MANOEUVRE F.P.S.

23.01.2018

Mohammad Hamzah Kirmani
Walled City Infotech
WalledCityInfotech@gmail.com

INDEX

Thank You!

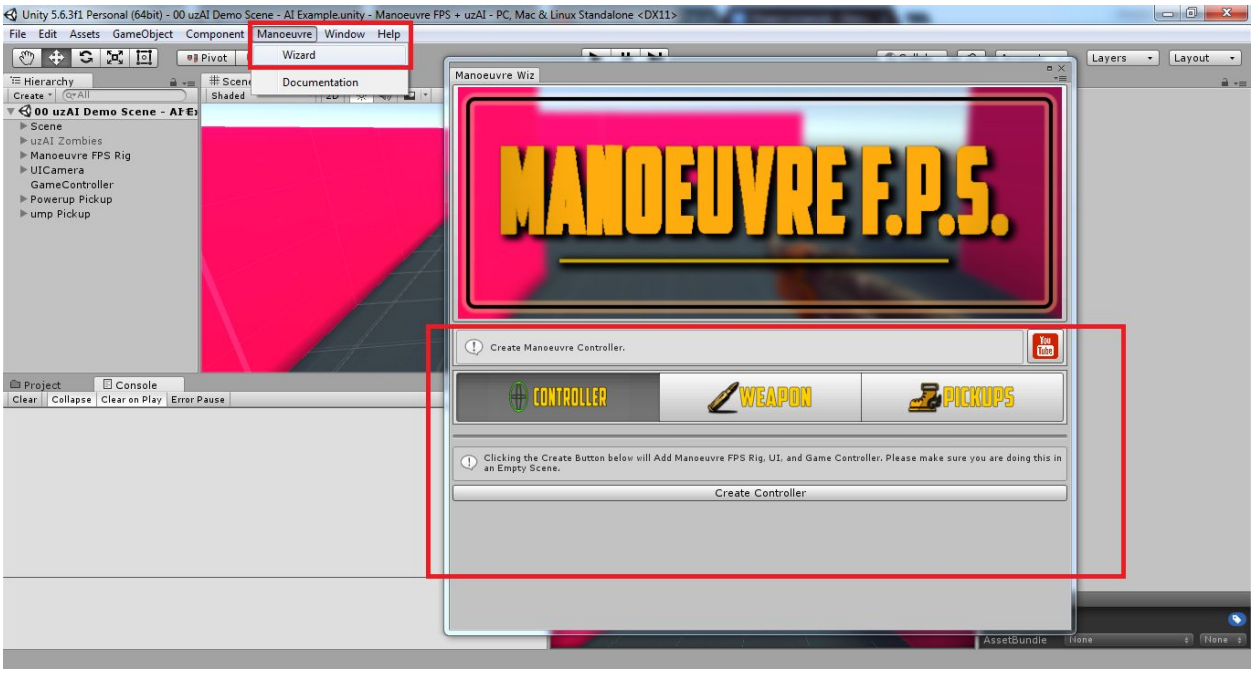
Thanks a lot for choosing **Manoeuvre FPS Controller**. With this controller you can seamlessly create a FPS Controller with Weapons, Powerups, MiniMap, Inventory etc.

How to Create a Controller from Scratch?

> Open ***Wizard*** by going to *Manoeuvre* > *Wizard*.
Make sure you are in an empty scene and also that you have selected the *Controller* Tab.

> Click on the **Create Controller** Button!

> That's it! :)



You might have already noticed that there are 3 objects added in your scene

- Manoeuvre FPS Rig
- UI Camera
- Game Controller

Awesome! You got yourself a working FPS controller in less than 10 Seconds ! Now let's explore these 3 in detail.

Manoeuvre FPS Rig

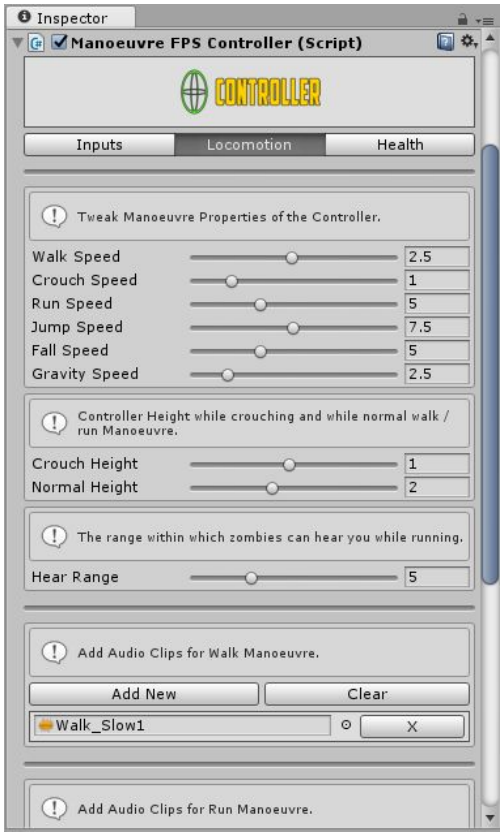
Before reading below, I really want you to go ahead and play the scene and make yourself comfortable with Manoeuvre FPS.

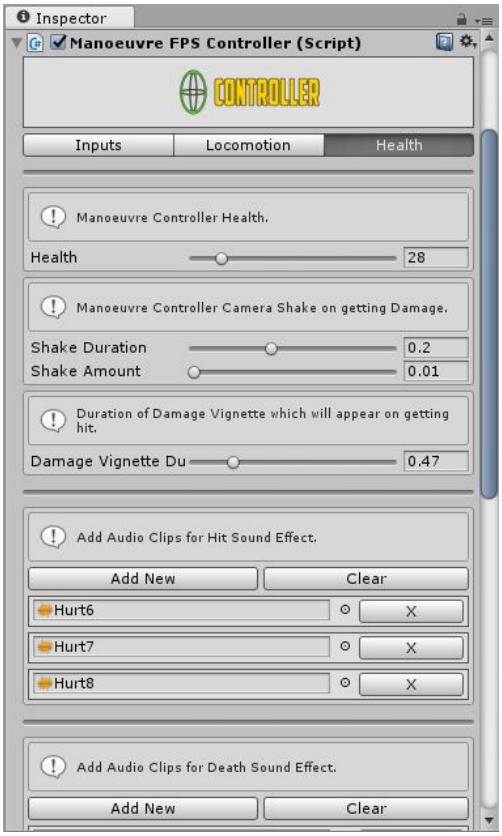
The **Manoeuvre FPS Rig GameObject** has a script named Manoeuvre FPS Controller.

The script's Inspector is designed by keeping in mind the ease of customization because it contains a **LOT** of properties and with the default Inspector it will become really hard to customize these properties.

There you will see, the whole controller script is broadly divided into 3 categories

- Inputs
- Locomotion
- Health





> Inputs -

The whole Inputs Tab is divided into 7 sections, but there will be more Inputs in future as this **Controller will be updated at least twice a month.**

- Axis Inputs : Horizontal and Vertical are used in player movement and Mouse ScrollWheel Axis is used in equipping next or previous weapon.

- Jump Input : Either define a button or a Key, if you will define both, Controller will use the Button. This Input is used to Jump.

- Crouch Input : Either define a button or a Key, if you will define both, Controller will use the Button. This Input is used to Crouch.

- Run Input : Either define a button or a Key, if you will define both, Controller will use the Button. This Input is used to Run.

- Weapon Inputs : All the inputs related to Weapon.
    - Shoot Button/Key : Define which Button/ Name you'd like to use for shooting.
    - Ironsight Key : Weapon will enter ironsight with this key.
    - Reload Button/Key : Reload the weapon if there's ammo left.
    - Next Weapon Button/Key : Un-equip current Weapon and equips **Next** weapon.
    - Previous Weapon Button/Key : Un-equip current Weapon and equips **Previous** weapon.

- Inventory Input : Opens the Inventory as long as you hold this Input and closes on release.

- Minimap Input : Zoom In and Out Inputs to zoom in and zoom out the Minimap Camera.

> Locomotion -

The Locomotion Tab is divided into 2 categories

- Controller's various speeds and heights Properties
- Controller's various sounds for both run and walk Manoeuvre.

- Controller Speeds
    - Walk Speed : How fast FPS Controller can move while Walking.
    - Crouch Speed : How fast FPS Controller can move while Crouching.
    - Run Speed : How fast FPS Controller can move while Running.
    - Jump Speed : How fast FPS Controller can Jump.
    - Fall Speed : How fast FPS Controller Fall after a Jump.
    - Gravity Speed : How fast FPS Controller return to ground while in Air.

- Controller Heights
    - Crouch Height : Controller's Height while crouching.
    - Normal Height : Controller's Height while normal Walk / Running.

