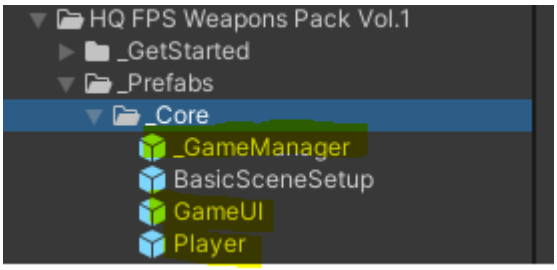
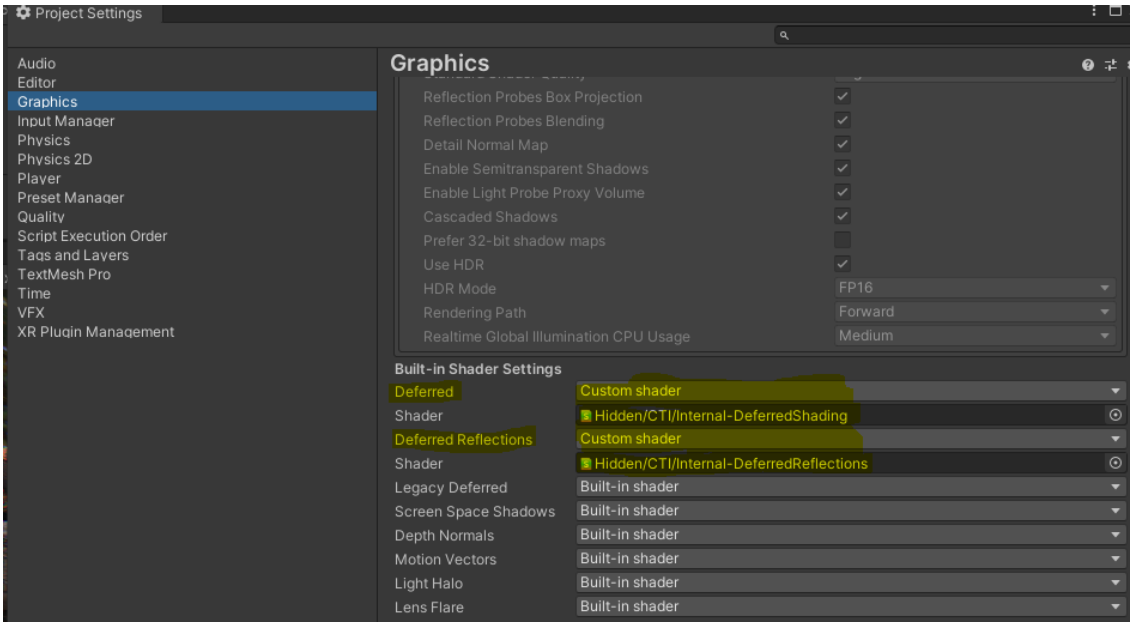
HQ FPS Template

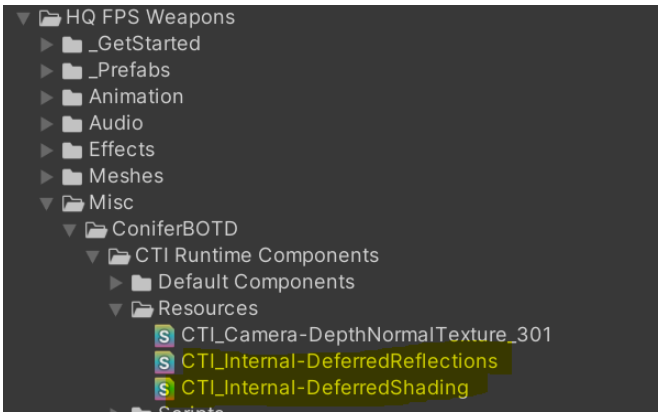
Version: 1.3.0.1
Email: Lucianpavel728@yahoo.com
Discord: Polymind Games

Get Started:
These are all of the prefabs that you need to have in your scene for the asset to function:



- If the trees are not looking good, replace these shaders with the ones from *CTI_ConiferBOTD*





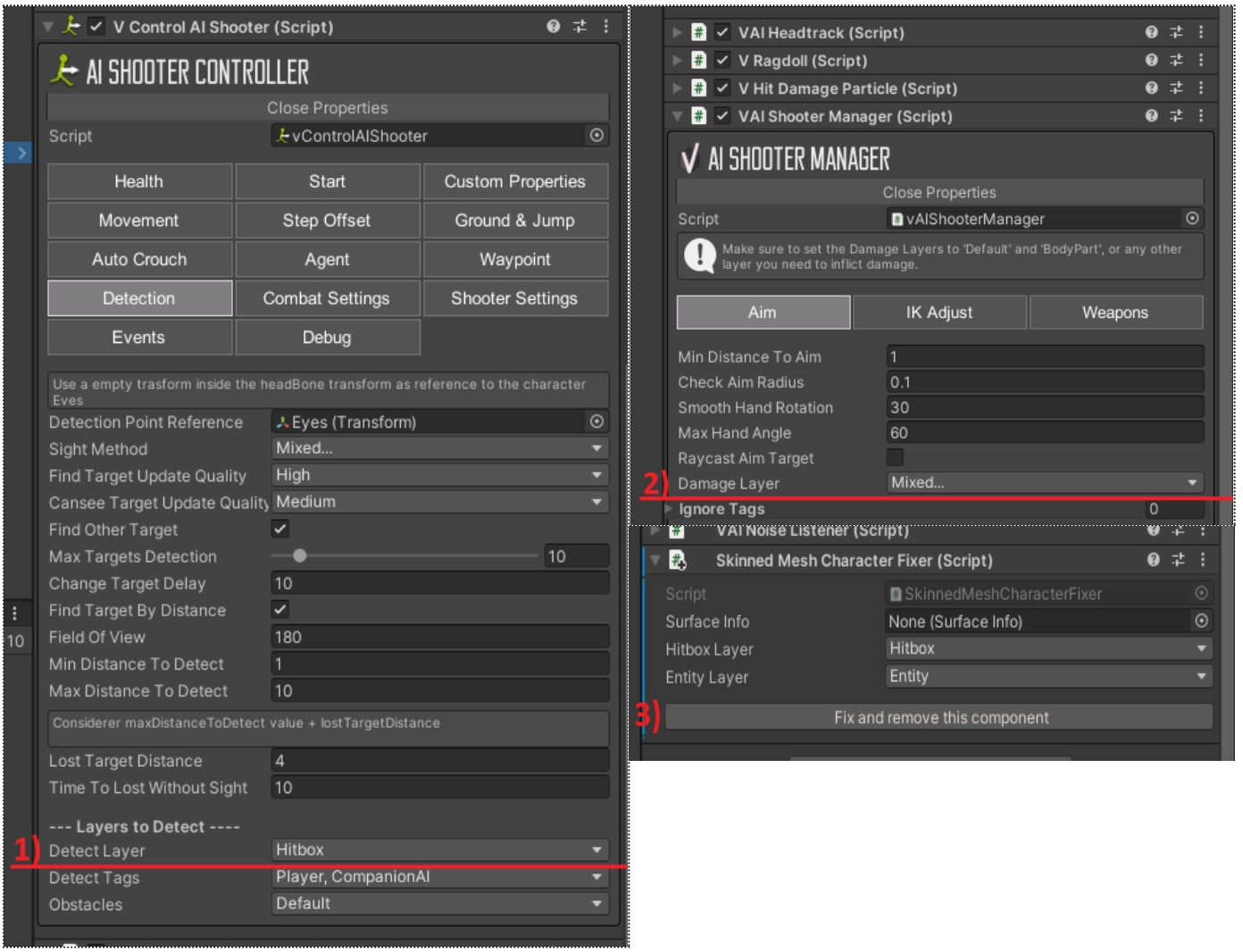
Integrations:

[FSM AI Template]

Enemy AI Setup (You'll have to follow these steps for every enemy you add):

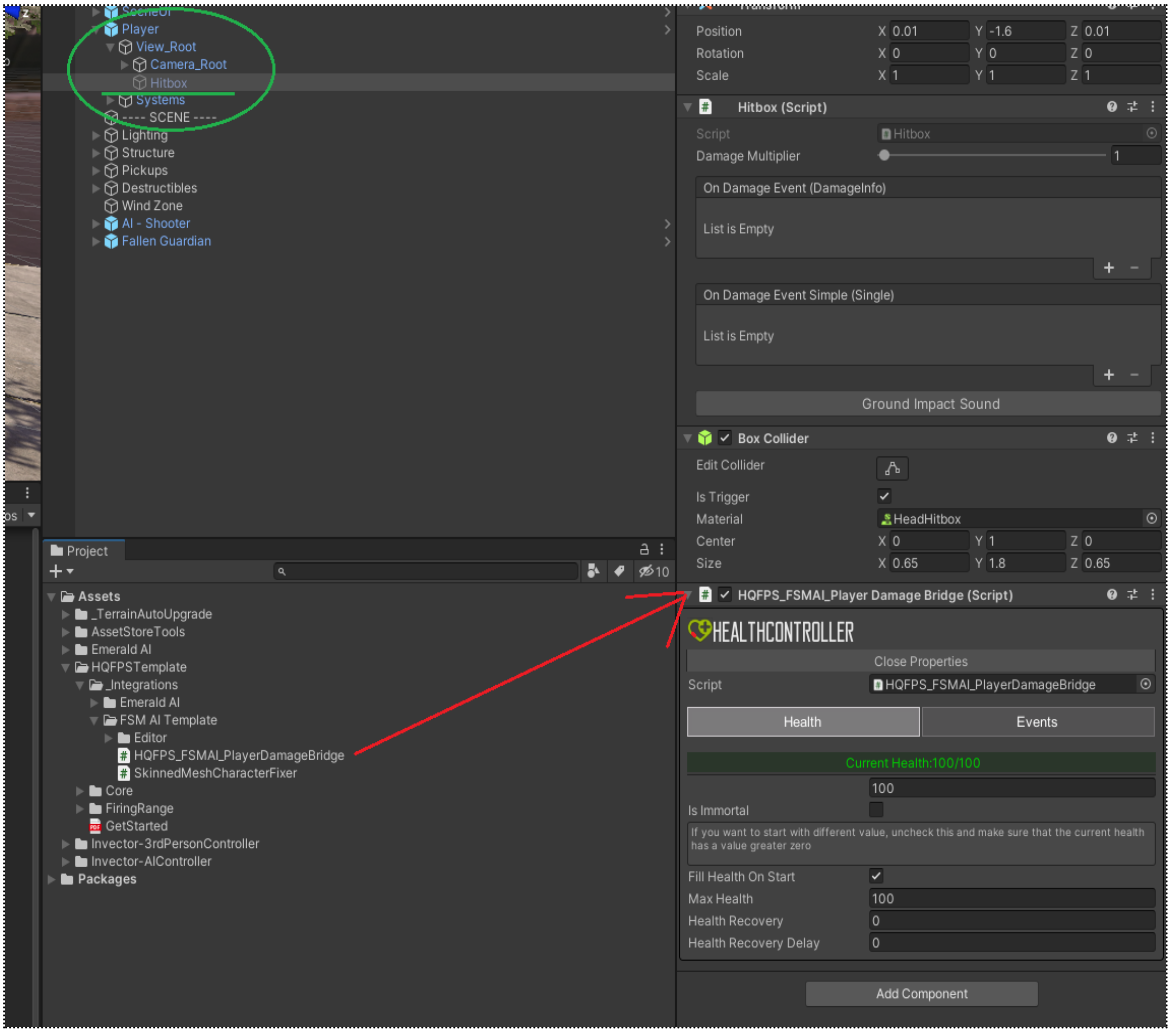
[Note: this was tested on the "AI - Shooter" enemy available in Invector's FSM AI Template]

