawn at the initial location if the health count reaches zero. Shooting a zombie will harm it and will die after enough damage is dealt. The weapons also conduct varying amounts of damage.

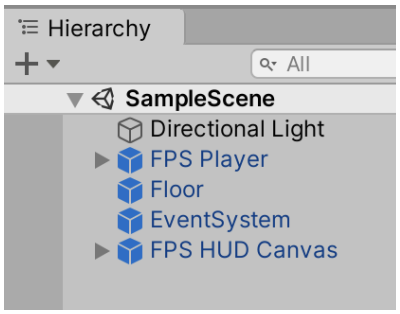


Create A New Scene

Step 1

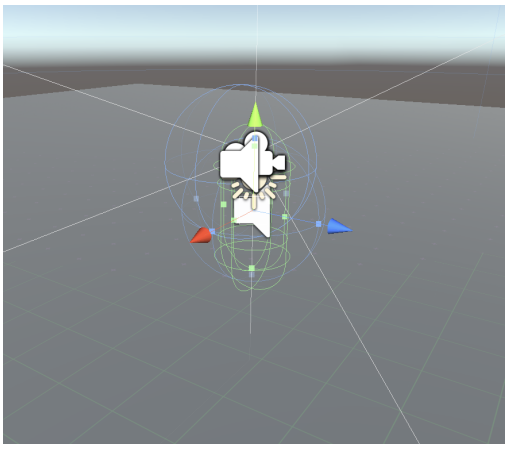
In the new scene:

1. Remove Main Camera from the Hierarchy tab.
2. In the Project Tab, under Assets > GG FPS Integration Tool > Scene Prefabs, drag EventSystems, FPS HUD Canvas and FPS Player Prefabs into the Scene tab.



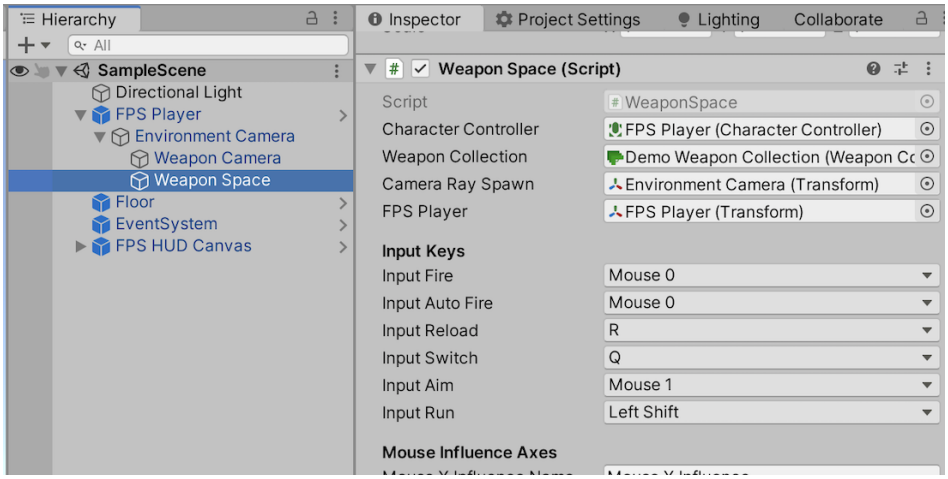
3. Drag in the Floor Prefab into the Scene Tab to create a Floor GameObject.

4. Reposition the FPS Player GameObject so it is over the Floor GameObject in the Scene.



Step 2

Select the Weapon Space GameObject from the Hierarchy tab under FPS Player > Environment Camera, to expose its WeaponSpace Component in the Inspector tab.

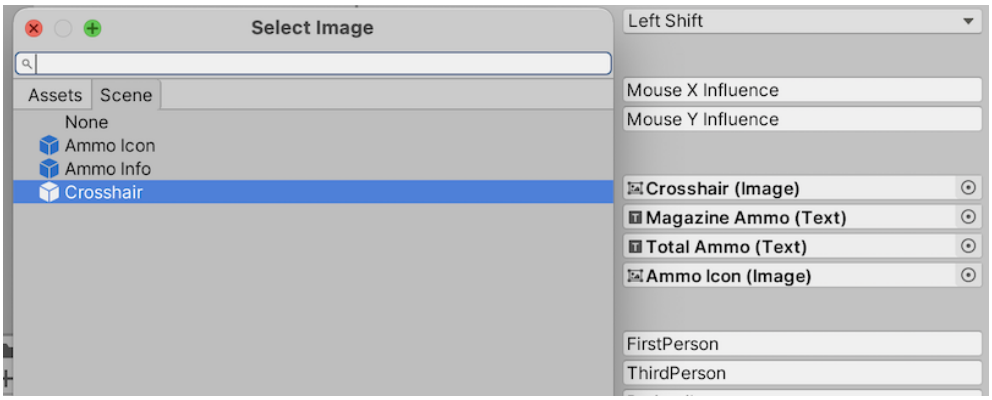


Please note, the **Weapon Space** GameObject is different to the WeaponSpace Component.

Step 3

Apply the FPS HUD Canvas GameObjects to the WeaponSpace Component fields via the Inspector tab:

1. Assign the FPS HUD Canvas > Crosshair GameObject to **UI Crosshair Space** field.

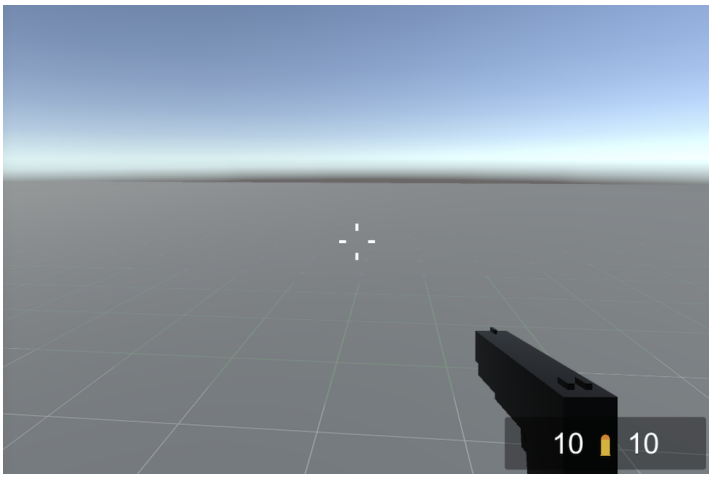


2. Apply FPS HUD Canvas > Ammo Info > Magazine Ammo GameObject to the Mag Ammo Count field.
3. Apply FPS HUD Canvas > Ammo Info > Total Ammo GameObject to the Total Ammo Count field.
4. Apply FPS HUD Canvas > Ammo Info > Ammo Icon GameObject to the Ammo Icon Space field.

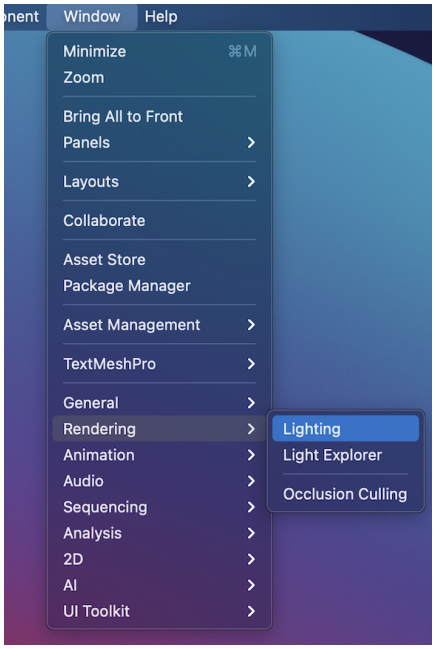
Step 4

If not done already, configure thet **Tags and Layers,** Physics and Input Manager in the Project Settings tab as this is essential for the project to function adequately. Refer to the Set-Up the Project to Play the Demo Scenes section.

At this point when running the project in Play Mode, you should have an FPS experience where the player can move around an area and fire a wielded weapon. The HUD should also be displayed and its values should update.



If the Scene appears dark, enable Auto Generate from the Lighting Tab. Do this by navigating from the Toolbar options Window > Rendering > Lighting, and tick the box next to **Auto Generate** towards the bottom of the Lighting tab. The scene should render thus lighting it up.



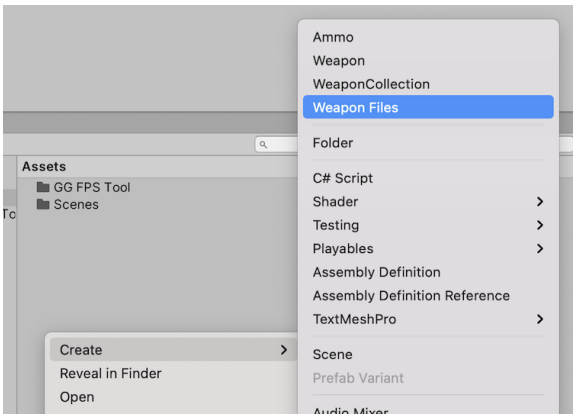
Please note that the lighting in the Shooting Range Demo and Zombie Room Demo scenes is baked, therefore enabling Auto Generate is not required.

Create a Weapon from Weapon Files

Step 1

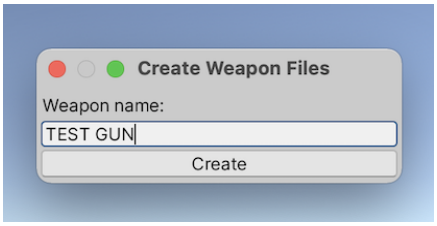
Create a new weapon:

1. Right-click in the Project tab within the Assets folder and select Create > Weapon Files. For more information on the Weapon Files option, check the Weapon Files section.

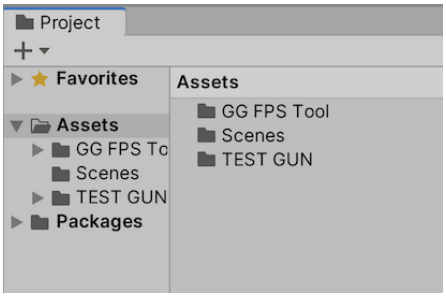


2. A pop-up menu should appear within the Scene tab, then type in the new weapon's name into its field. The start and end characters of the name must not be spaces.

3. Then click the Create button.



You should have a folder in Assets with the name of your weapon.



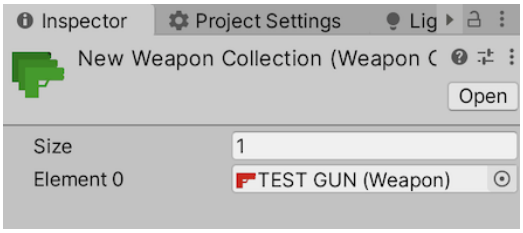
Step 2

Create a WeaponCollection ScriptableObject:

1. Right-click in the Project tab within the Assets folder and select Create > WeaponCollection.

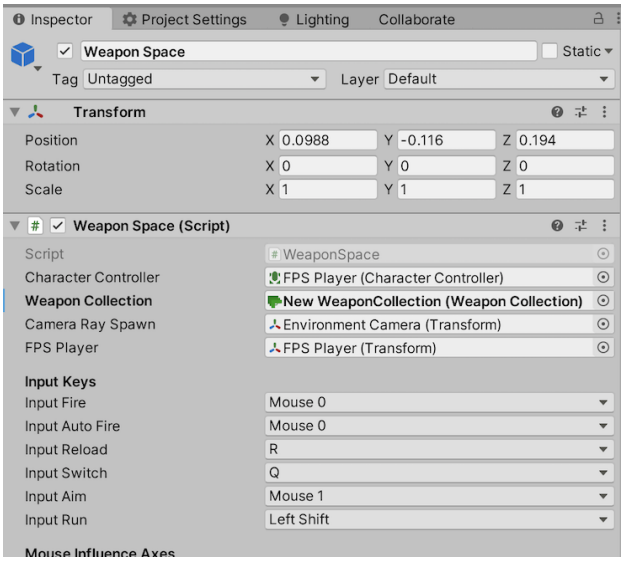
2. Select the new WeaponCollection ScriptableObject.

3. Set the Size field to 1.
4. Apply the Weapon ScriptableObject of your newly created weapon to the **Element 0** field.

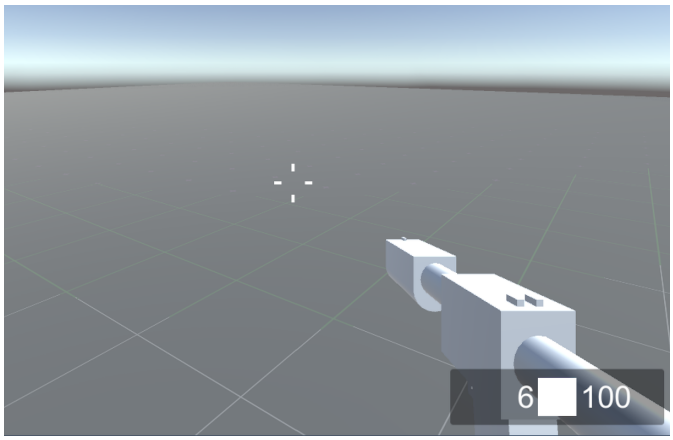


Step 3

Implement the new WeaponCollection ScriptableObject in the Weapon Collection field of the WeaponSpace Component.



In Play Mode, you should have an FPS experience that features a generated copy of the Default Weapon involving standardised attributes.



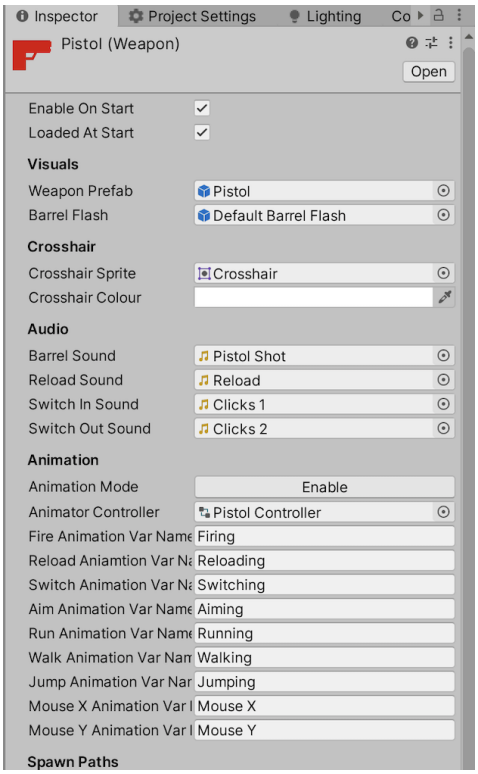
Edit Weapon Attributes

Step 1

Access the folder of your newly created weapon, and select its Weapon ScriptableObject.

Step 2

In the Inspector tab, the attributes of the ScriptableObject are displayed and can be changed to control the behaviour of the affected weapon.



For information on the Weapon ScriptableObject's fields, check the Weapon ScriptableObject section.

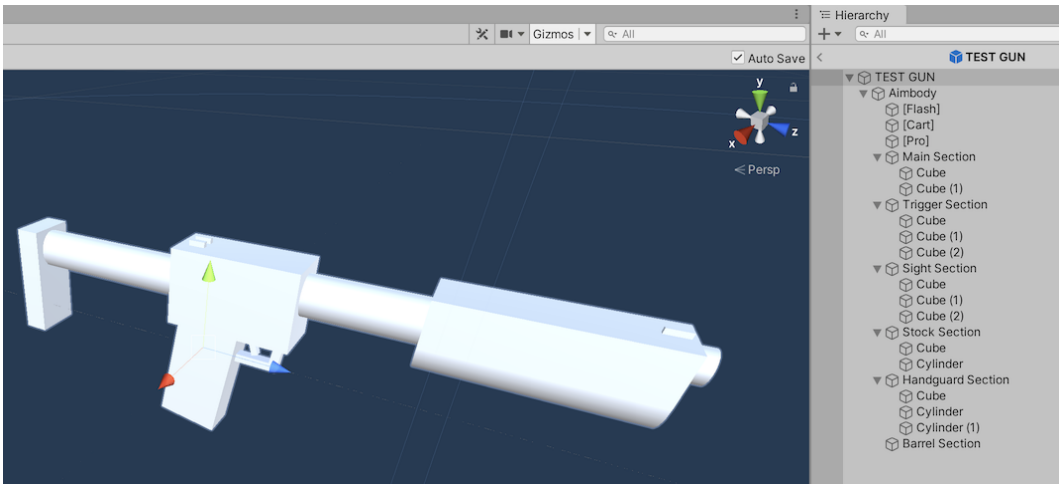
You should have a Weapon ScriptableObject with some adjusted attributes, and in Play Mode, you may notice differences in the weapon's behaviour.

Edit Weapon Prefab

Step 1

Within the new Weapon folder, double-click the Weapon's Prefab.

Doing so will focus the Hierarchy tab and Scene tab around the Weapon Prefab, allowing you to edit it alike adjusting in the Scene tab.



Step 2

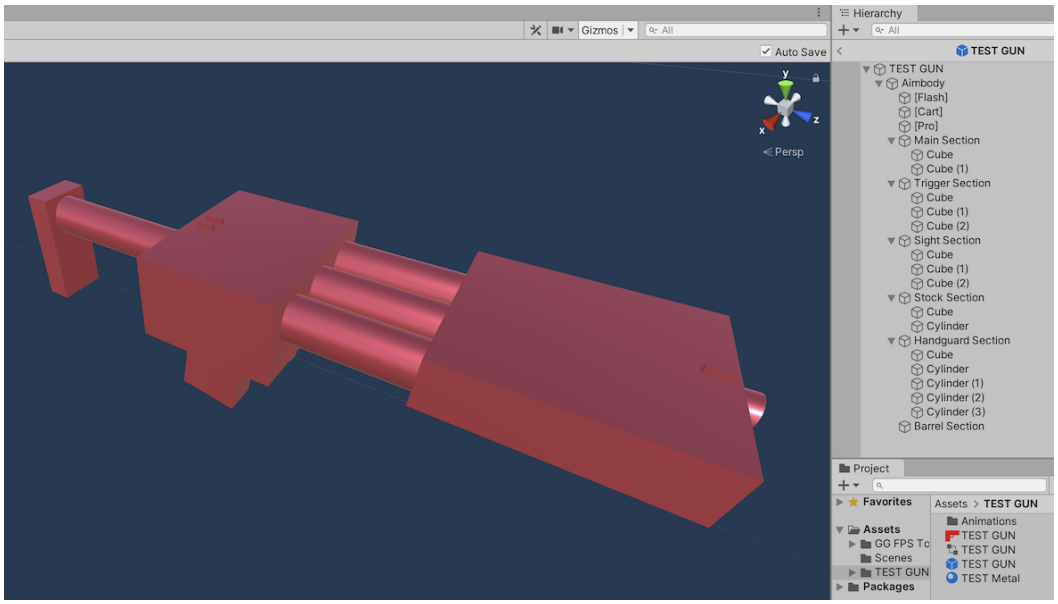
The structure of the Prefab is important to consider, the following is noted as it is in the Hierarchy tab with descriptions:

- **Weapon Name** - The Prefab itself, with the name ‘TEST GUN’ in this case.
    - Aimbody - Needed for aiming animations.
        - [Flash] - The spawn point of the barrel flash with sound.
        - [Cart] - Spawn point of the cartridge ejection.
        - [Pro] - Spawn point of projectiles when firing.
        - Section - Grouping smaller GameObjects together into sections assists with managing the Prefab.
            - Primitive - An individual part of the weapon; it can also involve Meshes.
            - … - Any number of parts can be included.
        - … - Any number of sections can be used.

Anything other than the Sections and Primitives should not be edited, to allow easy implementation and smooth running of the weapon’s animations and behaviour.

Step 3

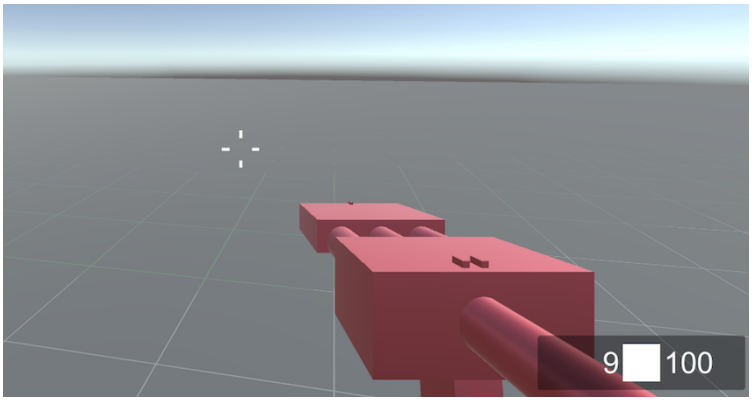
Experiment with selecting and changing the parts of the generated Weapon Prefab.



Prefab editing is automatically saved if the Auto Save box is ticked, at the top right of the Scene tab.

After customising the Weapon Prefab, run the project to view the changes to the weapon.

You now should have a weapon with an altered Prefab.



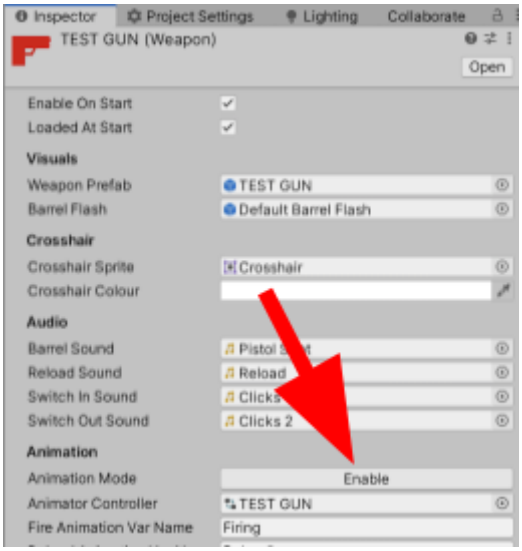
Models can also be imported from the .FBX format and be incorporated in a similar way as primitives.