- Hear Range [uzAI Specific field]

  A sensor trigger collider will be created whose radius will be changed according to Player's Run state. This radius will be the hearing Range of the nearby Zombies.

- Audio Clips

  - For Walk Manoeuvre - By Clicking the Add Button you can assign as many audio clips as you can. These will be played while you are **walking**.

  - For Run Manoeuvre - By Clicking the Add Button you can assign as many audio clips as you can. These will be played while you are **Running**.

> Health -

Similar to Locomotion Tab, the Health Tab is also divided into 2 categories.

- Controller's Health and its Properties
- Sound Effects for damage.

- Health and other Properties

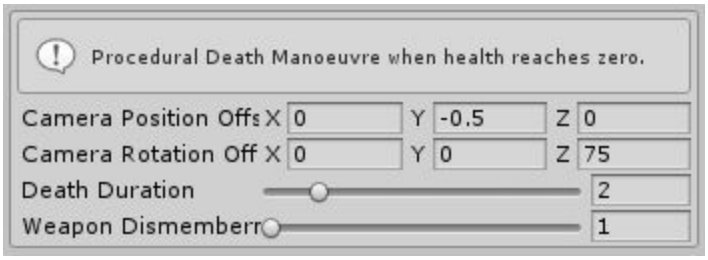
  - Health : Define the Total Health of the Controller

  - Shake Duration : How long Camera will keep shaking on receiving Damage.

  - Shake Amount : How much Camera will shake on receiving Damage.

- Hit Sound Effects : Click the Add Button and you can Add as many sounds as you want. These will be **played on receiving damage**.
- Death Sound Effects : Click the Add Button and you can Add as many sounds as you want. These will be **played on Death**.

- Procedural Death Manoeuvre : Instead of animating the camera, I have added a unique mechanic to the controller, this is a smooth lerp of Manoeuvre FPS camera's current position and rotation to the defined position and rotation which gives a nice effect of death. This is completely procedural and thus you can just define the Offsets and the death Manoeuvre will be automated accordingly.

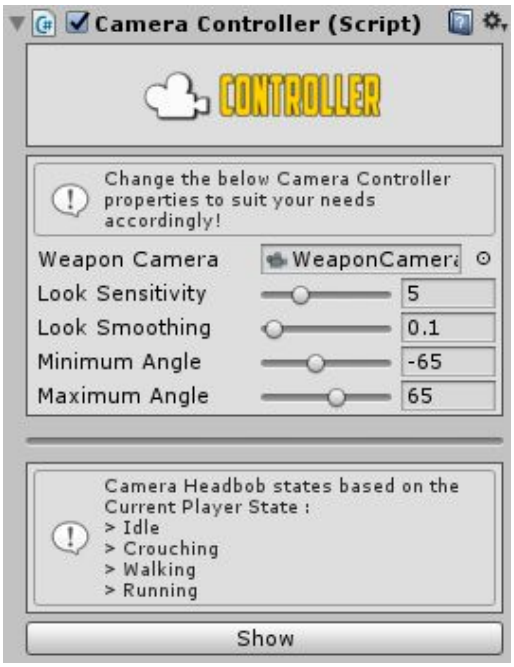


Here **Death Duration** is the lerp duration of Death Manoeuvre and **Weapon Dismemberment** is the force applied to the current equipped weapon (if there's any) to make it fly away from the Player.

Now if you will drill down into the Hierarchy of the Manoeuvre FPS Rig, you will see **Three Cameras** : FPS Camera, Weapon Camera, Minimap Camera.

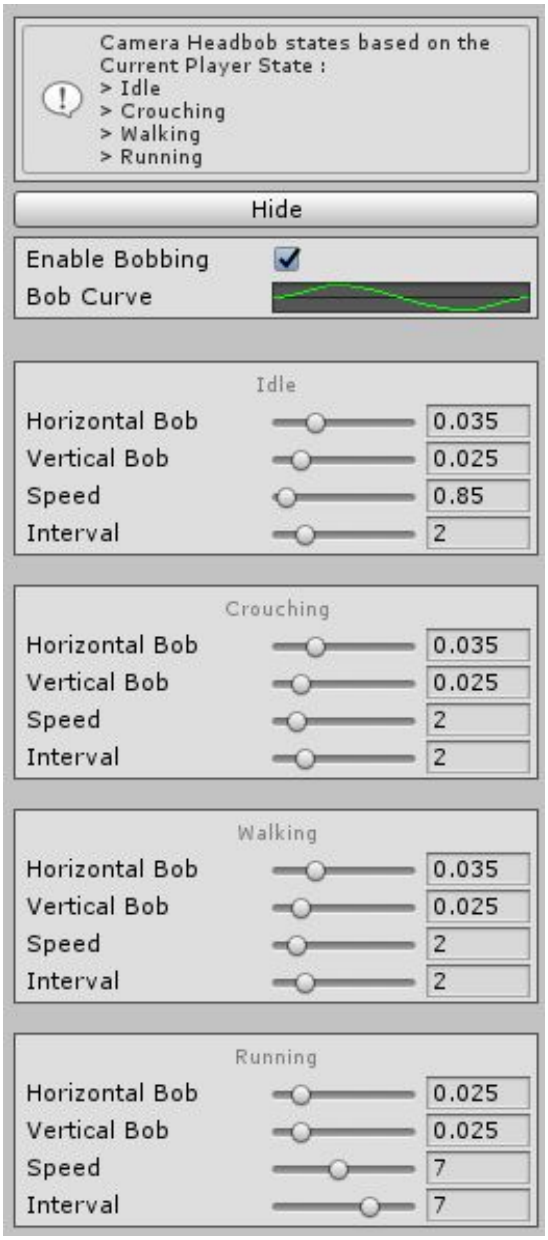
The FPS Camera is having the Main Camera Controller Script attached which controls all the three cameras. It is divided into 2 Parts, Camera Properties and Camera Headbob Properties.

- **Camera Properties**



- Look Sensitivity : Camera Look speed.
- Look Smoothing : Camera Look smoothing value.
- Minimum Angle : Camera Look's Minimum Clamping Value
- Maximum Angle : Camera Look's Maximum Clamping Value

- **Camera Headbob Properties** : Click the Show button to reveal these properties in the Inspector.



We are Bobbing the camera using our awesome **Universal Bob Script**. This is the same script we are using in **Weapon Bobbing** as well, this time we have changed the curve and the below properties.

Camera's bobbing will only happen if the **Enable Bobbing is true.**

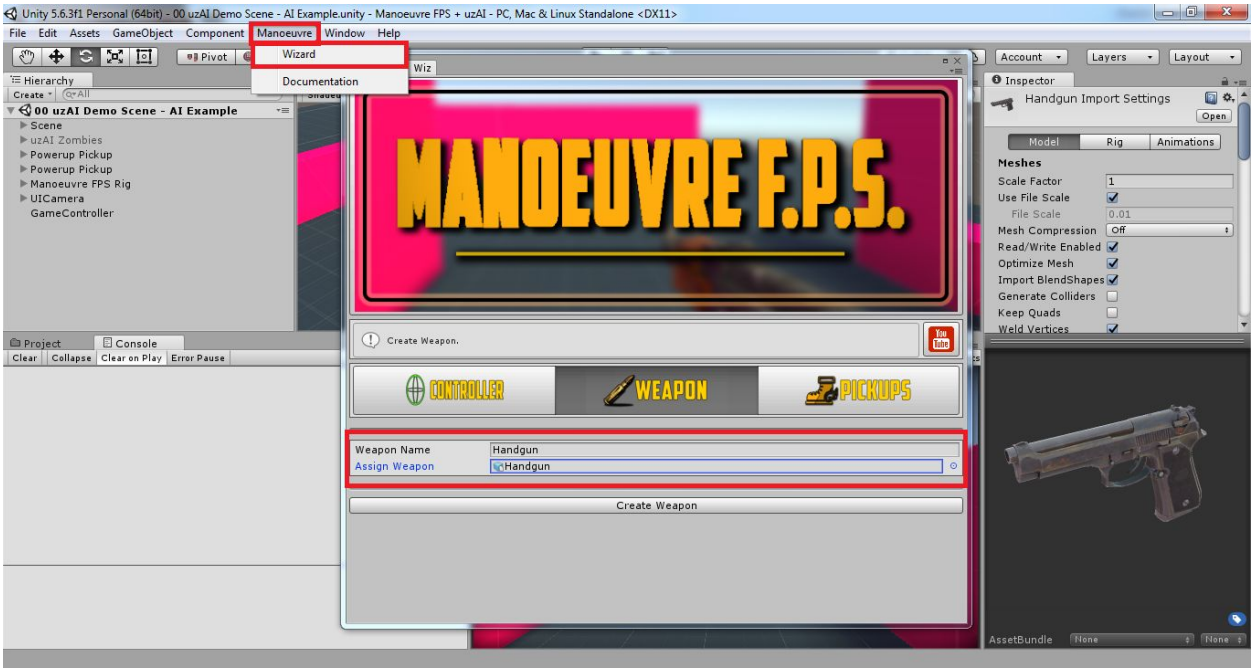
Bob Curve is the curve in which Camera bob will follow.

