



Third Person Controller - Shooter Template

(v2.5.0 - 26/11/2019)

Thank you for supporting this asset, we develop this template because a lot of developers have good ideas for a Third Person Game, but build a Controller is really hard and takes too much time.

The goal of this template is to deliver a top quality controller that can help those who want to make a Third Person Game but are stuck trying to make a controller.

With this template, you can setup a 3D Model in just a few seconds, without the need of knowing advanced scripting or wasting time dragging and drop game objects to the inspector, instead you can just focus on making your game.

--- Invector Team ---

Ps* This Documentation will cover the Shooter Features, there is another documentation for the Basic Locomotion and Melee Combat on their respective folders.

Summary

FIRST RUN	3
CREATING A NEW SHOOTER CONTROLLER	5
HOW TO ALIGN A SHOOTER WEAPON (RIGHT HAND)	7
HOW TO ALIGN THE LEFT HAND IK FOR ALL WEAPONS	10
IK ADJUST FOR FINE TUNING OR CREATE NEW POSES	12
HOW TO APPLY DAMAGE TO A TARGET	14
HOW TO APPLY DAMAGE TO THE PLAYER	16
ITEM MANAGER (INVENTORY)	19
COLLECTABLE STANDALONE (NO INVENTORY)	23
TOPDOWN / 2.5D / POINT & CLICK CONTROLLER / MOBILE	25
HEAD TRACK	27
LOCK-ON TARGET	29
THROW OBJECT	30
BODY SNAPPING ATTACHMENTS	31
WEAPON HOLDER MANAGER	34
DRAW/HIDE WEAPONS	36
AMMO MANAGER	37

FIRST RUN

IMPORTANT

This is a **Complete Project**, and as every complete project it includes a custom **InputManager**, **Tags**, **Layers**, etc... **Make sure that you import on a Clean Project.**

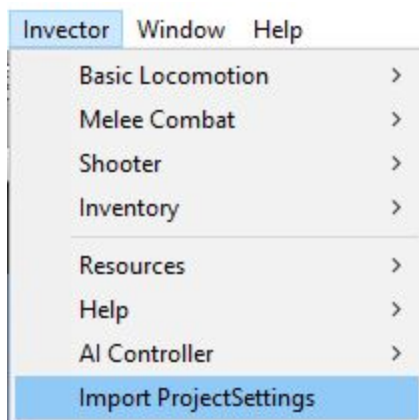


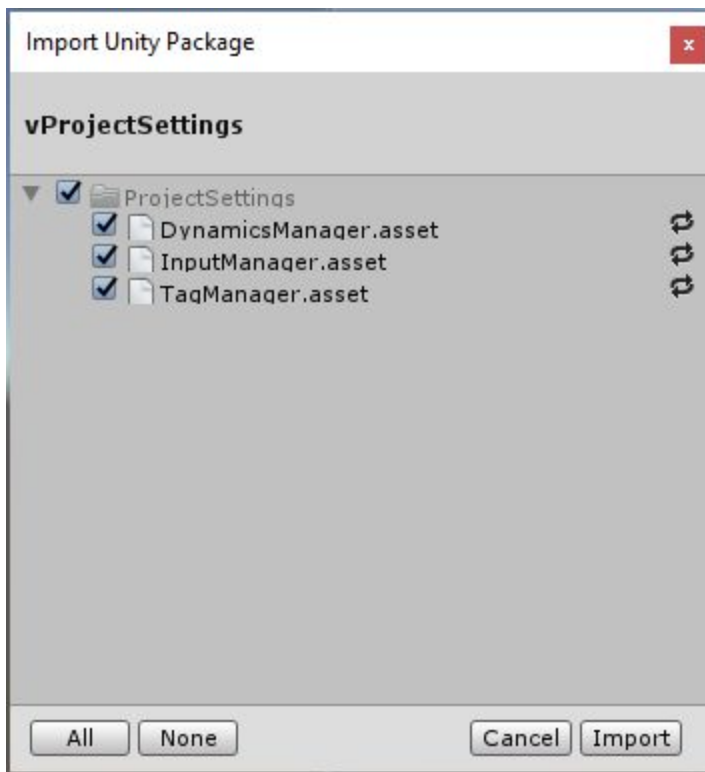
- *Importing on an existent project*

There are basically 3 files that are **extremely necessary for the correct functioning of this template.**

- **DynamicsManager.asset** - this will apply correct all the Collision Matrix of our Layers, for example we need the layer "Triggers" to not collide with the layer "Player".
- **InputManager.asset** - We have a custom input mapped with the Xbox360 controller, if you don't input those 2 files, the template will present errors and undesired behaviour.
- **TagManager.asset** - Includes all the necessary Tags and Layers for the project to work correctly.

After importing the template you can manually import those files by going to the tab **Investor** > **Import ProjectSettings.**



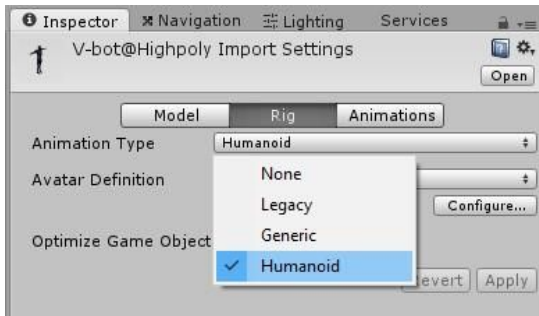


Now that you have imported the necessary files, you can explore the several demo scenes and figure it out what kind of Third Person Game you want to create.

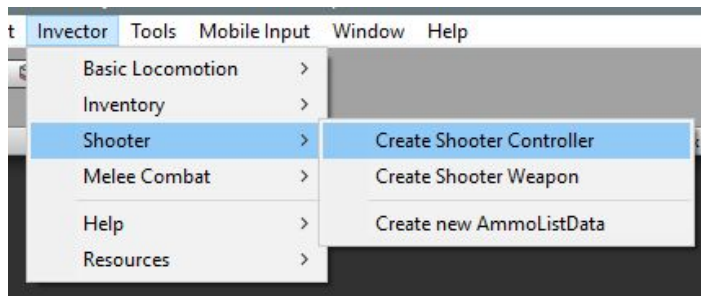
***Updates also need to be imported into a Clean Project, so MAKE SURE TO BACKUP your previous project and transfer the necessary files to your new project. ***

CREATING A NEW SHOOTER CONTROLLER

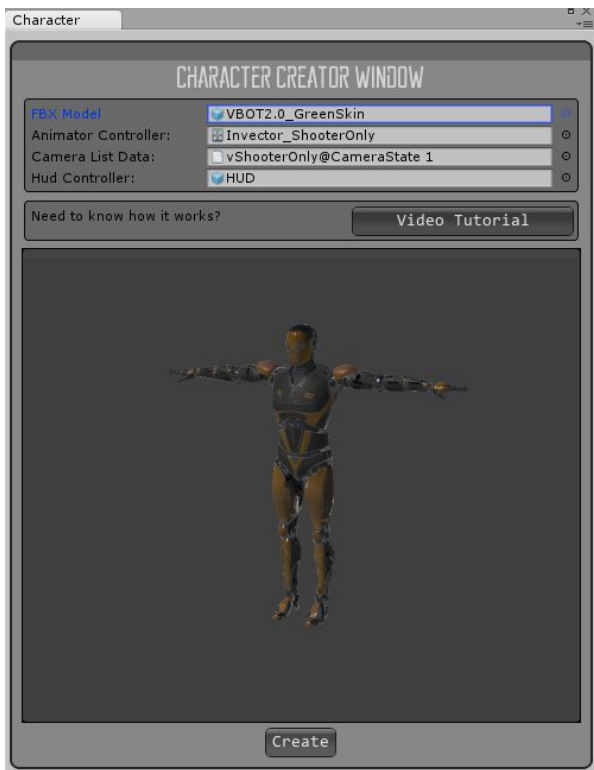
Make sure that your fbx character is set up as **Humanoid**



To setup a new character, go to the tab *Investor* > *Shooter* > *Create Shooter Controller*

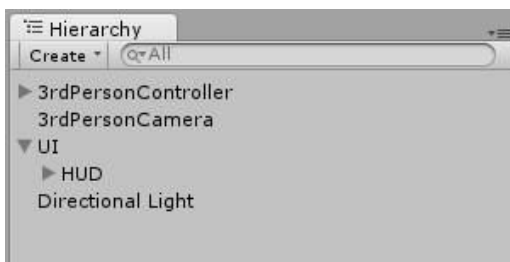


Make sure your Character is **Fully Rigged** and set up the FBX as a **Humanoid**, then assign the FBX to the field “FBX Model” and click on the button “Create”.



Ps Make sure to select the **Invector_ShooterOnly** or **Invector_ShooterMelee** if you want to use both shooter and melee as your Animator Controller, you can find the file at the folder: Shooter > Animator, or just click on the little circle icon.*

The **Character Creator** window will take care of all the hard work automatically and set up components such as capsule collider, layers, tags, rigibody, etc... It will create the **ThirdPersonController**, **ThirdPersonCamera** and a UI Canvas with a **HUD** to display health, stamina and other information.



Your Capsule Collider settings will be based on your model proportions, if the capsule gets the wrong size, make sure that you rig is correct, and that your **model is using the correct Scale Factor** the same goes if the ragdoll gets weird.

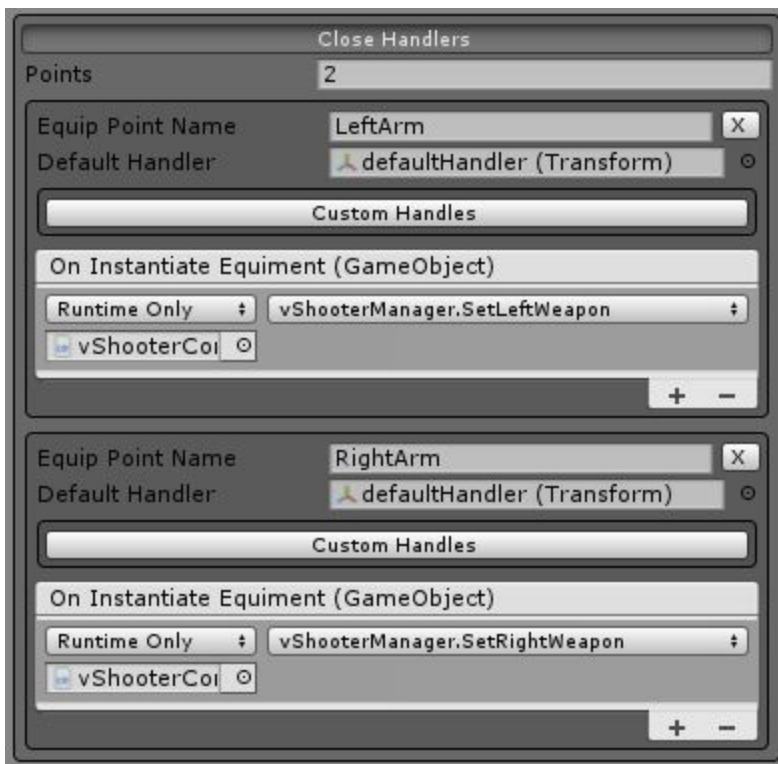
Hit Play and enjoy ☺

HOW TO ALIGN A SHOOTER WEAPON (RIGHT HAND)

After creating your character, you will need a *'handler'* to manage the position/rotation of the weapons you instantiate in game.

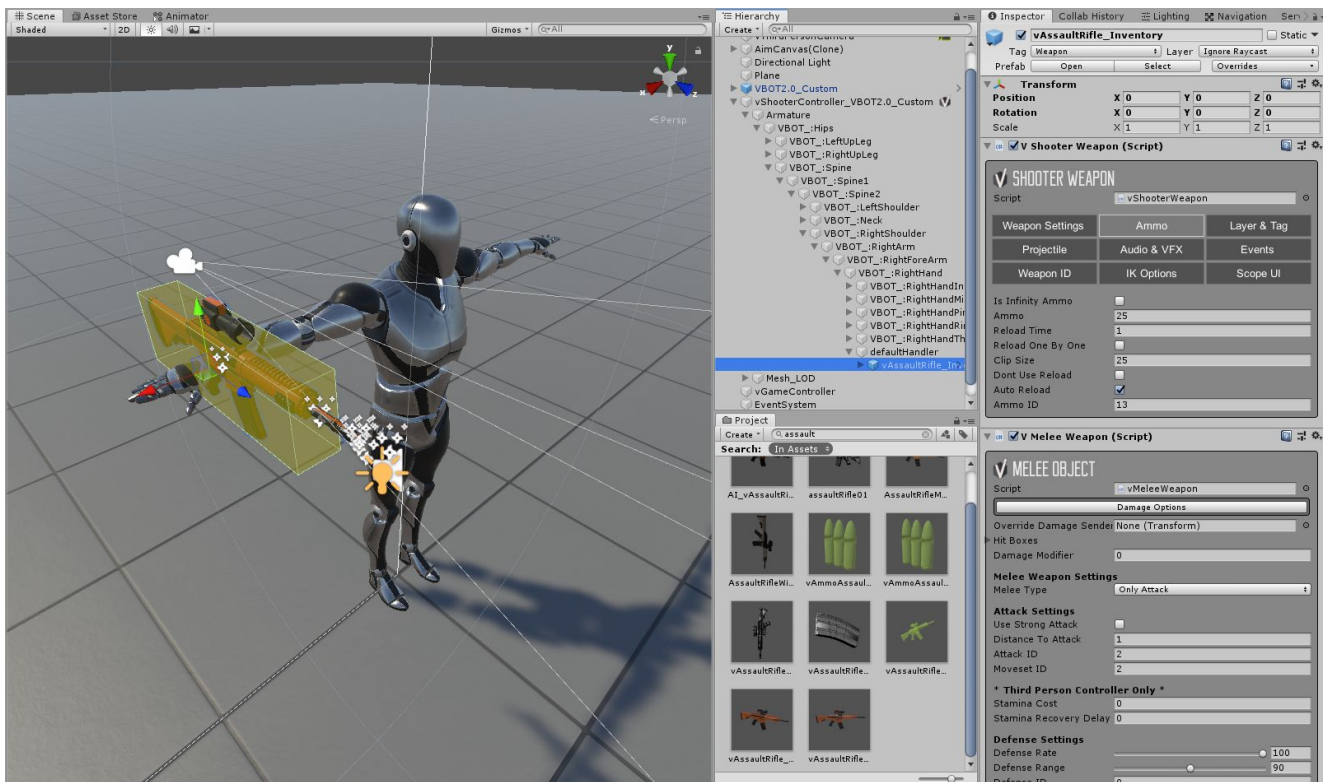
A **Handler** is basically an empty transform located inside your character's left and right hands, you will first align this handler into a position where a weapon looks correct in the characters hand and all weapons will be instantiated there when you pick up the collectible.

If you choose to use our **Item Manager**, both handlers will be automatically created for you once you attach the ItemManager to your character and you can see them or create customizable handlers for specific weapons in the "Open Handlers" tab.



Shooter Weapons will be equipped in the **RightArm defaultHandler**, so select this transform in the hierarchy and drag and drop any Weapon Prefab inside the defaultHandler.

For example search in the Project window to “vAssaultRifle_Inventory”, drag and drop inside the ‘defaultHandler’ and reset it’s position and rotation back to 0, 0, 0.



Now you can manually align the ‘defaultHandler’ to a position close to where the weapon should be located.

*** ATTENTION:** You should NOT move the Weapon Prefab itself, this gameObject should be at 0,0,0, **you will only move/rotate the Handler**, this way all the weapon prefabs that will be instantiated inside the handler will be aligned as well.



You can now hit **Play** to see if the **weapon position in the RightHand** looks good, it doesn't need to look perfect because we're still going to the **IK Adjustments** where we will **align the LeftHandIK**.



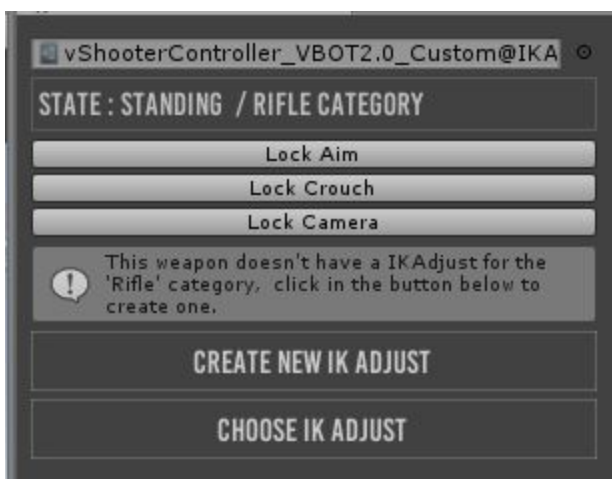
HOW TO ALIGN THE LEFT HAND IK FOR ALL WEAPONS

Go to your ShooterManager inspector and select the tab “IK Adjustment” and hit the button to create a new “IK Adjust List”.



A new IK Adjustment List will be created and automatically assign for you, a new button called “**Edit IK Adjust List**” will appear, click on it and a new window will open, you can dock this window anywhere you like.

Hit **Play** and select the **ShooterController** to see this menu:



Hit “Create New IK Adjust” and before we started playing around with the many cool IK options, we first need to align our LeftHand IK, you can do this by sliding the XYZ Left IK Rotation and Position Offsets.



As you can see in the infobox this offsets **will affect all weapons** just like the Handler position/rotation you previously set up. Now that we have a base alignment we can proceed to explore the **features of IK Adjust**.

Once aligned, you don't need to change the values of the LeftIK again since you now be able to create adjustments for each weapon and state.

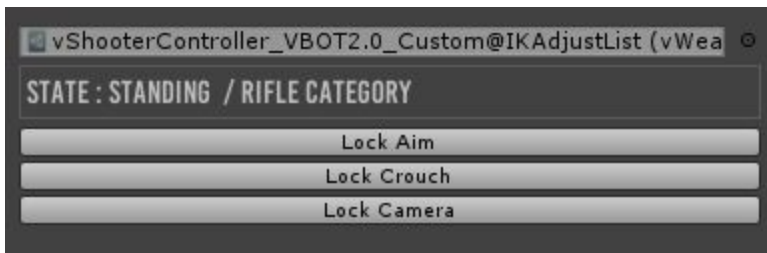
IK ADJUST FOR FINE TUNING OR CREATE NEW POSES

Now that we properly align the **defaultHandler** and the **LeftHandIK** for **all weapon prefabs**, we can start modifying the IK to create more **polished** or **even entirely new poses**, for **each state** and **each weapon category**.

The “State” will inform what state you are, there are basically 4 states that you can create unique IK modifications, those are:

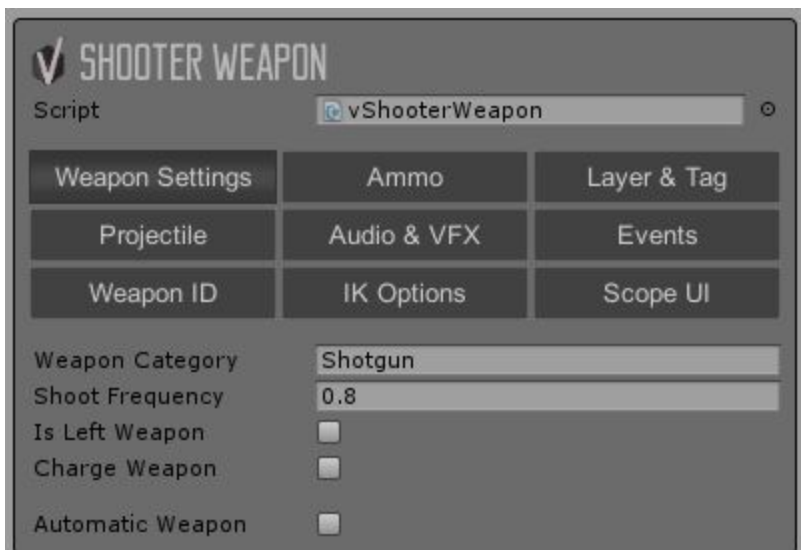
- Standing
- Standing Crouch
- Aiming
- Aiming Crouch

You can switch the state by pressing the buttons to **LockAim**, **LockCrouch** and **LockCamera**. Start by Locking the Camera so that the Headtrack doesn’t influence your pose.

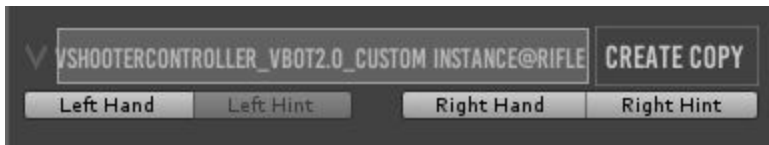


You can create different IKAdjusts for each weapon or use the same adjust if the weapon share the same **Category**, for example, if you have a large number of Pistols in your game that use the same pose, you don’t need to create an IKAdjust for each weapon, just set the same Category for all the ShooterWeapon prefabs.

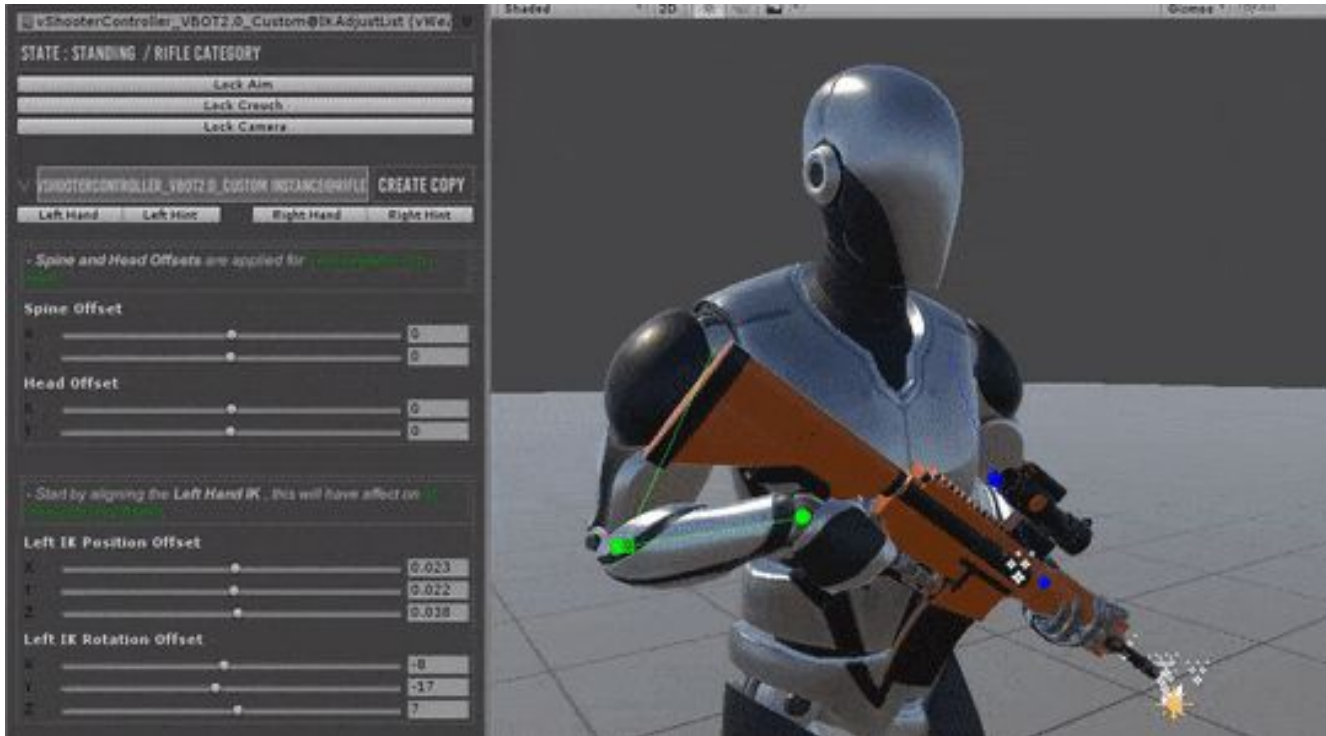
You can type any **Category** directly in the **ShooterWeapon** prefab:



You can create a **Copy** from another **IK Adjust**, for example if you have 2 rifles slightly different from each other, just make a copy and do the adjustment for the other rifle.



You can also select the gizmos handlers directly from the menu or click on the little sphere/square of each arm. Or reset the position/rotation by clicking in their respect buttons, the “Reset” button will reset both.



You can also create **Offsets** for the **Spine** and **Head** to help you set up the position.



- *This feature was designed to help you better align the weapon into your Character Pose, but if you want the best result as possible the ideal is to replace the animations to animations created using your Character Rig. We recommend the use of uMotion to quickly and easily adjust the animations to your character's rig.*

HOW TO APPLY DAMAGE TO A TARGET

The target must have a vHealthController and a Capsule Collider so that our Damage System can identify it as a living target and actually apply damage to it.



Shooter: You need to set the DamageLayer of your target in the ShooterManager.

If you want to apply damage to individual body parts you can create a Ragdoll and set the collider layers to BodyPart, each ragdoll collider comes with a DamageReceiver, you can even set a Damage Multiplier if you want to apply extra damage in the Head for example.

OR if you're creating a simple top down game for example, you can save performance leaving the enemies without a ragdoll and applying damage directly to the main capsule collider, in this case you can set the Damage Layer to Enemy.



MeleeCombat: You need to set the Tag of your Target in the MeleeManager / HitDamageTags field, you can assign several tags to hit different targets.

The image shows a software interface titled "MELEE MANAGER" with a close button. Below the title bar, there is a "Script" dropdown menu set to "vMeleeManager". Three buttons are stacked vertically: "Open Default Info", "Open Events", and "Add Extra Body Member".

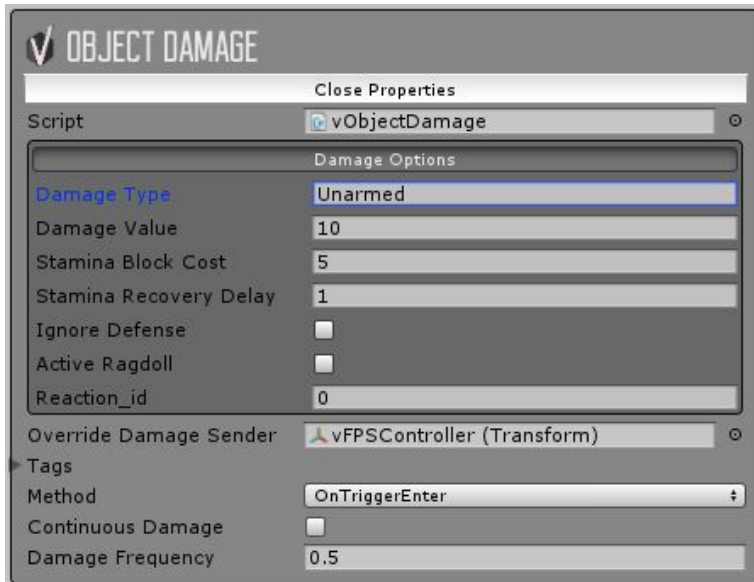
The "Body Members" section contains four buttons arranged in a 2x2 grid: "LeftLowerArm", "RightLowerArm", "LeftLowerLeg", and "RightLowerLeg".

The "Who you can Hit?" section is expanded, showing "Hit Properties". Under "Hit Damage Tags", there are several settings: "Size" is set to "1", "Element 0" is set to "Enemy", "Use Recoil" is checked, "Draw Recoil Gizmos" is unchecked, "Recoil Range" is a slider set to "90", and "Hit Recoil Layer" is set to "Default".

The "Weapons" section contains two dropdown menus: "LeftWeapon" and "RightWeapon", both currently set to "None (V Melee Weapon)".

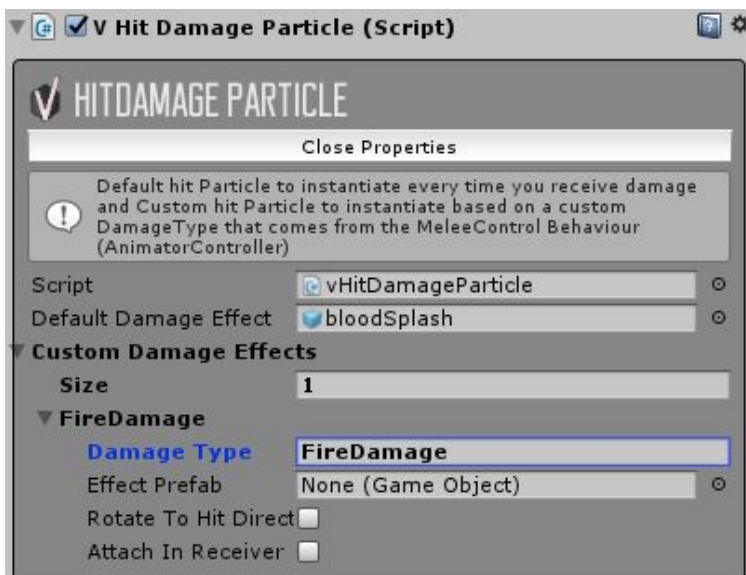
HOW TO APPLY DAMAGE TO THE PLAYER

We have a few examples on how to apply damage to the Controller, basically you need the **vObjectDamage** script which is attached to the Pendulum and Spikes.



- **DamageType**: Used together with the HitParticleDamage component, you can trigger different particles for different type of damage.

For example if you have an area with fire, you can add a vObjectDamage there and add a DamageType of "FireDamage", then add a Custom Damage Effect with the same DamageType to your **HitDamageParticle** on your Character and add the particle effect to burn your character.

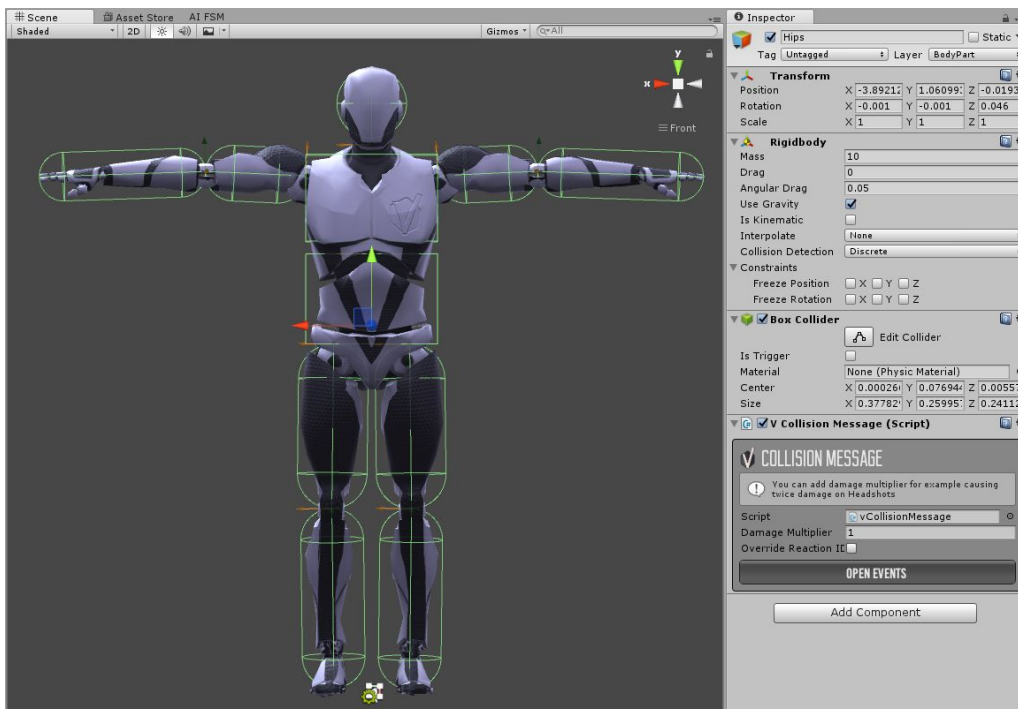


*This component is attached to the Controller or any object that contains the HealthController

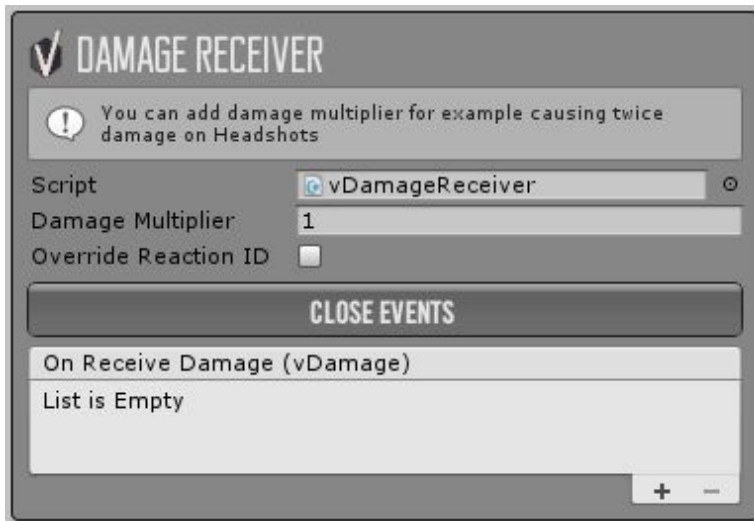
- **DamageValue:** How much damage it will be applied to the vHealthController of the target.
 - **Stamina Block Cost:** You can ignore that option, it's only for the ThirdPersonController.
 - **Stamina Recovery Delay:** You can ignore that option, it's only for the ThirdPersonController.
 - **Ignore Defense:** If you're using a MeleeCombat Controller, it will ignore the defense and apply damage anyways.
 - **Active Ragdoll:** It will active the ragdoll on your character, if it has one.
 - **Reaction ID:** You can trigger specific hit reaction animation, you can use -1 if you don't want to trigger any animation.
 - **Override Damage Sender:** Assign the root object otherwise the AI will target the object that has this component instead. For Example: If you apply the vObjectDamage to be a Hitbox of a LeftHand of a character, the vHealthController or AI will have the LeftHand as the target instead of the GameObject parent.
 - **Tags:** What tags you will apply damage to
 - **Method:** OnTriggerEnter or OnCollisionEnter
 - **Continuous Damage:** Useful for fire damage for example
 - **Damage Frequency:** Frequency to apply the damage, if Continuous Damage is enabled.
-

Using the Ragdoll Colliders as BodyParts to inflict precise damage.

SHOOTER > If you want to cause damage for each body member using the ragdoll colliders, **UNCHECK** the “Disable Colliders” and you can add damage multiplier on each member.



* You must use a different Layer in the Ragdoll Colliders like "BodyPart" and another for the main capsule collider like "Enemy", this way the Detection will detect the Enemy object as a target, but the ShooterManager will actually apply damage to the "BodyPart".



Body Parts use the Damage Receiver to receive damage.

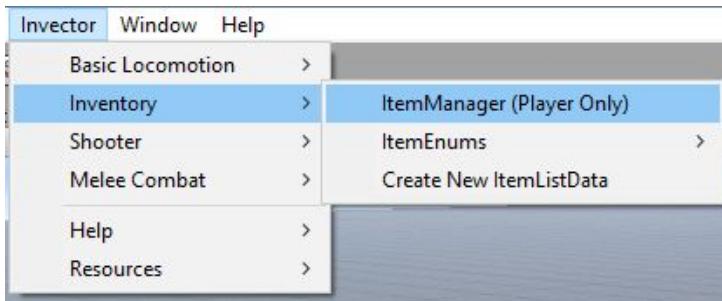
A DamageReceiver is attached to each ragdoll collider, this will allow to Player or AI to apply damage to each bodypart instead of the CapsuleCollider.

- **Damage Multiplier:** multiply the damage value
- **Override a Reaction ID:** Check this option to override the hit reaction animation to trigger a specific animation.

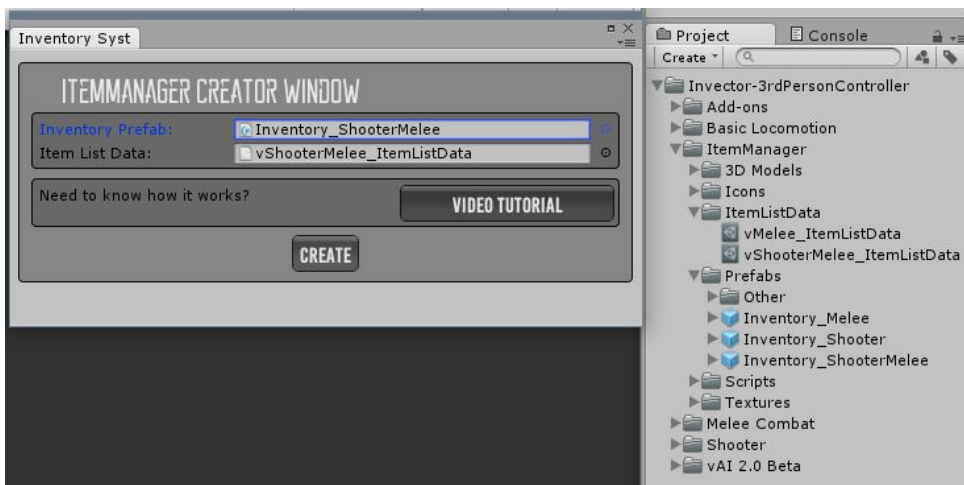
For example, if you want to cause 2x damage and trigger a specific reaction animation when shooting in the Head, simple change the values in this component.

ITEM MANAGER (INVENTORY)

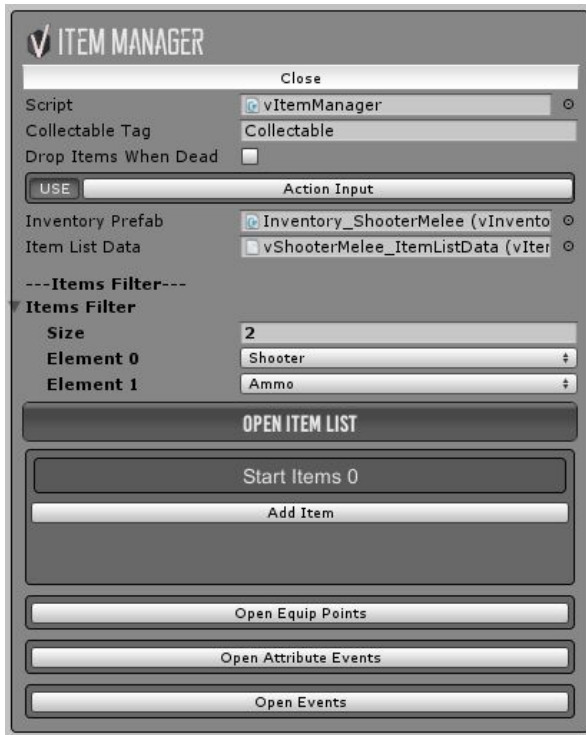
- Add the ItemManager into your Player from the menu **Invectore > Inventory > ItemManager**



- Select the Inventory Prefab from the **Project > ItemManager > Prefabs**
- and a **ItemListData > vShooterMelee_ItemListData**



- You can use the **ItemFilter** to filter the items that you want to show



We also added **Events** that are very useful, for example if you want to lock the input of the character while the Inventory is Open, just assign the Character and call the method LockInput from the vMeleeCombatInput.



You also have the option to Drop all your items when you die, it will instantiate the Prefab that you select to be the Collectable of your item.

Click in the **Open Item List** button, to manage or create new items



You can create new items or duplicate a current one, keep in mind that each item has a unique ID.

When creating a Weapon Item, you need to assign the **Original Object** (that instantiate into the Player with a `vMeleeWeapon` or `vShooterWeapon`) and a **DropObject** which we have a prefab called "**CollectableEquipment**" that you can use and it will automatically drop the item you assign or create a unique collectable with a mesh that matches your item.



ID 12
- Shotgun
- Shooter



Shotgun

EditName

Description
Tactical Shotgun

Item Type
Shooter

Stackable
☐

Icon
 shotgunIcon 

Original Object
vShotgun

Drop Object
vCollectableShotgun

Attributes

Add Attribute

Damage
50

AmmoCount
8

Custom Settings

Script
vItem

Two Hand Weapon
☒

Equipable Settings

Equip ID
3

Custom Equip Point
defaultPoint

Equip Delay Time
0.5

Don't forget to add the attribute Damage & AmmoCount of your weapon, this will allow you to drop and collect your weapon with the same amount of weapon, making it into a unique weapon.

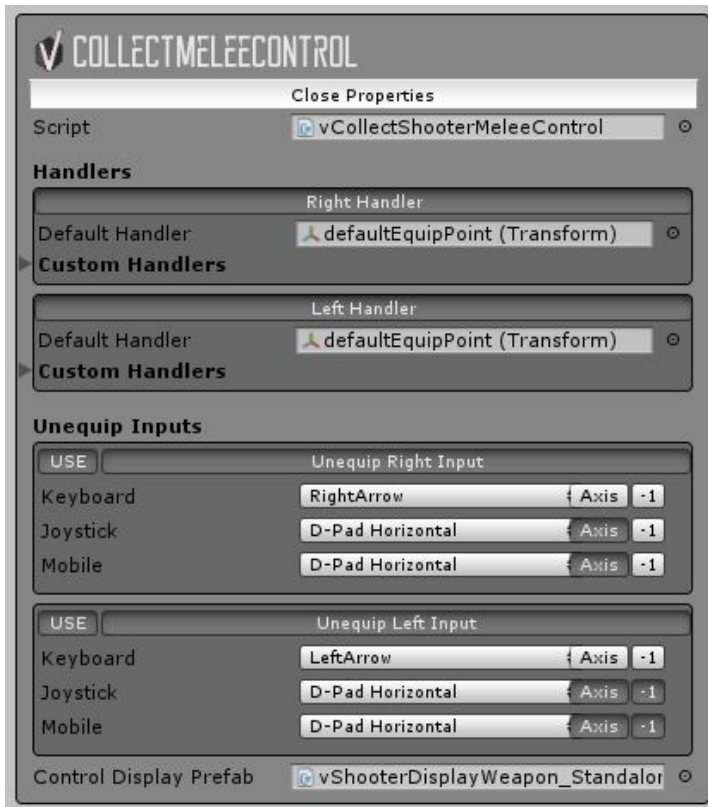
This Inventory Example goes further and further into options to customize, like consumable items, if is stackable or not, and much more that is better explained on video tutorials that you can watch on our [Youtube Channel](#).

COLLECTABLE STANDALONE (NO INVENTORY)

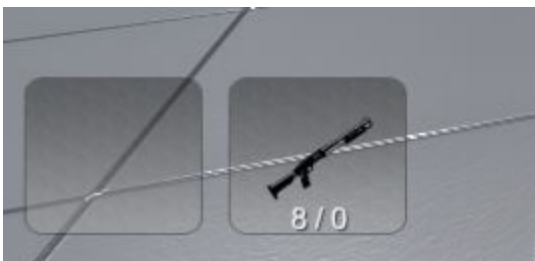
If you don't want to use the ItemManager to manage your items, we have another solution for 'on demand' collectibles, notice that you can only equip 1 item, once you try to equip another the current item will drop.

Take a look into the Demo Scene call "vShooterMelee_NOInventory", instead of adding the ItemManager component, now you will add the "vCollectShooterMeleeControl" component to automatically collect and equip weapons.


You need to create the defaultEquipPoint to equip weapons and assign inputs to drop them.



We also have a pretty simple example of a Display HUD to show what weapons you're equipped with, it's call "vShooterDisplayWeapon", search in the project folder and drag and drop the prefab into the scene.



For the ItemManager we need a prefab for the actual weapon that goes into the Player and another to be the Collectable, but in this case the CollectableStandalone is both. Take a look into one of the several example of collectables we have for both melee and shooter weapons.



The image shows a Unity-style inspector window for a component named **VCOLLECTABLESTANDALONE**. The window has a dark grey background with white text. At the top left is a small icon of a checkmark inside a circle. The title bar is at the top. The main area contains a list of properties on the left and their corresponding values on the right. The properties are: Script (vCollectableStandalone), Disable Collision (checkbox), Disable Gravity (checkbox), Reset Player Settings (checkbox), Play Animation (empty field), End Exit Time Animation (0.8), Avatar Target (Root), Match Target Mask (X 0, Y 0, Z 0), Match Target (None (Transform)), Start Match Target (0), End Match Target (0), Active From Forward (checkbox), Use Trigger Rotation (checkbox), Destroy After (checkbox), Destroy Delay (0), On Do Action Delay (0), Target Equip Point (defaultEquipPoint), Weapon (vShotgun_NoInventory), Weapon Icon (shotgunIcon), and Weapon Text (Shotgun). At the bottom is a button labeled **OPEN EVENTS**.

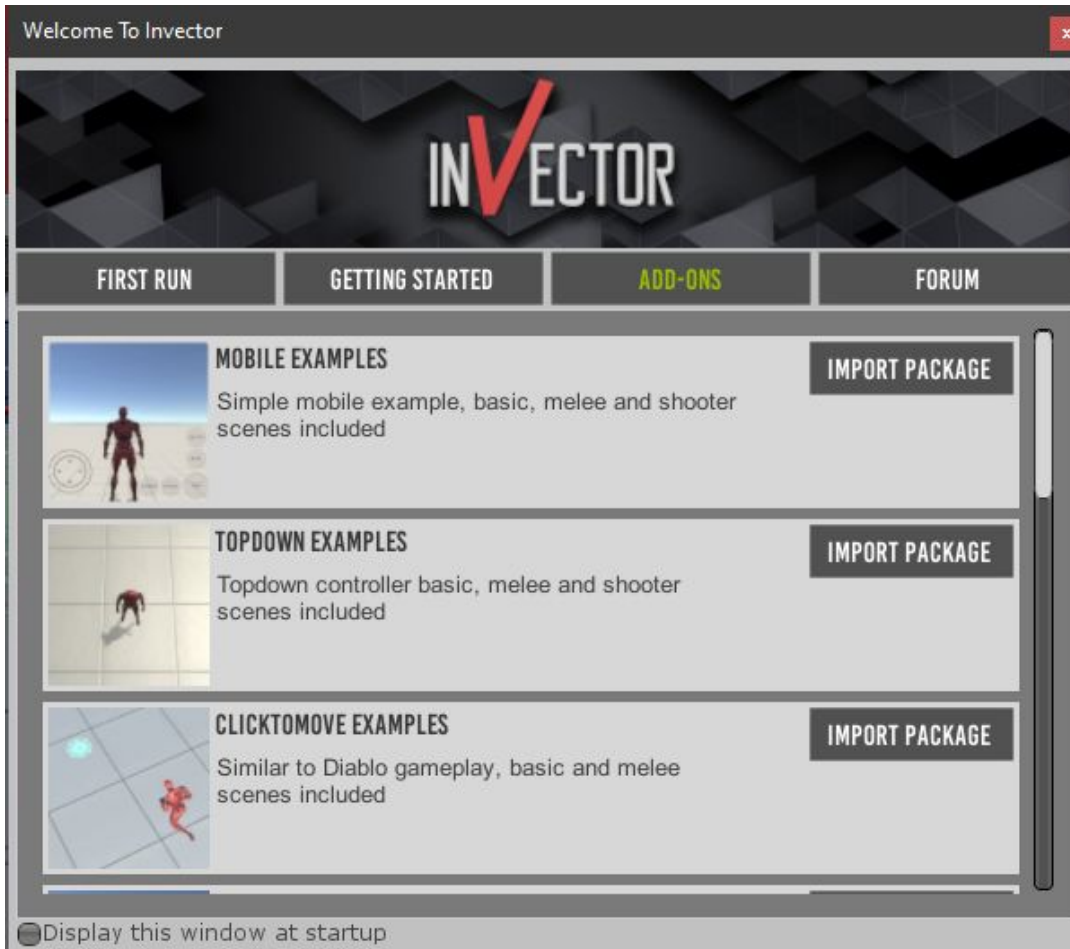
Property	Value
Script	vCollectableStandalone
Disable Collision	<input type="checkbox"/>
Disable Gravity	<input type="checkbox"/>
Reset Player Settings	<input type="checkbox"/>
Play Animation	
End Exit Time Animation	0.8
Avatar Target	Root
Match Target Mask	X 0 Y 0 Z 0
Match Target	None (Transform)
Start Match Target	0
End Match Target	0
Active From Forward	<input type="checkbox"/>
Use Trigger Rotation	<input type="checkbox"/>
Destroy After	<input type="checkbox"/>
Destroy Delay	0
On Do Action Delay	0
Target Equip Point	defaultEquipPoint
Weapon	vShotgun_NoInventory
Weapon Icon	shotgunIcon
Weapon Text	Shotgun

OPEN EVENTS

It's important to assign the correct gameobjects into the Events, we turn off the collision and gravity of the weapons when equipped and turn on when you drop them.

TOPDOWN / 2.5D / POINT & CLICK CONTROLLER / MOBILE

We have different types of controllers that are included in the project as a bonus, you can import those packages by going to *Invector* > *WelcomeWindow* > *Add-ons*



Each scene contains examples on how to set up each gameplay type.

To turn your Third Person Controller into a TopDown or Isometric controller just go into your ThirdPersonCamera and change the CameraState to **TopDown@CameraState**, **Isometric@CameraState** or **2.5@CameraState** depending on what controller you want.



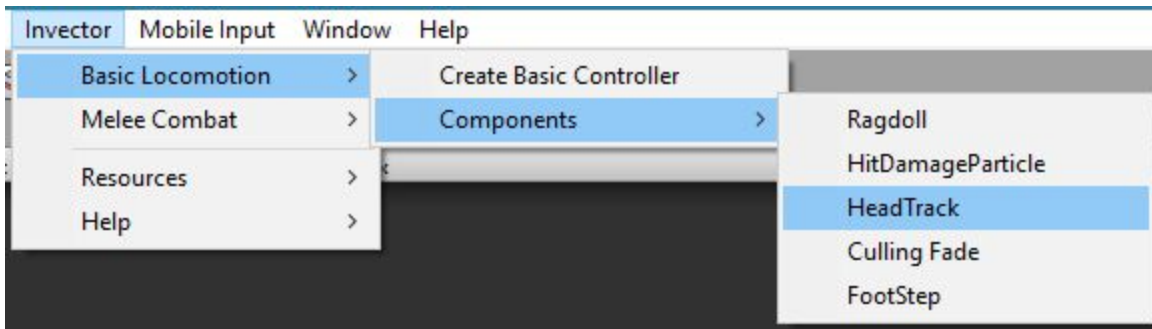
Go to the add-on folder you want and replace your current **vThirdPersonController** component for the **vTopDownController** or **2_5Dcontroller** in the Player Inspector.

To use the **Point&Click** you can still use the **vThirdPersonController**, but you will need to replace the Input to **vPointAndClickInput** or **vMeleePointClickInput** (if it's a melee character), for more information check the **Invector_Point&Click_Melee**.

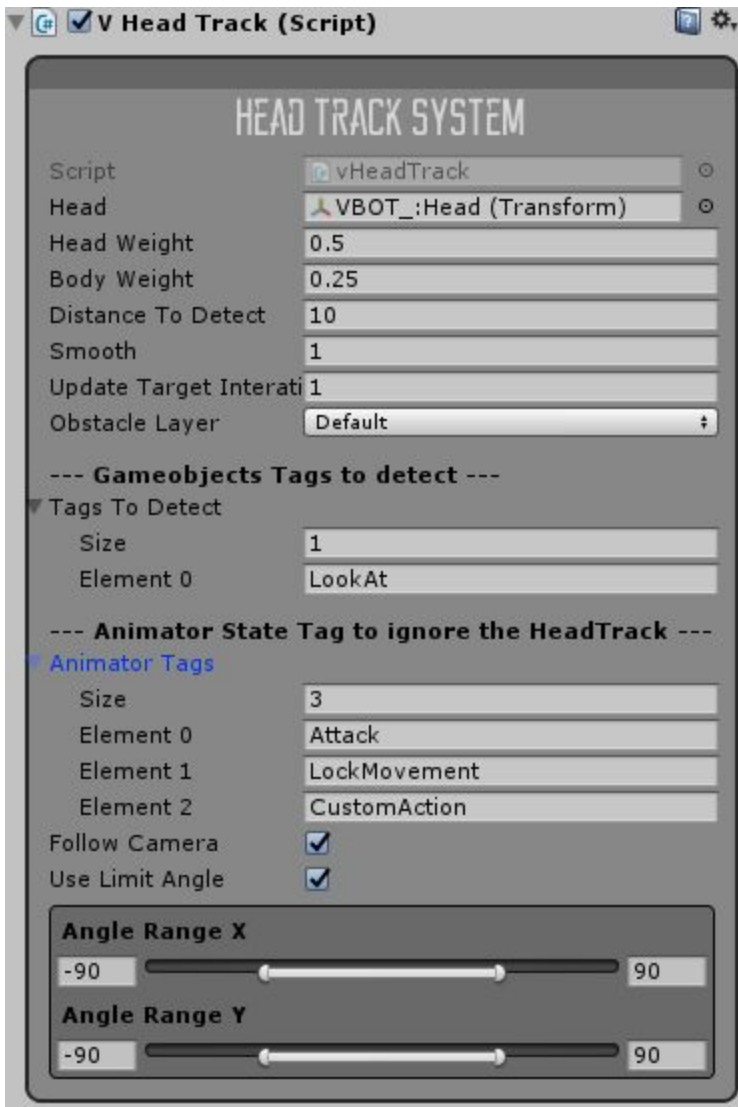
And for the **2.5DController** check the **2.5Demo** scene, you will need a **2.5Path** to navigate.

HEAD TRACK

***Shooter** - automatically add the headtrack in order to aim up/down

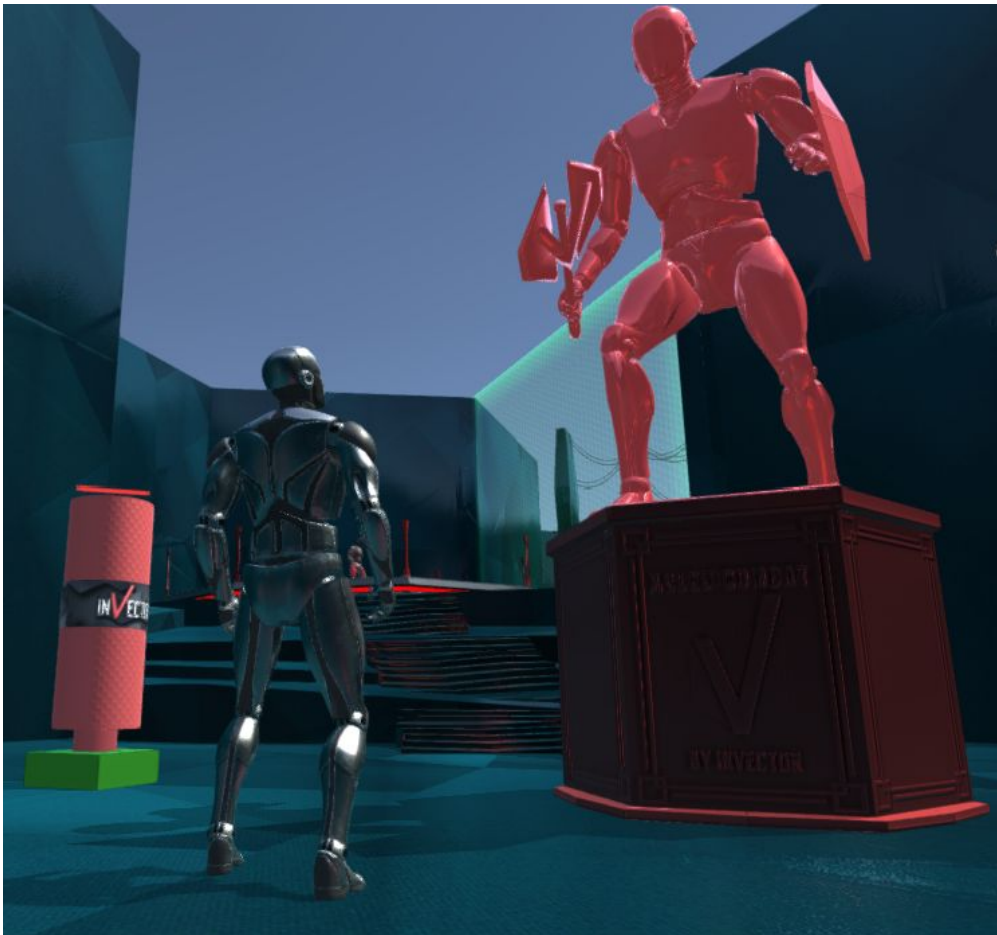


Now we have a lot more options and we can use the LookAt feature as well.



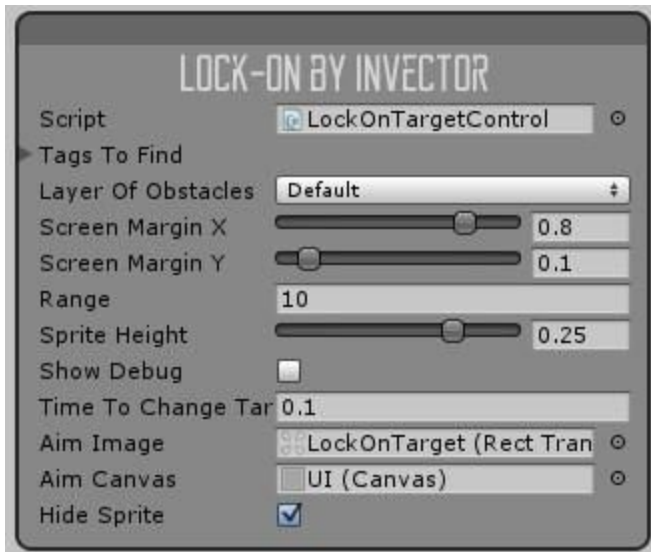
If you don't want the HeadTrack in a specific animation, you can add the Tag CustomAction into the animationState and the headtrack will turn off while this animation is playing.

To make the character look at an object, you need to add the component vLookTarget into the object, you can take a look at several examples in the DemoScenes.



LOCK-ON TARGET

You can add a Lock-on component into the Camera by opening the 3rd Person Controller menu > Components > Lock-On. The component will be ready to use, you can set up the input that activate the Lock-on in the **ThirdPersonController** script, at the method **LockOnInput**.



You can also display a **Sprite Image** into the Target by assigning an Image and Canvas.

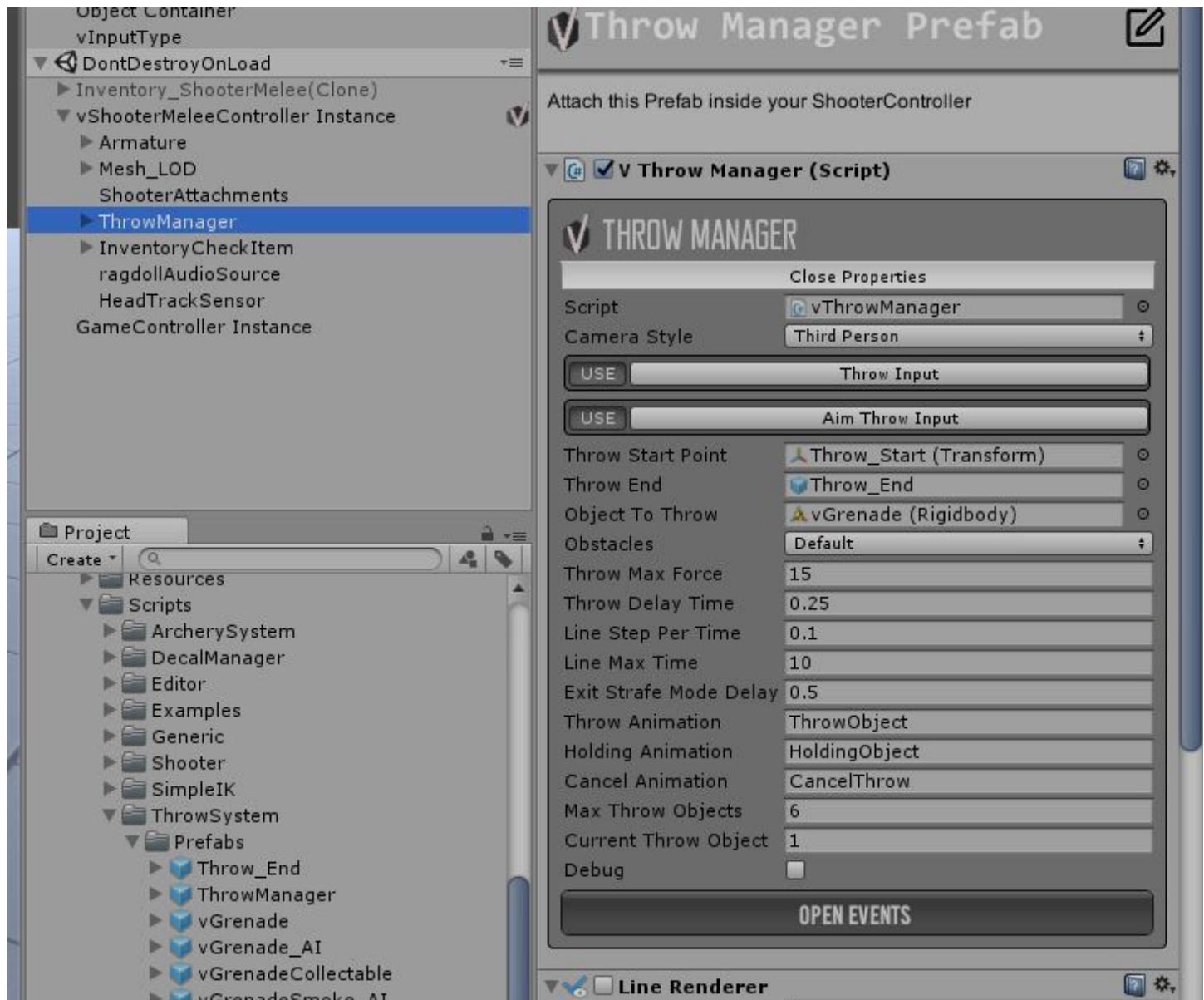
Hide Sprite will hide the sprite if the target if lock-on is false. Set off-set Y by changing the value of the **Sprite Height**.

This Lock-On currently works exclusively with our AI, it will not work out of the box with Non-Invector Characters because it needs the **vCharacter** interface to know if the target is alive. You can assign a **vCharacterStandalone** script into your gameobject, it contains health and a **TakeDamage** method to receive damage.

Shooter - You can use the Lock-On by checking the “Use Lock-On” option on the ShooterManager.

THROW OBJECT

The Throw System is pretty **Plug & Play**, just drag and drop the **ThrowManager** prefab inside your ShooterController, you can call the Event ForceToHideWeapons from the DrawHideWeapons to hide your weapons while is using the throw aiming.



BODY SNAPPING ATTACHMENTS

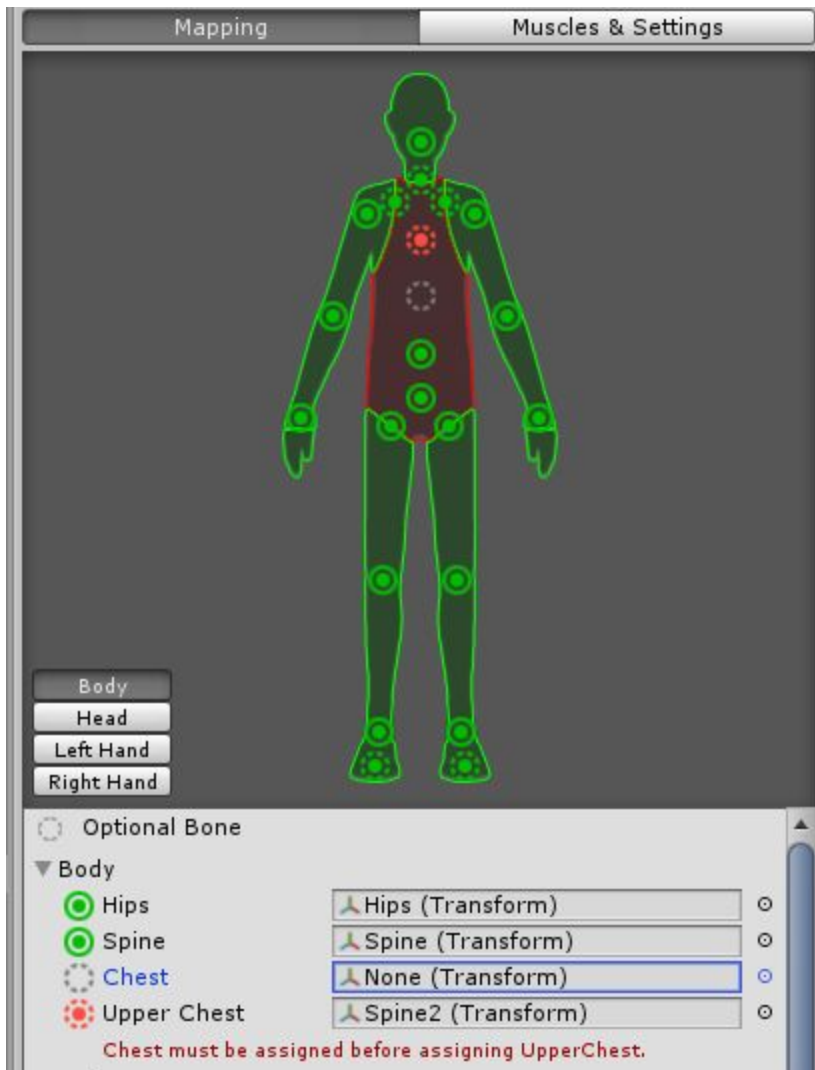
We created this feature to make it easier to transfer attachments from one controller to another.

This means that you can create a Prefab of a Character Attachments and quickly add to another character, without the need of adding attachments one by one on each bone.

First, create an Empty GameObject inside your character, add the “**vBodySnappingControl**” and hit the “Create New BodyStruct”.



If your Avatar is already set up as Humanoid and all the bones are correctly mapped, it will all be automatically assigned for you, in some cases Unity doesn't recognize a Spine or Chest, so you need to fix by going to your Avatar and assigning the correct Bone, example:



Now going back to our Character BodySnap Control, you can add all your character attachments such as particles that activated on a specific bone, itemManager Handles, anything that you may use and assigned to a specific bone, once you hit Play that GameObject will be attached to the bone you assigned.

BodySnap Attachments



With this feature you can set all the attachments your character in one prefab, making it easier to transfer attachments from one character to another.

Once you hit Play the attachments will go to the desired bones.

V Body Snapping Control (Script)



BODY SNAPPING CONTROL

Script

Create New BodyStruct

Load Bones

Body Struct

Show Labels ☒

Bone Snapping List (20)

Orientation

On Snap ()

List is Empty

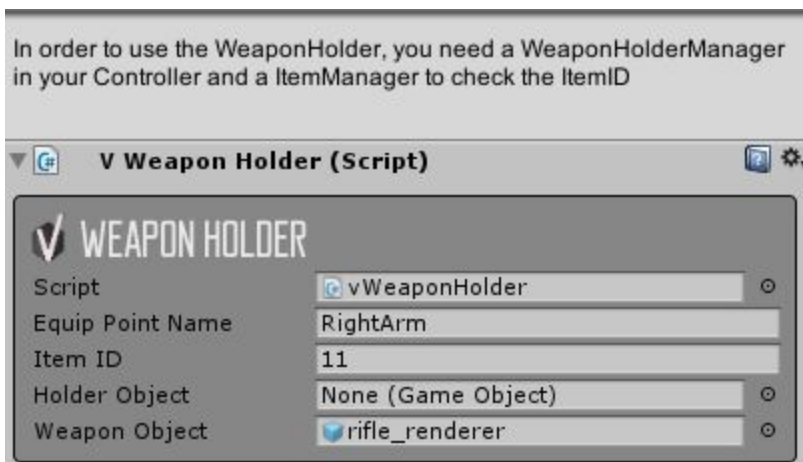
WEAPON HOLDER MANAGER

*This feature requires a ItemManager

Add this component to your Shooter Controller and there is no need to setup anything here:

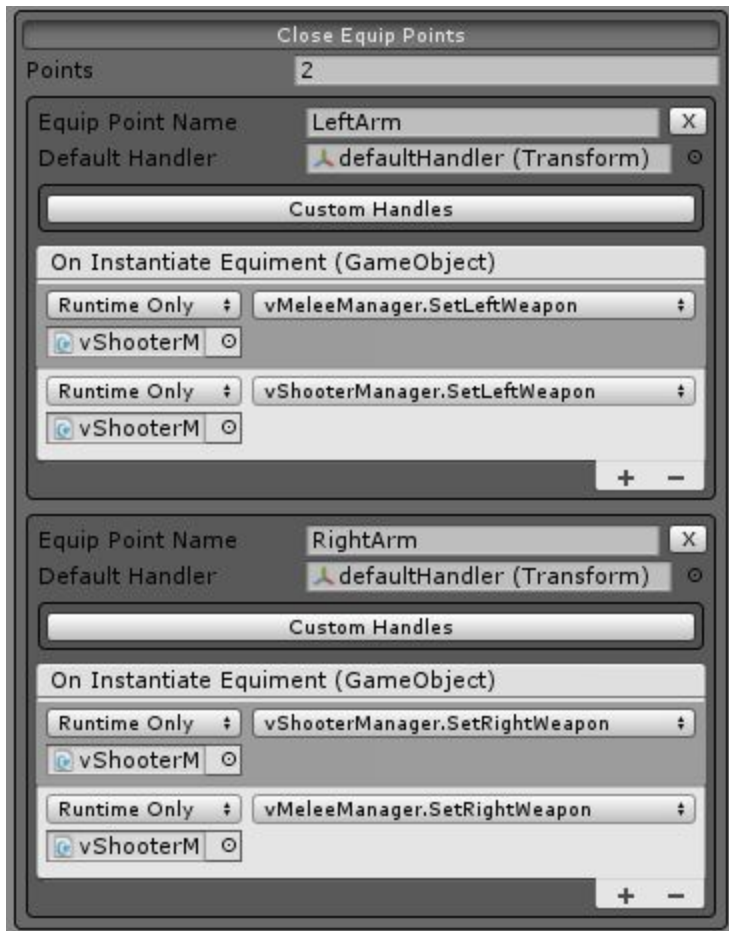


Now go inside your controller, create an empty gameObject and add the vWeaponHolder component.

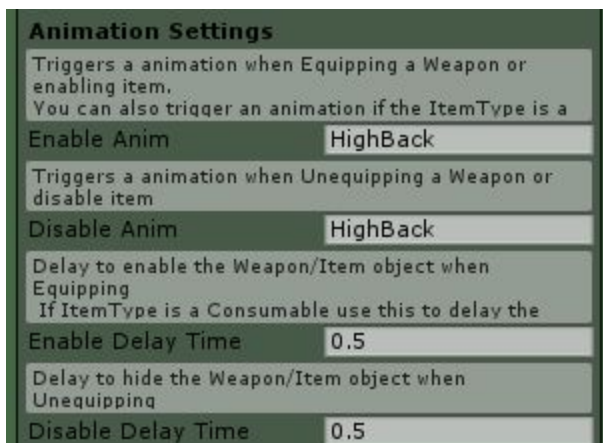


The **EquipPointName** can be found in the **ItemManager** > EquipPoints.

By default it comes with 2 EquipPoints, LeftArm and RightArm but you also create your own custom equipPoints.



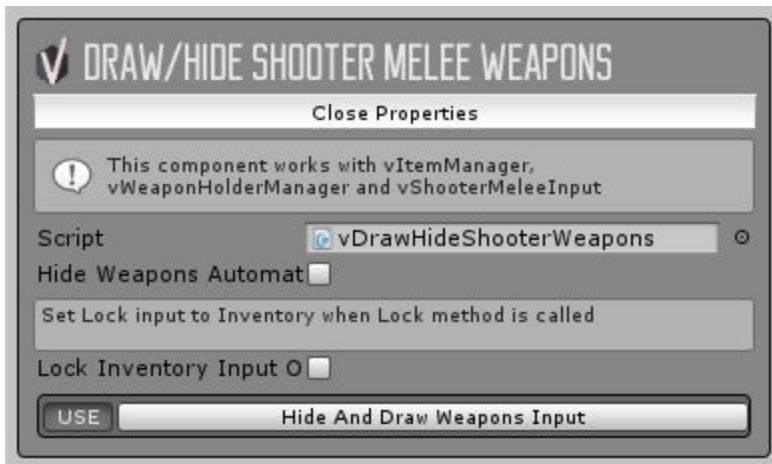
The **ItemID** can be found in the ItemManager **ItemListData**, there you can also set what animation will be played when you equip/unequip your weapon.



The **HolderObject** is in case your weapon has a holster or something. And the **WeaponObject** is just the 3D model without any scripts or collision, you can add all of this inside the character and leave it disabled. (Check our demo scenes to see examples)

DRAW/HIDE WEAPONS

This feature is to automatically or via input hide weapons without unequipping them.



It's pretty straightforward to use, simply add the component to your ShooterController, it must already have a ItemManager and a WeaponHolderManager already setup.

By checking the option Hide Weapons Automatically a field will appear so you can set a timer, for example after 5 seconds with the weapon equipped if will hide or you can leave it unchecked and use an Input to draw/hide the weapon.

AMMO MANAGER

To create a new AmmoListData you can duplicate the original .asset and don't forget to assign to your AmmoManager in the inspector



The screenshot shows the 'AMMO MANAGER LIST DATA' inspector window. It has a 'Script' field set to 'vAmmoListData'. Below it, the 'Item List Datas' section is expanded, showing a 'Size' of 1 and 'Element 0' set to 'vShooterMelee_ItemListData (vIt'. At the bottom, there are two buttons: 'NEW AMMO' and 'LOAD AMMOS'. Below these buttons, there are six ammo types, each with an 'ID' and a 'Count' field, and a small 'x' button to the right of the count field.

Ammo Type	ID	Count
Rifle Ammo	13	25
Handgun Ammo	14	15
Shotgun Shells	17	8
RPG Ammo	18	10
Sniper Ammo	19	15
Arrows	21	15

Here you can create new AmmoID for new weapons, just select your Weapon Prefab and assign the new AmmoID



The screenshot shows the 'SHOOTER WEAPON' inspector window. It has a 'Script' field set to 'vShooterWeapon'. Below it, there are three tabs: 'Weapon Settings', 'Ammo', and 'Layer & Tag'. The 'Weapon Settings' tab is selected, showing a grid of settings. The 'Ammo' tab is also visible, showing a grid of settings. The 'Layer & Tag' tab is also visible, showing a grid of settings. Below the tabs, there are several settings: 'Is Infinity Ammo' (checkbox), 'Ammo' (25), 'Reload Time' (1), 'Reload One By One' (checkbox), 'Clip Size' (25), 'Auto Reload' (checkbox), and 'Ammo ID' (13).

Setting	Value
Is Infinity Ammo	<input type="checkbox"/>
Ammo	25
Reload Time	1
Reload One By One	<input type="checkbox"/>
Clip Size	25
Auto Reload	<input type="checkbox"/>
Ammo ID	13

The 'Ammo' is how much your weapon already starts with loaded, the ClipSize is how much the clip of this weapon can storage, and finally the AmmoID is the ID you created in the AmmoListData.