1. Set the detect layer of the AI to "Hitbox".
2. Set the Damage Layer of the AI to Default, Hitbox and NoCollision.
3. Add a "Skinned Mesh Character Fixer" component on the root of the AI (same gameobject as the one that "V Control AI" is) and press on the "Fix and remove this component" button.



Player Setup:

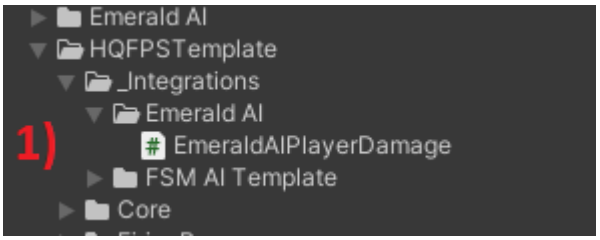
1. Add a HQFPS_FSMAI_PlayerDamageBridge component to the head hitbox of the Player.

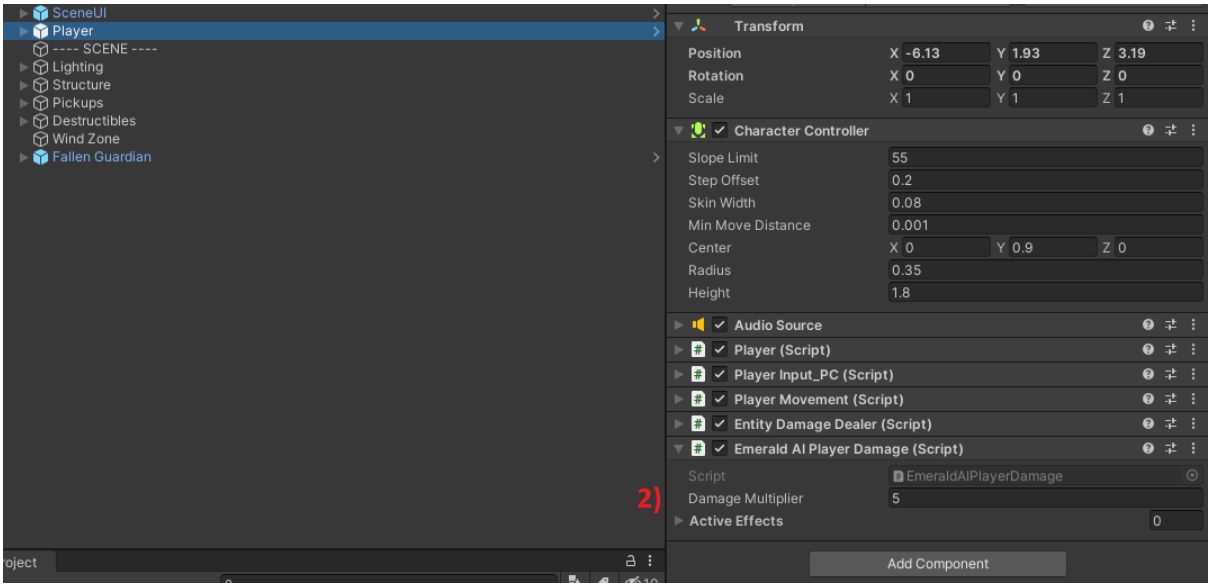


[Emerald AI]

Player Setup:

1. Replace the “EmeraldAIPlayerDamage” script with the one included in the package.
2. Add the EmeraldAIPlayerDamage component to the Player root (note: you can increase the Damage Multiplier if the incoming damage is too low).

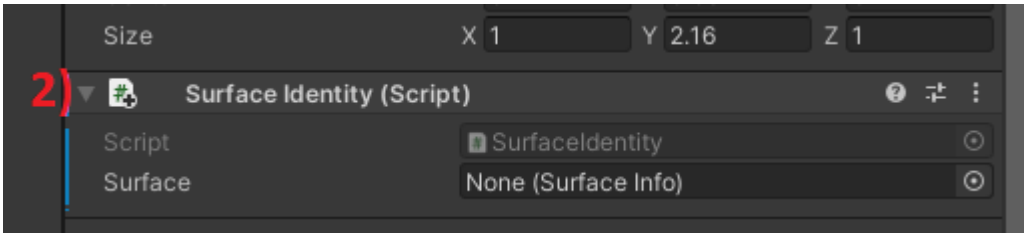




Enemy AI Setup (You'll have to follow these steps for every enemy you add):

[Note: this was tested on the Fallen Guardian enemy prefab available in Emerald AI]

1. Set the detection layers to "Entity".
2. Set the layer of the Enemy to "Hitbox".
3. Add a Surface Identity component to the enemy.



Structure (some parts are outdated, the whole documentation is getting overhauled in update 1.3.1)

Project Structure:

PREFABS:

1. **!Core...**

   - Game Manager: Contains things like Player Input, Surface Manager, Pooling Manager etc.

   - **Basic Scene Setup:** Contains 2 Post Processing Volumes and a Directional Light (Place it in your scene if you don't have any Post Processing Volumes).

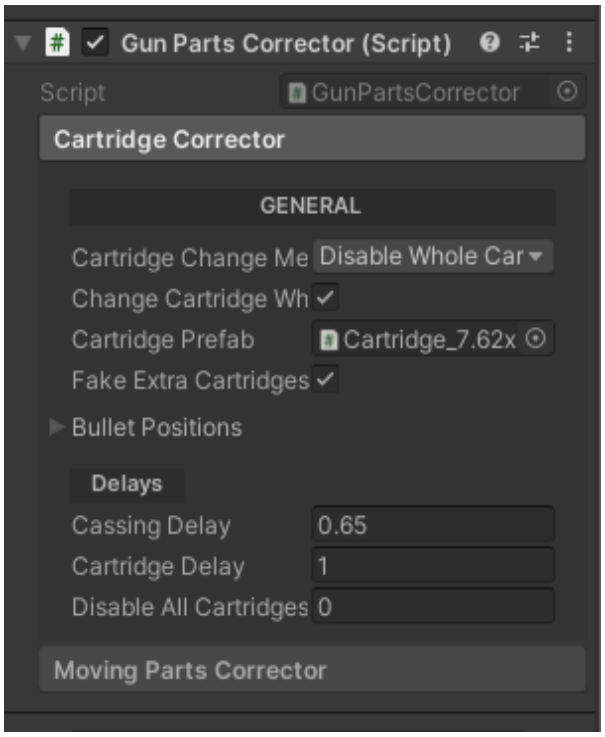
   - **Game UI:** Contains Every Piece of UI, from the Pause Menu to the HUD.

   - **Player:** Player Prefab. (Everything included Weapons, Inventory, Cameras etc.)

2. **Ammunition...**

   - Casings: Casings that are spawned from the weapons after shooting. First Person Cartridges: These Prefabs are the visible cartridges used for the guns.

   - First Person Cartridges: These Prefabs are the cartridges visible in the gun. Gun Parts Corrector spawns them immediately after playing the game. That script disables the visible cartridges if the magazine is empty, the Player will be able to see that when empty reloading.



   - Pickups: Contains all the Ammo Pickups. (Bullets, Magazines, Clips, Boxes etc.)

   - Projectiles: These are the non-Raycast Projectiles. (e.g Flare)

3. **Effects…**

   All The effects in this project from Surface Effects (e.g. **Decals**) to Weapon Effects (e.g. **Muzzle Flash**).

4. **Environment…**

   Contains all of the "**Firing Range**" map modular parts.

5. **First Person…**
   Arms & Accessories that the "**Equipment Manager**" cycles through if you press the "Change Arms" button.
6. **Others…**
   Extra prefabs that you can use, e.g. Item Spawner, Crates.
7. **Weapon Pickups…**
   All of the weapon pickups. (**Note**: the **explosive grenade** & the **molotov** pickups is the same as the thrown object)

ANIMATION: All of the **animations** (except for first person ones) and **animators**.

AUDIO: All of the **Sound Effects** in the asset (e.g. shoot, foley, weapon handling etc.)