Now, for all the 4 states i.e Idle, Crouching, Walking and Running, the Camera Bobbing's different properties are defined.

- Horizontal Bob : Bob amount in X-Axis.
- Vertical Bob : Bob Amount in Y-Axis.
- Speed : Bob Speed.
- Interval : How many Bobs in a single cycle.

How to Create a Weapon?

Creating a Weapon as easy as it sounds. Just Open the **Wizard** from Manoeuvre > Wizard, select the **Weapons Tab.**



Write the Weapon's Name - Make sure it is unique and no 2 weapons share the same name!

Now Drop the Weapon object in **Assign Weapon** field.

Now hit Create Weapon and that's it! :)

Understanding the created Weapon

The weapon just created contains 2 scripts : **Weapon Shooter** and Weap**on Procedural Manoeuvre.**

> Weapon Shooter

Weapon Shooter is a very complex script as it contains tons of properties and again, to make is ultra easy, I have written the whole Inspector from scratch and in that process I have divided all the Properties into 7 major categories

- Shooter
- Force
- Muzzle
- Bullet Hit
- Recoil
- Sounds
- Ironsight

But... on the very top of it is a "**Save**" button. You are advised to use that to save the transform position and rotation **at RUNTIME.**

Weapon Name : It is the unique identifier of this weapon, **please make sure no 2 weapons have the same name.**

Weapon State : This will be updated at runtime to whether Idle, Firing or Reloading.

Now let us discuss all these 7 categories in detail.

Shooter : This contains all the shooting/firing related properties of this weapon.

- Hit Mask : Assign Layers which this weapon can shoot.
- Shooting Range : How far this weapon can shoot?
- Fire Rate : How many Bullets this weapon can fire per seconds.
- Shake Camera : If true, camera will shake while shooting.
- Shake Amount : How much Camera should shake while shooting?
- Hear Range : [*uzAI Specific variable*] Define from how far **zombies** can hear this weapon's Shooting sound.

Force : Add forces to rigidbodies.

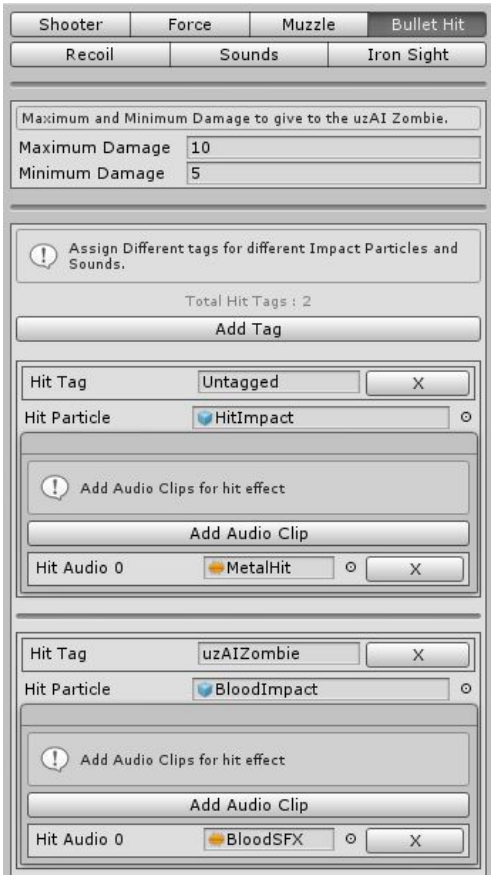
- Apply Force : If true, then only the forces will be applied to the rigidbodies.
- Force Amount : How much force a single bullet of this weapon can apply on rigidbody.

Muzzle : Muzzle flash to show while firing.

- Muzzle Flash : Muzzle Particle to show while firing.
- Muzzle Flash Location : There is a green gizmo in the scene view. Please adjust its position and **assign that** as the Muzzle Flash Location **if** it's not already.

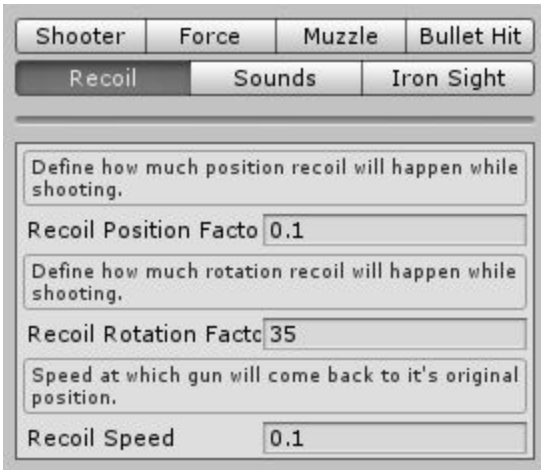
Bullet Hit : Damage and different Impact properties for differently tagged objects.

- Maximum Damage : The max amount of damage you can cause with a single bullet. **[uzAI Specific]**
- Minimum Damage : The min amount of damage you can cause with a single bullet. **[uzAI Specific]**
- **Add Tag button** will add a new entry. Now you can specify the tag and hit particle, for this tag, you can add infinite sound fx by clicking the **Add Audio Clip button**. Everytime a random sound fx will be played. By default there are 2 Hit Tags, each labeled as **Untagged** with a HitImpact particle and metal hit sound fx and **uzAIZombie** for uzAI zombies. There's also a BloodImpact particle and similarly a blood sound fx.



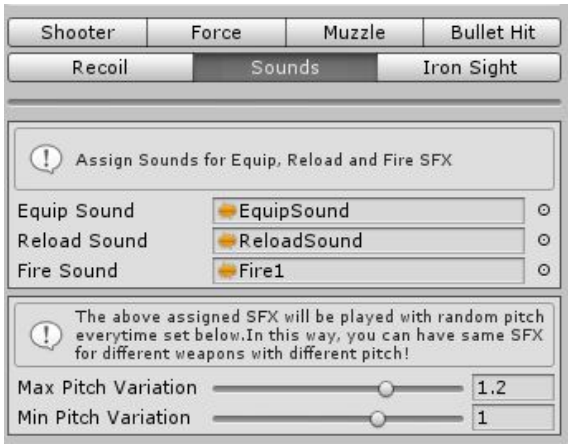
Recoil : A procedural recoil spring for the Weapon.

- Recoil Position Factor : How much recoil while shooting in respect of position.
- Recoil Rotation Factor : How much recoil while shooting in respect of rotation.
- Recoil Speed : Speed at which gun will come back to its original position. If you find your gun going back too much try increasing the Recoil Speed.



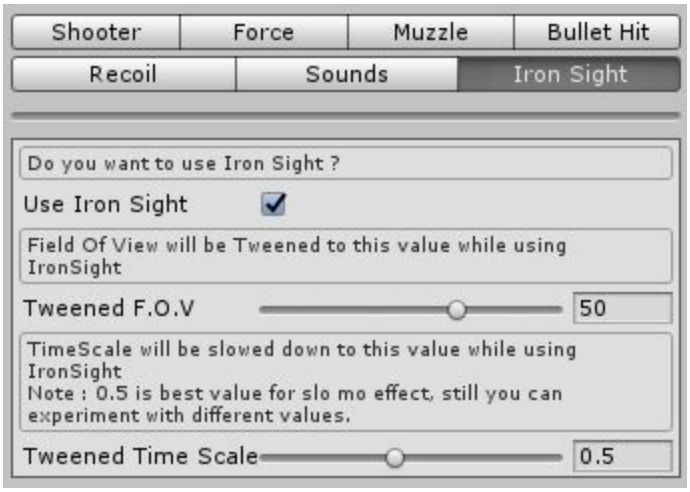
Sounds : These sounds will be played while firing, reloading and equipping or unequipping the weapon.

- Equip Sound : Sound to be played while equipping and unequipping.
- Reload Sound : Sound to be played while Reloading.
- Fire Sound : Sound to be played while Firing.
- These sound fx will always be having a different pitch and you can adjust that value by assigning the **Max Pitch Variation** and **Min Pitch Variation.**



Iron Sight : A unique Ironsight mechanism which no other template offers. While holding down the input button / key of the Ironsight, the game will go into the slo-mo and a Vignette effect will be enabled and the FPS Camera's FOV will be tweened.

- Use Iron Sight : If true, then only player can have an Ironsight.
- Tweened FOV : While using Ironsight, FPS Camera's **current** FOV will be tweened to the value defined. By default it's 50.
- Tweened Time Scale : Define how slow you want time to be while using Ironsight. 0.5f is the value I use personally for my games, still I recommend experimenting with different values.



Weapon Procedural Manoeuvre : Now after Weapon Shooter script we come to another script of equal Importance which is attached on the same object with Weapon Shooter. As the name suggests, this is the main brain of the complete procedural behaviour of the weapon. *More Behaviours will be added to this script in future on user requests and personal analysis.*

This script is also pretty much complex so I wrote it's Inspector from scratch for ease of use. Almost every Property is well commented in the Inspector as well. :)

For easy understanding and implementation, I have divided all the properties majorly in 4 categories :

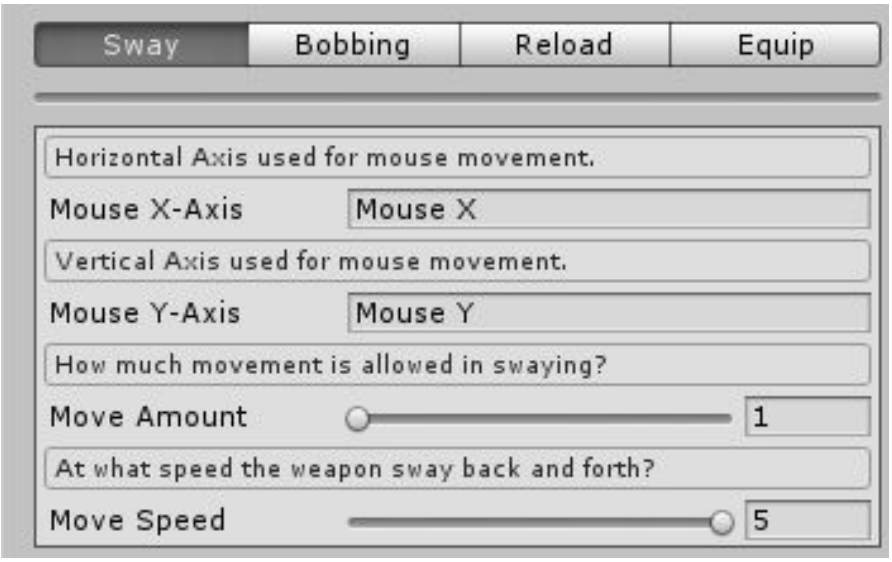
- Sway
- Bobbing
- Reload
- Equip

On top of the Tabs, there is a Debug Transform boolean, if it is true, all 4 properties will be disabled and weapon will enter into debug mode. Now you can adjust the position and rotation of weapon.

Now let's see these 4 Procedural behaviours in detail

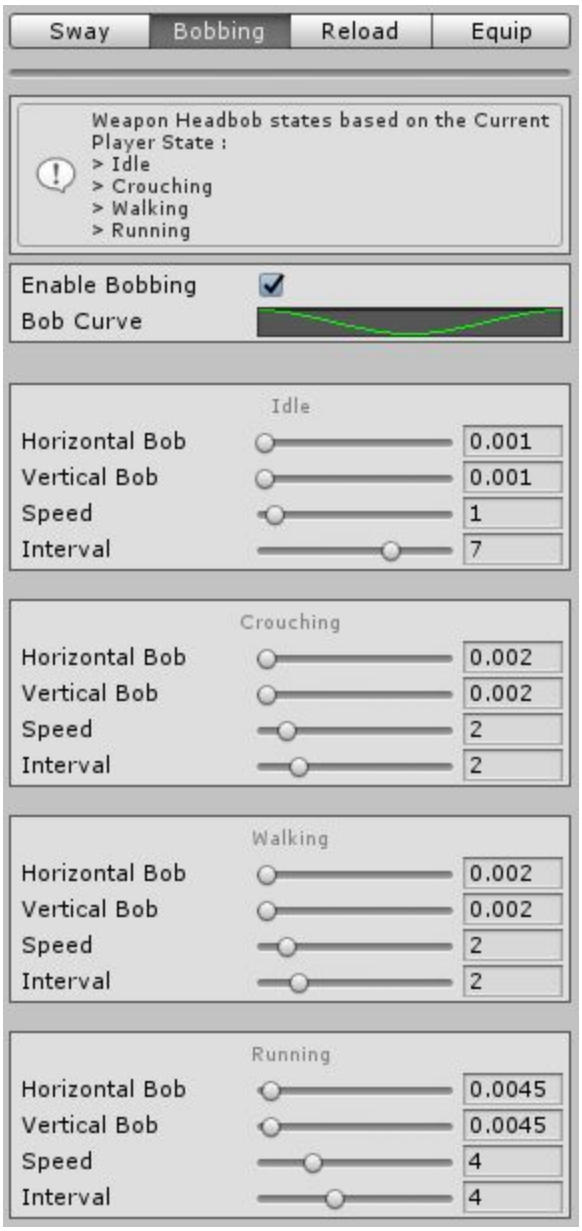
Sway : Weapon's position will be lerped towards the movement of the camera.

- Mouse X-Axis : Axis used for Horizontal movement of mouse.
- Mouse Y-Axis : Axis used for Vertical movement of mouse.
- Move Amount : How much gun should move with the mouse movement.
- Move Speed : How fast it should move with the mouse and gets back to the original Position. Keep this speed at max value for realistic behaviour, but I recommend experimenting with this value.



Bobbing : Realistic Weapon movement according to current player state i.e Idle, Crouching, Walking and Jumping. It is made possible by our **Universal Bob script** which I have explained earlier in **Camera Bobbing.**

- Enable Bobbing : Camera will only bob if this is true.
- Bob Curve : Define the movement path curve of the Weapon Transform.



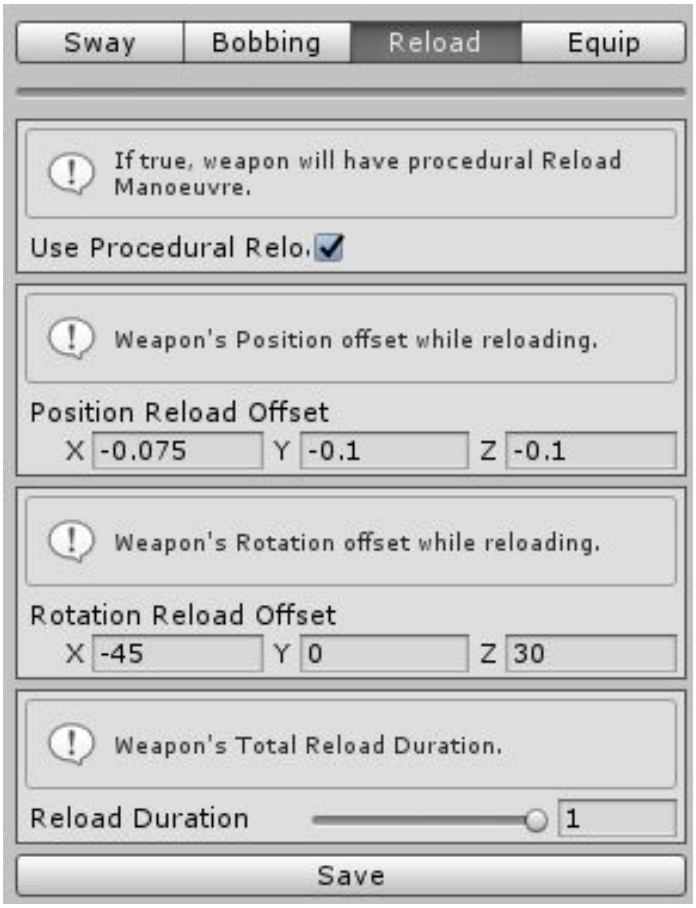
For all the 4 states, just simply change the below 4 properties and the Weapon will move accordingly.

- Horizontal Bob : Bob amount in X-Axis.
- Vertical Bob : Bob Amount in Y-Axis.
- Speed : Bob Speed.
- Interval : How many Bobs in a single cycle.

Reload : This is a procedural Reload manoeuvre where it makes a procedural animation of the weapon being reloaded.

Note : Please edit the below properties at Runtime and **don't forget to hit that Save Button** once you are happy with the procedural behaviour!

- Use Procedural Reload : If true, the weapon uses procedural reload Manoeuvre.
- Position Reload Offset : Weapon current position will be offsetted to this position.
- Rotation Reload Offset : Weapon current rotation will be offsetted to this rotation.
- Reload Duration : Total duration of this Reload Manoeuvre.
- Save Button : Save whole settings [only works in RUNTIME]



Equip : Procedural Equip Behaviour similar to that of reload Manoeuvre. The difference is in the Reload weapon transform **moves towards** the offset pos and rot, but in Equip, the weapon transform **moves from** the offset pos and rot towards the original pos and rot.

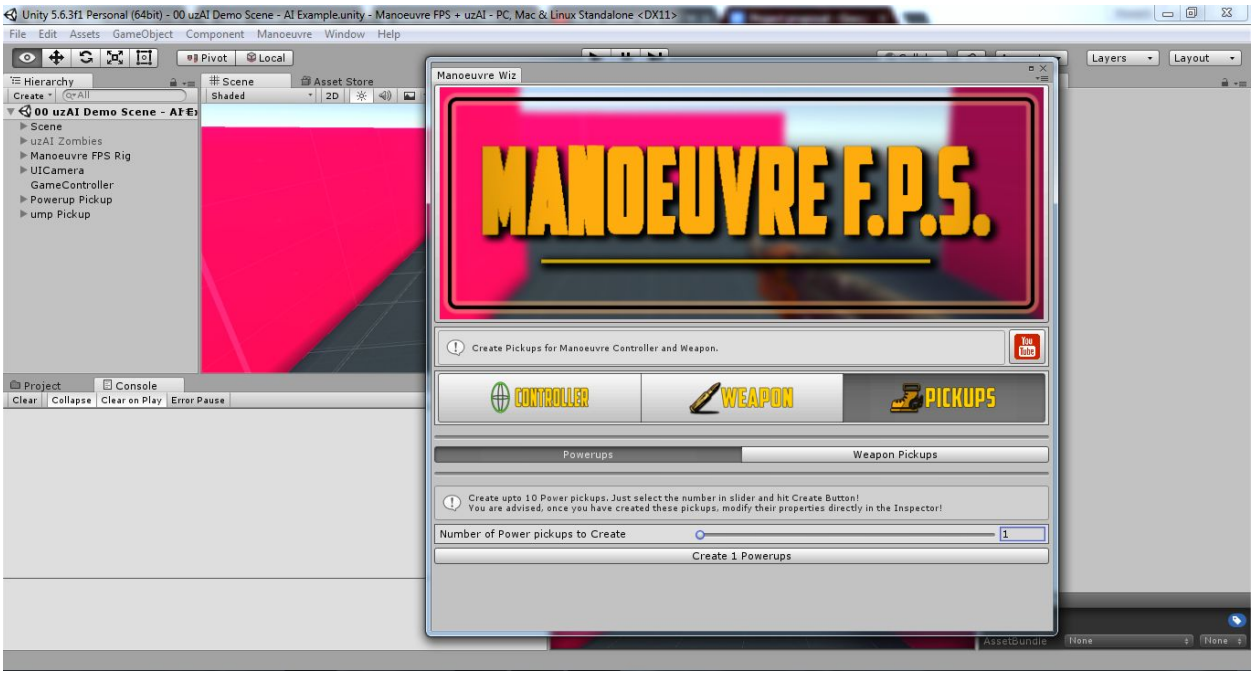
Note : Please edit the below properties at Runtime and **don't forget to hit that Save Button** once you are happy with the procedural behaviour!

- Equip Position Offset : Offset position from where the weapon moves to its original position.
- Equip Rotation Offset : Offset rotation from where the weapon rotates to its original rotation.
- Equip Duration : Total Equip / UnEquip Manoeuvre Duration.

How to Create a Power-Ups Pickup?

> Open ***Wizard*** by going to Manoeuvre > Wizard.
Make sure you have created a Controller already by following previous steps.

> Click on the **Pickups** Button!



> Now select **Powerups** Tab.

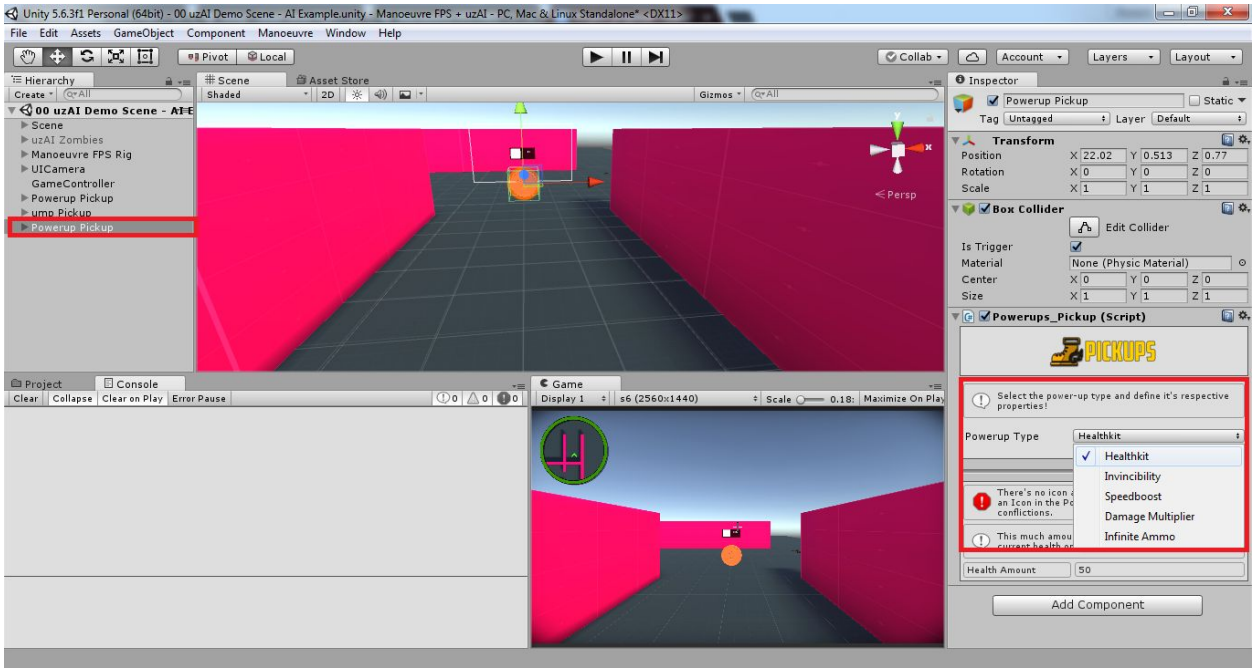
> Now, specify how many pickups you want to have in your scene. You can create upto 10 Powerups' pickups.

> Once specified, just hit the Create **Powerups Button.**

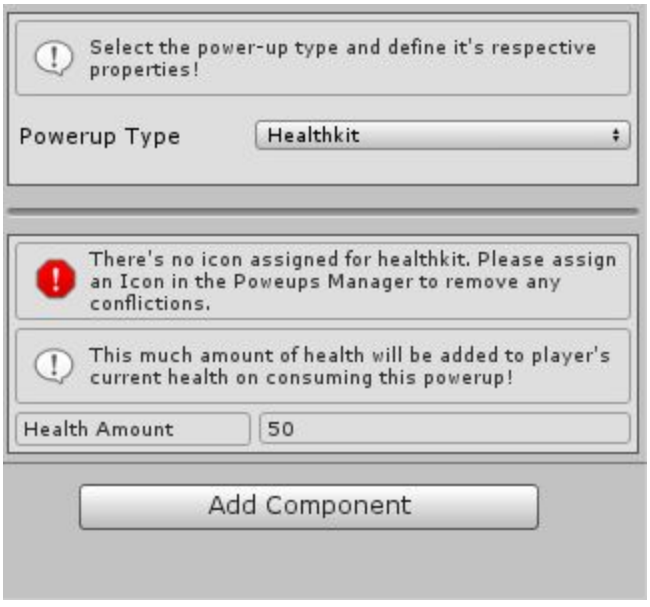
How to Customize Created Powerups?

> After you have created the Powerups by following previous steps, select it in the Hierarchy.

> In the Inspector select the **Powerup Type** and choose between any five from the dropdown**.**



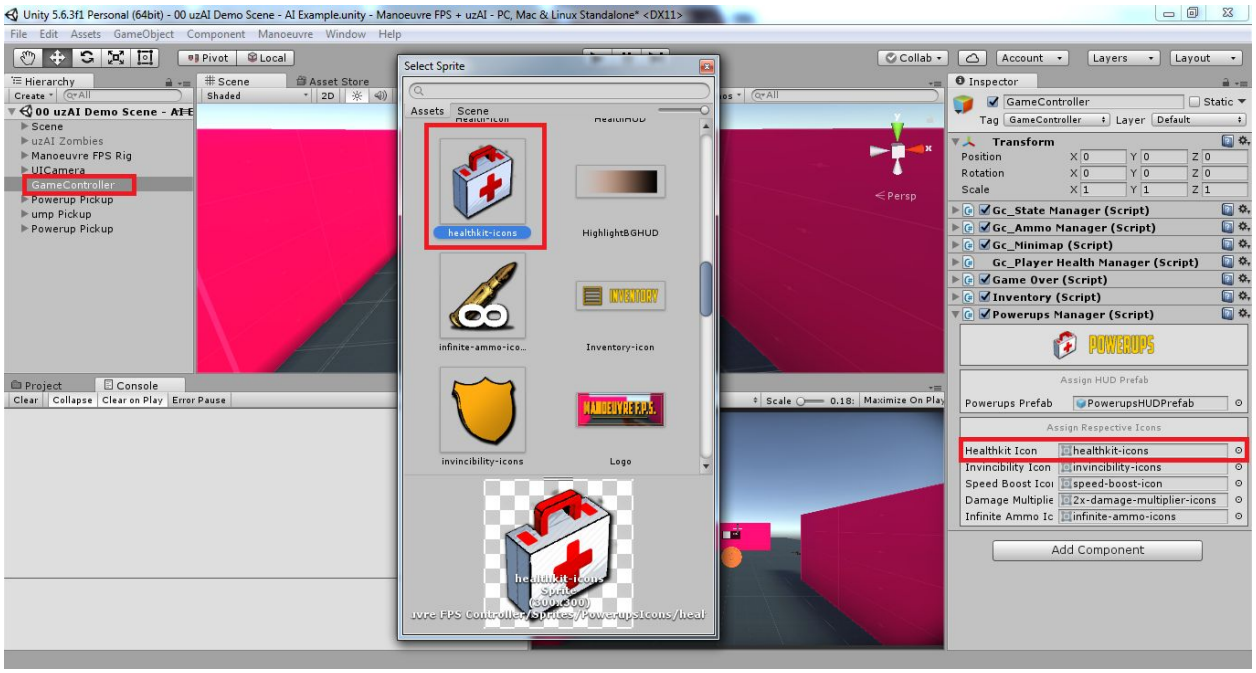
> After you have selected (example **Healthkit**) the PowerUp type from the dropdown, you ***might*** see an error :



> It simply means that this power has no Icon assigned. This **icon will be displayed in HUD and Inventory** whenever you **consume** this power in Game.

> For easy re-skinning and debugging purposes, I have made an error to pop whenever there's no Icon assigned so that you don't have to spend hours finding what's missing, where's missing, etc.

> To remove this error, simply Select the **GameController** in the Heirarchy and Assign the Respective icons to the PowerUp under Powerups Manager Script.



> You **must Assign all the icons** in Powerups Manager. There's plenty of them which comes with Manoeuvre FPS as an awesome Placeholder!

What are these PowerUps ?

> There's a total of 5 PowerUps which comes with Manoeuvre FPS as of now. I will be surely adding more on user requests.

- Healthkit
- Invincibility
- Speedboost
- Damage Multiplier
- Infinite Ammo

Healthkit -

> **Healthkit** is a health pickup which Player can consume whenever he wants.

> It will **add the Amount** you have specified as **Health Amount** in **current health of the player.**

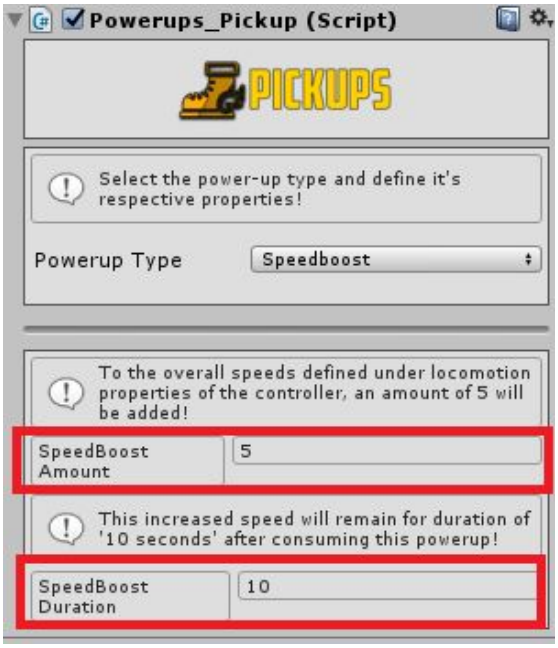
Invincibility -

> Player will remain Invincible to all incoming damages for the Duration you have specified as **Invincibility Duration** after consuming this powerups.

Speedboost -

> Player will be having an increased amount of walk / crouch / run speed as soon as he consumes this Power.

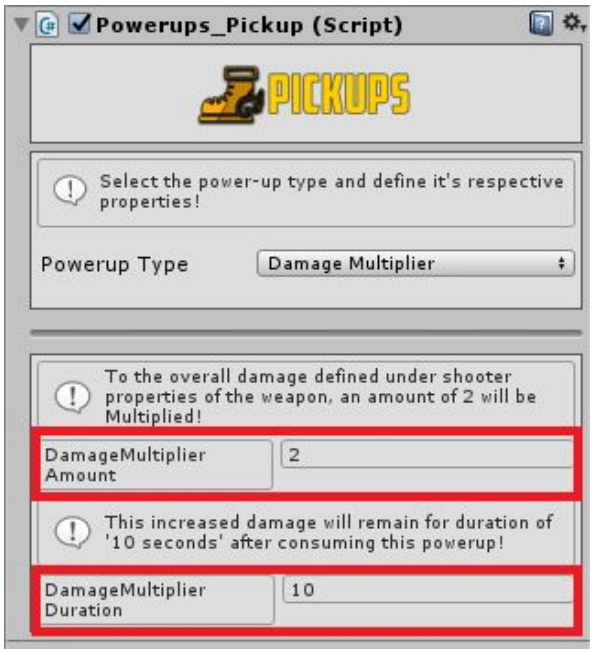
> Specify the **Speedboost amount** to be added in **the speed of the Player** and the **duration** of this power in **Speedboost duration.**



Damage Multiplier - [uzAI specific PowerUp]

> To the overall **Damage** defined under the Bullet Hit Properties of the Weapon , **Damage Multiplier Amount** will be added.

> This power will remain active for the **Damage Multiplier Duration** you have specified.



Infinite Ammo -

> Once this power is consumed, for the **Infinite Ammo duration** you have specified in the Inspector, the weapon will be having no loss in its Ammo Count.

Game Controller

Game Controller is the brain of the Manoeuvre FPS.

The scripts it contains handles all sorts of tasks like managing Player and Weapon States, weapons and their ammo, minimap, health, Inventory, etc. Please see below all the scripts Game Controller contains, their definitions, properties and what they are capable of.

State Manager

Script Location : Scripts/Controller/Game Controller/gc_StateManager.cs

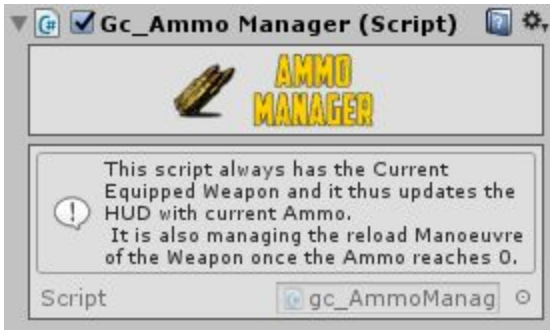
Description : This script looks and monitors Players State i.e Idle, Landing, Crouching, Walking, Running and Jumping.

It also looks at current equipped Weapon's State i.e Idle, Firing and Reloading.

Ammo Manager

Script Location : Scripts/Controller/Game Controller/gc_AmmoManager.cs

Description : This script holds the Current Equipped Weapon and keeps count of its current Ammo, capacity left, its clip count and simultaneously updates the HUD as well.



Minimap

Script Location : Scripts/Controller/Game Controller/gc_Minimap.cs

Description : Minimap can be seen inside the Health HUD. This Minimap is very simple yet very effective and looks very cool and add that extra touch in the Game.

However, it is completely up to you to include in your game or not. You can go ahead and remove it and it won't affect the game in any way.

Minimap contains functionality of zoom in and out and custom tagged objects you want to show in Minimap. I have divided this script in 2 parts for ease of use.

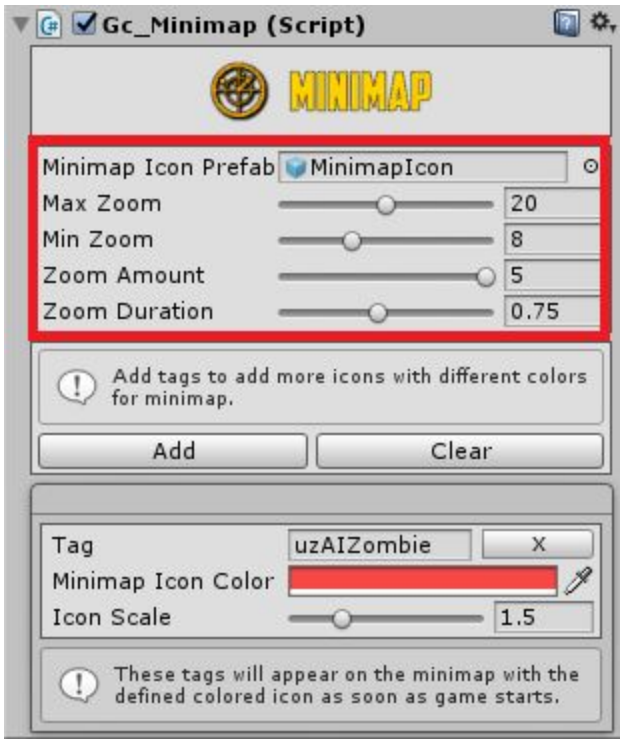
Part 1

> Max Zoom : Maximum zoom minimap can do.

