



Third Person Controller - Melee Combat Template

(v2.5.1 - 01/05/2020)

Thank you for supporting this asset, we developed this template because a lot of developers have good ideas for a Third Person Game, but building 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 set up a 3D Model in just a few seconds, without the need of knowing advanced scripting or wasting time dragging and dropping game objects to the inspector, instead you can just focus on making your game.

--- Invector Team ---

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

Summary

FIRST RUN	3
CREATING A MELEE CONTROLLER	5
MELEE MANAGER	7
CREATING A MELEE WEAPON	10
HOW TO APPLY DAMAGE TO A TARGET	12
HOW TO APPLY DAMAGE TO THE PLAYER	14
ITEM MANAGER	17
INVENTORY	22
COLLECTABLE STANDALONE (NO INVENTORY)	25
TOPDOWN / 2.5D / POINT & CLICK CONTROLLER / MOBILE	27
CREATING A ENEMY AI	29
LOCK-ON TARGET	31
WAYPOINT SYSTEM	32
CREATING A COMPANION AI	33
BODY SNAPPING ATTACHMENTS	34
WEAPON HOLDER MANAGER	37
DRAW/HIDE WEAPONS	39

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.**

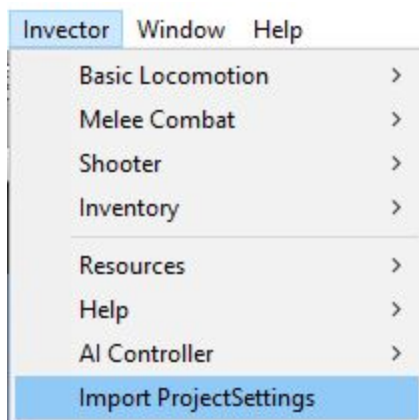


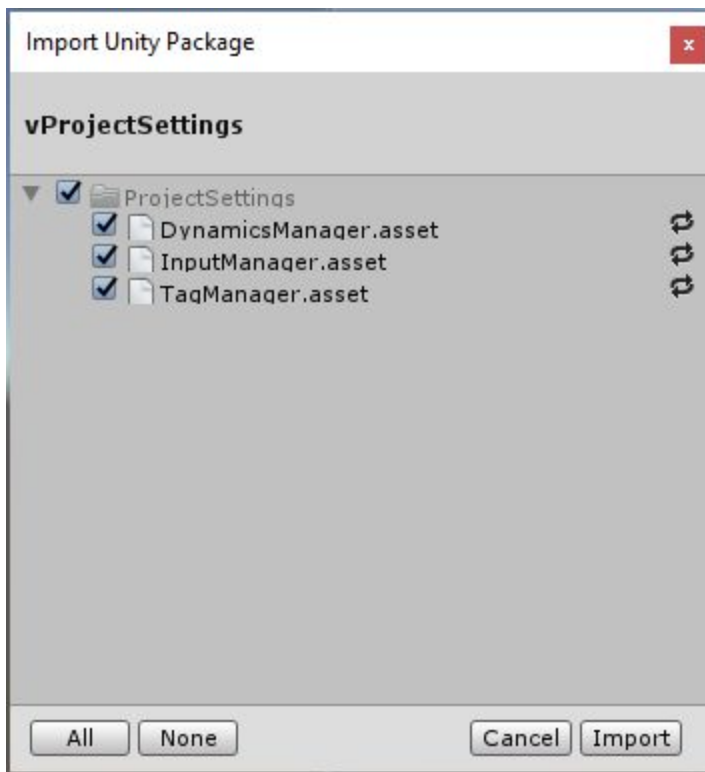
- *Importing on an existent project*

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.**



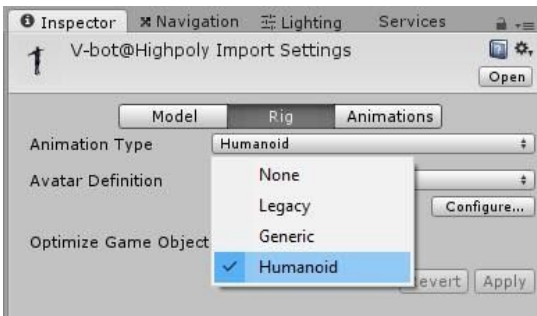


Now that you have imported the necessary files, you can explore the several demo scenes and figure it out what kind of Third Person Game you want to create.

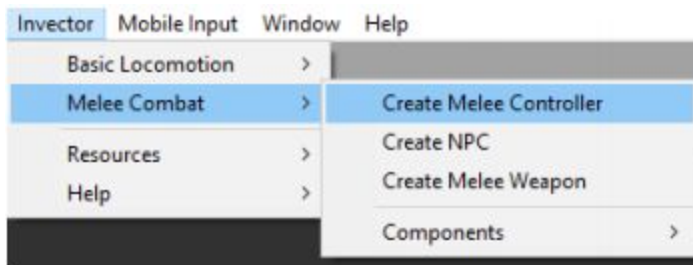
***Updates also need to be imported into a Clean Project, so MAKE SURE TO BACKUP your previous project and transfer the necessary files to your new project. ***

CREATING A MELEE CONTROLLER

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



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

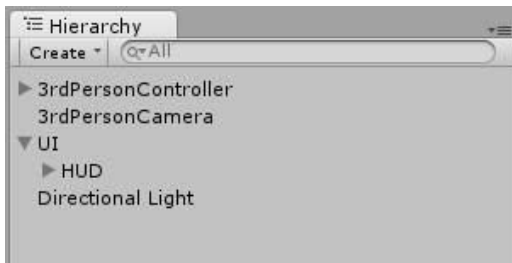


Make sure your Character is **Fully Rigged** and set up the FBX as a **Humanoid**, then assign the FBX to field “Humanoid” and click on the button “Create”.



Done.

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



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 😊

MELEE MANAGER

V2.0 - You can add a **Melee Manager** Component by opening the Invector tab > Melee Combat > Component

Open Default Info: here you can setup the default values for Hand to Hand Combat

Open Events: here you can add generic events like trigger something when you make damage

Add Extra Body Member: If you need an extra hitbox for example a Head Hitbox for a zombie, you can add

Who you can Hit > Important this is the tag that will receive Damage, so if you are using this component on the Player, assign the Tags of the gameObjects that you want to apply damage (the receiver need to have the method TakeDamage).

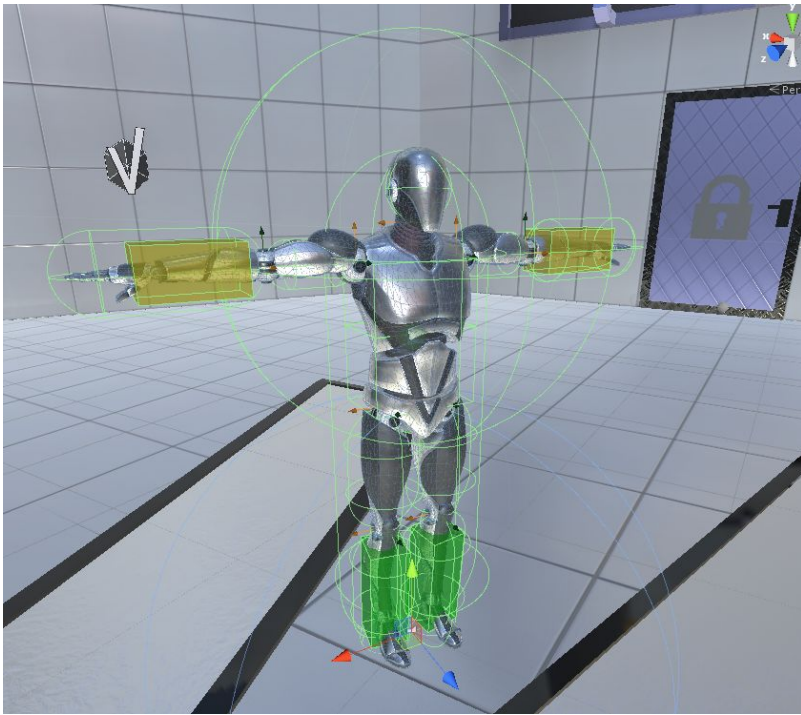
Use Recoil > Check if you want the character to trigger a recoil animation when hit a wall

Recoil Range > max angle to allow trigger the recoil animation

Hit Recoil Layer > the layer that will affect the recoil (usually it's the Default layer)



When you assign the **MeleeManager** component into your character, it will automatically create default hitboxes for both hands and legs, you can add an extra hitbox if you need.

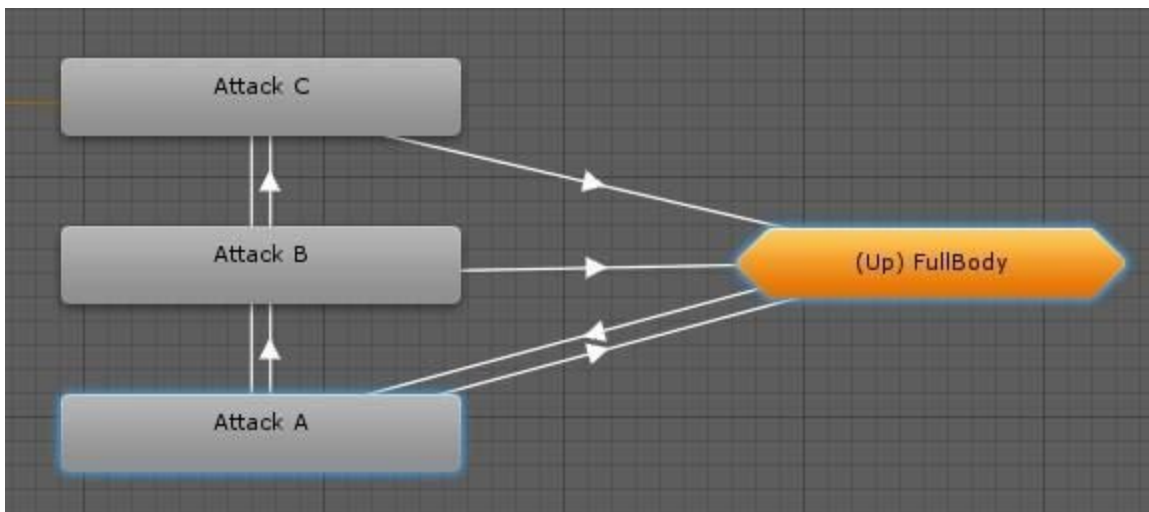


The animations for the hand to hand combat can be set up in the **Unarmed** state machine, trigger by the **ATK_ID 0** and the defense **DEF_ID 1** on the UpperBody Layer, **Default Defense**.

The **Basic Attack** State Machine is just an example, you can have as many State Machines you need, just remember to set up the **ID** to the corresponding weapon.

You can use **UpperBody** to attack as well, this way you can move the character and attack at the same time.

You can set up as many combos as you want, just put the attack animation and apply a transition.



Every Attack State need to have a **vMeleeAttackBehaviour** script attached.

StartDamage > Time of the animation that you will apply damage

End Damage > Time of the animation that will stop trying to apply damage

~~Allow Movement At: free your character rotation during the attack animation~~

*Removed on update 2.4.2 use the **AnimatorTagAdvanced** *LockMovement* and *LockRotation* instead

Recoil ID > Trigger a Recoil animation if you hit a wall or an object

Reaction ID > Trigger a Reaction animation when you take damage

Melee Attack Type > Select Unarmed or Melee Weapon

Reset Trigger > Check this bool for the last attack, to reset the combo

Attack Name > You can write an Attack Name to trigger different HitDamage Particles on the Target, Ex: If your weapon has electric damage, you can match the Attack Name with the HitDamage Particle and instantiate a different particle for this specific weapon.

Ignore Defense: it will apply damage even if the target is blocking

Active Ragdoll: activate the target ragdoll

✓

MELEE ATTACK CONTROL

Make sure that the **Exit Time** of this state to the next one in your Combo is **lower** then the Exit Time to the Exit state, otherwise it will always exit first and never play the next animation.

! For Example if your Exit Time to the Exit State is 0.7 then your transition to the next state must be 0.6 or lower.

The same applies to the **End Damage**

Script

vMeleeAttackControl

Start Damage

0.25

End Damage

0.6

Damage Multiplier

0

Recoil ID

0

Reaction ID

0

Melee Attack Type

Unarmed

Damage Type

Ignore Defense

☐

Active Ragdoll

☐

Reset Attack Trigger

☐

Debug

☐

n°

BodyPart

00

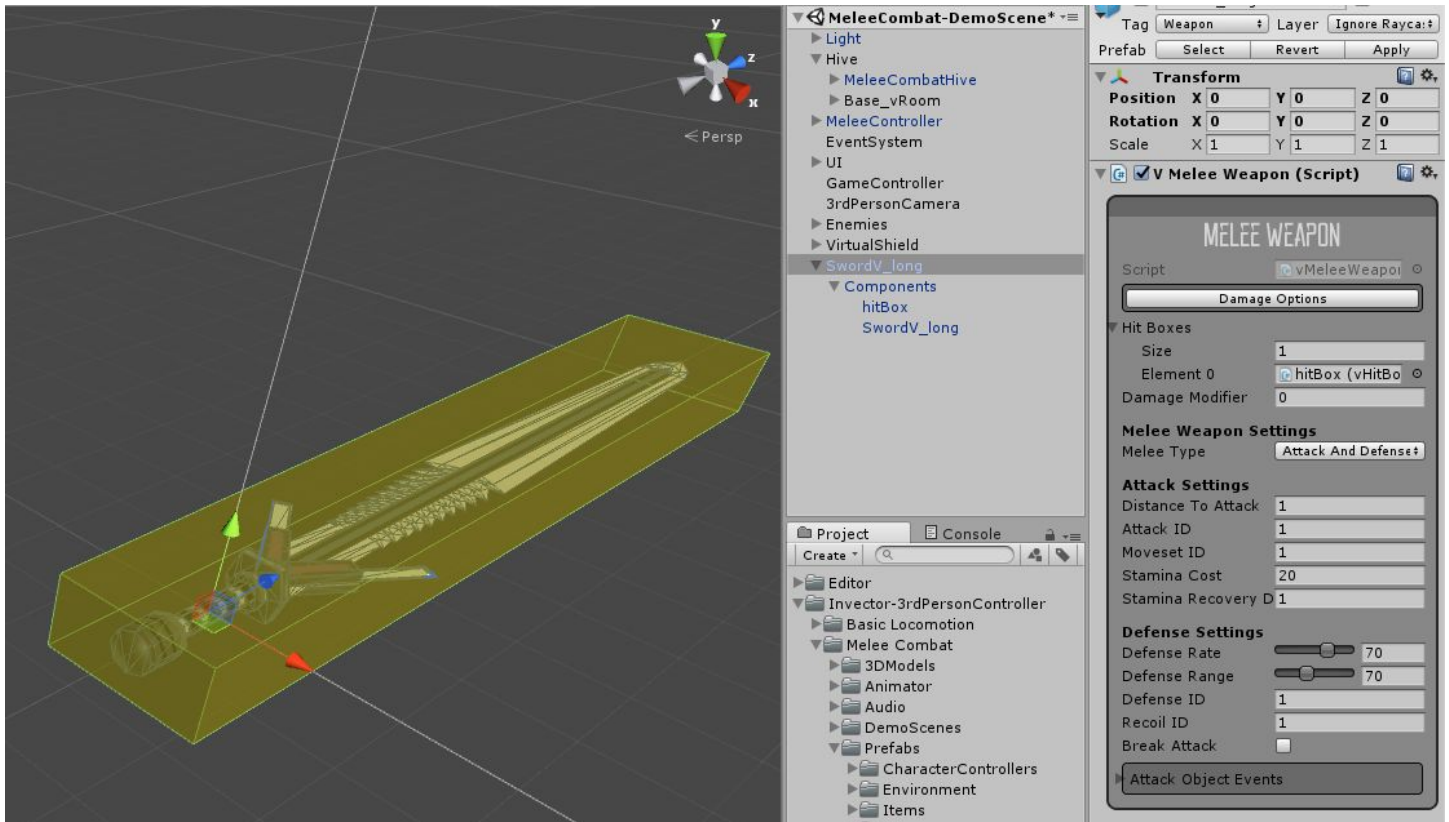
RightLowerArm

Add New Body Part

Ps* Don't forget to assign the limb member of your BodyPart to match the animation, this will trigger the correct HitBox, you can add new BodyParts if your attack use more than one member.

CREATING A MELEE WEAPON

To create a new weapon, you just need to select your weapon Mesh and go to the menu **Inspector > Melee Combat > Create Melee Weapon**.



After that your mesh will be transferred inside the Components gameobject, and the parent will have a **vMeleeWeapon** attached where you can set up your weapons settings.

A single hitbox will be created and if you need more you can just duplicate the first and assign into the Hitbox List into the **MeleeWeapon** component.

IMPORTANT - don't forget to set your **Weapon Layer to Ignore Raycast** and the **Tag to Weapon**, if you put a weapon into an Enemy or Player and change the Layer and children's, you need to set the weapon layer to Triggers again

After creating your weapon, you can just drag and drop inside a hand Bone of your character and hit Play, the **MeleeManager** will auto assign into the Weapon slot.

To change weapons ingame you will need a **ItemManager** assign into your Character.

Attack Settings:

[Damage Options]

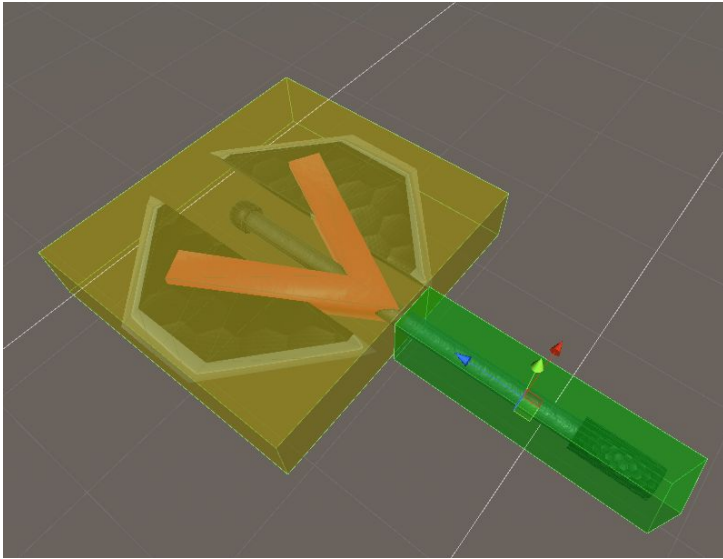
Value: Total damage of your weapon

Stamina Block Cost: How much stamina the target will lose when receive this attack while blocking

Stamina Recovery Delay: How much time will wait to start recover the stamina

Ignore Defense > Check if this weapon can pass through shield

Active Ragdoll > Check to make this weapon activate the Ragdoll of the target



Example of a weapon with 2 hitbox

HitBoxes List: Assign your hitboxes here

Damage Modifier: Extra Damage

Melee Type: Just Attack, Just Defense or Both;

(SOON) Use Two Hand > Check this if your weapon uses two hands (the left weapon will drop)

Distance to Attack > Used for AI only, to know the distance to attack if this weapon

ATK_ID > correspond to the Attack Animation State that will trigger

MoveSet_ID > it's the correct move set that the character will move when using this weapon

Stamina Cost > how much stamina the attack will cost

Stamina Recovery Delay > how much time will take to the stamina start recovery

Defense Settings:

DEF ID > ID of your defense animation

Recoil_ID > Trigger a recoil animation

Defense Rate > how much damage you can defend from an attack

Defense Range > When you select the shield, a Gizmos will appear to help you see how much of Defense Range you need.

Break Attack > Trigger a Recoil Animation on the Attacker

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 visible: "Open Default Info", "Open Events", and "Add Extra Body Member".

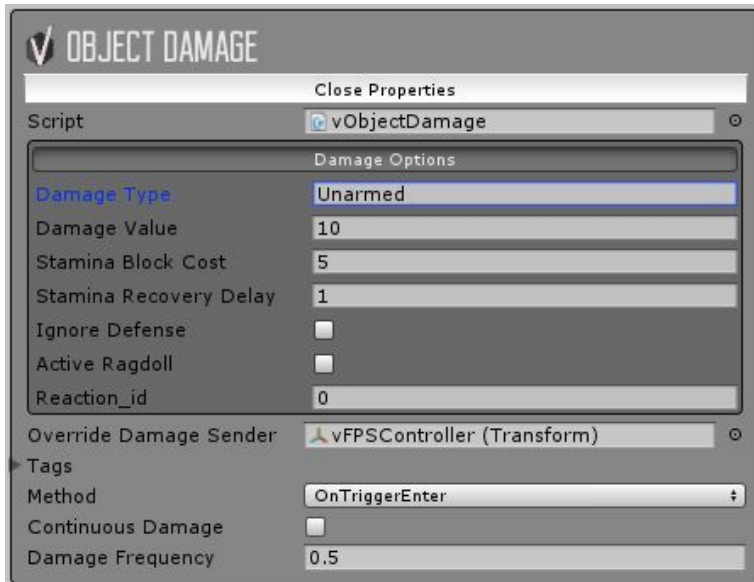
The "Body Members" section contains four buttons: "LeftLowerArm", "RightLowerArm", "LeftLowerLeg", and "RightLowerLeg".

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 set to "None (V Melee Weapon)".

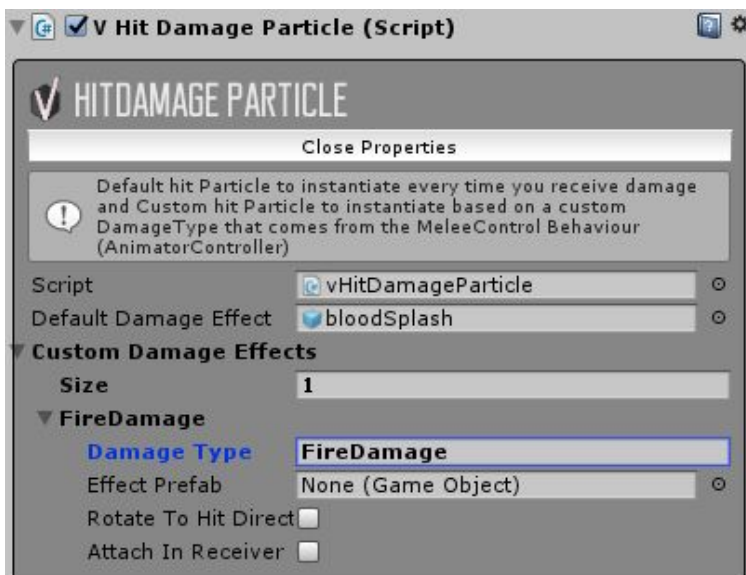
HOW TO APPLY DAMAGE TO THE PLAYER

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.

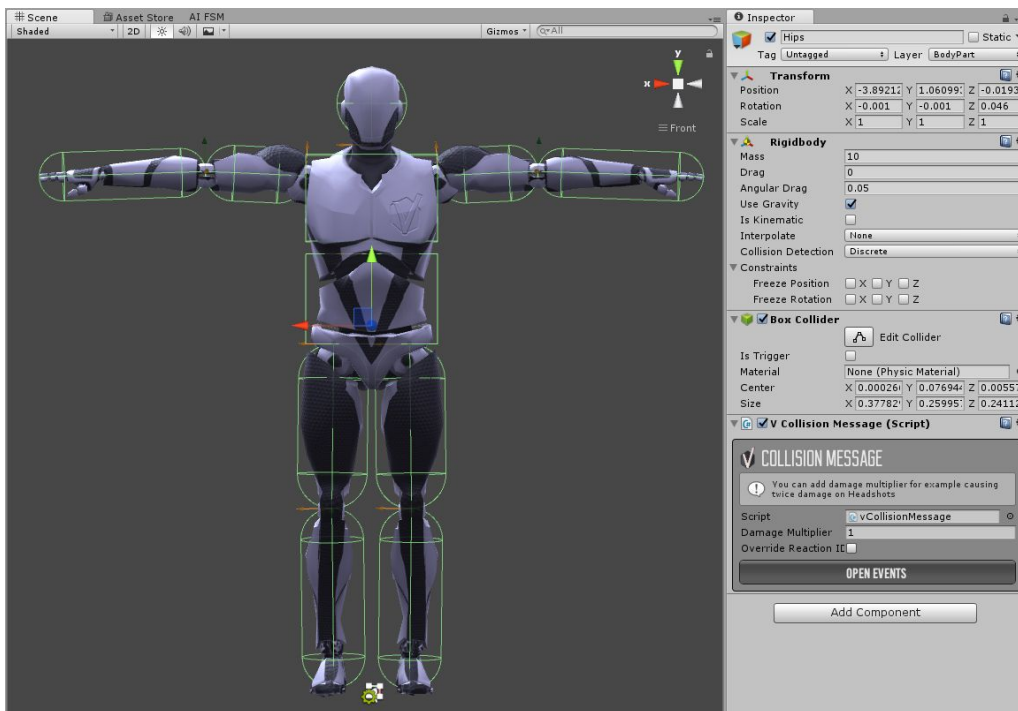


*This component is attached to the Controller or any object that contains the HealthController

- **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”.



Bodyparts use the Damage Receiver to receive damage.

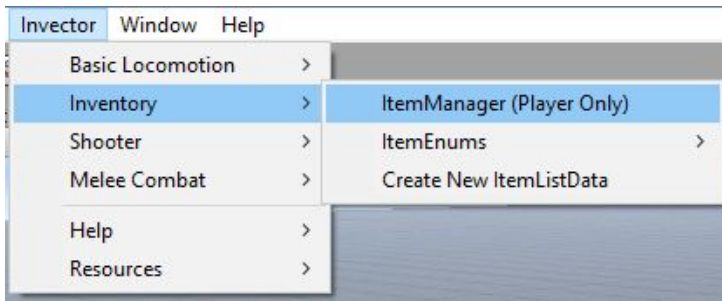
A DamageReceiver is attached to each ragdoll collider, this will allow to Player or AI to apply damage to each bodypart instead of the CapsuleCollider.

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

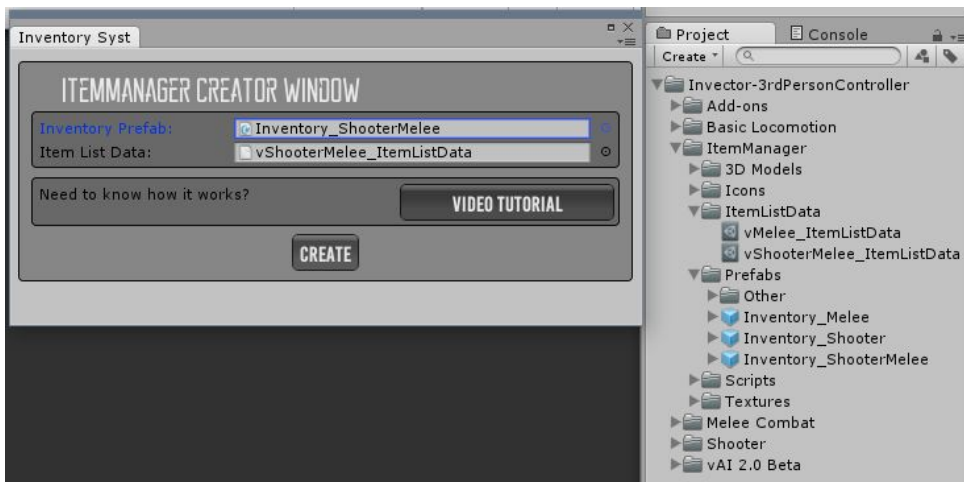
For example, if you want to cause 2x damage and trigger a specific reaction animation when shooting in the Head, simple change the values in this component.

ITEM MANAGER

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



- Select the Inventory Prefab from the **Project > ItemManager > Prefabs** then drag and drop inside your **Third Person Controller**, when you hit play it will be automatically assigned to the ItemManager.
- and a **ItemListData > vShooterMelee_ItemListData**



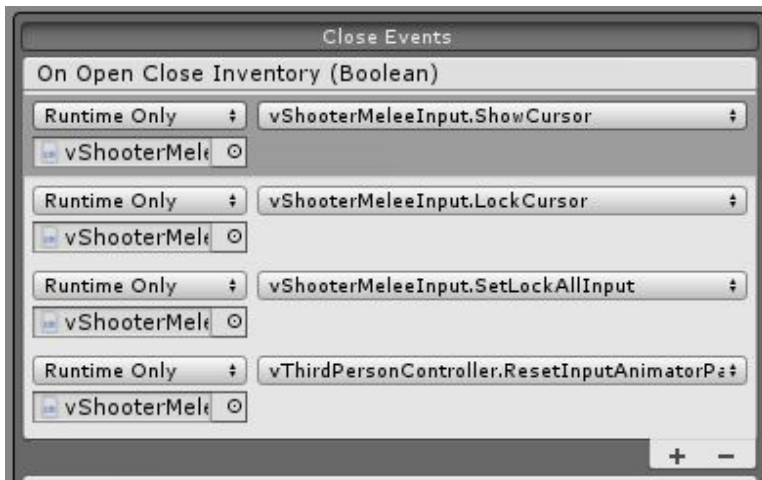
- You can also add the ItemManager manually using the Add Component button via inspector, but don't forget to assign an ItemListData.



In the tab **Events** you can call methods like lock the input of the character while the Inventory is Open, just assign the Character (from the scene hierarchy) and call the method `LockInput` from the `vMeleeCombatInput` or `vShooterMeleeInput`.

Here are the most used events you can use to:

- Hide/Show the mouse cursor
- Lock/Unlock the cursor at the screen center
- `SetLockAllInput` to lock all the input from the controller (basic, melee and shooter)
- `ResetInputAnimatorParameters` to reset back to zero the animator parameters, usefull when using the `CharacterCameraPreview` inside the Inventory, this way if you're walking and open the inventory, there character will be at Idle in the camera preview.

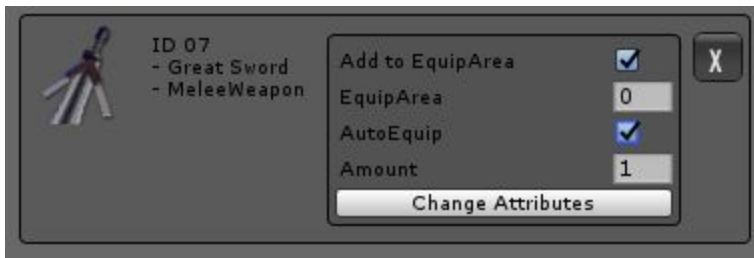


Click at the Add Item button to open the Filter, so you can search or filter exactly what ItemType you need:
You can add multiple ItemTypes to filter multiple items.



Once you add a Item in the list, you can use the option **Add To EquipArea** to automatically assign this item to a specific EquipArea (it will only be assigned if your itemSlots are currently empty, if one is already assigned, it will be equipped in the next one in the list)

The **AutoEquip** will auto equip this item to the EquipArea and also change the EquipmentDisplayWindow to where the Item is equipped.



Click in the **Open Item List** button, to manage, search 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.

The image shows a detailed configuration window for an item named 'Shotgun'. At the top, it displays 'ID 12' and a small icon of a shotgun. Below this, the name 'Shotgun' is in a text field with an 'EditName' button. The 'Description' field contains 'Tactical Shotgun'. The 'Item Type' is set to 'Shooter' in a dropdown menu, and the 'Stackable' checkbox is unchecked. The 'Icon' field shows 'shotgunIcon' with a small shotgun icon next to it. Below the icon field are two object selection fields: 'Original Object' set to 'vShotgun' and 'Drop Object' set to 'vCollectableShotgun'. The 'Attributes' section has an 'Add Attribute' button and two attributes: 'Damage' with a value of 50 and 'AmmoCount' with a value of 8. The 'Custom Settings' section includes a 'Script' dropdown set to 'vItem', a checked 'Two Hand Weapon' checkbox, and 'Equipable Settings' with 'Equip ID' set to 3, 'Custom Equip Point' set to 'defaultPoint', and 'Equip Delay Time' set to 0.5.

Attributes	
Damage	50
AmmoCount	8

Custom Settings	
Script	vItem
Two Hand Weapon	<input checked="" type="checkbox"/>
Equipable Settings	
Equip ID	3
Custom Equip Point	defaultPoint
Equip Delay Time	0.5

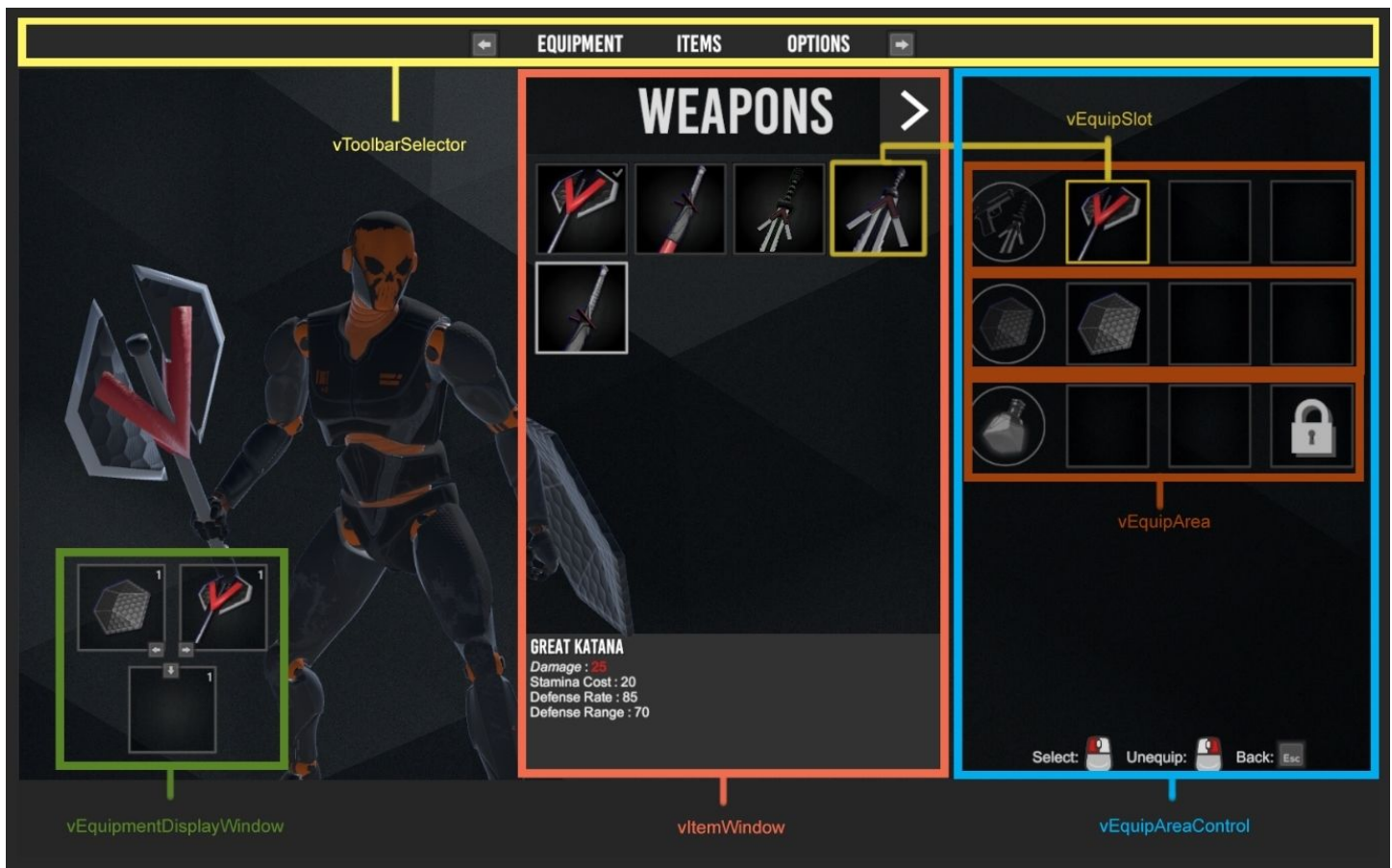
Don't forget to add the attribute Damage & AmmoCount of your weapon, this will allow you to drop and collect your weapon with the same amount of weapon, making it into a unique weapon.

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](#).