EFFECTS: All of the materials & textures that the Particle Systems use. (e.g. **Muzzle Flashes** textures **&** materials**)**

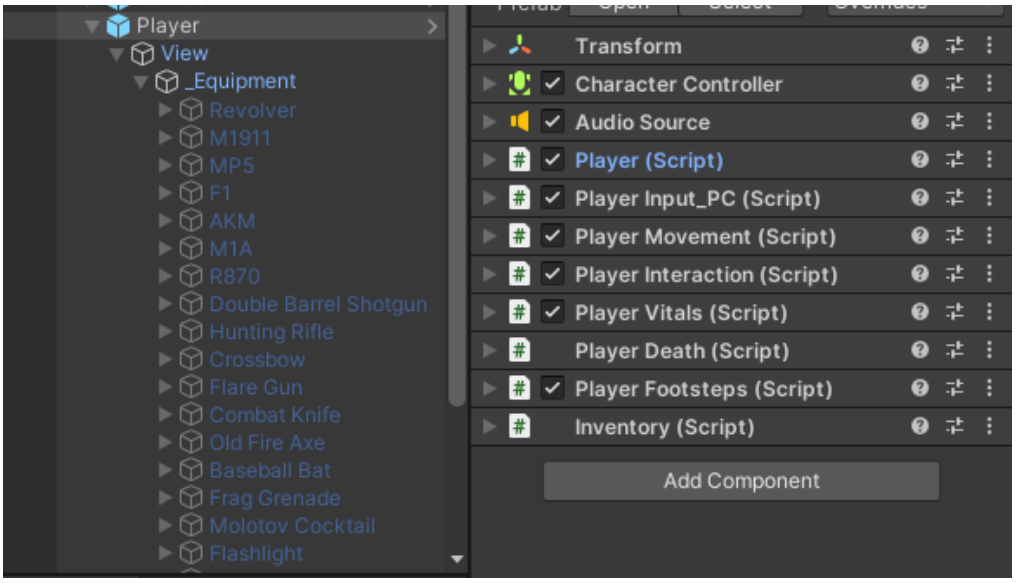
MESHES: Contains all of the Meshes(e.g. **weapons**, **arms**, **ammo**, **map pieces** etc.) and the **first-person weapon animations**.

SCRIPTS: All of the animations (except for the weapons) and animators.

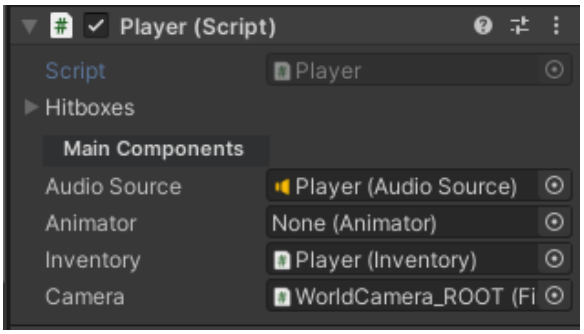
SHADERS: All of the shaders used this asset.

SPRITES: All of the **sprites** used in this asset. (e.g. **HUD** elements, Panels etc.)

Player Components Explained:



➔ The **Player script** handles all of the events that the Player takes place in such as running, crouching, using equipment item etc.




```
/// <summary></summary>
public Attempt UseOnce = new Attempt();

/// <summary></summary>
public Attempt ChangeArms = new Attempt();

/// <summary></summary>
public Attempt ChangeFireMode = new Attempt();

/// <summary></summary>
public Attempt UseContinuously = new Attempt();

/// <summary></summary>
public Value<float> ScrollValue = new Value<float>(0f);

/// <summary></summary>
/// <summary></summary>
public Activity Run = new Activity();

/// <summary></summary>
public Activity Crouch = new Activity();

/// <summary></summary>
public Activity Jump = new Activity();
```

Note: the “attempts, values & activities are just “custom” events.

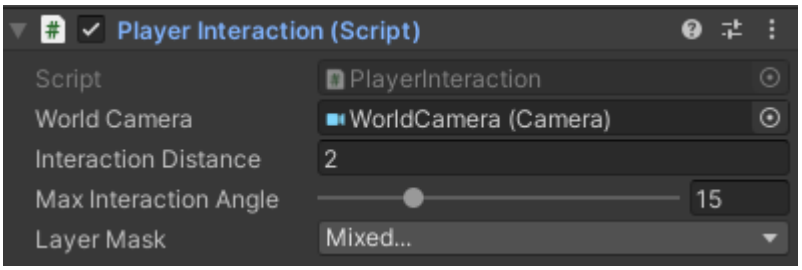
➔ The **Player Input_PC script** handles all of the input for the Player. (e.g. pressing the use button starts the "use" event on Player)

```
// Jump.
if (Input.GetButtonDown("Jump"))
    Player.Jump.TryStart();

if (Input.GetButtonDown("Drop") && !Player.EquippedItem.Is(null) && !Player.Reload.Active && !Player.Healing.Active)
{
    var item = Player.EquippedItem.Get();
    Player.Inventory.TryDropItem(item, true);
}

// Run.
bool sprintButtonHeld = Input.GetButton("Run");
bool canStartSprinting = Player.IsGrounded.Get() && Player.MoveInput.Get().y > 0f;
```

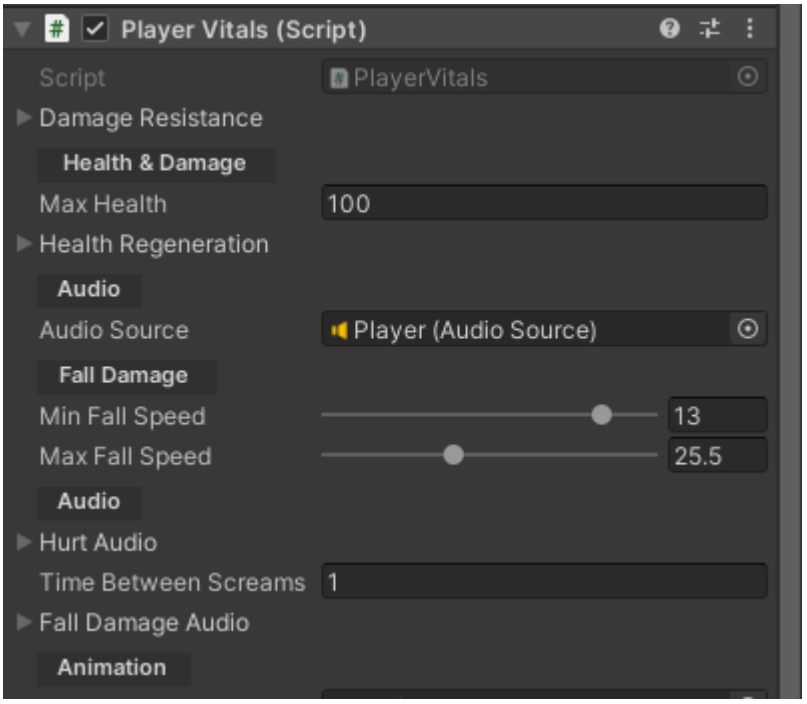
➔ The **Player Movement script** moves the Player through space by controlling the “Character Controller”.


```
1 reference
private bool TryRun()
{
    if (!m_EnableRunning || Time.time < m_NextTimeCanRun || Player.Stamina.Get() < 15f)
        return false;

    bool wantsToMoveBack = Player.MoveInput.Get().y < 0f;

    return Player.IsGrounded.Get() && !wantsToMoveBack && !Player.Crouch.Active && !Player.Aim.Active;
}

1 reference
private bool TryJump()
{
    // If crouched, stop crouching first
    if (Player.Crouch.Active)
    {
        Player.Crouch.TryStop();
        return false;
    }
```

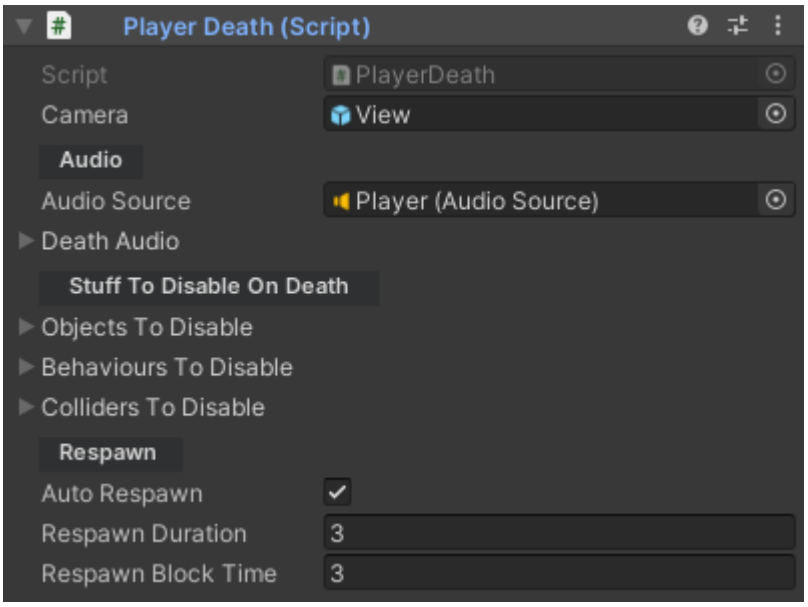
➔ The **Player Interaction script** checks for interactable objects that are near the Player and in the “Max Interaction Angle”.



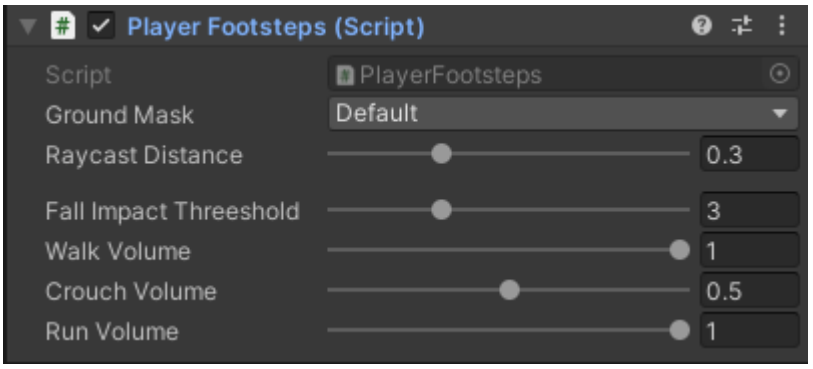
➔ The **Player Vitals script** contains all the data for Health, Damage Resistance, Stamina, etc.