> Min Zoom : Minimum zoom minimap can do.

> Zoom Amount : While zooming, how much amount to be added or subtracted.

> Zoom Duration : How long the zoom in and zoom out tween lasts.



Part 2

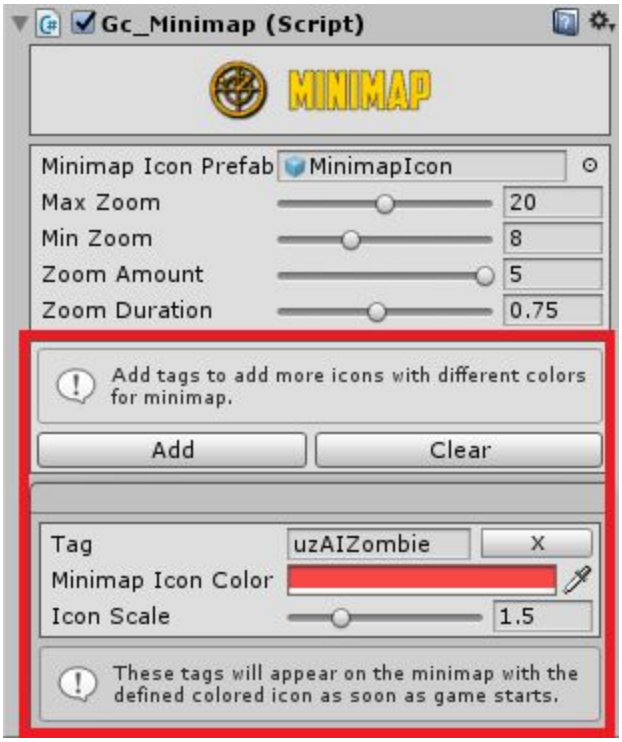
> The next part contains awesome interface for you to add or remove a new tag in the Minimap list.

> Simply hit the “Add” button to add a new tag.

> After adding a new Tag in the list, write the object’s tag **on whom you want to see the Minimap Icon**.

> Minimap Icon Color is the color with which icon which will appear in the Minimap HUD.

> Icon Scale is the scale of Icon with which it will appear in the Minimap. Increase the size if it’s an important objective or reduce the size if it’s a side quest. Imagination is the only limitation ;)



Player Health Manager

Script Location : Scripts/Controller/Game Controller/gc_PlayerHealthManager.cs

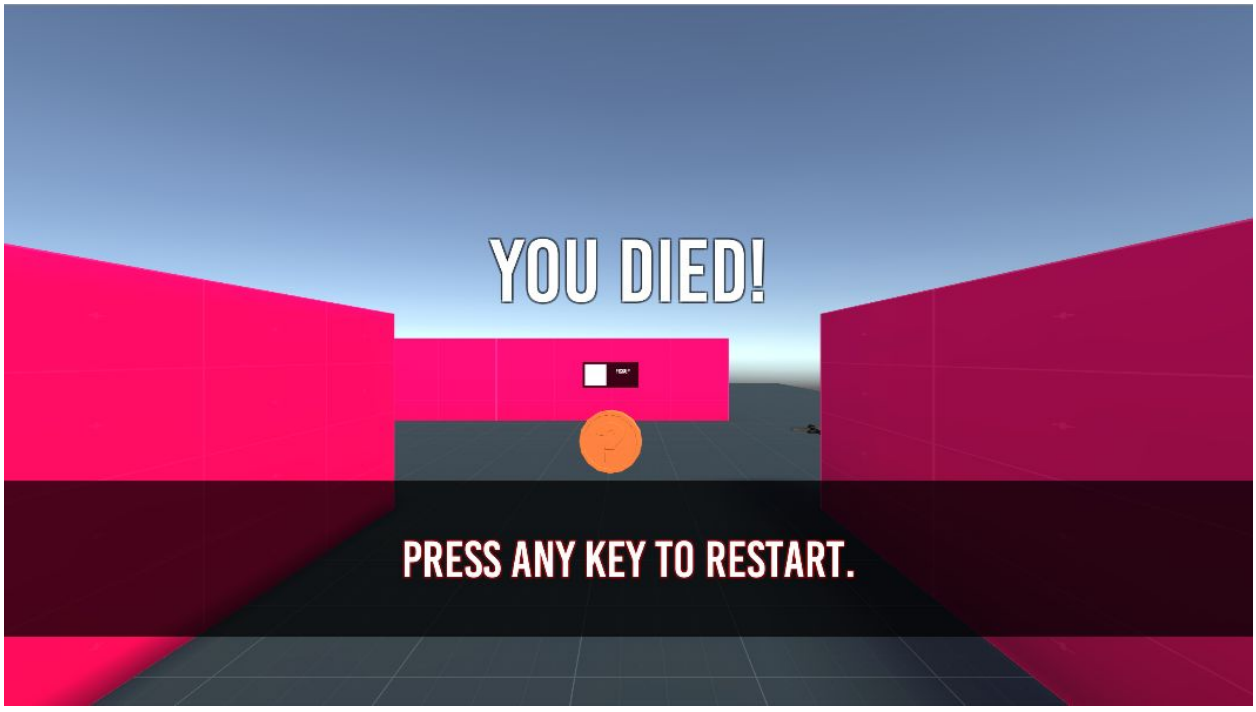
Description : It monitors the current and maximum player health and updates the HUD accordingly.

Damage Lerp duration is how fast the Red Slider follows the green slider when player receives a damage.

Game Over

Script Location : Scripts/UI/GameOver.cs

Description : It is a basic Game Over script which will enable Game Over UI as soon as player dies.



This script is just a placeholder and a great for prototyping and an awesome start for you to build your own Game Over script on top of it!

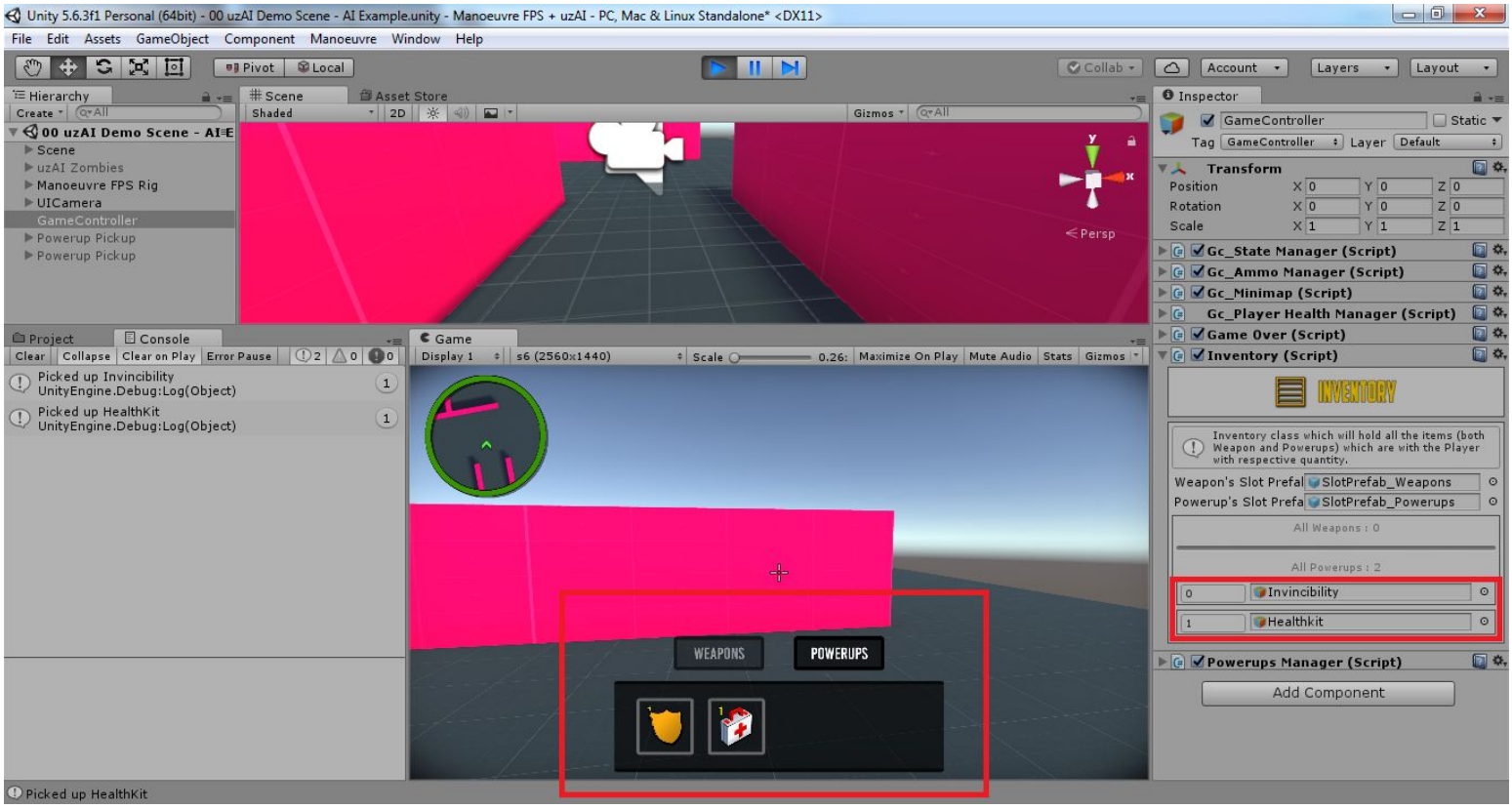
Moreover, it contains a Restart Text Delay property which you can tweak to define after how many seconds Player can restart the scene.

Inventory

Script Location : Scripts/Inventory/Inventory.cs

Description : An Inventory where all the Weapons and Powerups Player currently have are stored with respective quantities and are shown with respective icons.

Note that the contained objects can only be seen at **runtime**.



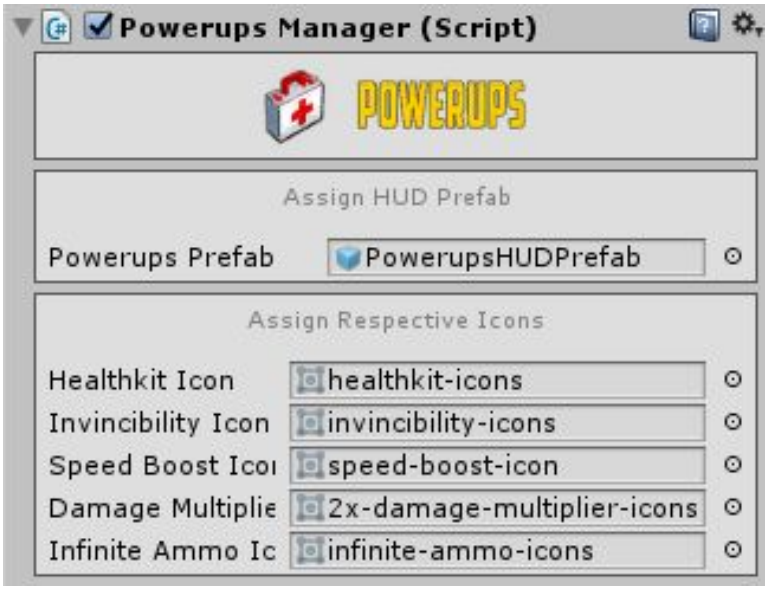
This Inventory is opened with the **Inventory Button / Inventory Key** you have defined in the Manoeuvre FPS Input properties tab.

Powerups Manager

Script Location : Scripts/Powerups/PowerupsManager.cs

Description : This is the main power house of all the powerups. It contains all the PowerUps logic and for better understanding I recommend going through it. :)

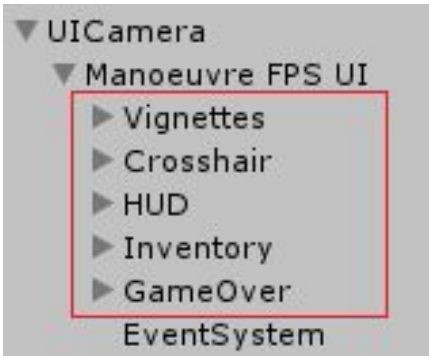
However in the **Inspector** it is highly recommended that you must assign all the icon fields with their corresponding power ups.



Manoeuvre FPS UI

The UI Solution that comes in this package contains 5 main Panels.

- Vignettes
- Crosshair
- HUD
- Inventory
- Gameover



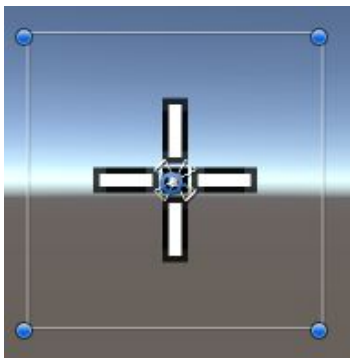
> Vignettes

It contains 2 UI Elements named Screen Vignette and Damage Vignette.

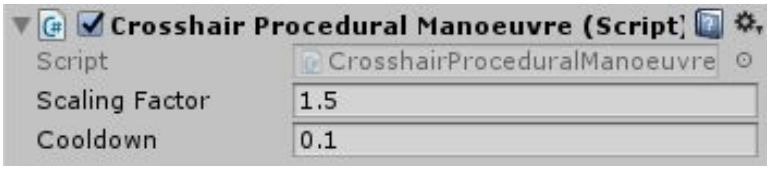
Screen Vignette is a highly optimized trick to have an awesome looking black vignette effect with almost 0 performance impact. It's scale is increased from 1 to 1.2 and vice versa in order to achieve this effect. **It is enabled whenever Player enter into Ironsight.**

Damage Vignette is yet another way to have a cool looking damage effect which work on the same principle of Screen Vignette but is red in color. **It is enabled whenever Player receives a Damage.**

> Crosshair



Dynamic Crosshair HUD works on the current player and weapon's state. It accordingly stretch and contracts based on Scaling Factor and Cooldown. Tweak these properties to achieve the desired effect you want.



> HUD

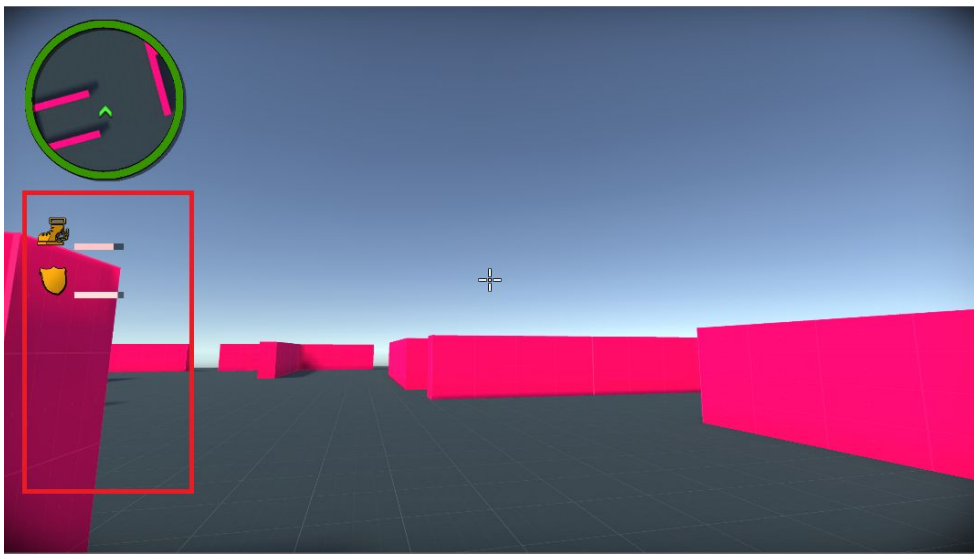
Hud consists of 4 UI Elements.

- Minimap : It contains a Raw Image with a render texture which renders the Minimap Camera's view.

- Health HUD : It contains a damage slider and health slider whose **Fill Amount** has been lerped to current Player's health value.

- Weapons HUD : It contains 3 UI Elements, 1 Image for Current Equipped Weapon's Icon and 2 Text Elements for its current Ammo and total Ammo left.

- Powerups HUD : It is empty Rect Transform but with **Grid Layout Group** attached. It is filled with the Powerups HUD Prefabs at **runtime** whenever Player consumes a power. These Powerups HUD Prefabs looks like this :



Since there's a Grid Layout Group attached, whenever these Prefabs are added they are auto arranged! :)

> Inventory

Inventory UI contains 4 Elements

- BG - Background Image
- Weapons Button - This enables the **Item Container/StotsContainer_Weapons** and display all the weapons Player currently hold.
- Powerups Button - This enables the **Item Container/StotsContainer_Powerups** and display all the powerups Player currently hold.

> Game Over

Game Over screen when is enabled all the other Inputs are locked / disabled. Game Over UI contains 2 elements

- Game over Text - This text display right after Player dies.
- Black Panel - This Image Element contains a Restart Text which is enabled a few seconds later (which you have defined in Game Over script). Player can only restart the game once this text component is enabled.

-- END --