INVENTORY

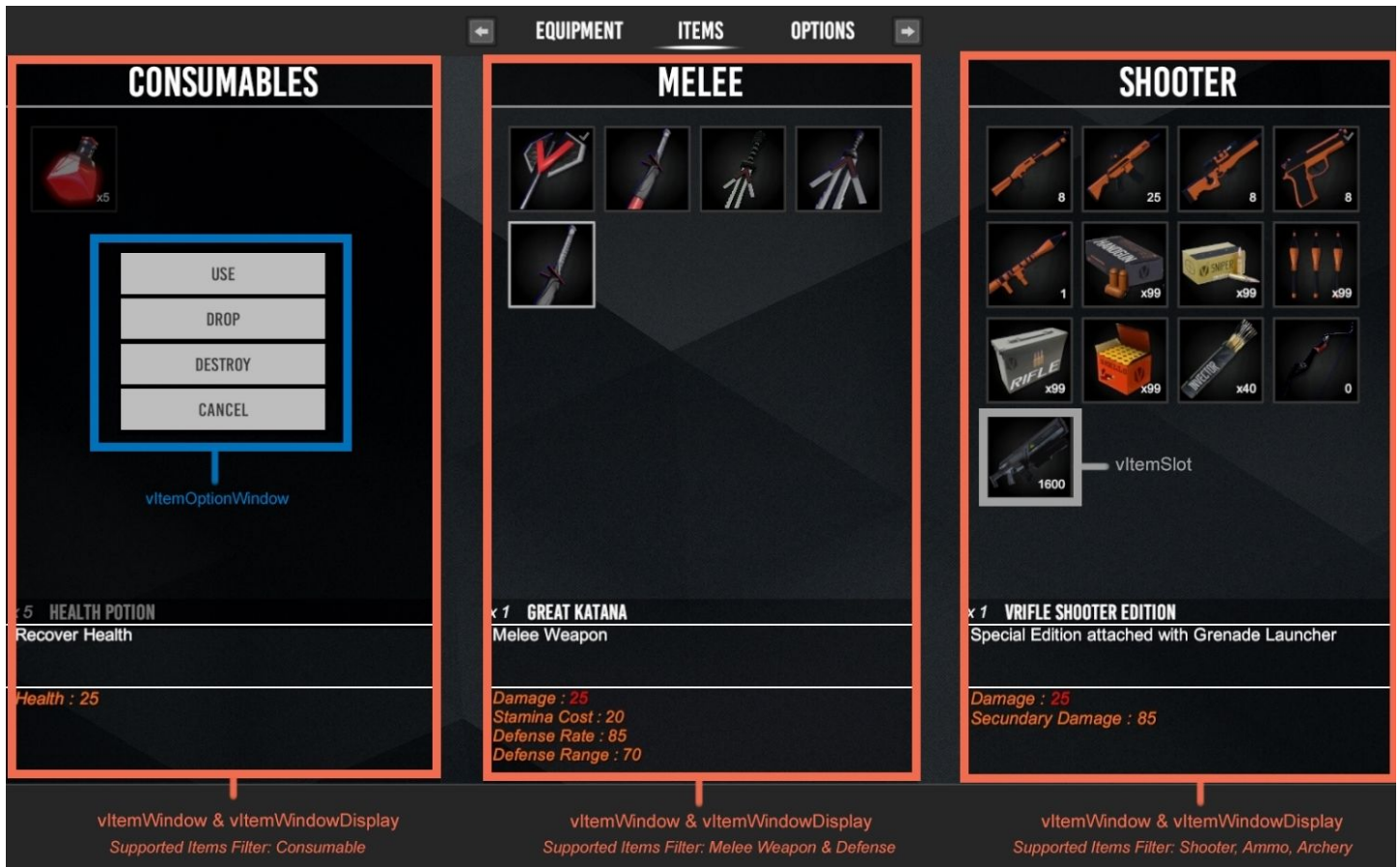
We have basically 3 Inventory Examples in the project, you can find them in the folder ItemManager/Prefabs. Just drag and drop inside your character to test it, make sure to also use the corresponding ItemListData at your ItemManager according to what Inventory you're using.

For ex: ShooterOnly requires the vShooterMelee_ItemListData, MeleeOnly requires the vMelee_ItemListData



- **ToolbarSelector:** It's basically a menu, you can set the input to switch tabs at your Inventory
- **Equipment Display Window:** It shows the current slot of a EquipArea, you can have multiple slots and multiple EquipAreas at the same time and rotate the slots via input
- **EquipArea:** A row of EquipSlots
- **Equip Area Control:** Responsible to inform the EquipmentDisplay
- **EquipSlots:** Display a specific Item, you can use the ItemType to filter what item can be displayed on this slot
- **ItemWindow:** A window that display Items, you can filter what ItemType will be displayed using the filter 'Supported Items'

This is an example of a window that displays several different Items based on their Types.



- ItemWindowDisplay: Manage what happens with the ItemWindow when using, drop, destroy and cancel
- ItemOptionWindow: Buttons that buttons UI to use, equip, drop, destroy and cancel

You can see a **MasterWindow** component on each main window of the Inventory, you can use the MasterWindow to manage your windows for example if you open a window and open another, you can press the back button to close the current window and get back to the previous, it's basically a window manager.

You can also use the Events OnEnable/OnDisable to for example, turn on / off specific objects, call animations, sounds, etc...



CHECK IF ITEM IS EQUIPPED

You can use the component *vCheckItemsEquipped* to check if a specific Item or ItemType is currently equipped.

The screenshot shows the configuration window for the 'CHECK IF ITEM IS EQUIPPED' component. The window has a title bar with a checkmark icon and the text 'CHECK IF ITEM IS EQUIPPED'. Below the title bar, there are several sections for configuration:

- Script:** A dropdown menu showing 'vCheckItemIsEquipped'.
- Item Manager:** A dropdown menu showing 'None (V Item Manager)'.
- Get In Parent:** A checkbox that is checked.
- Item ID Events:** A section with a 'Size' field set to '1'. Below it is a 'DoSomething' section with a 'Name' field set to 'DoSomething'. Under 'DoSomething', there is an 'Items ID' section with a 'Size' field set to '3'. Below 'Size', there are three fields: 'Element 0' set to '8', 'Element 1' set to '10', and 'Element 2' set to '33'.
- On Is Item Equipped ():** A section with a 'List is Empty' checkbox.
- On Is Item Unequipped ():** A section with a 'List is Empty' checkbox.
- Item Type Events:** A section that is currently collapsed.

You can attach this component directly to the character or inside the character (check the option “GetInParent” if it’s inside the Player).

Name - just the name of your event, so it’s easier to identify what it does

ItemsID - You can add one or more items to trigger something specific, for example, when equipped with Potions, you can enable the UI Button to use those items, when unequipped simply disable the object.

You could either filter the ItemID’s of your Potions or set the ItemType Consumable in this example.

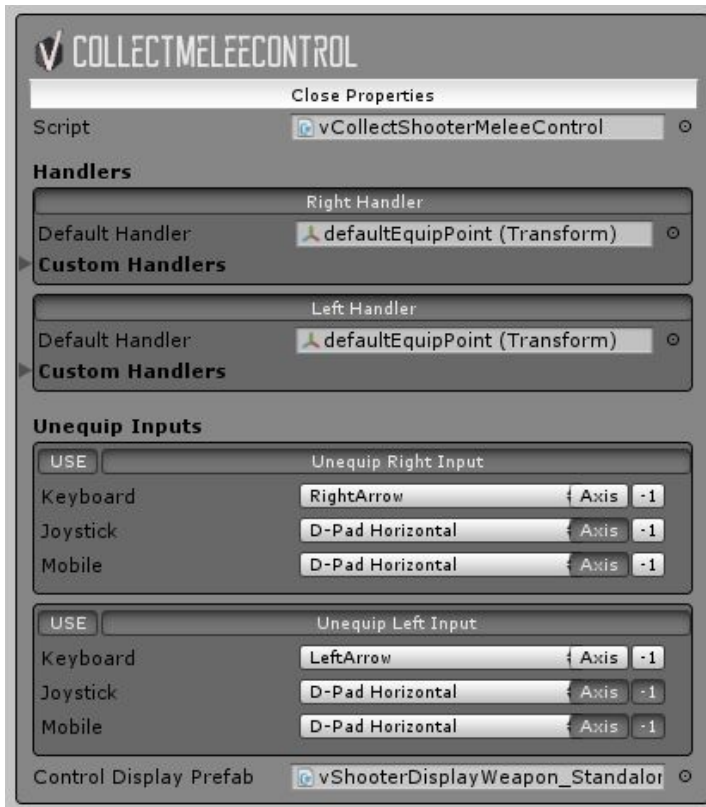
The screenshot shows the 'Item Types' configuration section. It has a 'Size' field set to '1'. Below it, there is an 'Element 0' field set to 'Consumable'.

COLLECTABLE STANDALONE (NO INVENTORY)

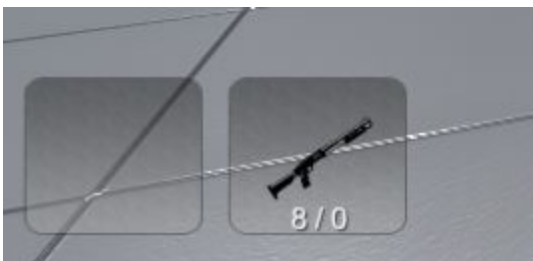
If you don't want to use the ItemManager to manage your items, we have another solution for 'on demand' collectibles, notice that you can only equip 1 item, once you try to equip another the current item will drop.

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 called "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 examples of collectables we have for both melee and shooter weapons.

V

COLLECTABLESTANDALONE

Script

vCollectableStandalone

Disable Collision

Disable Gravity

Reset Player Settings

Play Animation

End Exit Time Animation

0.8

Avatar Target

Root

Match Target Mask

X 0Y 0Z 0

Match Target

None (Transform)

Start Match Target

0

End Match Target

0

Active From Forward

Use Trigger Rotation

Destroy After

Destroy Delay

0

On Do Action Delay

0

Target Equip Point

defaultEquipPoint

Weapon

vShotgun_NoInventory

Weapon Icon

shotgunIcon

Weapon Text

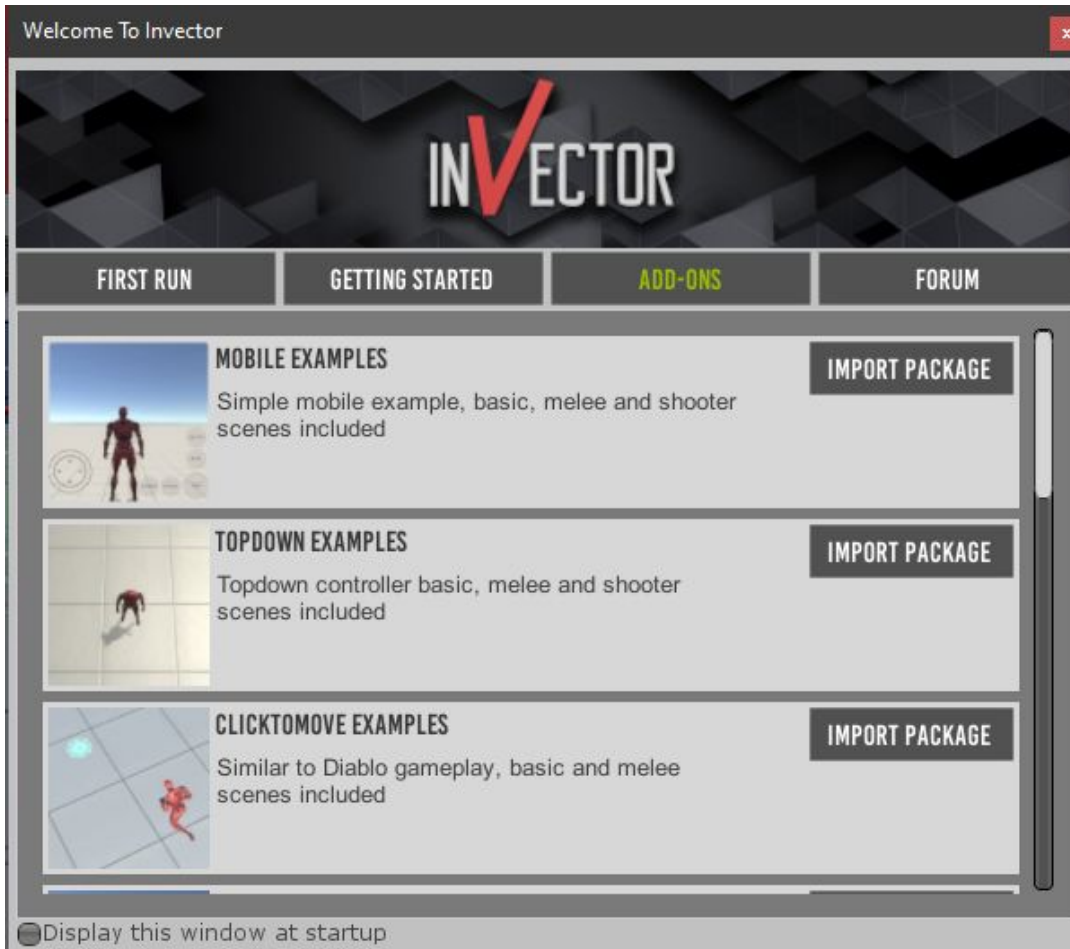
Shotgun

OPEN EVENTS

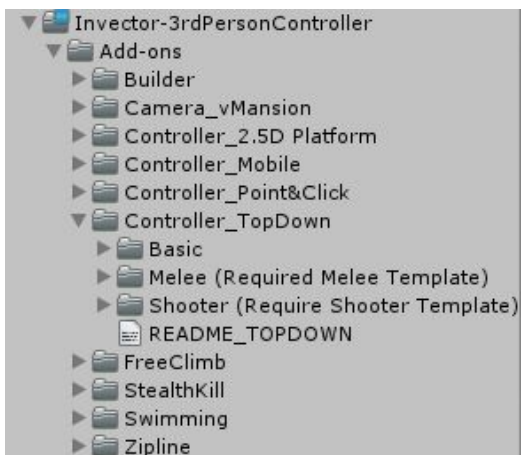
It's important to assign the correct gameobjects into the Events, we turn off the collision and gravity of the weapons when equipped and turn on when you drop them.

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*



If you don't own the Melee or Shooter templates, you don't need to import their folders.



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.

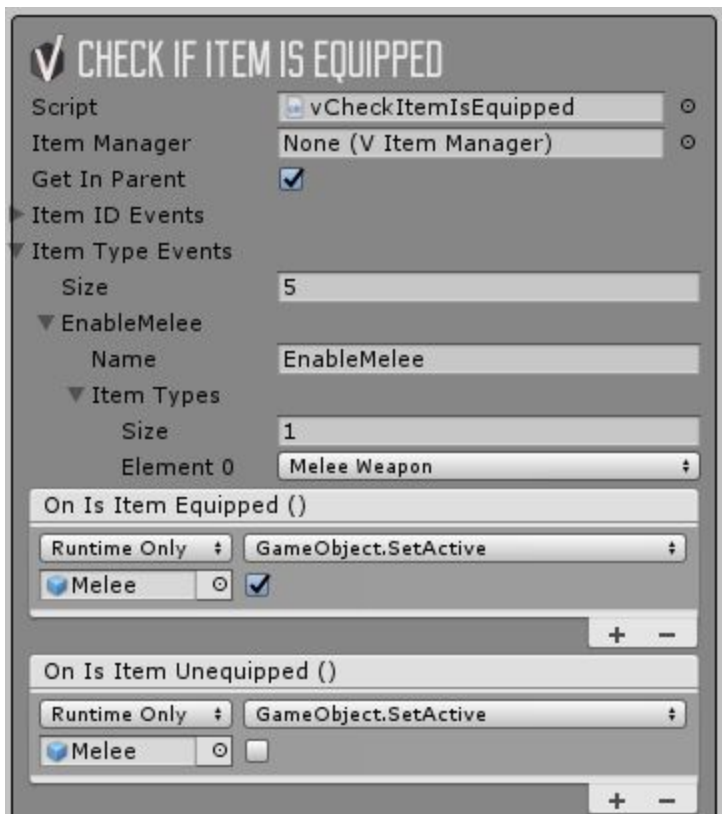
MOBILE CONTROLS

After importing the mobile package, you can create a regular `ThirdPersonController` using the Character Creator Window and after setting up the controller and aligning the handlers, search for the Mobile UI Controls prefabs:

MobileUI_ShooterMelee_Inventory - if you want to use Inventory

MobileUI_ShooterMelee_NoInventory - if you don't want to use Inventory, check the demo scenes to see how it works and choose one style for your game.

If you're using the Inventory you can use the component **vCheckItemsIsEquipped** to turn on/off the mobile buttons:



If you're not using the Inventory, you can use the **vCollectShooterMeleeControl** Events to manage the buttons, it identifies if you're using a shooter weapon or a melee weapon only.



CREATING A ENEMY AI

Inspector > Melee Combat > Create NPC and change the **Character Type** to **Enemy AI**

After hitting the Create button, our scripts will handle all the most time consuming stuff and make the AI almost done to hit Play, you just **need to BAKE a NavMesh on the Scene.**

Locomotion - It works pretty much the same as the Character Controller, you still need to set up the

Layers - just like the Player, you need to set up a Layer for the AI (Enemy) and a layer for the Ground (Default). If you equip a Weapon on the character, don't forget to assign the Layer Triggers for this weapon.

Combat - Here you will have a lot of options to make very different combat behavior, you can add a chance to block (if equipped with a defense weapon), chance to roll, chance to defend an attack, change the attack frequency, strafe around the target, etc...

Waypoints - You can add waypoints for the AI to follow in sequence or activate the option to Random.

We manage to get better results with the NavMesh using this set up, but of course this will depend on your scene, terrain, meshes, etc...

Object

Bake

Areas

Baked Agent Size

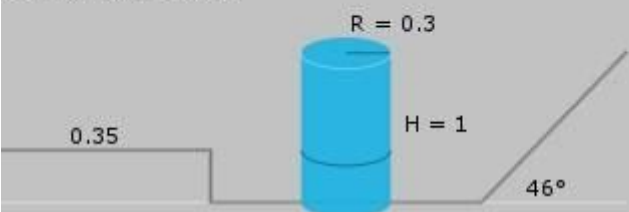


Diagram illustrating the baked agent size. The agent is represented by a cylinder with radius $R = 0.3$ and height $H = 1$. The agent is positioned on a slope with a maximum angle of 46° . A step height of 0.35 is shown on the left.

Agent Radius

0.3

Agent Height

1

Max Slope

46

Step Height

0.35

Generated Off Mesh Links

Drop Height

4.1

Jump Distance

3

▼ Advanced

Manual Voxel Size

☐

Voxel Size

0.1

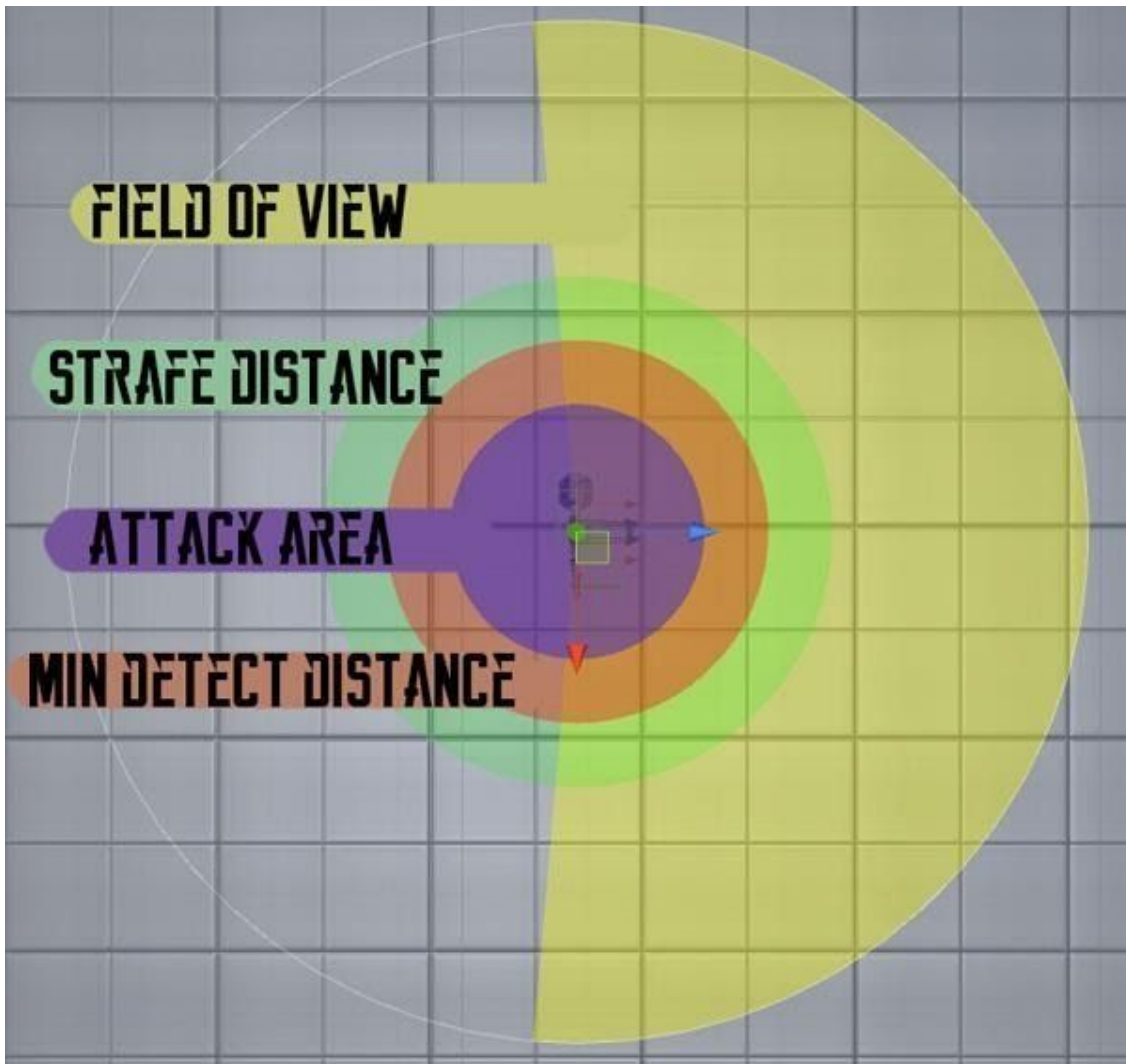
3.00 voxels per agent radius

Min Region Area

0.5

Height Mesh

☐



Field of View > total range to detect the Player

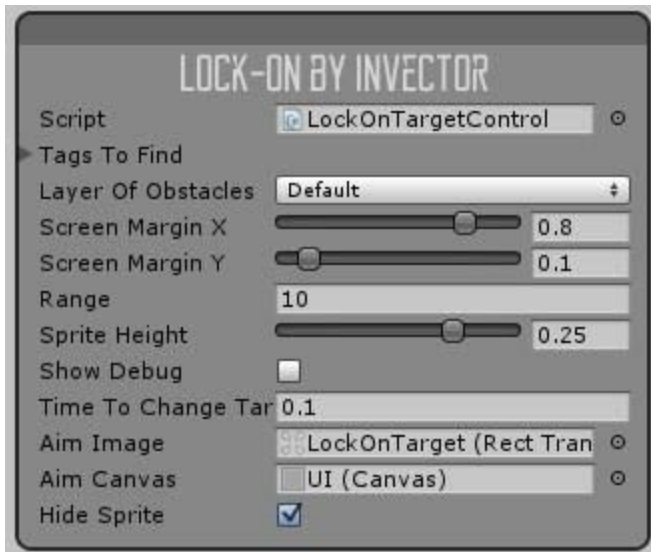
Strafe Distance > once in combat, the character will move Strafing

Attack Area > total range to attack

Min Detect Distance > Min distance to detect the player, even if the player is outside the Field of View range.

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.

WAYPOINT SYSTEM

You can create a Waypoint Area by opening the 3rd Person Controller > Component > New Waypoint Area. To create a new **Waypoint**, just hold *Shift + Left Click* on any surface with a collider, to reposition the same waypoint hold *Shift + Right Click*. The same goes to create Patrol Points, but you will hold Ctrl instead of Shift. You can assign this Waypoint Area to many AI as you want, and limit the area / limit of AI that will access.

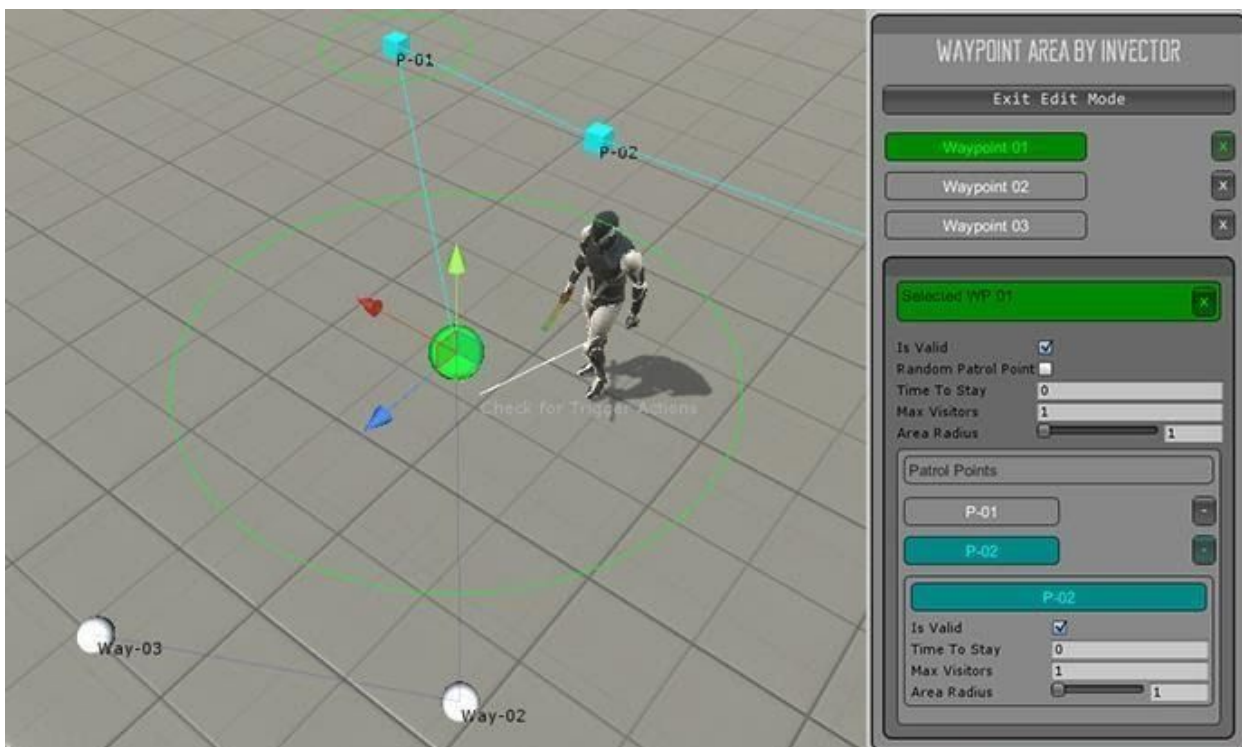
Patrol Points are points of interest that one waypoint has, for example if you have a corridor with 3 rooms, you can create 1 waypoint in the middle of the corridor and 3 patrol points with **Max Visitors of 1**, this means that if an AI is already on a room, the other AI will not come to the same room, he will go to the next one.

Time to Stay is how much time the AI will stand there.

isValid is a bool that you can turn on/off to disable a waypoints/patrol point in real time.

You can make the AI walks randomly at waypoints by selecting the option **Random Waypoints** on the AI Inspector. To make random patrol points, select the option **Random Patrol Point** on the Waypoint Inspector.

Waypoints are represented by Spheres and Patrol Points are represented by Cubes.



CREATING A COMPANION AI

Same process as creating a Player or EnemyAI, just select CompanionAI on the Character Type.

The Companion has the same variables as the EnemyAI, plus:



If you open the `v_AICompanion` script, you will see that we have a method call `CompanionInputs` and you can customize for your needs, this method contains the basic commands like Follow, Stay, Aggressive/Passive and MoveTo (you can send the AI to a specific spot by an `Vector3`)

Default Inputs:

- Stay
- Follow
- Aggressive/Passive
- Move to (`moveToTarget`)

*Notice that the transform target height can be no higher than 0.5f from the navmesh, otherwise he can't find a path to go.

BODY SNAPPING ATTACHMENTS

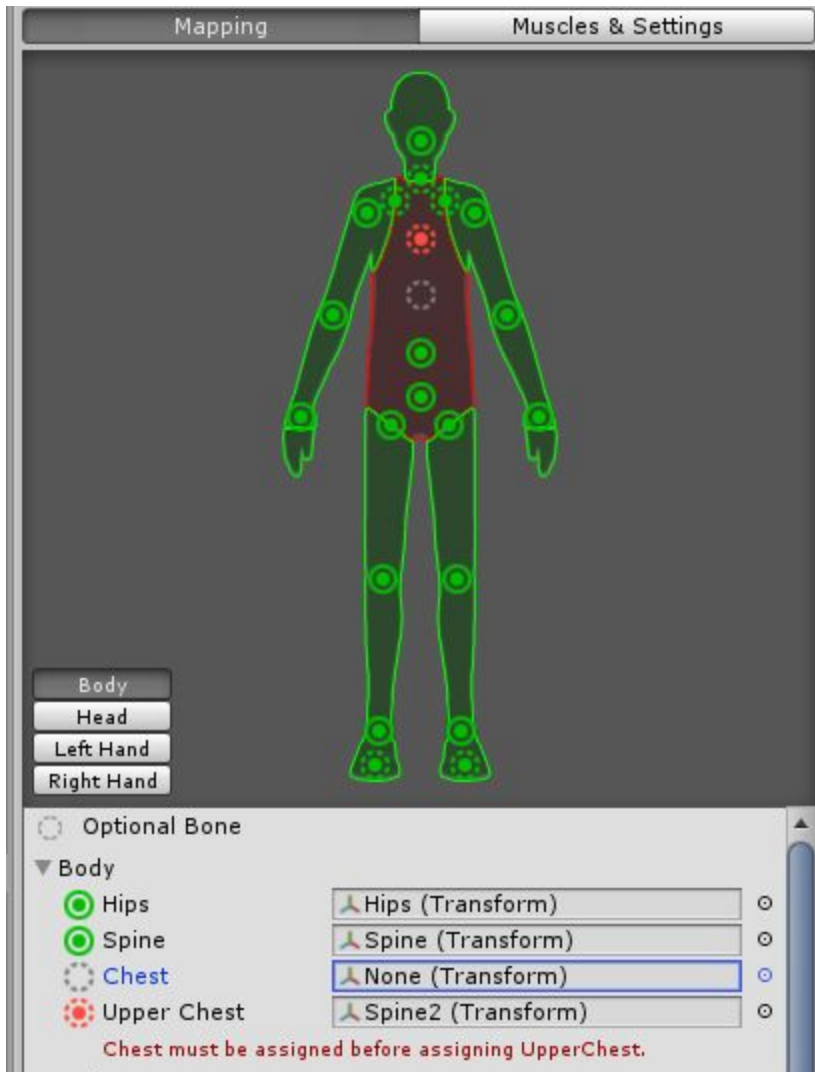
We created this feature to make it easier to transfer attachments from one controller to another.

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.

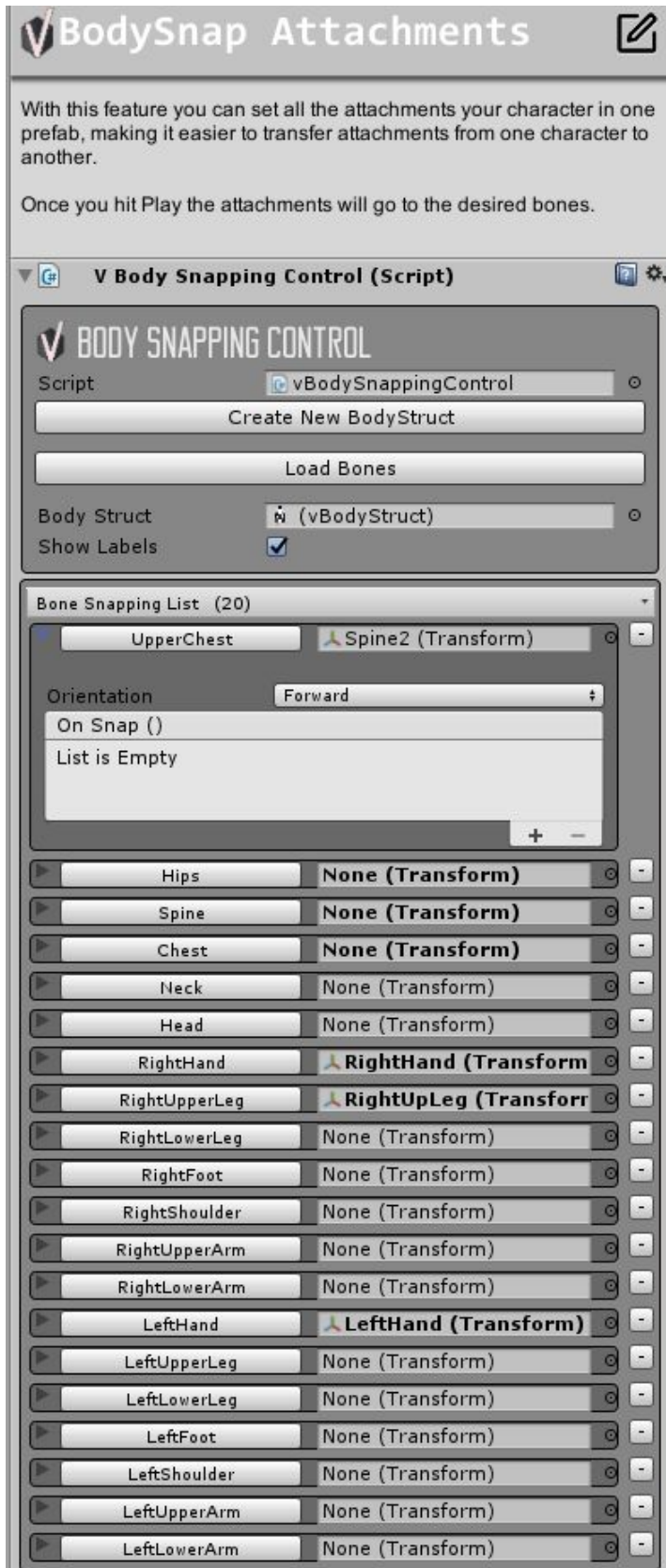
First, create an Empty GameObject inside your character, add the “**vBodySnappingControl**” and hit the “Create New BodyStruct”.



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.



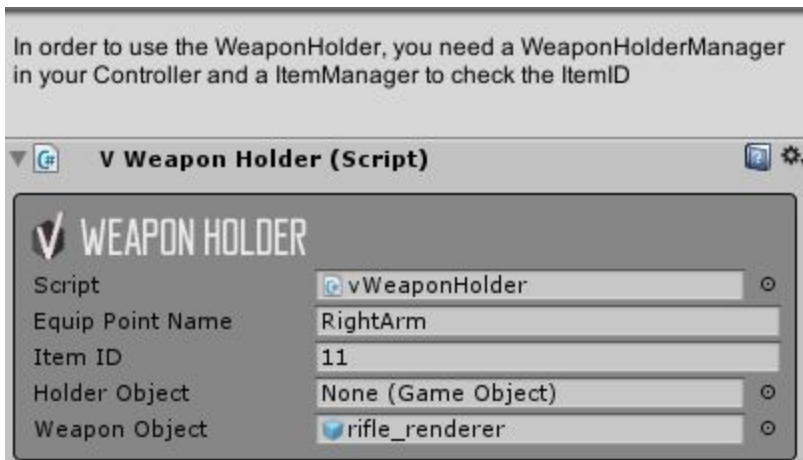
WEAPON HOLDER MANAGER

*This feature requires a ItemManager

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

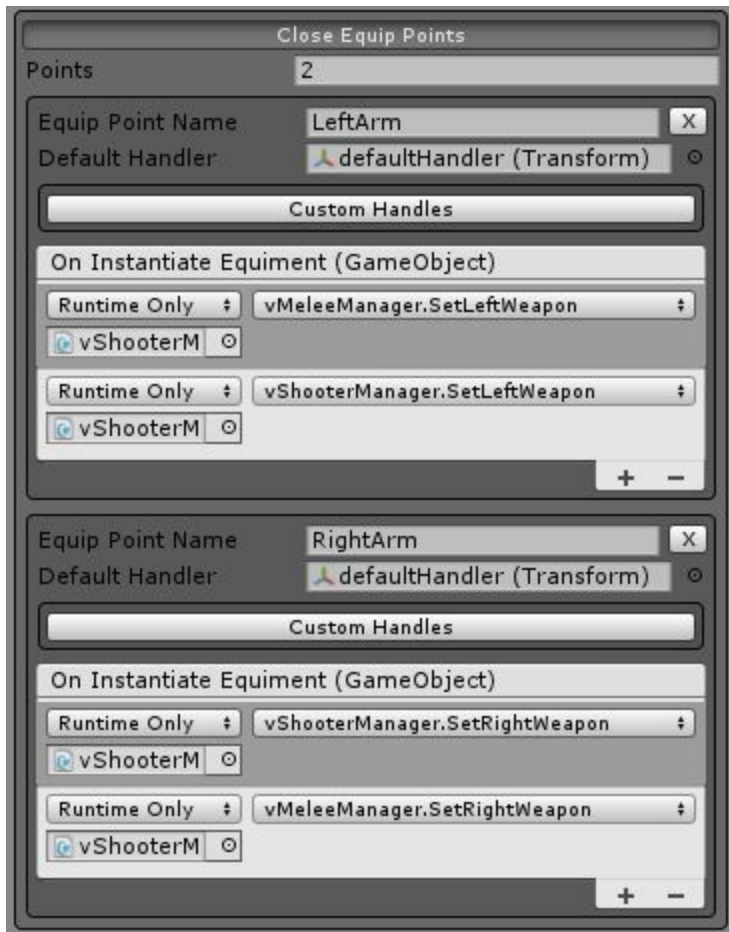


Now go inside your controller, create an empty gameObject and add the vWeaponHolder component.

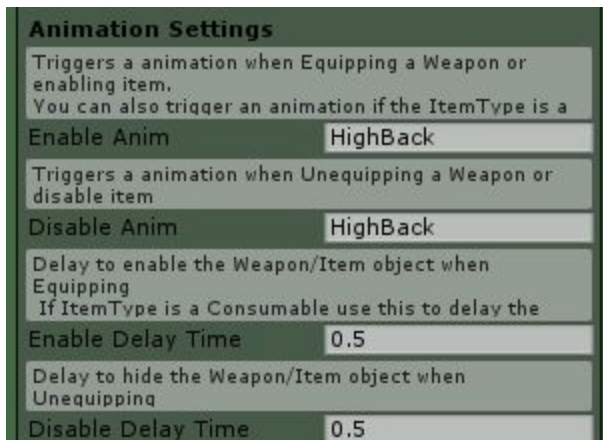


The **EquipPointName** can be found in the **ItemManager** > EquipPoints.

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.