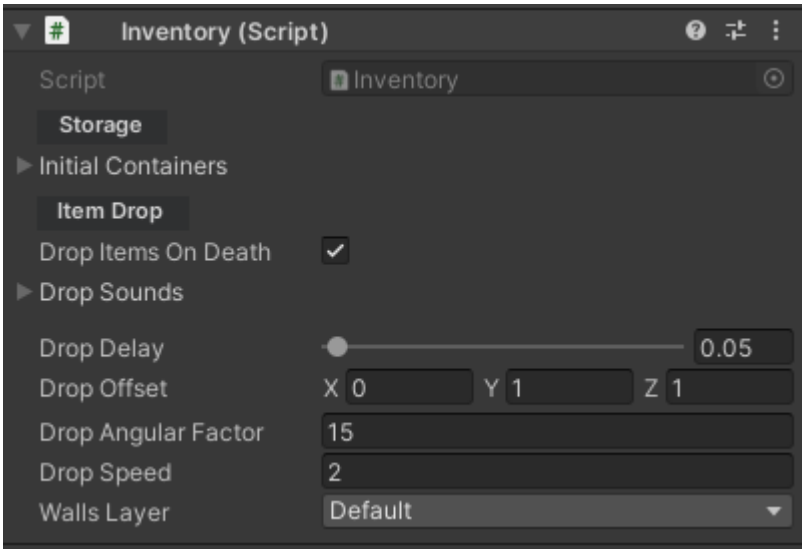
➔ The **Player Death script** handles the Respawn and the Behaviours that’ll get disabled if the Player dies (e.g. it disable the Player Movement Script).



➔ The **Player Footsteps script** plays the corresponding footstep sounds if the Player moved enough.

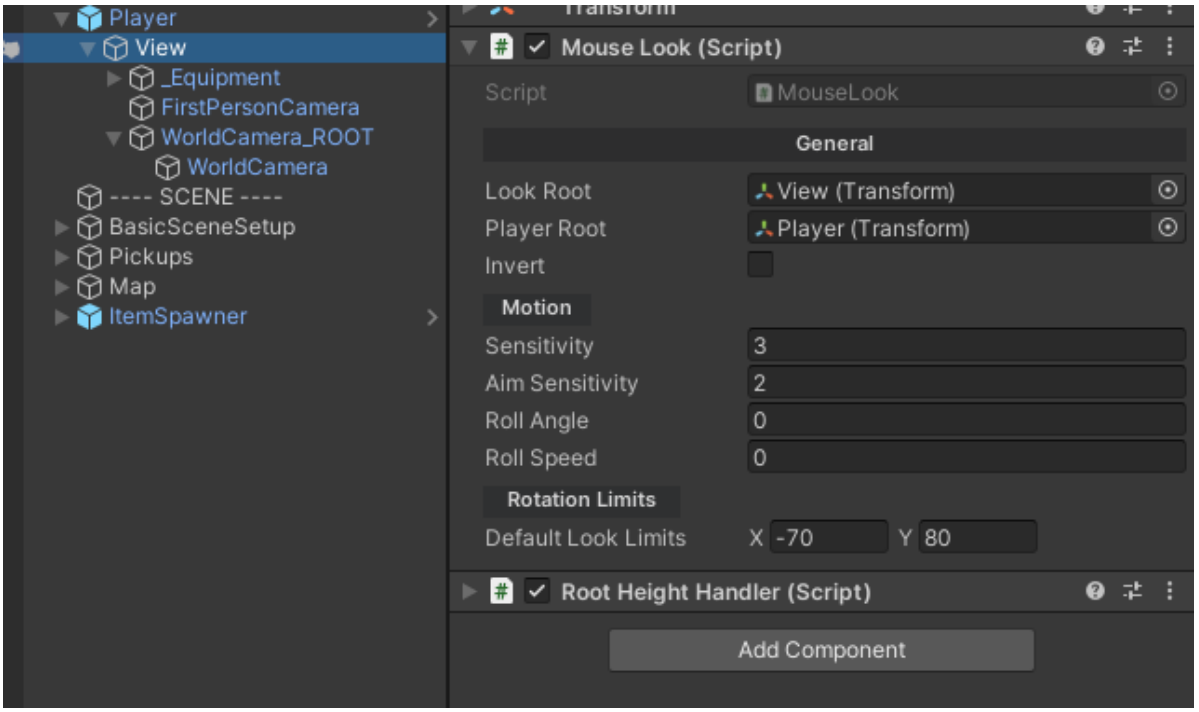


➔ The **Inventory script** contains all of the data for the current Player Items (e.g. weapons, ammo, heals etc.)
You can also change the “Initial Containers” to make the Player spawn with items already in their inventory.



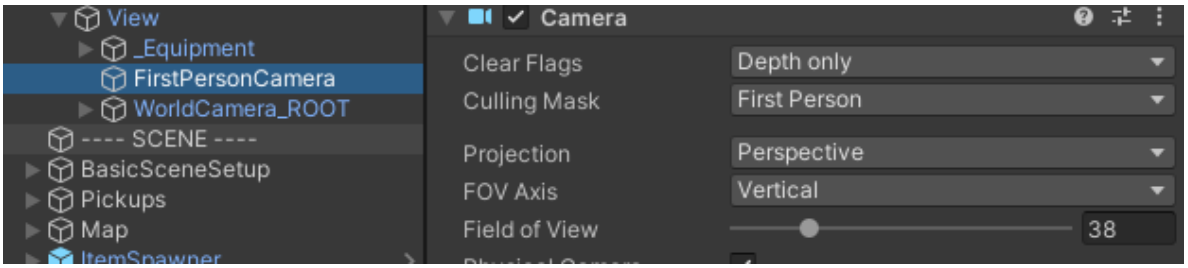
Player Camera Explained:

➔ **View** has the “Mouse Look” component attached to it, here you can control the mouse sensitivity or invert the mouse look.

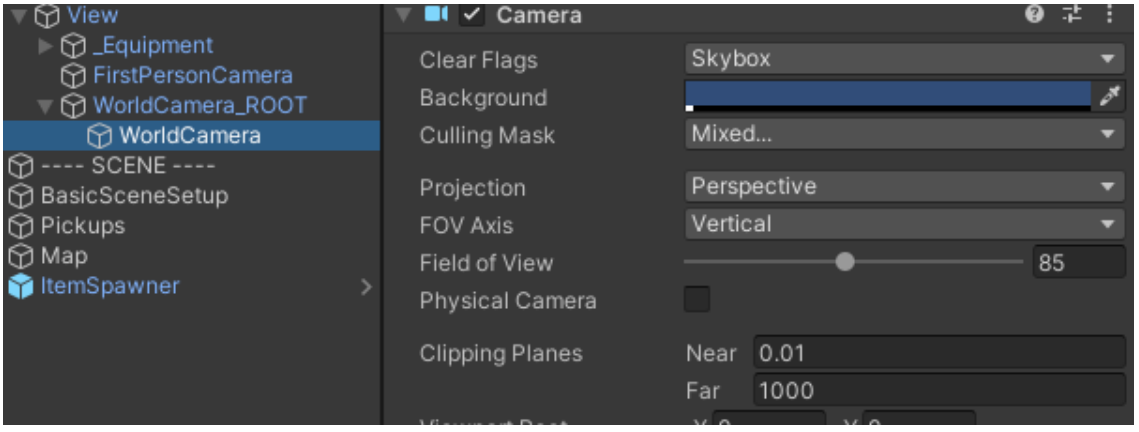


➔ This asset, as of right now, uses a **dual camera setup**, which means that the weapons are rendered on a different camera than the rest of the world, so that they don’t clip in objects and can have their own **FOV**.

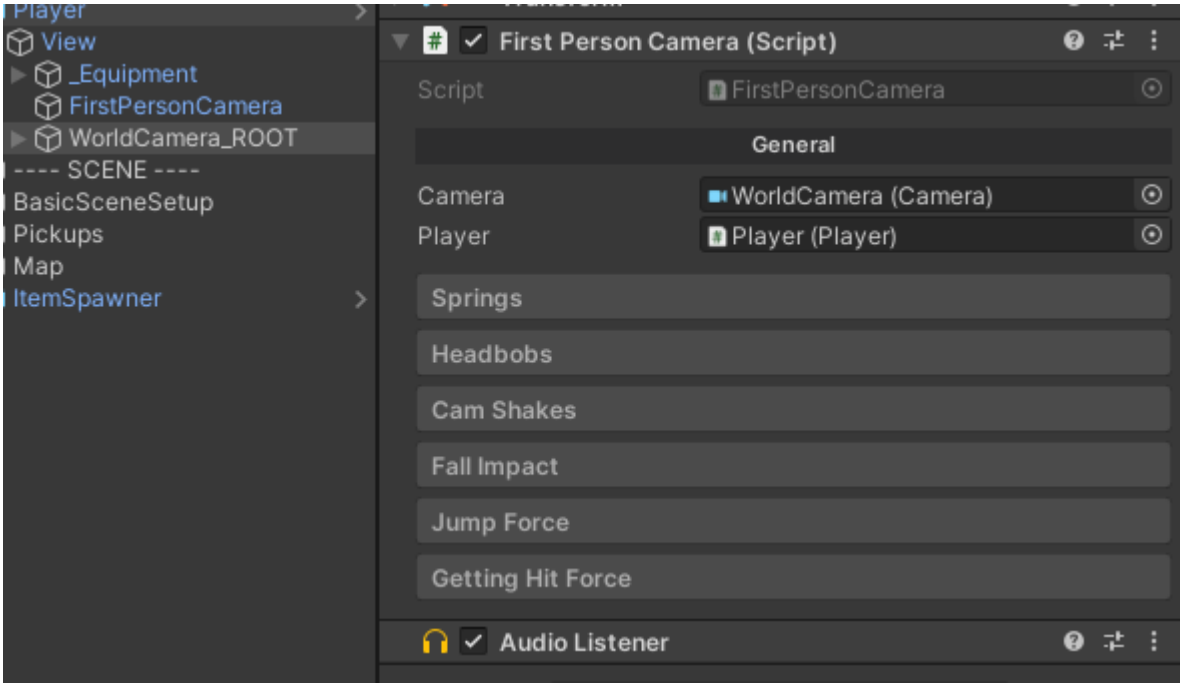
First Person Camera:



World Camera:

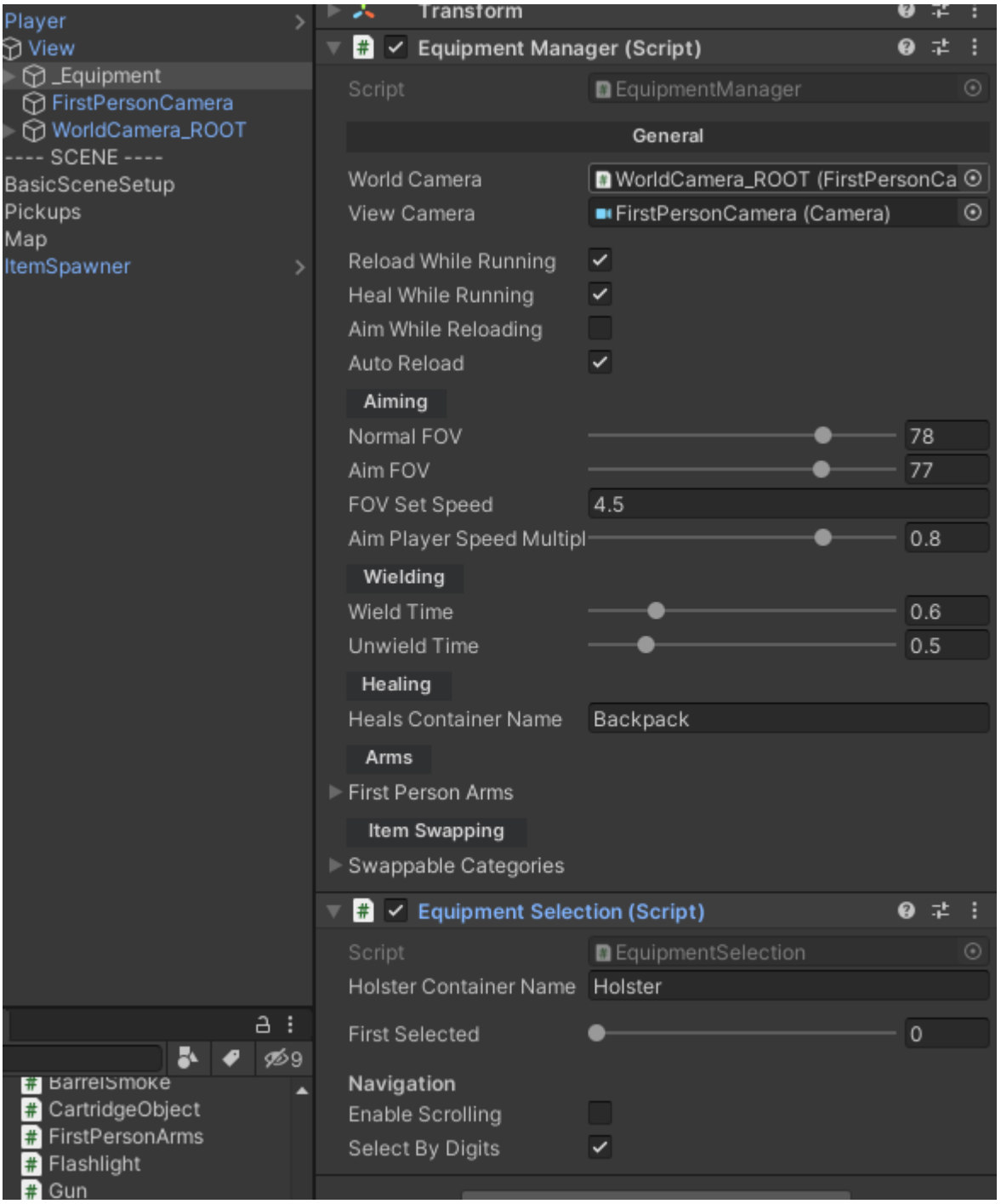


➔ The **First Person Camera Script** It's used to move the camera with spring-based procedural animations, things like walk Head bobs, run Head bobs, grenade explosion cam shakes can be tweaked here.

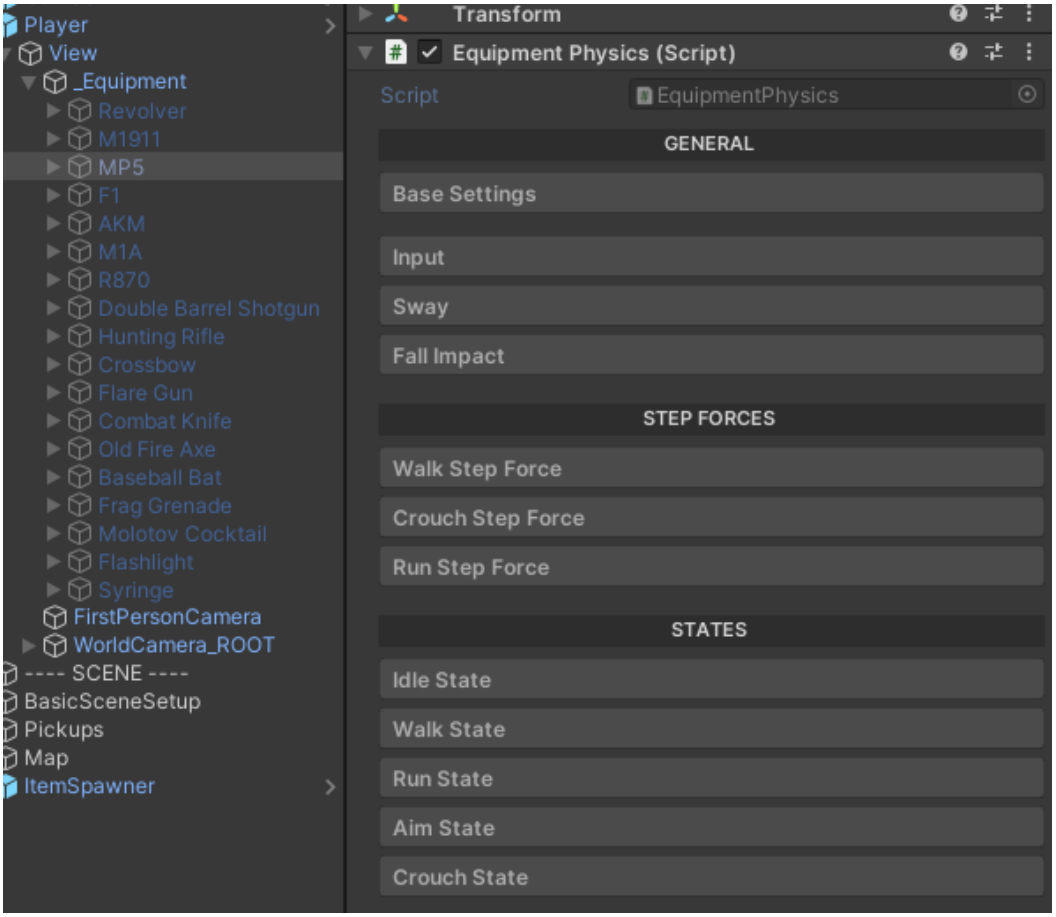


Weapon Components Explained:

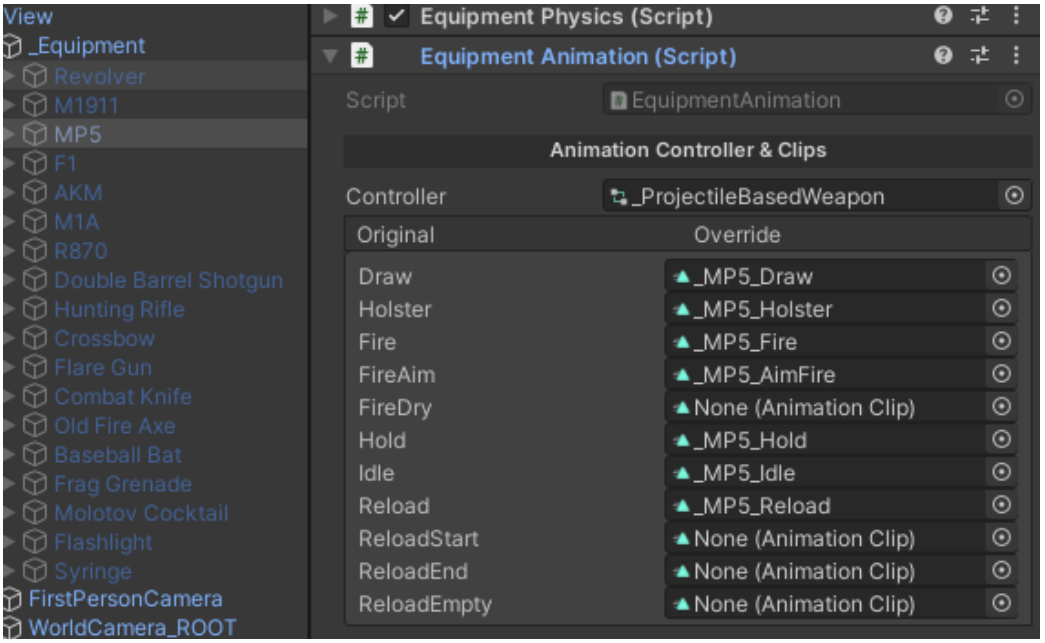
➔ The **Equipment Manager** It's the one who controls the weapon equipping, switching, aiming and a few other things.