Edit Weapon Animations

Step 1

Prepare the Weapon Prefab for animating:

1. Select the Weapon ScriptableObject of the weapon you want to edit.
2. In the Inspector tab, click the button marked Enable that is labelled Animation Mode.

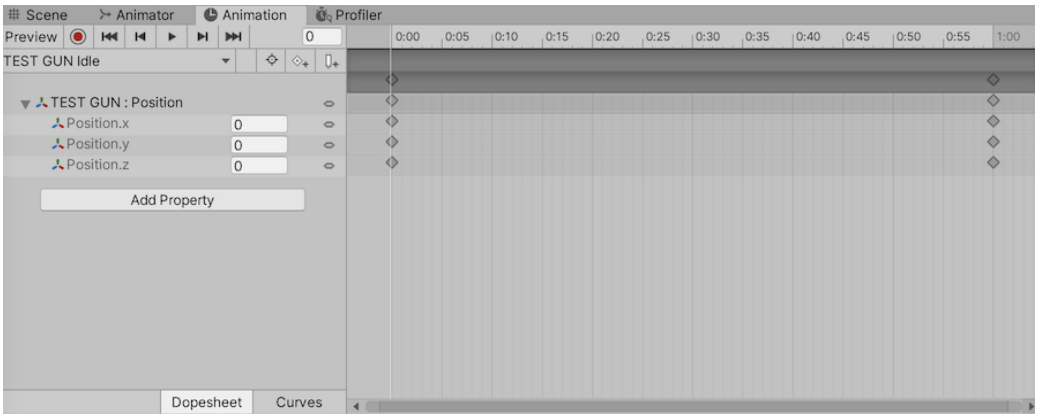


This spawns the weapon's Prefab in the Weapon Space GameObject and applies the weapon's Animator Controller to the Animator Component of the Weapon Space GameObject.

Step 2

Prepare animation system:

1. From the Hierarchy tab, select the weapon GameObject under Weapon Space. This ensures animations will be recorded correctly.
2. Open the Animation tab, which includes the Timeline interface.



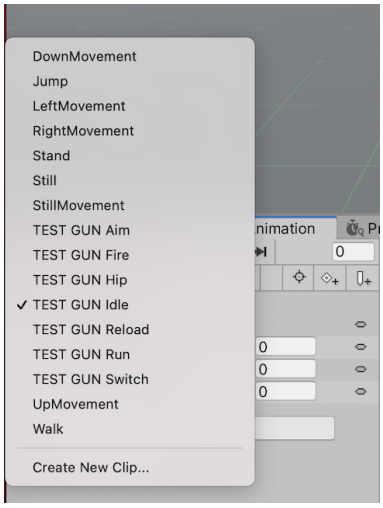
For more information on creating animations in Unity generally, review docs.unity3d.com/Manual/AnimationSection.html.

Step 3

There are important guidelines to consider when creating animations for the FPS IntegrationTool and are described in the Animations Guidelines section.

Introducing the animations:

1. The animations for the Weapon have already been generated through the Create > Weapon Files option.
2. Ensure the weapon GameObject under Weapon Space is still selected.
3. At the top right of the Animation tab, click the Animation Clip name to select another from the drop down.

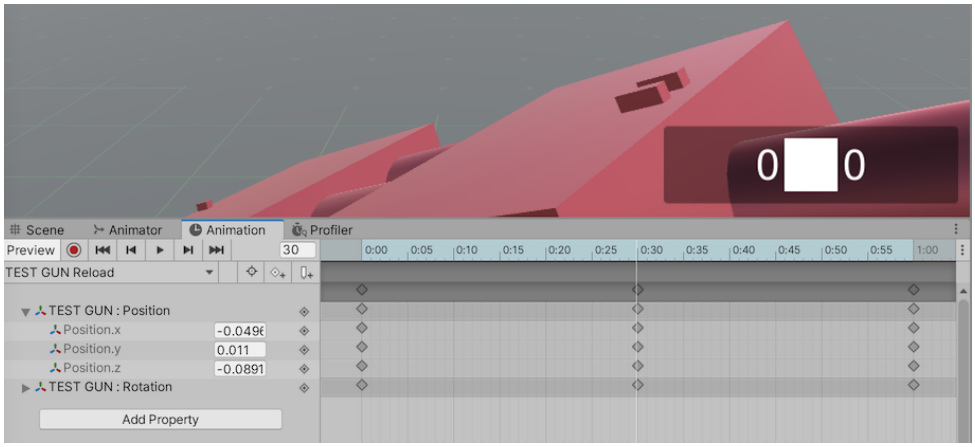


4. Click the play icon to play the looping animation. Alternatively, click (or drag across) the time bar (which displays timing numbers) to show the animation at specific times.

Step 4

Recording animations in general:

1. Utilise the Keyframe Recording Mode to easily produce animations through moving GameObjects via the Scene tab.
2. When adjusting GameObjects below the Weapon Space GameObject in the Hierarchy Tab, the modification is recorded over the current position of the white line in the Animation tab's Timeline.



3. Move this white line around the Timeline to affect different parts of the animation.

You now should have a weapon which plays adjusted animations.

Creating Weapon Collectables

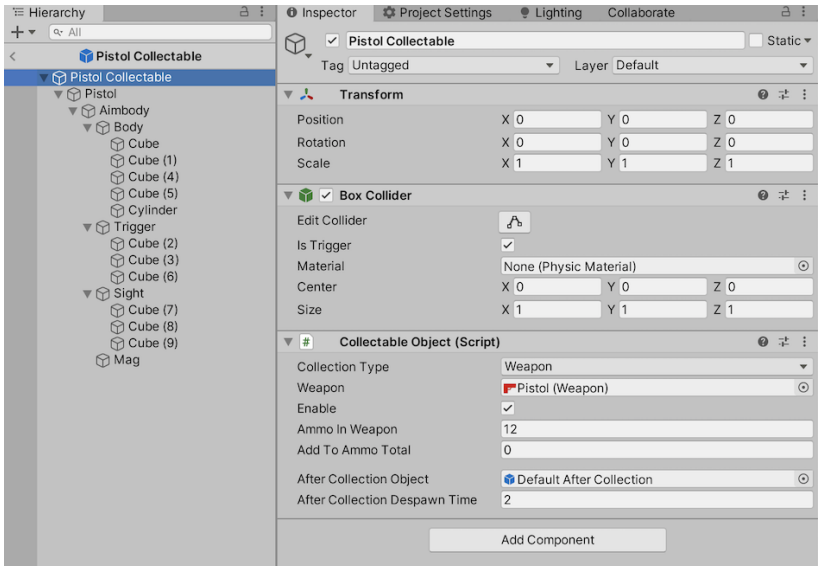
Step 1

Pick-ups provide the FPS Player GameObject with weapons and/or ammo when they intersect each other.

Creating and setting-up the GameObject:

1. Create a new empty GameObject.
2. Attach the CollectableObject Component to it.

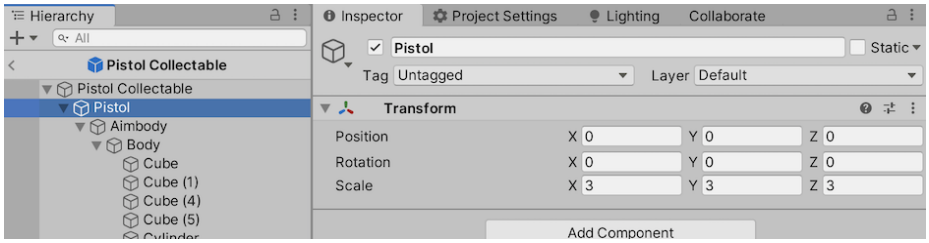
3. Attach a Box Collider Component to it.



Step 2

Applying and adjusting the model:

1. Assign the weapon's Prefab as a child of the Collectable GameObject.
2. Consider rescaling the weapon GameObject in the collectable (**3** in all axes should be suitable). This is needed as generated Weapon Prefabs are small in nature for use in first-person views.



3. Remove the child GameObjects of the Weapon GameObject which have square-bracket names such as [Flash], [Cart] or [Pro]. This prevents barrel flash particle effects, cartridges and projectiles from spawning via the collectable GameObject.
4. Ensure the weapon GameObject's and its children's Layer is set to Default, not FirstPerson.

Step 3

Set collectable GameObject attributes:

1. Select the collectable GameObject.
2. Inside the Inspector tab, set the Weapon field to the correct Weapon ScriptableObject.
3. Set the After Collection Object field to the Default After Collection Prefab.
4. Set the After Collection Despawn Time field to around 2.

You should have a collectable GameObject which enables the chosen weapon (if Enabled On Start within the Weapon ScriptableObject is unticked) on collision with the FPS Player GameObject.

Introducing Zombies to your Scene

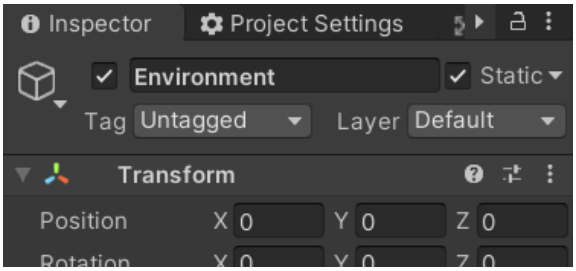
The NonPlayer Zombie is an entity that can be applied within the scene to behave as an enemy against the player. It is fully animated, has audio, seeks the player to inflict damage, and can receive damage and be killed as well.

Including the NonPlayer Zombie requires these steps:

Step 1

For the NonPlayer Zombie to function appropriately with the environment, the level should be constructed before introducing these entities. Ensure that the environment's GameObjects use Collider Components that have their Is Trigger tickbox unticked.

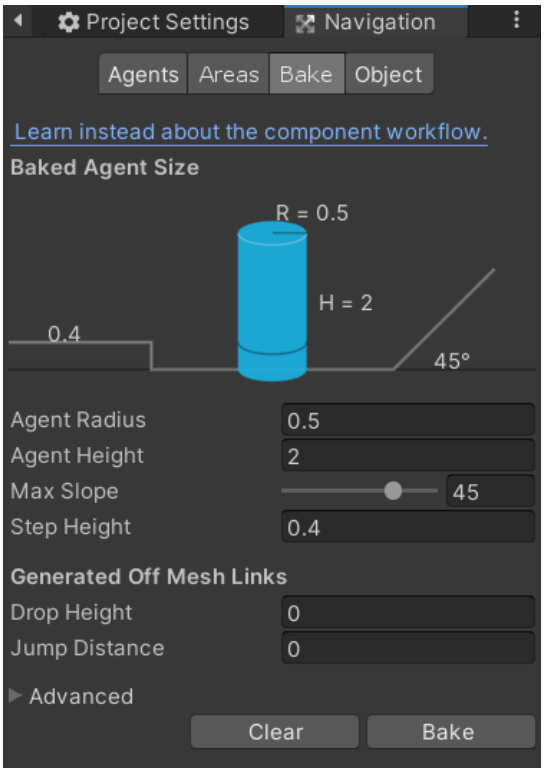
Once the environment is complete, define all the GameObjects that make the environment and that will not move during Play Mode as Static, by selecting them and clicking the Static tickbox from the top of the Inspector tab. This is required for the AI navigation system to function.



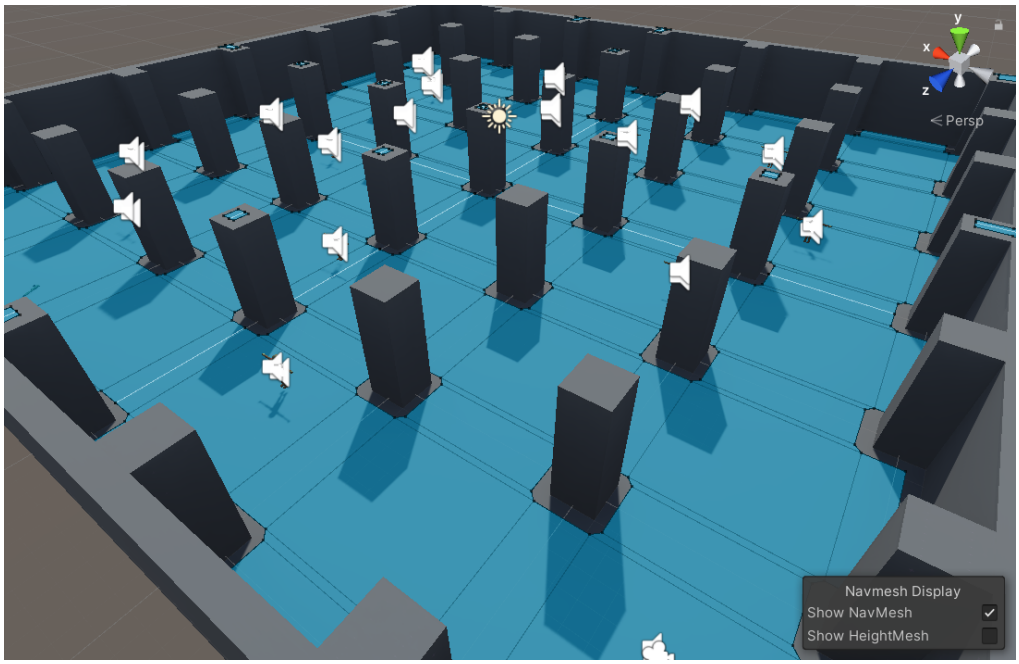
Step 2

From the Navigation tab, press the Bake button to define the area that the AI Agents can navigate to:

1. Open the Navigation tab via Window > AI > Navigation
2. Select the Bake tab button at the top of the new Navigation tab
3. Press the Bake button towards the bottom of the selected Bake tab



At this point, blue surfaces should appear over the top of the static GameObjects.



Step 3

Apply the NonPlayer Zombie to the scene and ensure that they are above the floor and not intersecting with any other GameObjects. This stops the zombie entities from falling through the floor.

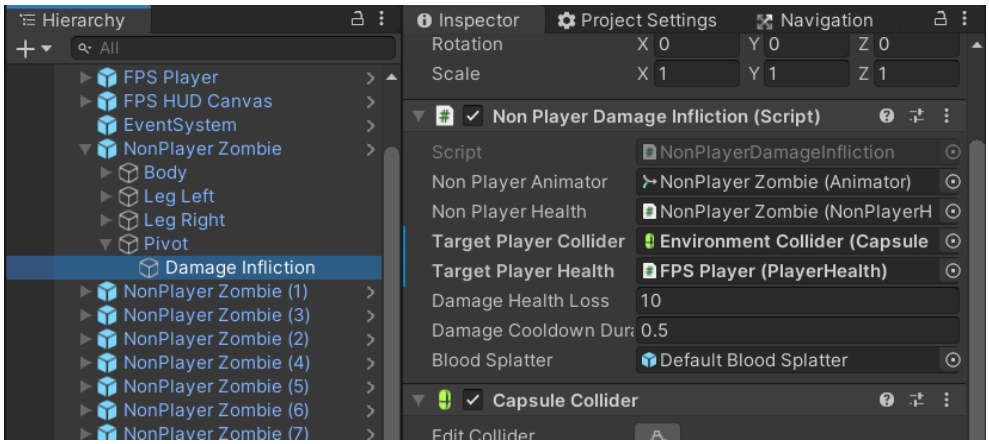
Step 4

With the NonPlayer Zombie selected, in the NonPlayerController Component, assign the FPS Player GameObject from the scene to the Target Player Transform field.

Step 5

Within the NonPlayer Zombie's Damage Infliction GameObject, assign FPS Player's Environmental Collider GameObject and its PlayerHealth Component to the Target Collider and Target Player Health fields respectively:

1. Via the Hierarchy tab, expand the NonPlayer Zombie and its Pivot GameObject then select Damage Infliction.
2. Assign the Environmental Collider GameObject, that is under the FPS Player, to the Target Collider field in the NonPlayerDamageInfliction Component.
3. Also within NonPlayerDamageInfliction, assign the FPS Player's PlayerHealth Component to the Target Player Health field.



When in Play Mode, the zombie entity should navigate toward the player unless completely obstructed, while producing noises and displaying a walking animation. The zombie and player should be able to harm and kill each other.

To involve additional zombies, consider duplicating the one that has already been set-up, saving you from repeating the previous steps.

Adjusting the Zombies

The NonPlayer Zombie is highly customisable as appearance, animations, audio and attributes can be altered.

Remember that implementing complex model-related improvements to the NonPlayer Zombie could cause an increase in overhead where the computer running the scene in Play Mode, or as a built game, will have to work harder.

Speed

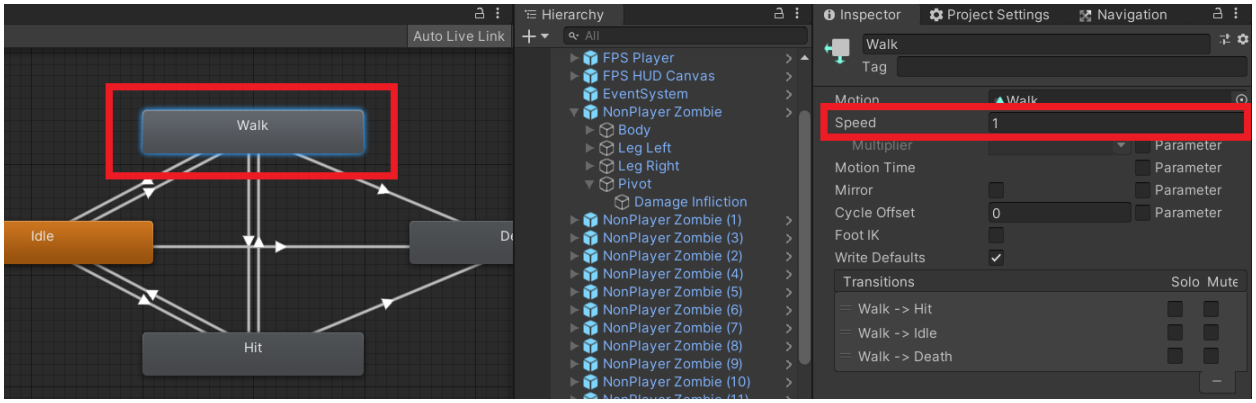
Step 1

To adjust the movement speed of the NonPlayer Zombie GameObject in the scene, access the NavMeshAgent Component and change the value of the Speed field where it is set to 1.2 by default.

Step 2

To maintain some realism, the animation speed of the NonPlayer Zombie has to be changed in synchronisation:

1. In GG FPS Integration Tool > Animations > NonPlayer Zombie, double-click the Zombie Animator Controller to view it in the Animator tab.
2. In the Animator tab, select the Walk Node, then in the Inspector tab, alter the Speed field to the same proportion as the Nav Mesh Agent Component's Speed.



For example, adjusting the speed from 1.2 to 2.4 in the NavMeshAgent means changing the Walk Node speed from 1 to 2. However, minor differentiations could be necessary to improve the accuracy of the animation.

Damage To Player

To modify the damage the NonPlayer Zombie deals to the player, access its NonPlayerDamageInfliction Component and edit the Damage Health Loss field to define how much health is deducted each time it attacks.

Also the Damage Cooldown Duration field can be used to control how many seconds until the attack is performed again.

However, the Hit Node's Speed field would need adjusting proportionally, except that as the Node's Speed field resembles a speed multiplication rather than a duration in seconds, the value of the conversion **Speed = 1/Seconds** should be used in the Node's Speed field.

Damage From Player

Although this does not involve editing the NonPlayer Zombie, to control the damage dealt to the zombie entity:

- Configure the Damage Per Round field from within the relevant Weapon ScriptableObject that uses the Output Type of Ray.
- Otherwise if using the Output Type of Projectile, the explosion GameObject featuring the Explosion Component contains the customisable Maximum Damage field.

Health

To amend the health of the NonPlayer Zombie, access the NonPlayerHealth Component in the NonPlayer Zombie and change the Maximum Health field.

WeaponSpace Animation Guidelines

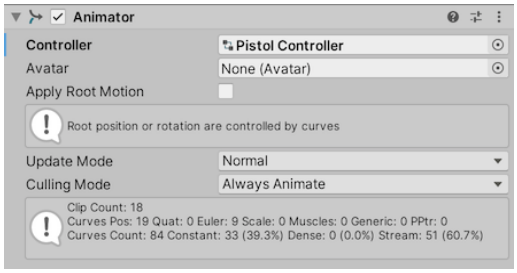
Although Unity's animation system is complex, it offers great flexibility and power for producing animations, which is why it is utilised by the FPS Integration Tool.

However, for animations to function appropriately with the FPS Integration Tool, animations should follow the guidelines below.

Terminology

To avoid confusion with the elements that form Unity's animation system, the terms are explained:

- Animation Tab - Is the window for creating animations which features a Timeline.
- Animator Tab - Also a window, but used for structuring how a collection of animations are executed, which involves a flow diagram.
- Animator - Often refers to the Component of a GameObject which applies the behaviours of an Animator Controller into the GameObject.

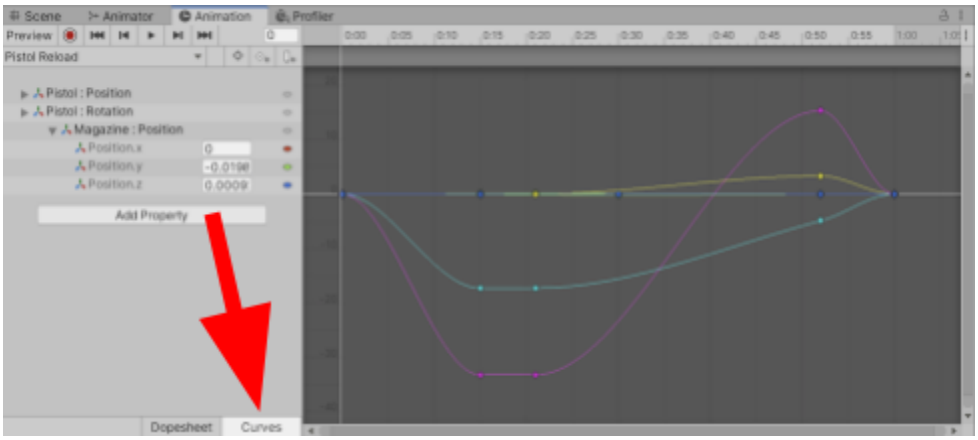


- Animator Controller - Is used in an Animator Component to provide an animation system onto the GameObject.
- Animation Clip - Holds a particular animation to be used within an Animator Controller.

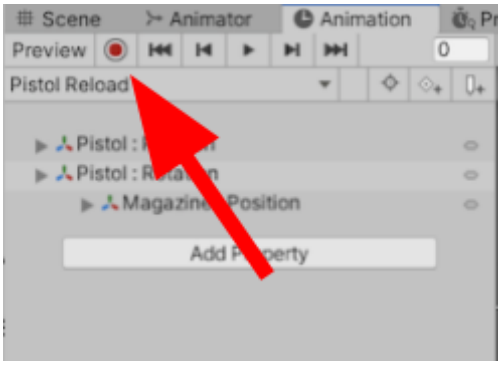
Animation Tab

Terms related to the inner workings of the Animation tab:

- Dopesheet - The Timeline interface that displays Keyframes as dots.
- Curves - The Timeline interface that uses curves between Keyframes to describe characteristics of the transitions.



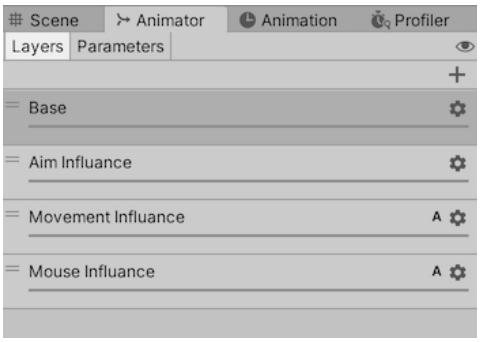
- Keyframe - A point in the Animation tab Timeline that marks a state of a GameObject, such as a position or rotation. Gradual transitions occur between Keyframes as the animation is played.
- Keyframe Recording Mode - The red recording button at the top-left of the Animation tab that allows GameObjects' states to be recorded if altered in the Scene tab. Often used for animation development involving GameObject Transforms such as position, rotation and scale.



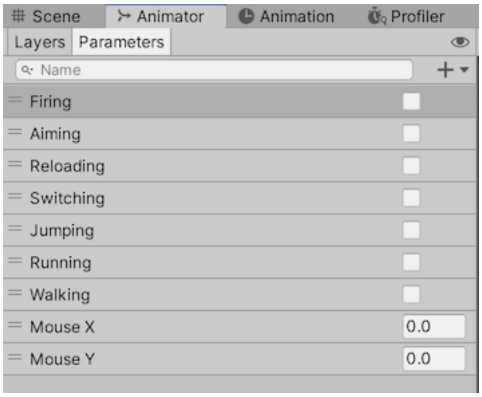
Animator Tab

Terms for the Animator tab involves:

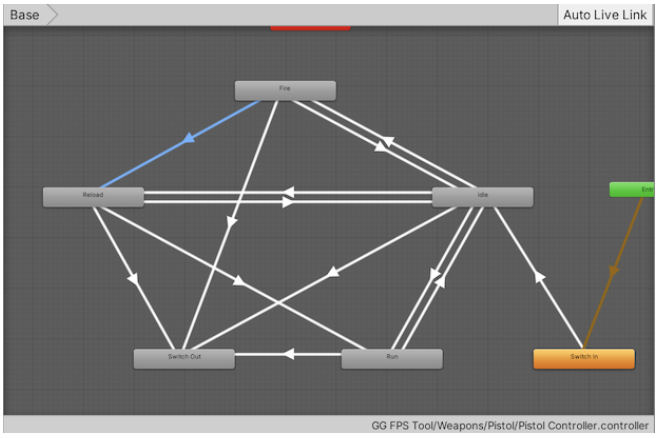
- Layers - Levels of animations that can influence others in additive or overriding ways, useful for combining animations which may need to play in parallel during runtime.



- Parameters - Variables of certain types that are often used to trigger transitions between animations.



- Nodes - States in the Animator Controller that plays a certain animation when executed.
- Transitions - Links between two Nodes that allows animations to transition once the specified Parameter conditions are met.



Layers (Animation)

In general, animations used with the weapon can be categorised as:

- Base - Where most of the animations related to the weapons' actions lie such as Fire, Switch and Reload, and where most animation editing takes place.
- Aim Influence - Features Hip and Aim animations which hold a constant position and uses transitions for switching between the two states, this is also often edited.
- Movement Influence - Supports animations that are controlled by the movement of the FPS Player GameObject.
- Mouse Influence - Is for animations controlled by mouse movements which also turns the FPS Player GameObject.

Remember, Layers for animations differ from Layers for GameObjects.

For more information on Layers for GameObjects, check the Layers (GameObject) section.

Hierarchy

To allow parallel animations provided by the Animator Controller Layers to affect the weapon correctly, the Weapon Space GameObject contains levels of GameObjects under it within the Hierarchy tab which are controlled by specific Layers:

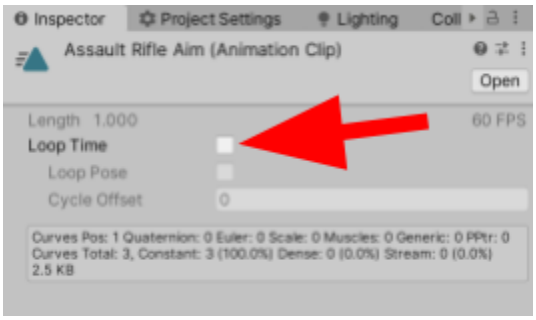
- Weapon Space - Used by Mouse & Movement Influence Layers.
    - Weapon GameObject - Is spawned from the prefab during Play Mode, and is controlled by Base Layer animations.
        - Aimbody - Affected by the Aim Influence Layer.
            - Weapon Sections/Primitives - Are the parts of the weapon which can be individually animated within Base and Aim Influence Layers. Useful for creating more advanced animations, for example, moving the magazine during reloading.

General

1. Animations should be 1 second long for easy management of speed within the Animator Controller Nodes.
2. With the Weapon ScriptableObject, ensure the correct Animation Clips and Animation Controller are applied to the fields.

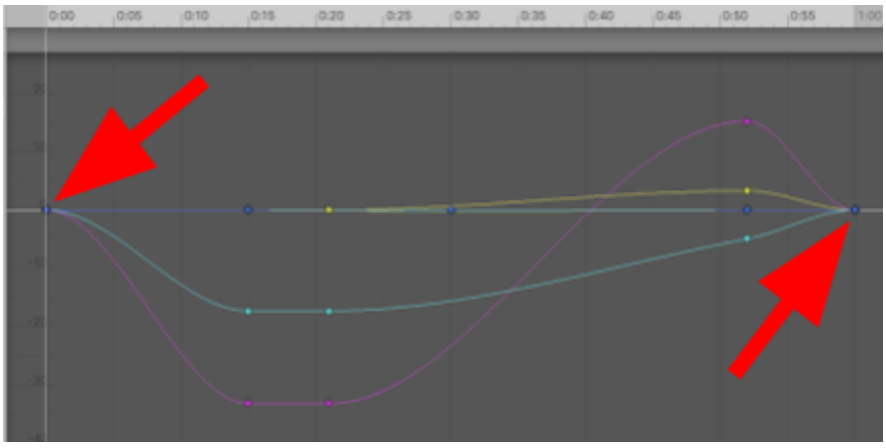
3. Animation Timing Fields in Weapon ScriptableObjects should contain relatable values to certain Animator Controller Transitions to ensure Unity's animation system and the FPS Integration Tool are synchronised to prevent disordered outcomes.

4. Animation Clips' Loop Time should be unticked, unless the animation should repeat like for Partial Repeat reloading weapons.



5. If an Animator Controller becomes severely disrupted through editing, consider copying and pasting the Animator Controller of the Default Weapon to retry. Though ensure the Default Weapon files remain unchanged to prevent system malfunctions.

6. For creating realistic animations, the weapon's animated GameObjects should return to their initial position, rotation and scale by the end of the animation. Some types of animations are exempt from this rule, such as Switch animations.



7. Avoid editing the Animator Controller Parameters naming and Layer settings.

Idle Animations

Idle animations just involve the Weapon GameObject in its normal position throughout the animation.

Fire Animations

Often, firing animations rely on the incoming transition to replicate recoil, as firing often pushes the weapon back suddenly. When animating this, the animation should start with the weapon positioned slightly backwards then gradually returns to the initial position by the end.

Inconsistent Animations

Sometimes, firing animations may not play every time a round is fired even if the Speed field of the Fire Node matches the Fire Rate field in the corresponding Weapon ScriptableObject, this is due to the transition delay and can be remedied by slightly increasing the Node's speed.

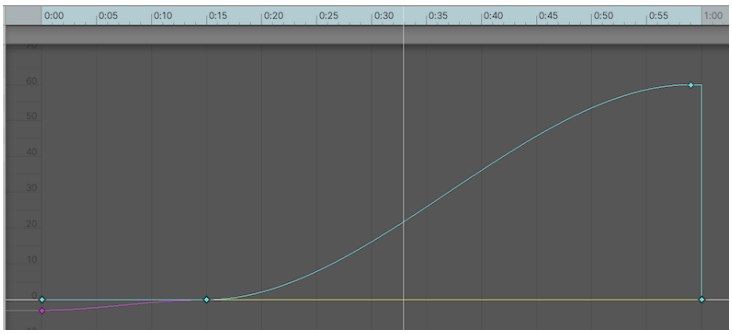
Revolving Chambers

Producing something like the Grenade Launcher's firing animation which involves the chambers partially rotating can cause difficulty with realistically animating the Weapon's GameObjects to return to their initial Transforms.

The solution to this involves:

1. In the Timeline, set the Transforms to their initial stats at the animation's end (1:00 second).
2. Animate the rotating chambers and ensure they rotate to a similar-appearing state. Consider calculating the amount of rotation by the equation **360 / Number of Rotational Symmetry = Degrees of Rotation**.
3. Position the rotation's ending Keyframe just before the resetting Keyframe (0:59 second), so there is minimal time between the rotation's and animation's end.
4. Switch the Animation tab from Dopesheet to the Curves setting.
5. Find the second to final Keyframe Node (on what should be the most expressive curve) then right-click it and select Broken from the Pop-Up Menu.
6. Move the Handle Node on the right side of that Keyframe Node downwards so it is directly below the Keyframe Node. Doing this turns the curve into a right angle, meaning the transition will be sudden instead of gradual.

The result should be a firing animation featuring a revolving section which rotates to the next chamber without any apparent reverting change at the animation's end.



Reload Animations

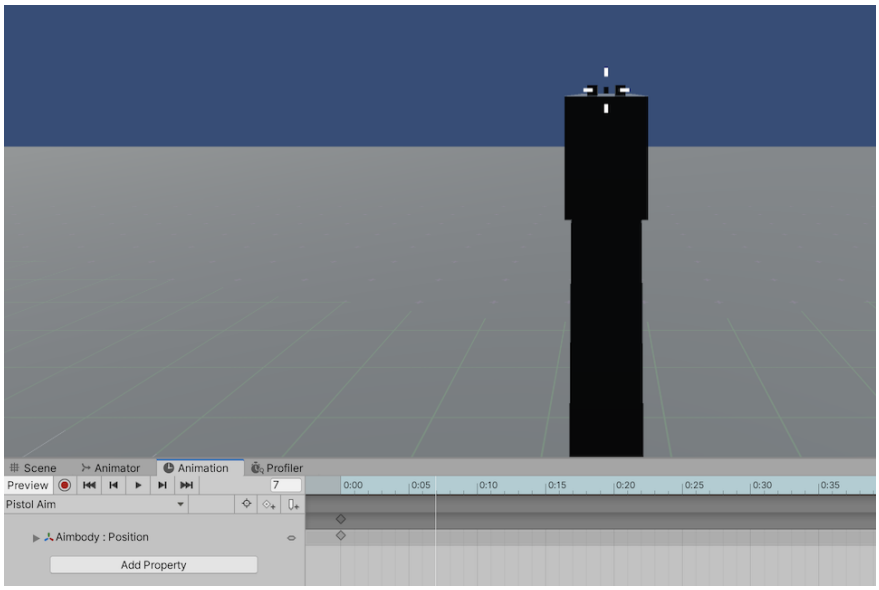
Reload animations should begin and end at the original idle positions, unless it uses **Partial** or **Partial Repeat** Reload Types enabled within the Weapon ScriptableObject, as this may need to repeat. Thus, the animations start and end should match each other and be positioned towards their target position instead of the idle position. Such animations utilise the Transitions to produce the idle to target movement.

Hip Animations

Hip animations are similar to the Idle animations as they feature the initial Transforms throughout.

Aim Animations

Aim animation involves positioning the Weapon GameObject where it's own sight can be used to aim without the crosshair. When producing the Aim animation, ensure the sight accurately overlaps the pre-applied crosshair to ensure the raycasting matches the weapon sight direction. The whole animation is also constant.



Switch Animations

Switch animations should begin with the weapon off from view and end when at the idle position, by definition this is a Switch-In. For ease of animating, Switch-In animations can be reused as a Switch-Out by setting their Node's Speed field to -1 (or the same number as the Switch-In Speed times -1).

Run Animations

Run animations are similar to Idle in that it resembles a constant position, but the Weapon GameObject is aimed away from ahead such as downwards. The jogging motion is managed automatically by the Movement Influence Layer.

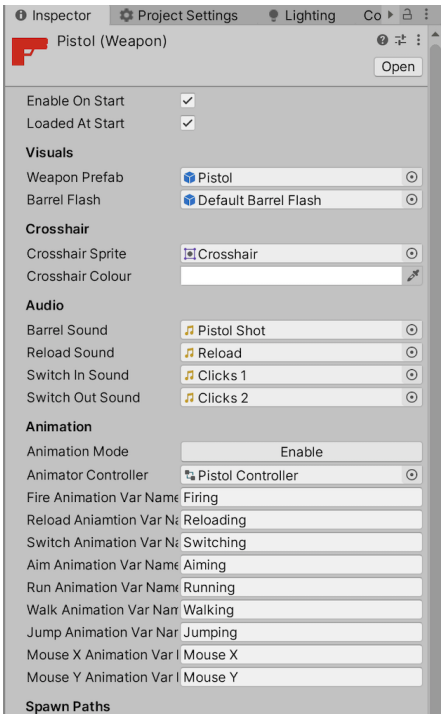
ScriptableObjects

ScriptableObjects are objects that behave like files rather than GameObjects and Prefabs. They are designed to hold values that can be modified via the Inspector tab, and can be applied to a field in another menu instead of being placed in the Scene.

The FPS Integration Tool utilises this feature and includes these types of ScriptableObjects:

Weapon

Weapon ScriptableObjects contain data relating to a specific weapon's characteristics.



They are often assigned to WeaponCollection ScriptableObjects and are created via the Create Options.

The Weapon ScriptableObject involves these fields:

Field Name / Heading	Data Type	Description
Enable On Start	Tickbox (Boolean)	If ticked (true), this weapon will be equipped from the very start. If not ticked (false), the weapon will need to be enabled, such as from a collectible, to be equipped.
Loaded On Start	Tickbox (Boolean)	**This field is displayed only when Enable On Start is ticked (true).** If ticked (true), this weapon is fully loaded from the start. Otherwise (false), the weapon is completely unloaded from the start.

Visuals		
Weapon Prefab	Prefab	The Prefab is instantiated when the weapon switches-in and is destroyed when switching-out. Decollide Message Box appears if the applied Prefab or its children contains any Collider Components, which could disrupt the Raycast firing. Clicking the Decollide button and again in the pop-up menu to resolve. Relayer Message Box appears if the applied Prefab or its children are not in the FirstPerson Layer, which could disrupt the weapon's rendering. Clicking the Relayer button and again in the pop-up menu to resolve.
Barrel Flash	Prefab	This Prefab is spawned at the referencing GameObject from the Barrel Flash Spawn Name field when firing the weapon.
Crosshair		
Crosshair Sprite	Sprite	The Sprite is applied to the Crosshair Space of the FPS HUD Canvas when switching-in the weapon.
Crosshair Colour	Colour	Tints the Crosshair Sprite in the defined colour.
Audio		
Barrel Sound	Audio	The audio that is played when firing the weapon.
Reload Sound	Audio	The audio that is played when reloading the weapon.
Switch In Sound	Audio	The audio that is played when switching in the weapon.
Switch Out Sound	Audio	The audio that is played when switching out the

		weapon.
Animation		
Animation Mode	Toggle Button	When the button is clicked while displaying Enable, the weapon's Prefab and Animator Controller is applied to the Weapon Space GameObject. When clicked while displaying Disable, the Weapon's Prefab and Animator Controller are removed from the Weapon Space GameObject. If clicking the button of another Weapon ScriptableObject's Animation Mode while the initial weapon is in Animation Mode, the Prefab and Animation Controller is automatically replaced.
Animator Controller	Animator Controller	The Animator Controller of the weapon that is applied to the Weapon Space Animator Component when the weapon is wielded.
Fire Animation Var Name	String	Name of the Firing Parameter of the Weapon's Animator Controller. The default name is 'Firing' and often does not need renaming.
Reload Animation Var Name	String	Name of the Reloading Parameter of the Weapon's Animator Controller. The default name is 'Reloading' and often does not need renaming.
Switch Animation Var Name	String	Name of the Switching Parameter of the Weapon's Animator Controller. The default name is 'Switching' and often does not need renaming.
Aim Animation Var	String	Name of the Aiming Parameter of the Weapon's

Name		Animator Controller. The default name is ‘Aiming’ and often does not need renaming.
Run Animation Var Name	String	Name of the Running Parameter of the Weapon’s Animator Controller. The default name is ‘Running’ and often does not need renaming.
Walk Animation Var Name	String	Name of the Walking Parameter of the Weapon’s Animator Controller. The default name is ‘Walking’ and often does not need renaming.
Jump Animation Var Name	String	Name of the Jumping Parameter of the Weapon’s Animator Controller. The default name is ‘Jumping’ and often does not need renaming.
Mouse X Animation Var Name	String	Name of the Mouse X Parameter of the Weapon’s Animator Controller. The default name is ‘Mouse X’ and often does not need renaming.
Mouse Y Animation Var Name	String	Name of the Mouse Y Parameter of the Weapon’s Animator Controller. The default name is ‘Mouse Y’ and often does not need renaming.
Spawn Points		
Barrel Flash Spawn Name	String	Is the Hierarchy directory under the Weapon Space GameObject of the Barrel Flash spawn. Usually follow the format: ‘Weapon Name’/Aimbody/[Flash]

		For example: MyWeapon/Aimbody/[Flash]
Projectile Spawn Name	String	Is the Hierarchy directory under the Weapon Space of the Projectile spawn. Usually follow the format: ‘Weapon Name’/Aimbody/[Pro] For example: MyWeapon/Aimbody/[Pro]
Cartridge Spawn Name	String	Is the Hierarchy directory under the Weapon Space of the Cartridge spawn. Usually follow the format: ‘Weapon Name’/Aimbody/[Cart] For example: MyWeapon/Aimbody/[Cart]
Attributes		
Output Type	Options	Defines the type of simulated projectile fired from the weapon with options: Ray / Projectile. If set to Ray, a Raycast is used to represent a projectile. If set to Projectile, a specified projectile is launched instead.
Ray Impact	Prefab	**This field is displayed only when Output Type is set to Ray.** The Prefab that is spawned at the Raycast Hit, often to simulate a bullet hole or equivalent.
Range	Float	**This field is displayed only when Output Type is set to Ray.** The maximum distance of the Raycast from the Camera Ray Spawn GameObject.
Impact Force	Float	**This field is displayed only when Output Type is set to Ray.**

		The force that is applied to a Rigidbody that was hit by the Raycast.
Damage Per Round	Float	**This field is displayed only when Output Type is set to Ray.** The amount of damage per round to inflict on GameObjects with PlayerHealth or NonPlayerHealth Components. Each Raycast applies the damage equal to Damage Per Round divided by Output Per Round.
Projectile Object	Prefab	**This field is displayed only when Output Type is set to Projectile.** The Prefab that is launched from the Projectile Spawn Name field.
Launch Force	Float	**This field is displayed only when Output Type is set to Projectile.** The forward force applied to the newly spawned Projectile GameObject.
Firing Type	Options	The type of firing response used when firing the weapon with options: Semi / Auto. If set to Semi, the weapon is fired per button/key press, but firing will never exceed Fire Rate. If set to Auto, the weapon fires repeatedly at the Fire Rate while the firing button/key is held down.
Rounds Per Burst	Integer	Number of shots fired per fire button/key press.
Fire Rate	Float	Maximum number of shots fired per second.
Output Per Round	Integer	Number of Raycasts/projectiles launched per shot. Exceeding the value of 10 is not recommended as it may cause unnecessary

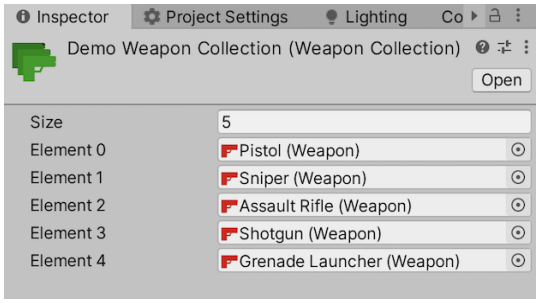
		overhead.
Aiming Spread	Float Range	The Euler angle offset of the firing direction when aiming. Lower values provide more accuracy. Aiming Spread is often the lowest spread value.
Hip Spread	Float Range	The Euler angle offset of the firing direction when not aiming. Lower values provide more accuracy. Hip Spread is often between Aiming Spread and Movement Spread values.
Movement Spread	Float Range	The Euler angle offset of the firing direction when not aiming while moving. Lower values provide more accuracy. Movement Spread is often the highest spread value.
Ammo	Ammo	The type of ammo used by the weapon.
Capacity	Integer	The maximum amount of ammo that can be loaded into the weapon.
Ammo Loss Per Round	Integer	Amount of ammo decreased from the Capacity per shot.
Reloading Type	Options	Defines the weapon's nature of reloading with options: Full / Partial / Partial Repeat. If set to Full, the weapon is normally fully reloaded. If set to Partial, the weapon's ammo is normally increased by a set amount during reloading. If set to Partial Repeat, the weapon's ammo is normally increased by a set amount but the reloading process repeats until the weapon is fully loaded.
Ammo Added Per Reload	Integer	**This field is displayed only when Reloading Type is set to Partial or Partial Repeat.**

		The amount of ammo normally added to the weapon during reloading.
Animation Timing		
Aiming Time	Float	Defines the duration taken from activating or deactivating the aiming to the effects being applied. The value of this field should be similar to the **Transition Duration (s)** field in the **Aim -> Hip** and **Hip -> Aim** Transitions of the Animator Controller's Aim Influence Layer.
Reloading Time	Float	Defines the duration of the reloading process in seconds. Should be proportionate to the Speed field of the Reload Node in the Base Layer of the Weapon's Animator Controller. As this Reloading Time and the Reload Node's Speed are based on Seconds and Speed Multiplication respectively, this conversion is required: Speed = 1 / Seconds Seconds = 1 / Speed
Switching Time	Float	Defines the duration of the switching process in seconds. Should be proportionate to the Speed field of the Switch-In and Switch-out Nodes in the Base Layer of the Weapon's Animator Controller. As this Switching Time and the Switch-In / Switch-out Node's Speed are based on Seconds and Speed Multiplication respectively, this conversion is required: Speed = 1 / Seconds Seconds = 1 / Speed
Running Recovery Time	Float	Amount of time firing is re-enabled after running. Should be similar to the **Transition Duration (s)**

		field in the Transitions exiting the Run Node in the Base Layer of the Weapon's Animator Controller.
Incremental Reload Recovery Time	Float	Amount of time firing is re-enabled after a Partial or Partial Repeat reloading is finished or interrupted. Should be similar to the **Transition Duration (s)** field in the Transitions exiting the Reload Node in the Base Layer of the Weapon's Animator Controller.
Cartridge Ejection		
Ejection Object	Prefab	The Prefab that represents an ejected cartridge.
Ejection Trajectory	Vector3	The direction for the ejected cartridge to be launched towards.
Ejection Force	Float	The force applied to the ejected cartridge when launched.

WeaponCollection

The WeaponCollection ScriptableObject holds an array of Weapon ScriptableObjects that are used in the WeaponSpace Component to define which weapons can be accessed in the FPS Player GameObject.



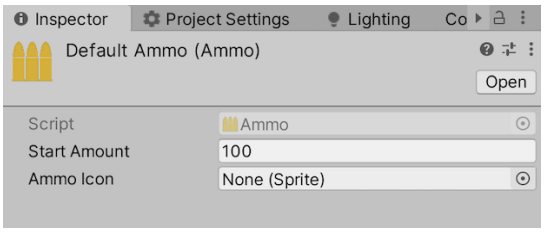
WeaponCollection ScriptableObjects are created via the Create Options.

The WeaponCollection involves these fields:

Field Name / Heading	Data Type	Description
Size	Integer	Number of weapons that can be assigned to this WeaponCollection, determining the amount of **Element *** fields available for Weapon ScriptableObjects.
Element * (Depends on array index)	Array of Weapon ScriptableObjects	Weapon ScriptableObjects used within this WeaponCollection. The order of Weapon ScriptableObjects in these fields will correlate with the order of accessing weapons when switching.

Ammo

The Ammo ScriptableObject stores information about a type of ammo that weapons can refer to.



Field Name / Heading	Data Type	Description
Start Ammo	Integer	Amount of this ammo available from the start.
Ammo Icon	Sprite	The Sprite that is applied to the AmmoIcon GameObject when a weapon using this ammo is switched-in.

Components

The FPS Integration Tool comes with a few MonoBehaviour Scripts which provide certain functionality to GameObjects being:

FPSPlayerController

This script is pre-attached to the FPS Player Prefab and provides movement mechanics with the related Character Component as well as audio-visuals.

Field Name / Heading	**Data Type**	**Description**
Character Camera	Camera	The camera that will be influenced with bobbing effects.
Movement Input Axes		
Movement X Name	String	Name of the axis in the Input Manager used for tracking left and right FPS Player GameObject movements.
Movement Y Name	String	Name of the axis in the Input Manager used for tracking forwards and backwards FPS Player GameObject movements.
Movement Input Keys		
Jump Key	KeyCode	The key or button to make the FPS Player GameObject Jump.
Run Key	KeyCode	The key or button to hold down to enable the FPS Player GameObject to Run.
Mouse Input Keys		
Mouse X Name	String	Name of the axis in the Input Manager used for tracking horizontal mouse movements to rotate the FPS Player GameObject left and right.

Mouse Y Name	String	Name of the axis in the Input Manager used for tracking vertical mouse movements to rotate the FPS Player > Environment Camera GameObject upwards and downwards.
Movement Speeds		
Walk Speed	Float	The movement speed of the FPS Player GameObject when walking.
Run Speed	Float	The movement speed of the FPS Player GameObject when running. Often is set to a greater value than the Walk Speed field.
Mouse Sensitivities		
Mouse X Sensitivity	Float	The sensitivity multiplier for horizontal mouse movements.
Mouse Y Sensitivity	Float	The sensitivity multiplier for vertical mouse movements.
Aero Mobility		
Jump Force	Float	The upwards vector force applied to the FPS Player GameObject when jumping. A higher value leads to higher jumping.
Aero Mobility Multiplier	Float	The multiplier for effecting forwards, backwards, left and right FPS Player GameObject movements when it is airborne.
Gravity Multiplier	Float	The multiplier of the downwards vector force applied to the FPS Player GameObject when airborne.
Camera Bobbing Effects		

Walk Cycle Rate	Float	The per second rate of the walk cycle used for camera bobbing and footstep audio while walking or running.
Is Camera Bobbing Enabled	Tickbox (Boolean)	If ticked (true), camera bobbing will be activated. If unticked (false), camera bobbing will not occur.
Camera Bob Intensity Moving	Float	Defines the walk cycle intensity of the camera bobbing when moving.
Camera Bob Intensity Landing	Float	Defines the intensity of the camera bob when landing after being airborne.
Audio		
First Footstep Audio	Audio	The audio that is played in sequence of the walk cycle to resemble the initial footstep sound.
Second Footstep Audio	Audio	The audio that is played in sequence of the walk cycle to resemble the alternative footstep sound.
Jump Audio	Audio	The audio that is played when jumping.
Land Audio	Audio	The audio that is played when landing after being airborne.

WeaponSpace

This is pre-attached to the Weapon Space GameObject of the FPS Player Prefab and provides it with weapon related capabilities while allowing other vital systems to connect with it via the available fields.

Field Name / Heading	Data Type	Description
Character Controller	Character	The Character Controller component of the FPS

	Controller	Player GameObject.
Weapon Collection	WeaponCollection	The WeaponCollection ScriptableObject that is implemented to the FPS Player GameObject.
Camera Ray Spawn	Transform	Transform of the GameObject used as a camera, where the firing ray is projected in a forwards direction.
FPS Player	Transform	Transform of the FPS Player GameObject. Needed for the WeaponSpace Component to access details of the FPS Player GameObject.
Blood Splatter Impact	Prefab	To instantiate at the Raycast impact when shooting a GameObject of the EffectFromPlayer Layer.
Input Keys		
Input Fire	KeyCode	The key or button used to fire the wielded weapon if its Firing Type field is set to Semi.
Input Auto Fire	KeyCode	The key or button used to fire the wielded weapon if its Firing Type field is set to Auto. Often set the same as Input Fire.
Input Reload	KeyCode	The key or button for reloading the wielded weapon.
Input Switch	KeyCode	The key or button to switch the current weapon for the next.
Input Aim	KeyCode	The key or button for aiming the wielded weapon.
Input Run	KeyCode	The key or button to hold down to enable running. Used with the input axes defined in the **Movement X Axis Name** and **Movement Y Axis Name** fields.

Mouse Influence Axes		
Mouse X Influence Name	String	Name of the axis in the Input Manager used for tracking horizontal mouse movements to effect Weapon Animations when rotating the FPS Player GameObject.
Mouse Y Influence Name	String	Name of the axis in the Input Manager used for tracking vertical mouse movements to effect Weapon Animations when rotating the FPS Player GameObject.
UI Objects		
Crosshair Space	Image	The Image Component that holds the crosshair Sprite of the current weapon.
Mag Ammo Count	Text	Text that is updated to display the amount of ammo loaded in the equipped weapon.
Total Ammo Count	Text	Text that is updated to display the total amount of ammo in possession for the equipped weapon.
Ammo Icon Space	Image	The Image Component that holds the ammo icon Sprite of the current weapon.
Layer Names		
Fire Raycast Ignorable Layer Names	Array Dropdown	
Size	Integer	Number of **Element *** fields available for Strings.
Element * (Depends on array index)	Array of Strings	The Layers that should not be hit by the firing Raycast.

CollectableObject

This Component is designed to be applied to GameObjects that simulate collectables. When in contact with the FPS Player GameObject, the collectable will despawn while enabling a weapon and/or increasing the total ammo amount, depending on its configuration.

Field Name / Heading	**Data Type**	**Description**
Collection Type	Options	Defines the purpose of the collectable with options: Weapon / Ammo. If set to Weapon, this collectable can enable/disable the equipping of a weapon and affect the amounts of ammo in possession. Designed for weapon pick-ups. If set to Ammo, the collectable can only control total ammo amounts. Designed for ammo box pick-ups.
Weapon	Weapon ScriptableObject	**This field is displayed only when Collection Type is set to Weapon.** The weapon that will be affected by the collection.
Ammo	Ammo ScriptableObject	**This field is displayed only when Collection Type is set to Ammo.** The ammo that will be affected by the collection.
Enable	Tickbox (Boolean)	**This field is displayed only when Collection Type is set to Weapon.** Defines whether the weapon should be enabled/disabled on collection if not already in such status. If ticked (true), enable the specified weapon. If not ticked (false), disable the specified weapon.

Ammo In Weapon	Integer	**This field is displayed only when Collection Type is set to Weapon.** Amount of ammo pre-loaded into the newly enabled weapon on collection. If this value exceeds its Capacity field, the value will be capped.
Add To Ammo Total	Integer	Amount of ammo to be added to the total ammo in possession.
After Collection Object	Prefab	The GameObject that will be temporarily spawned once the collectable is collected.
After Collection Despawn Time	Float	Time in seconds until the After Collection Object despawns after spawning.

Grenade

The Grenade Component can be used on projectile Prefabs that need to detonate, like a grenade.

Field Name / Heading	Data Type	Description
Explosion Object	Prefab	The GameObject that spawns when the projectile GameObject despawns. Such assigned GameObject is often used to simulate an explosion.
Detonation Time	Float	The time from launching until the projectile detonates.
Is Detonate On Collision	Tickbox (Boolean)	If ticked (true), the projectile will detonate before the Detonation Time if it contacts another collider. If unticked (false), the projectile will only detonate through the Detonation Time.

Explosion Existence Time	Float	The amount of time the Explosion Object will exist until it is destroyed.

Explosion

The Explosion Component is for use with GameObjects that simulate an explosion involving Particle Effects. It also applies specified forces to nearby Rigidbody GameObjects when spawned.

Field Name / Heading	Data Type	Description
Force	Float	The amount of sudden outwards force applied to Rigidbody GameObjects within the Radius.
Radius	Float	The radius of the force effect from the central point of the explosion.
Upwards Modifier	Float	Offsets the centre of the explosion force downwards with values over 0. This forces Rigidbody GameObjects in a more upwards nature.
Particle Effect Size	Float	The size multiplier of the involved Particle Effects.

PlayerHealth

This Component is designed to manage the health of player GameObjects, assists with respawning the player and applying visual effects to the HUD to express damage being inflicted.

Regarding the respawning, if the current health of the player reaches zero, the player is repawned at its start position and rotation.

Field Name / Heading	Data Type	Description
Maximum Health	Float	The most health the player can possess and will begin with in Play Mode.

Health Text	Text	The Text that is updated to display the amount of health possessed by the player.
Low Health Overlay	Image	Image that resembles the HUD overlay that becomes more visible as health is reduced.
Damage Overlay	Image	The Image used for the HUD overlay that flashes when damage is being inflicted onto the player.
Maximum Damage Overlay Opacity	Float Range	The starting opacity of the Damage Overlay when damage infliction occurs before gradually fading.
Damage Overlay Fade Duration	Float Range	Duration in seconds of the Damage Overlay fading out after damage is inflicted.

NonPlayerHealth

NonPlayerHealth is for tracking the health and death of the NonPlayer GameObject especially for the NonPlayer Zombie.

Field Name / Heading	Data Type	Description
Maximum Health	Float	The most health the non-player character can possess and will start with in Play Mode.

NonPlayerController

This Component coordinates the movement of the NonPlayer GameObject through guiding its navigation AI. This is also designed for the NonPlayer Zombie.

Field Name / Heading	Data Type	Description
Target Player Transform	Transform	The Transform of the player GameObject
Non Player Damage	NonPlayerDa	For the NonPlayerDamageInfliction Component of

Infliction	mageInfliction	the non-player GameObject.

NonPlayerAudio

NonPlayerAudio manages the audio of the non-player character and influences the variability of the audio in relation to pitch and timing.

Field Name / Heading	Data Type	Description
Periodic Audio Clip	AudioClip	The AudioClip that is played in random intervals.
Damage Audio Clip	AudioClip	The AudioClip that is played when damaged.
Death Audio Clip	AudioClip	The AudioClip that is played during death.

NonPlayerDamageInfliction

This Component is applied within non-player GameObjects where a Collider with Is Trigger set to true is present and allows them to cause damage onto the player.

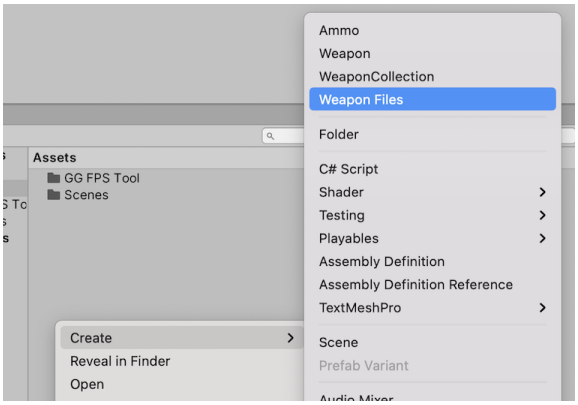
Field Name / Heading	Data Type	Description
Non Player Animator	Animator	The Animator of the parent GameObject.
Non Player Health	NonPlayerHealth	The NonPlayerHealth of the parent GameObject.
Target Player Collider	Collider	For the Collider of the player GameObject.
Target Player Health	PlayerHealth	For the PlayerHealth of the player GameObject.
Damage Health Loss	Float	The amount of damage that is inflicted onto the target player.

Damage Cooldown Duration	Float	The delay in seconds until the damage can be applied or reapplied onto the target.
Blood Splatter	Prefab	For a prefab with blood splatter particle effects.

Create Options

The Create Options allow ScriptableObjects and file systems to be easily created.

Access these options by right-clicking within a Project tab, hover over Create from the pop-up menu, then select one of the options towards the top of the second menu.



The ScriptableObjects that can be created using this feature involves:

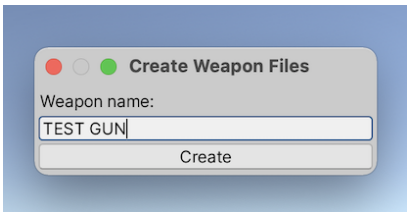
- Weapon - Creates a new Weapon ScriptableObject.
- WeaponCollection - Creates a new WeaponCollection ScriptableObject.
- Ammo - Creates a new Ammo ScriptableObject.

Weapon Files

The Weapon Files option works differently compared to the individual ScriptableObject creation, it generates all the files that the new weapon depends on and links them together automatically.

This is achieved by this system duplicating content from the Default Weapon folder while adjusting the naming, similar to templating.

After navigating to and selecting Weapon Files within the Create Options, a window menu should appear over the Scene tab titled Create Weapon Files with the field labelled **Weapon name:**.



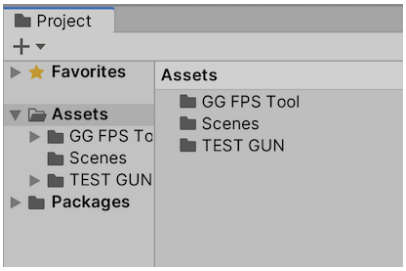
This window can be closed by clicking the usual window close icon.

Enter the name of your weapon and ensure the start and end characters of the string are not spaces.

This should reveal the Create button below the field, which if pressed will generate the weapon's files.

This will create a folder named after the specified weapon name and contains these files:

- Animations folder - Contains the weapon's animations used in Base and Aim Influence Layers.
- The weapon's ScriptableObject
- The weapon's Animator Controller
- The weapon's Prefab

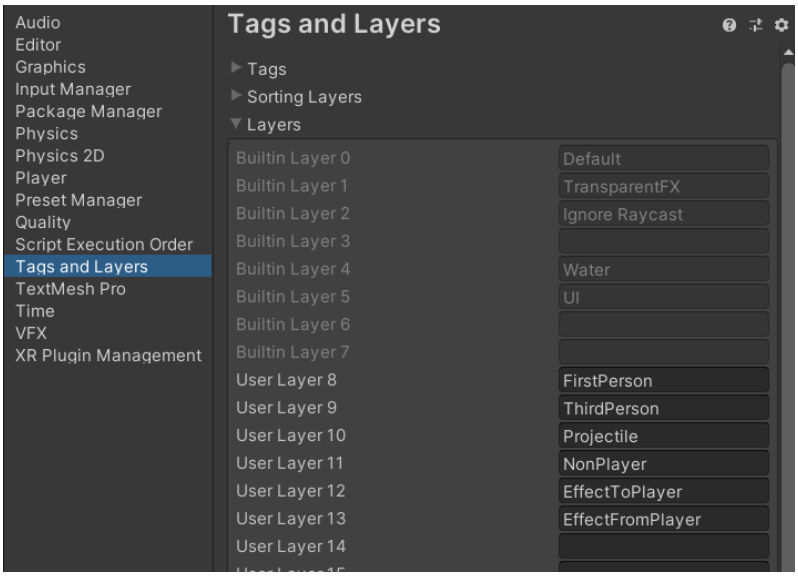


Layers (GameObject)

Layers are designed to easily categorise GameObjects and can be defined via **Tags and Layers** in the Project Settings Tab.

The FPS Integration Tool involves these Layers:

- Builtin Layer 0: **Default** - Used in the standard scenario.
- Builtin Layer 5: **UI** - Used by Canvases and their child GameObjects.
- User Layer 8: ‘FirstPerson’ - For Culling Masks within the Environment Camera and Weapon Camera and is used on Weapon Prefabs.
- User Layer 9: ‘ThirdPerson’ - Is applied to the FPS Player Prefab and is used for the collisions with NonPlayer and EffectToPlayer.
- User Layer 10: ‘Projectile’ - For preventing projectiles from colliding with the FPS Player GameObject and other projectiles.
- User Layer 11: ‘NonPlayer’ - Used for simulating contact between non-player entities and the player.
- User Layer 12: ‘EffectToPlayer’ - For the colliders with Is Trigger enabled that deal damage to the player using the NonPlayerDamageInfliction Component.
- User Layer 13: ‘EffectFromPlayer’ - For the closely fitted colliders of non-player GameObjects which detects Raycasts from the player to simulate damage.



Defining the User Layer names in different fields from the ones described above may cause incorrect Layers being assigned to GameObjects and Prefabs, leading to issues with the gameplay.

To correct this malfunction through using the specified User Layer fields:

1. Delete the GG FPS Integration Tool folder.

2. Apply User Layer names to the correct fields as shown above.
3. Reimport the GG FPS Integration Tool folder by importing the package again.

If the intended User Layer fields are already taken by other Layers from your project, manually changing the Layers of the affected GameObjects and Prefabs would be the next best solution.

Intended use of the FPS Integration Tool

There are some conditions to consider when using this tool:

- This tool can be used in commercial and non-commercial projects so long as the product of such project is Built (where the project hosted in the Unity application is exported as an application itself known as a Build) from the Unity application.
- The tool cannot be used partly nor wholly for creating other tools that would be published (whether it is via the Unity Asset Store or not).
- The tool should be credited within the credits (or with the listed contributors of the project) of the Built application, where the sentence '**Made with the Grey Gear FPS Integration Tool available on the Unity Asset Store**' is displayed.
- Promoting the use of the FPS Integration Tool with the product in general (such as on websites) is recommended and greatly appreciated. The sentence '**Made with the Grey Gear FPS Integration Tool available on the Unity Asset Store**' can also be used for this case.
- Writing an honest review of this tool on the Unity Asset Store would also be fantastic!

Thank you for cooperating and using the FPS Integration Tool!