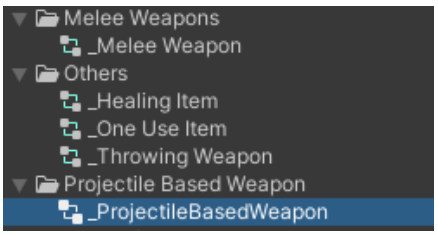


➔ The **Equipment Physics** It's the place where you define the procedural animations for a weapon (e.g. running, walking, aiming etc.). It's using a spring system, the same as the camera movement.

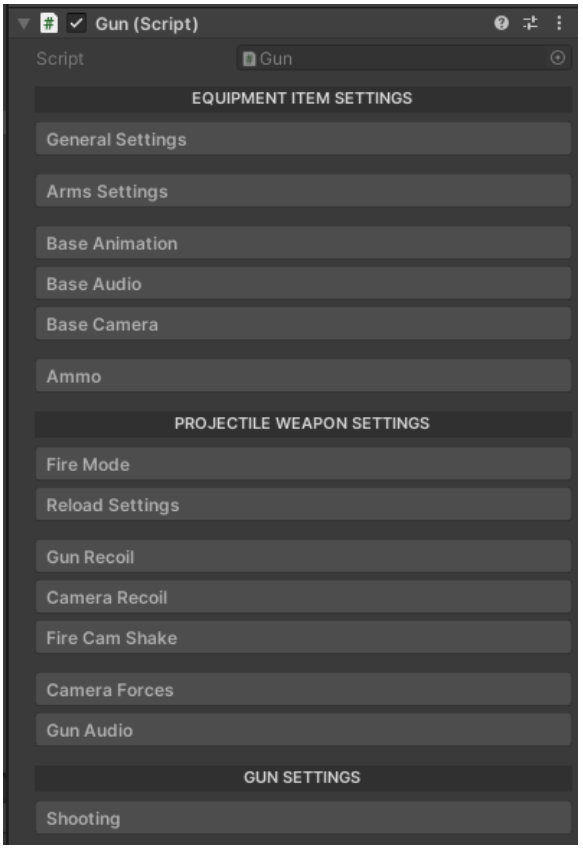


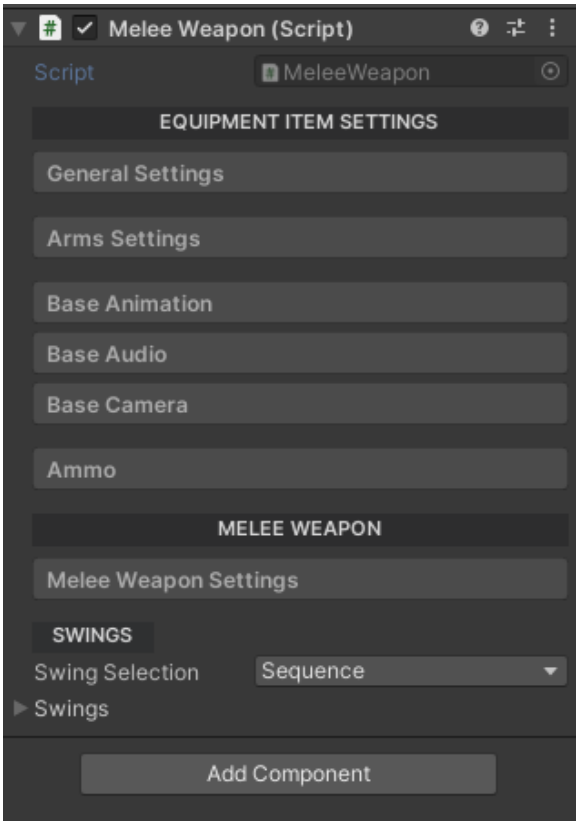
➔ The **Equipment Animation** component It's where you assign the animations for each weapon. (**Note**: every weapon type has a different controller, "**_ProjectileBasedWeapon**"(Used for every Gun), "**_MeleeWeapon**", etc.)



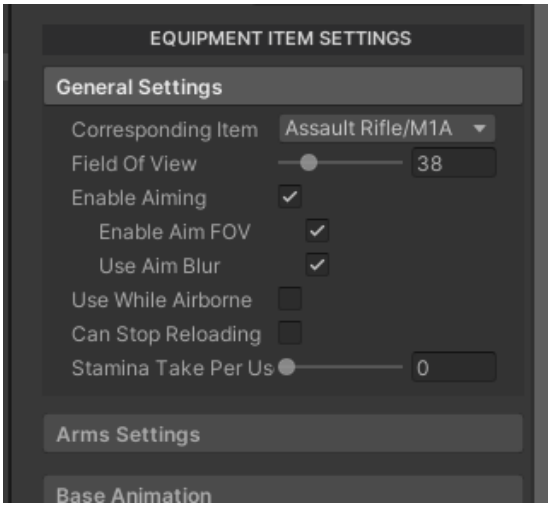


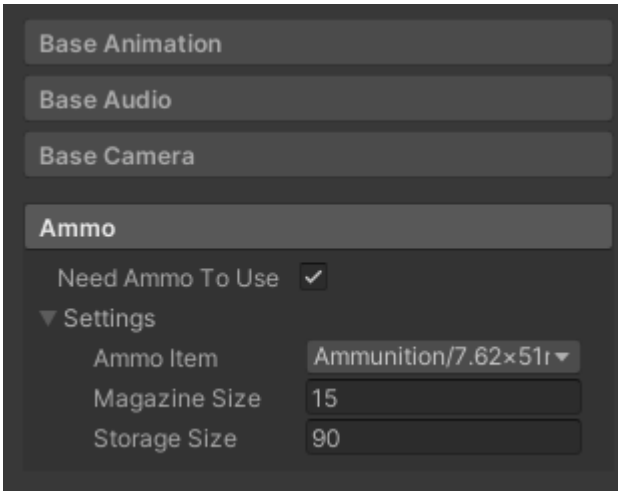
➔ The **Gun, Melee Weapon, Launcher Weapon, Tool, Throwing Weapon and Heal Item** are all a type of Equipment Item with their own features. (e.g. reloading for guns & launchers, fire modes, swings for melee weapons etc.)





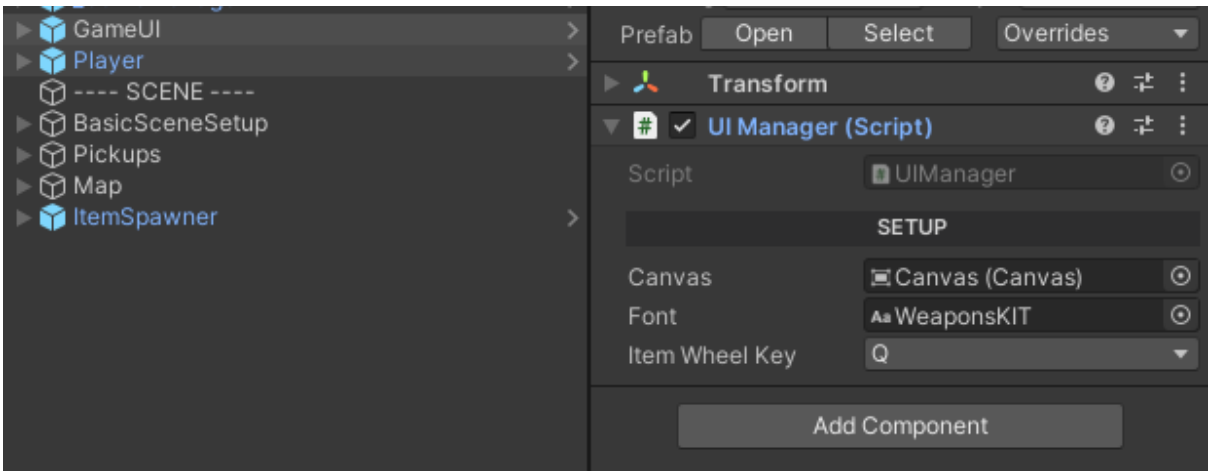
"**Equipment Item Settings**" has all of the generic settings that each first person item should have: corresponding item, if aiming is enabled, if it uses some kind of ammo, etc.



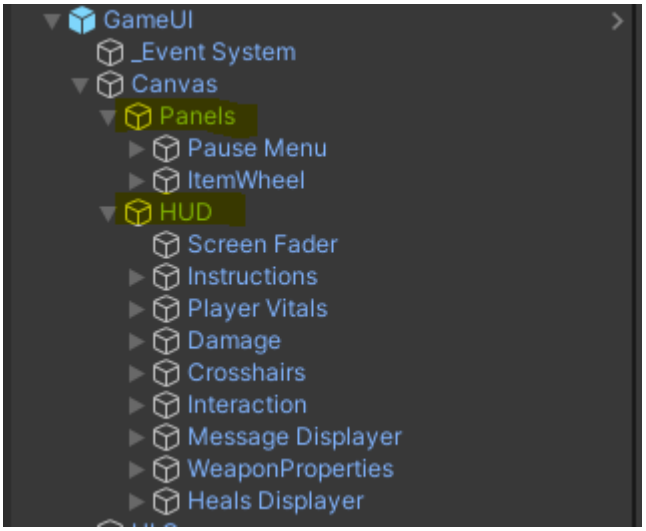


Player UI Components Explained:

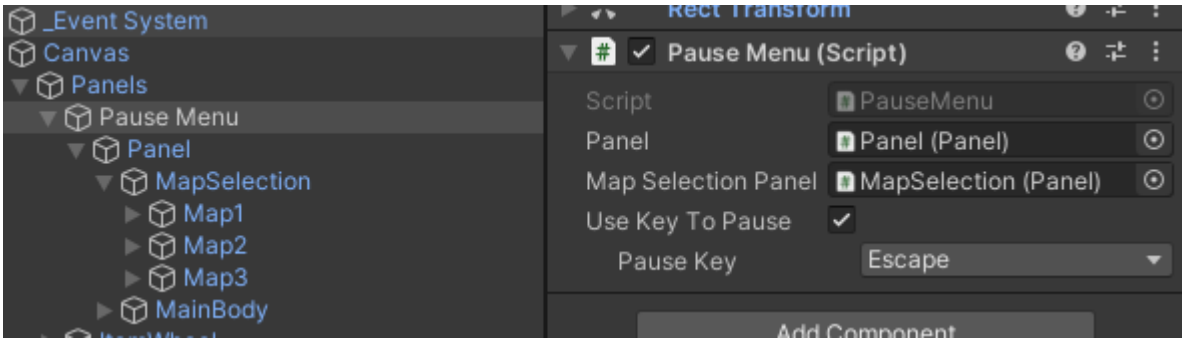
➔ UI Manager:



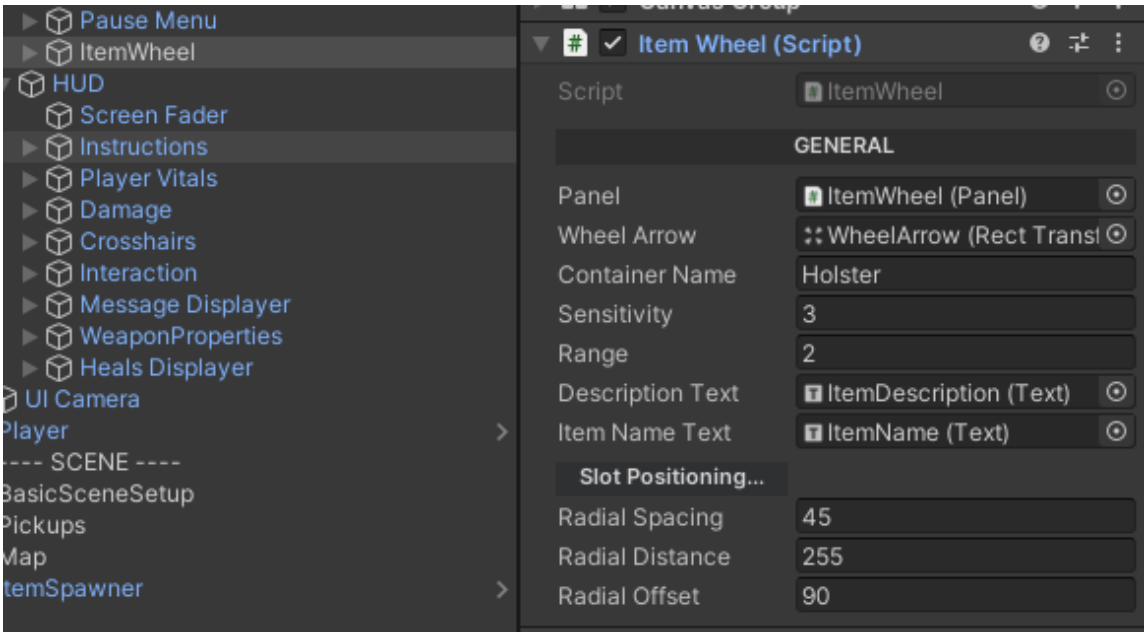
➔ Panels:



- **Pause Menu** pauses the game and can select map

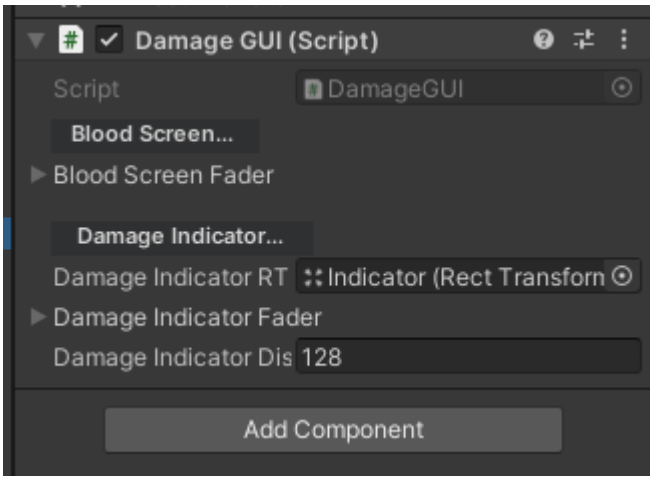


- **Weapon Wheel** is used to select items.



➔ **HUD:**

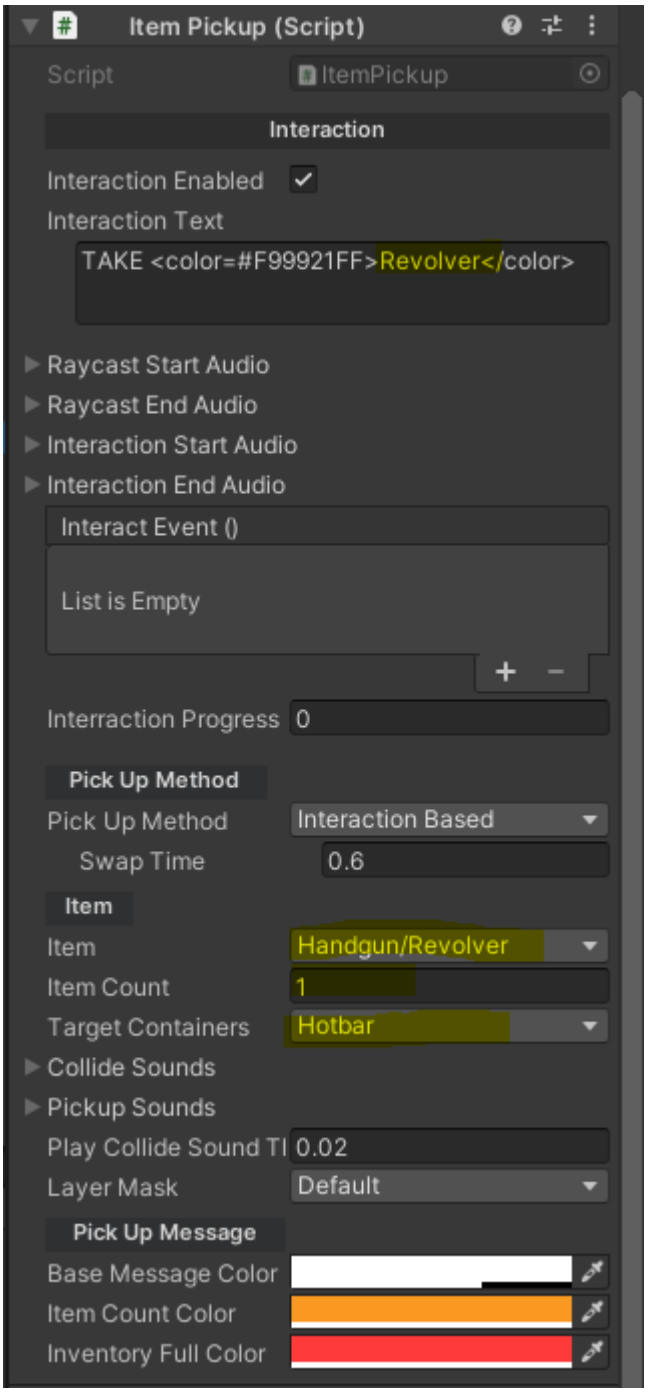
1. **Screen Fader** (Fades in the screen when starting the game)
2. **Instructions** (The instructions text in the top right, can be deleted without any side effects)
3. **Player Vitals** (Shows the current amount of Health & Stamina of the Player)
4. **Damage** (Fades in a Blood screen and damage indicators if the player took damage)



5. **Crosshairs** (Set a custom crosshair width, size, etc. for each weapon)
6. **Interaction** (Interaction panel that appears when the Player is looking at an interactable item (e.g. Item Pickup, buttons etc.).
7. **Message Displayer** (Displays a message to the screen, e.g. Player has picked up an “x” item or “Inventory Full”).
8. **Weapon Properties** (Displays the current amount of ammo, the currently selected fire mode and related things).
9. **Heals Displayer** (Displays the amount of heals the Player has and related Healing things)

Pickups Explained:

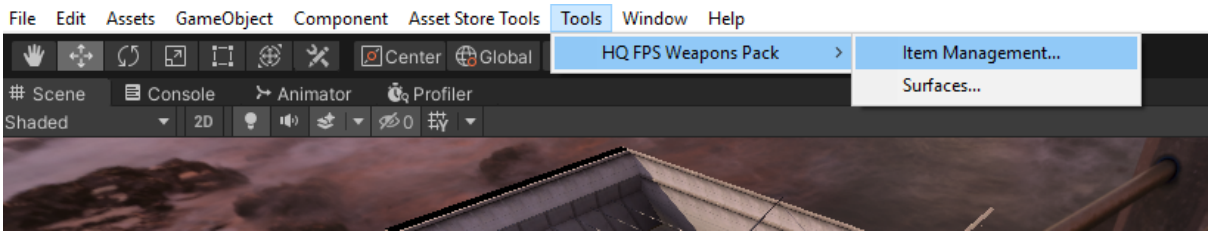
➔ **Item Pickup:**



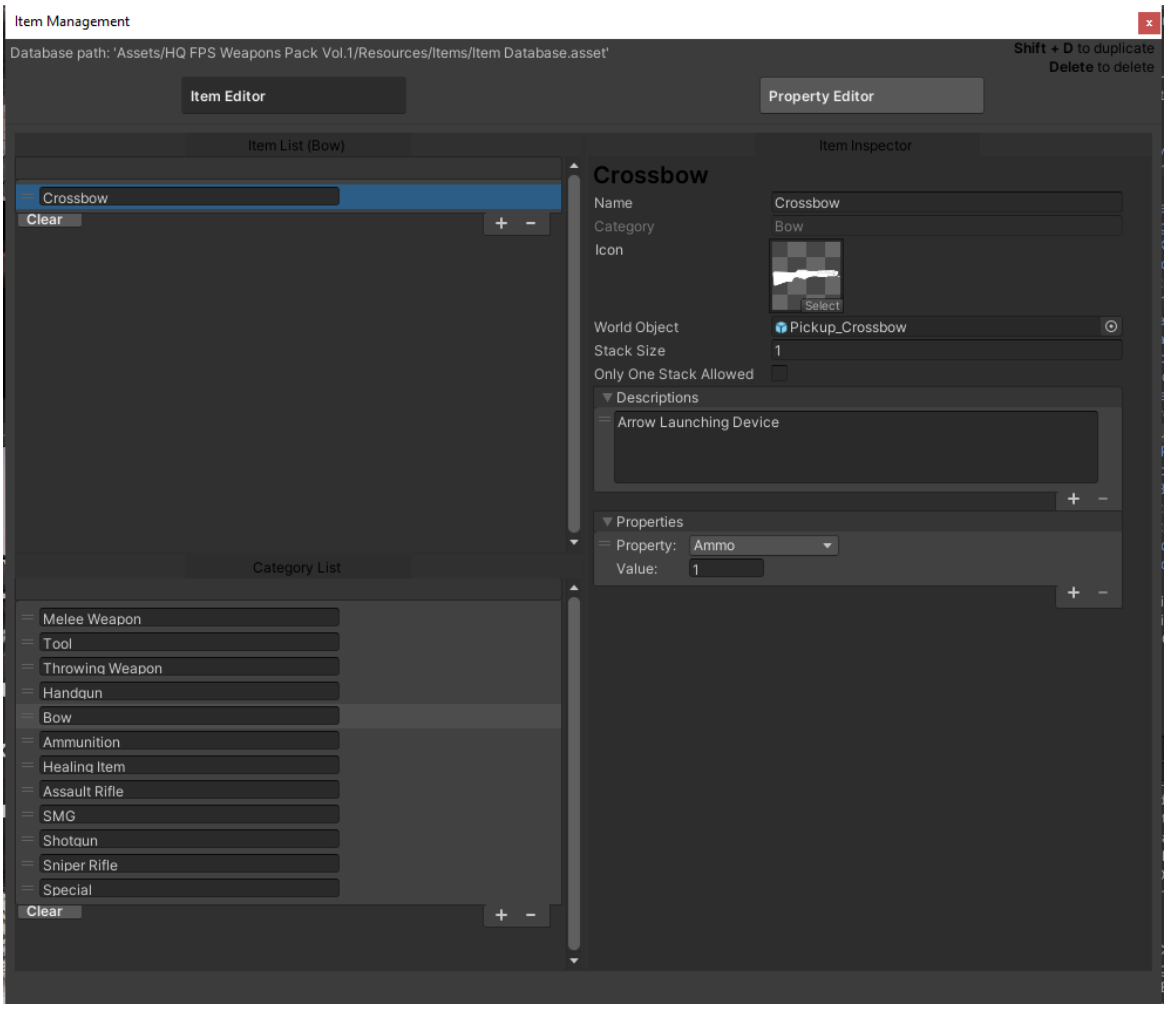
1. **Interaction Text** (Name of the Pickup)
2. **Item** (What item is going to be added to the Player Inventory if It's picked up).
3. **Target Containers** (Where in the inventory will this item be added, e.g. Weapons should be added in the "**Hotbar**", ammo in "**AmmoPouch**" and heals in "**Storage**".

How to use the Item Management:

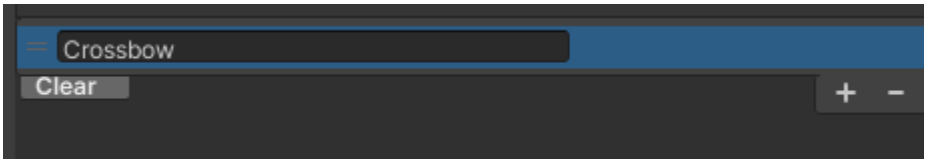
1. To access the **item management** settings go here:



2. Here you can add your own **items** or modify the ones that are already in the asset:



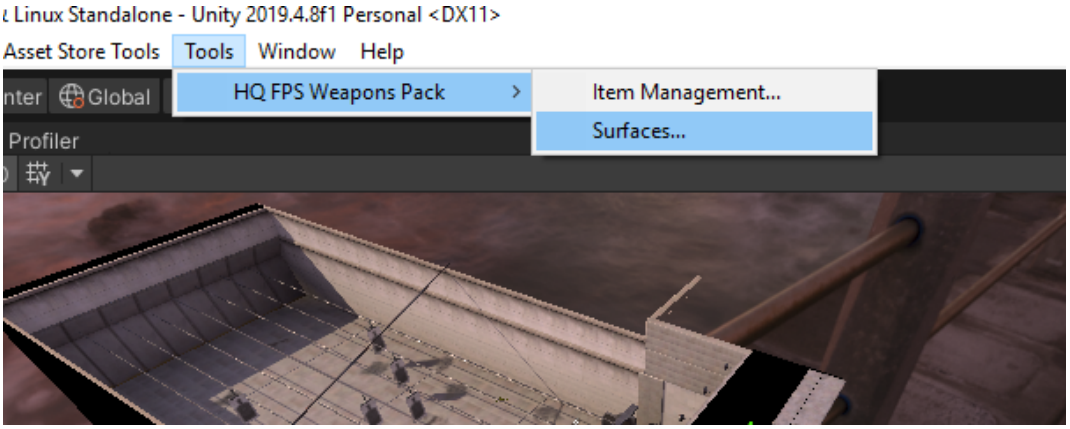
To add an item you'll need to select/create a category from the bottom then press "+" button at the top.



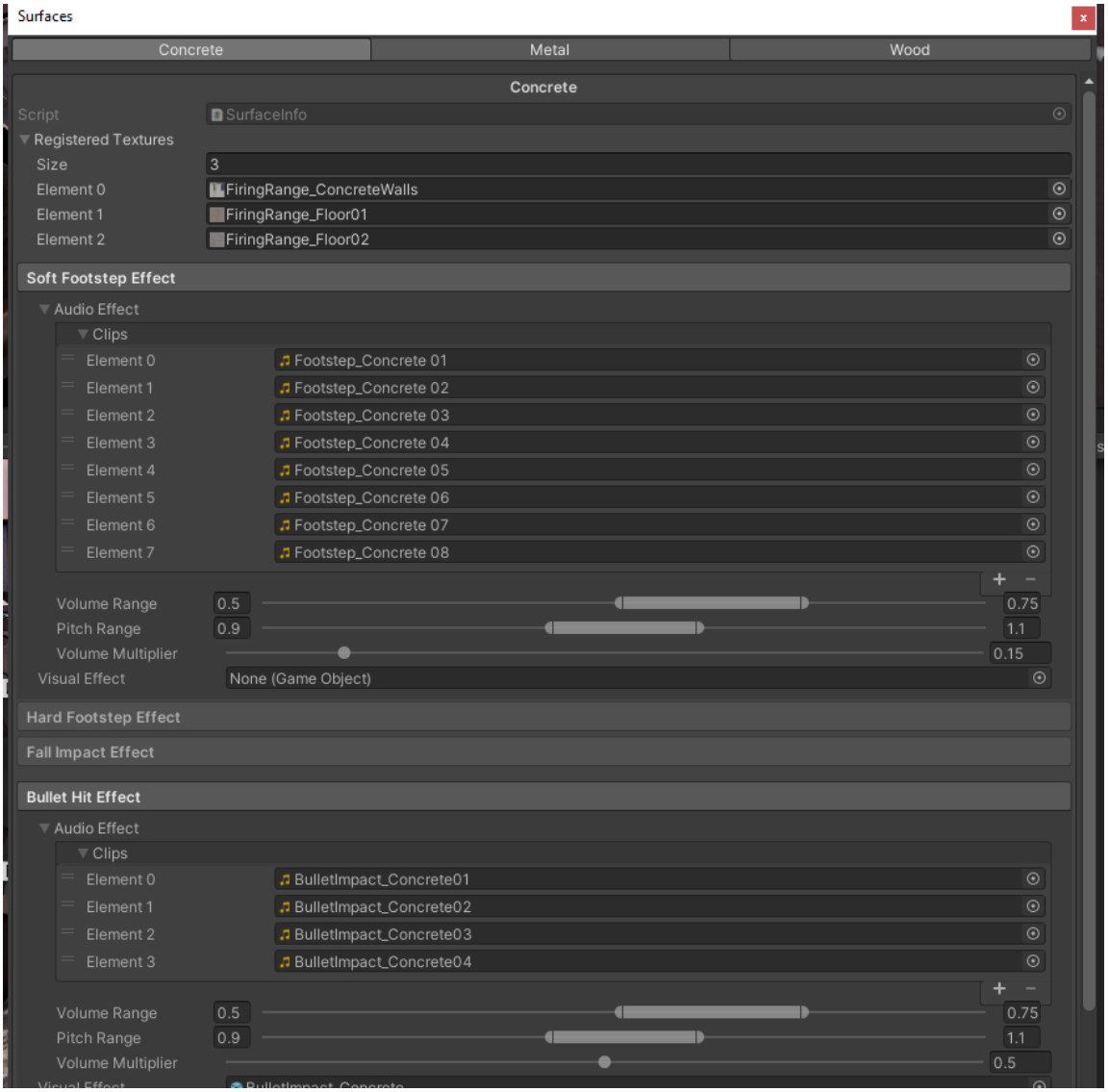
From there you can modify It's properties (name, icon, description etc.) in the right side of the window. You can also add custom properties, such as "ammo", durability etc.

How to use the surface decals system:

1. To access the **surface management** settings go here:



Here you can select from all of the surfaces available (e.g. Concrete, Metal and Wood). After that you can change/add your own sound effects, particle System prefabs (e.g. bullet decal).



How to change the Damage Of a Gun: