Third Person Controller – Shooter Template

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 ---

FIRST RUN

IMPORTANT

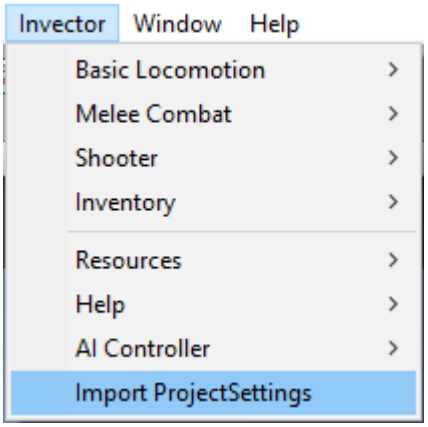
This is a **Complete Project**, and as every complete project it includes a custom **InputManager**, **Tags**, **Layers**, etc... **Make** sure that you import **on** a **Clean Project**.

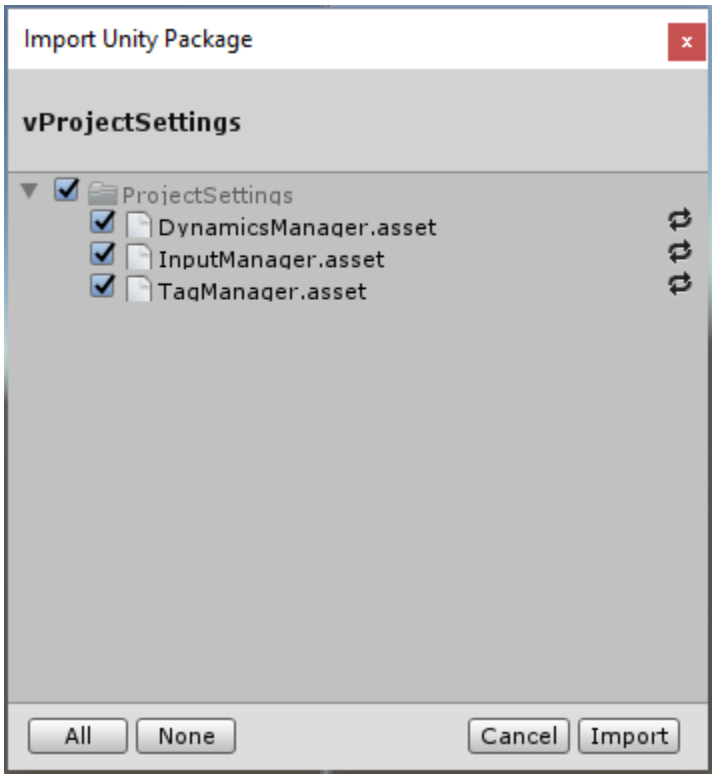
- *Importing on an existent project*

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

- ***InputManager**.asset* - We have a custom input mapped to support the Xbox360 controller, without it you will receive errors about missing input.
- ***TagManager**.asset* - Includes all the necessary Tags and Layers for the project to work correctly.
- ***DynamicsManager**.asset* - this will update the Collision Matrix of our Layers, for example we need the layer "Triggers" to not collide with the layer "Player".

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



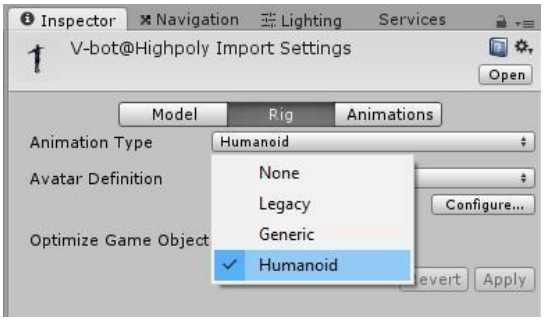


Now that you have imported the necessary files, you can explore the several demo scenes and figure out what kind of Third Person Game you want to create.

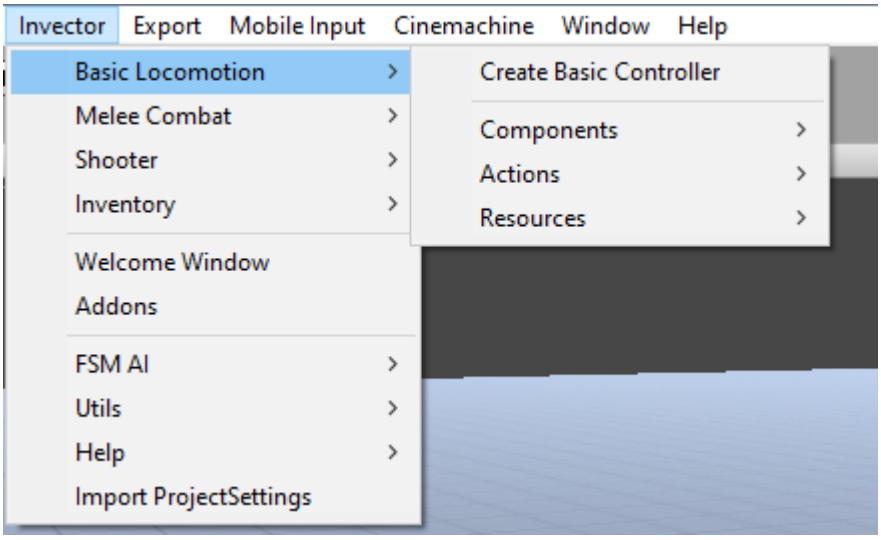
***Updates also need to be imported into a Clean Project, so MAKE SURE TO BACKUP your previous project and transfer the necessary files to your new project. ***

CREATING A CHARACTER CONTROLLER

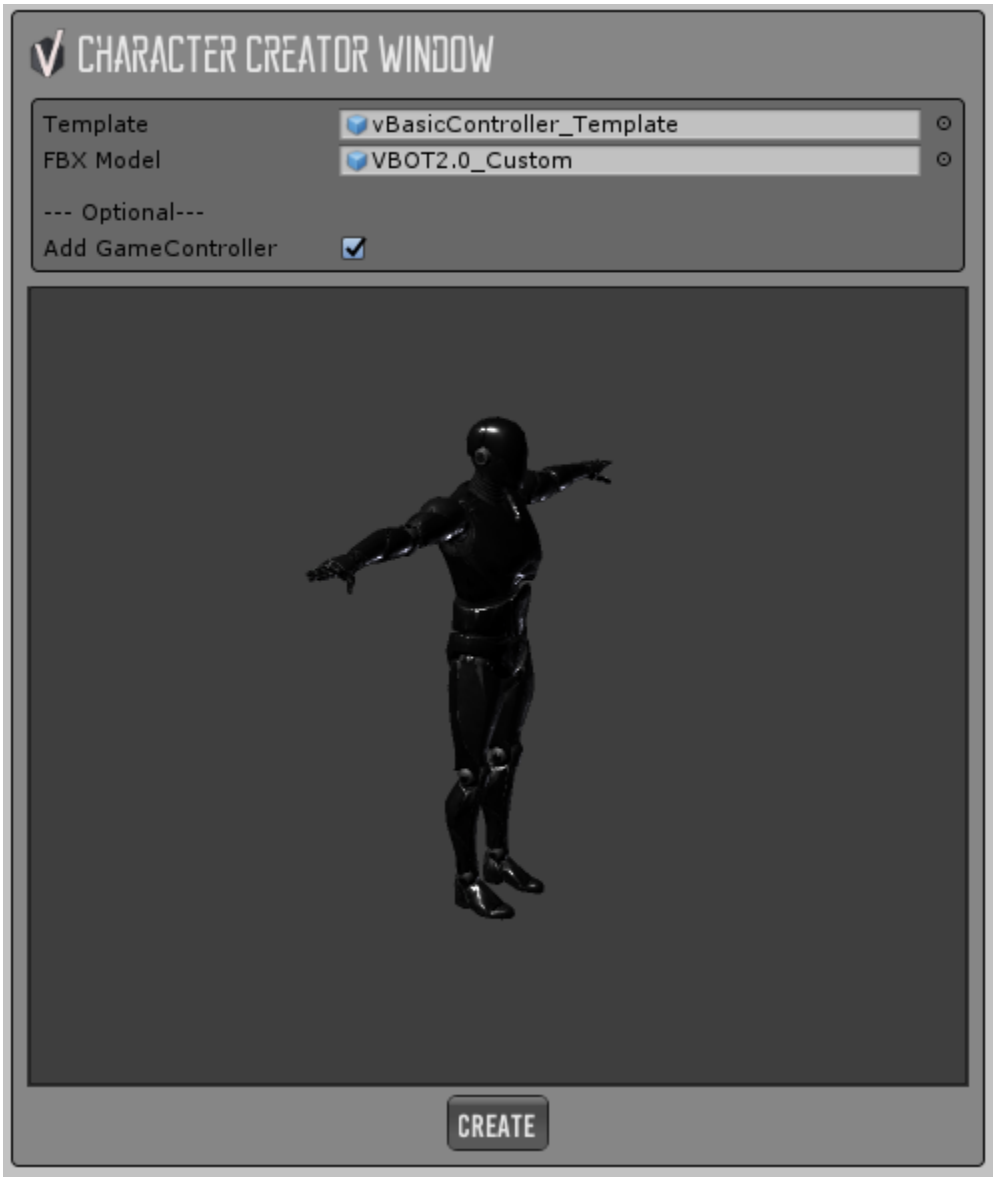
Make sure that your fbx character is set up as **Humanoid**



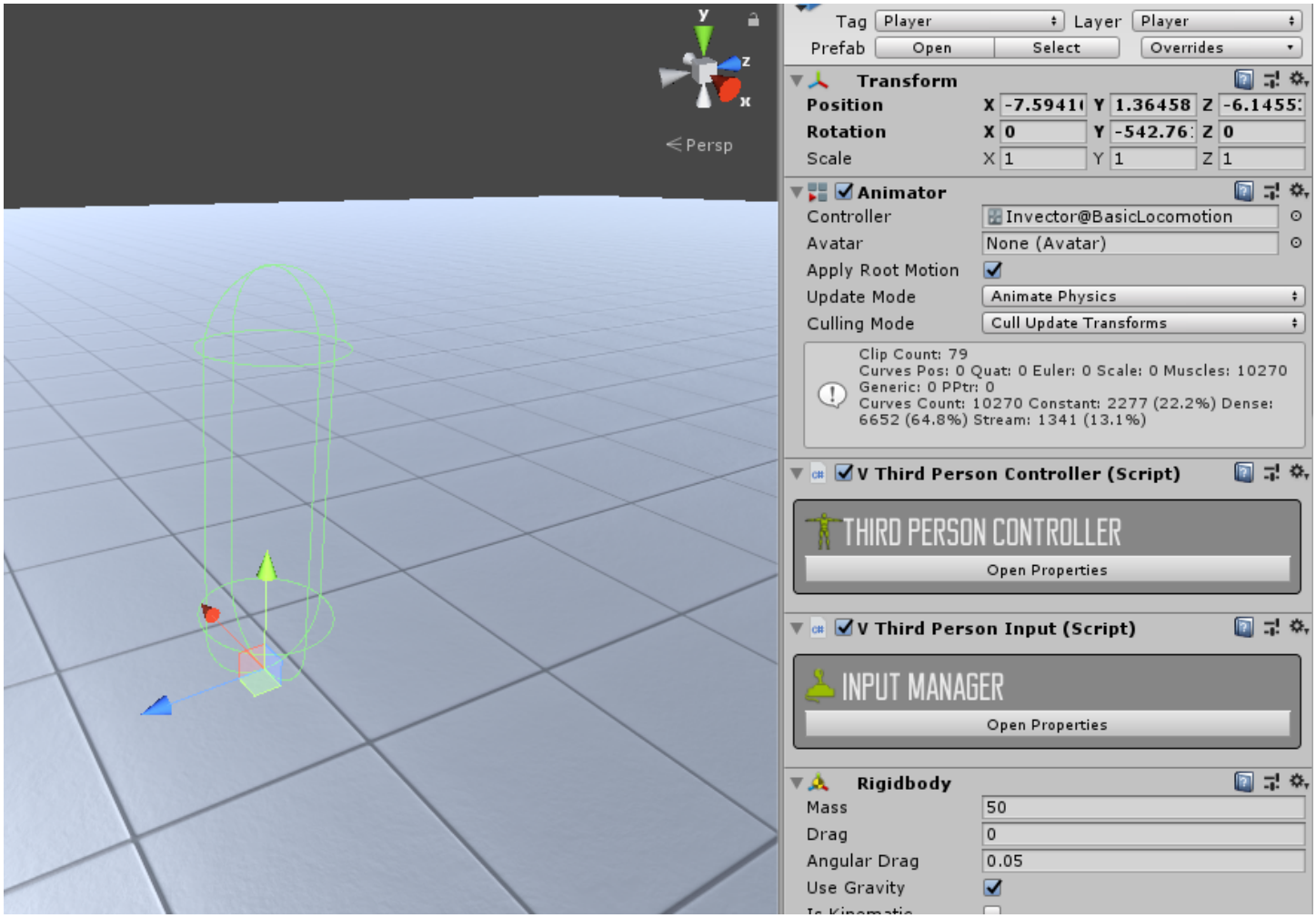
To setup a new basic character, go to the tab *Invector* > *Basic Locomotion* > *Create Basic Controller*
To setup a new basic character, go to the tab *Invector* > *Melee Combat* > *Create Melee Controller*
To setup a new basic character, go to the tab *Invector* > *Shooter* > *Create Shooter Controller*



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

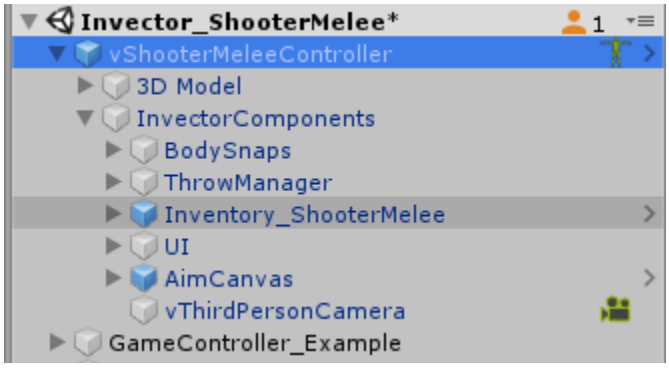


The **vBasicController_Template, vMeleeCombat_Template and vShooter_Template** depending on what controller you're creating, is already assigned in the Template field and is a Controller Prefab that contains everything ready to go, you just need to assign the model to update the Animator Avatar.



The **Character Creator** window takes care of all the hard work for you.

Depending on what Controller you created (Basic, Melee or Shooter) you will find different components inside, such as the Inventory or AimCanvas (Shooter requirement)

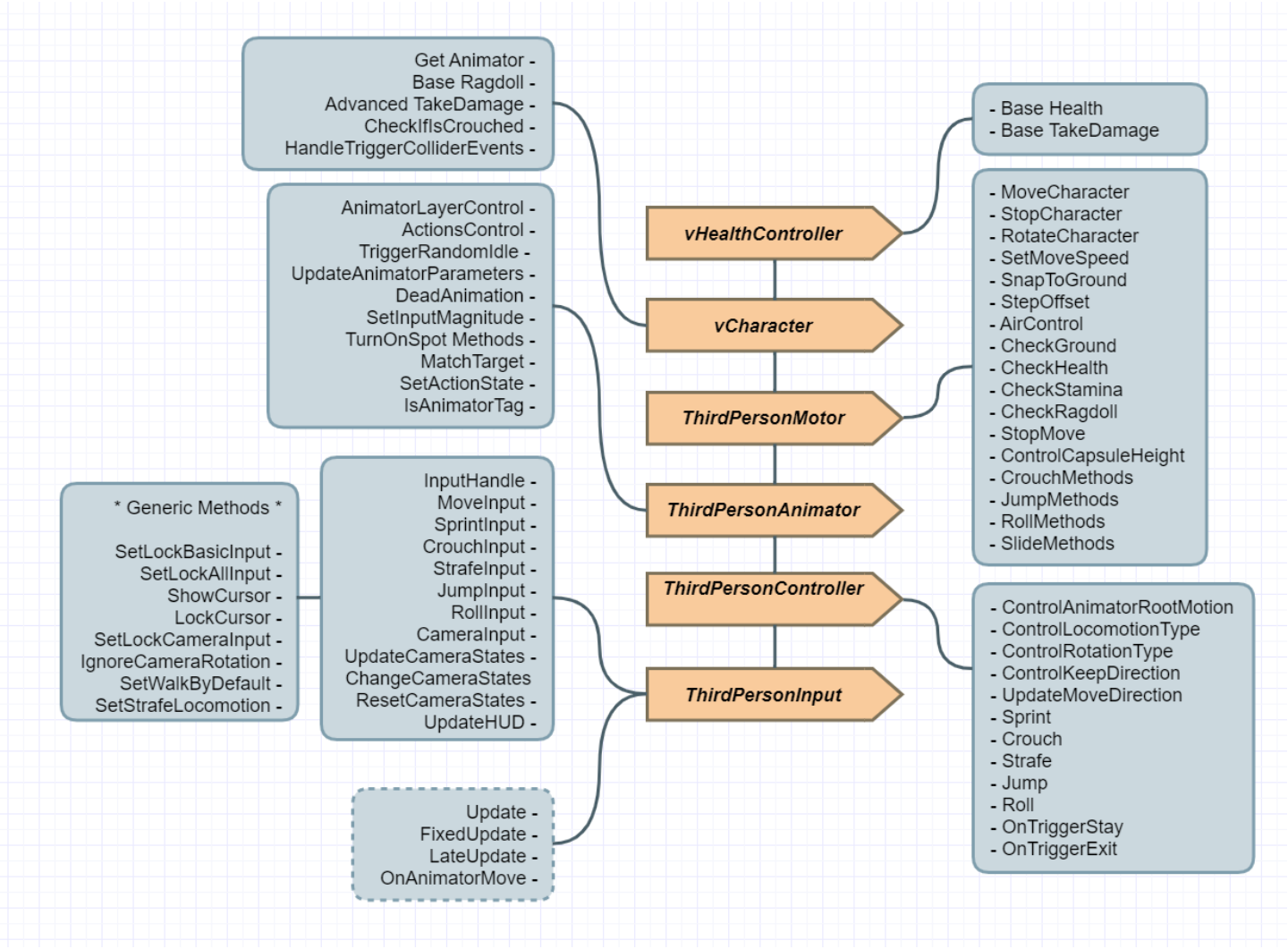


Hit Play and enjoy ☺

HOW IT WORKS?

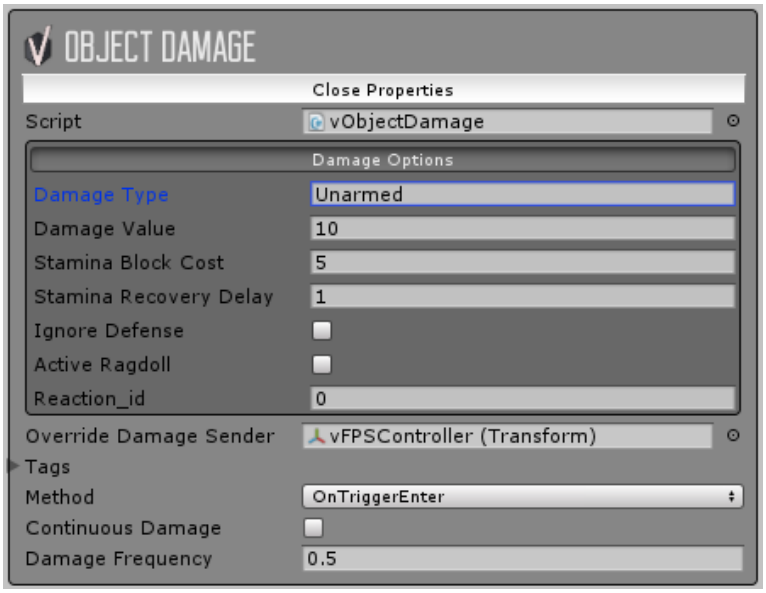
The Controller works with **six main scripts**:

1- **vHealthController** takes care of Health/Stamina and has the method TakeDamage to apply damage.
2- **vCharacter** - it prepares the vHealthController to be a vCharacter using our animator parameters, ragdoll and action system.
3- **vThirdPersonMotor** handles all the information of rigidbody, colliders, verifications of ground distance, stepoffset, slope limit, etc…
4- **vThirdPersonAnimator** is responsible to control the behavior of animations
5- **vThirdPersonController** manages methods like sprint, crouch, roll, jump, etc…
6- **vThirdPersonInput** receives all the input and calls every method of the other scripts on Updates.



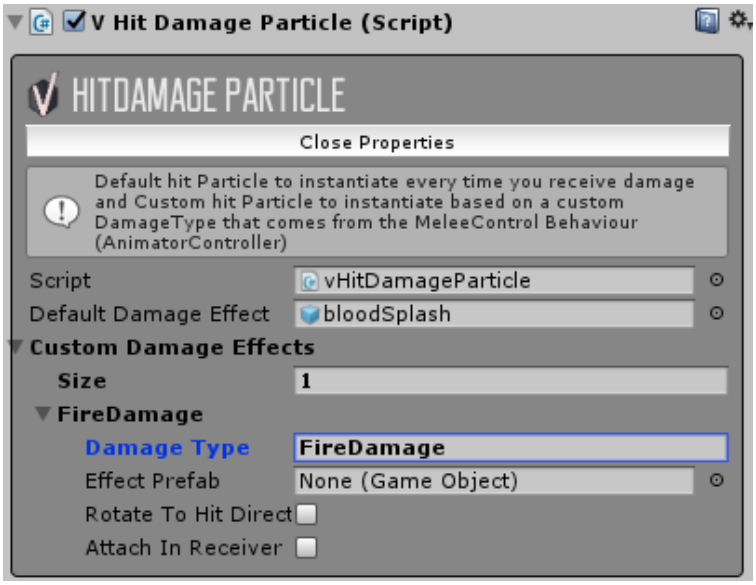
HOW TO APPLY DAMAGE TO THE PLAYER

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, an Object Damage can send damage to any object that uses a **vHealthController**.



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

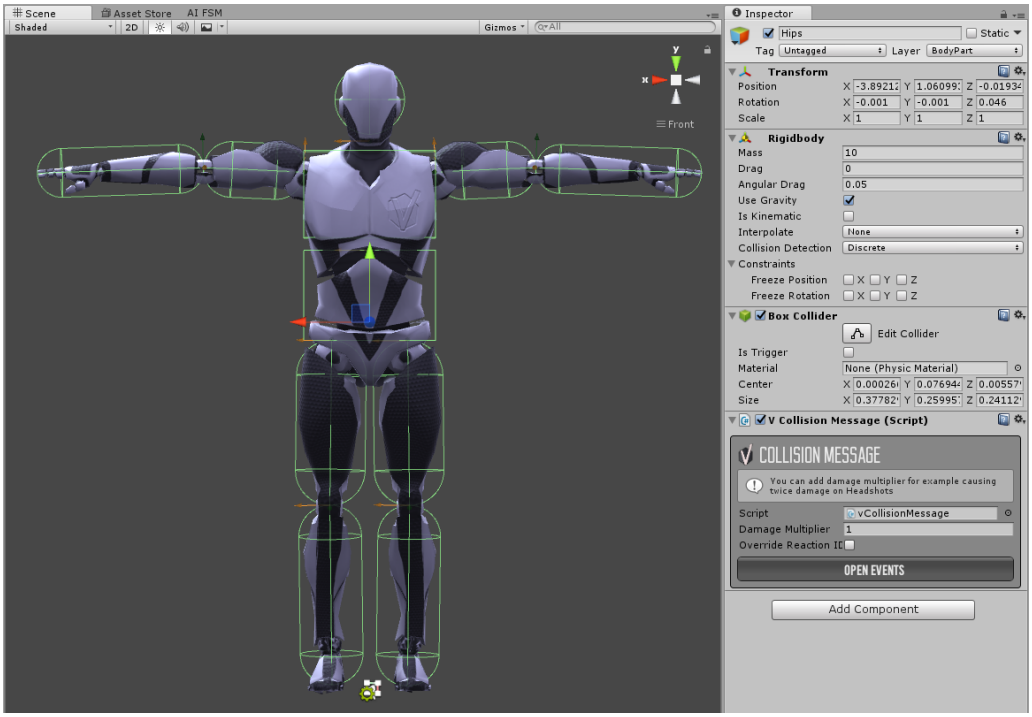


*This component is attached to the Controller or any object that contains the HealthController

- **DamageValue**: How much damage 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 activate 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.

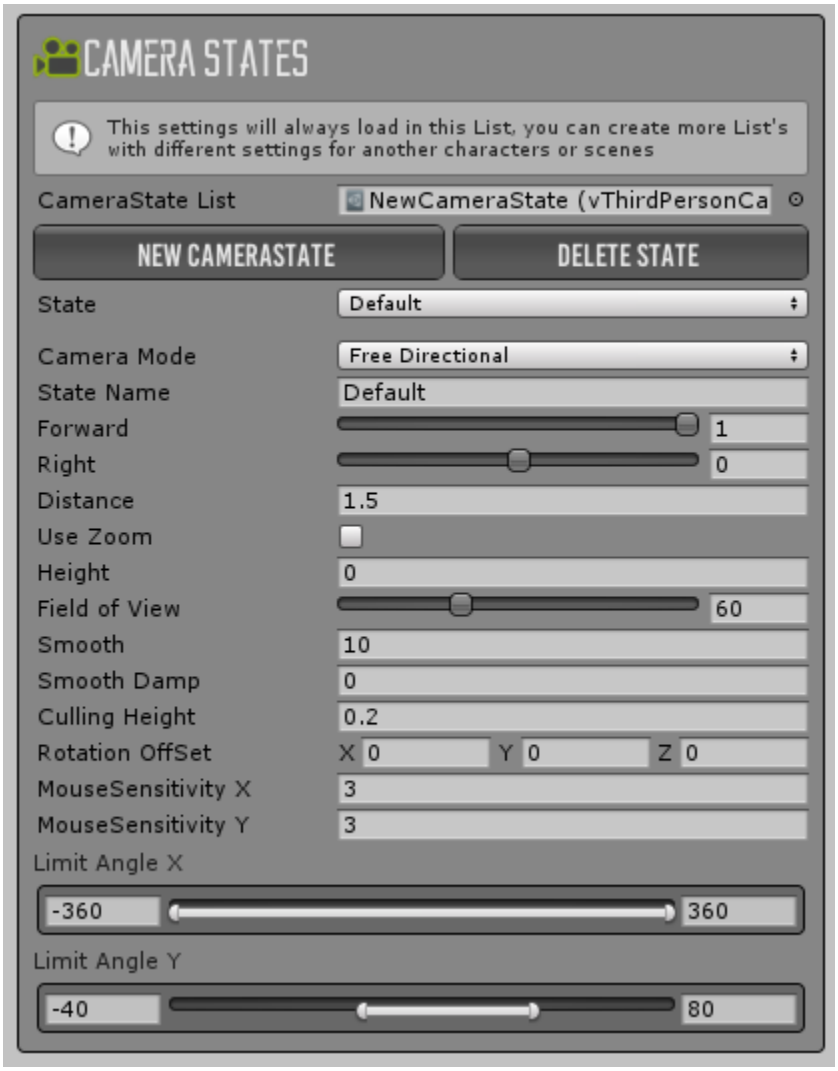
A DamageReceiver is attached to each ragdoll collider, this will allow the Player or AI to apply damage to each bodypart instead of the CapsuleCollider.

- **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, simply change the values in this component.

CREATING A NEW CAMERA STATE

With the Third Person Camera you can create new CameraStates to manage different values, states like “Default”, “Aiming”, “Crouch”, to set up new camera position, distance, height, etc.



You can call the method **ChangeCameraState()** from the tpInput via Triggers or Custom Scripts.

Example:

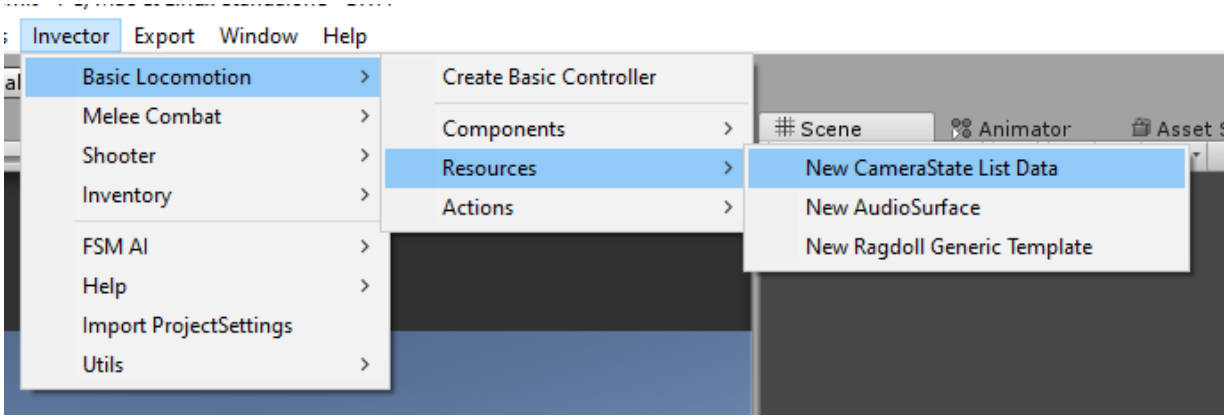
tpInput.ChangeCameraState(string cameraState, bool useLerp = true)

The first string value is the State Name that you created on the Camera Inspector, the second value is a bool, leave it true if you want a smooth transition to this state or false if not.

You can also call a method to reset to the defaultCameraState

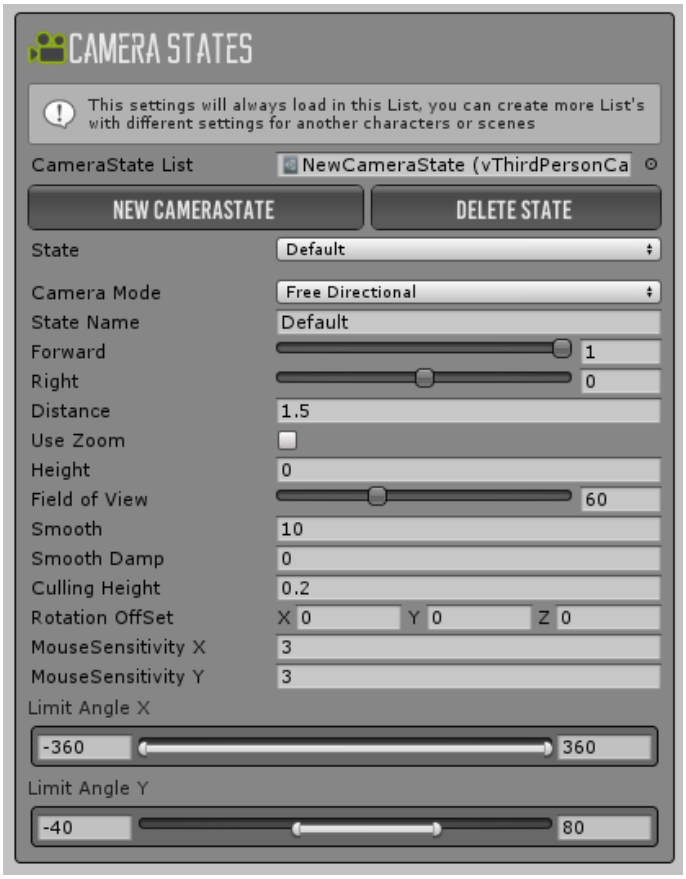
tpInput.ResetCameraState();

You can create a new **CameraState List Data** and assign it in the CameraState List field on TP Camera Inspector.



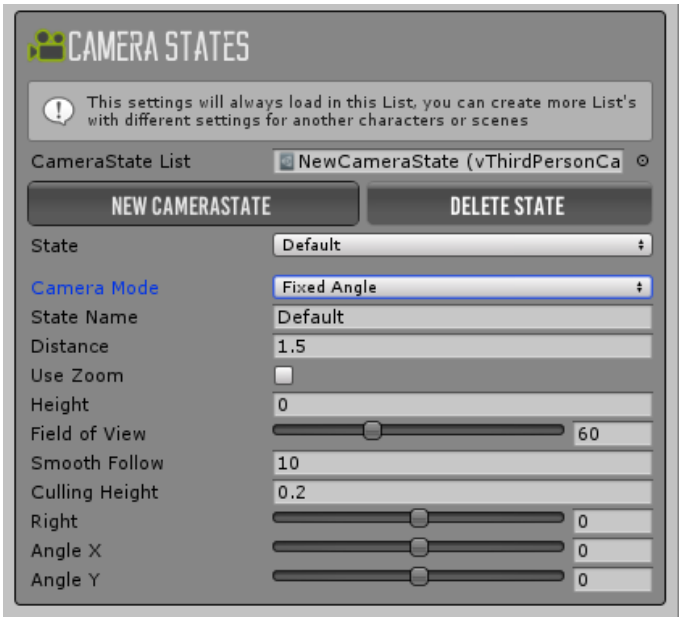
CameraMode - Free Directional

This CameraMode offers a free directional - orbital around the character, with a lot of options to customize and make over the shoulders, or above the character, zoom (mouse only) etc...



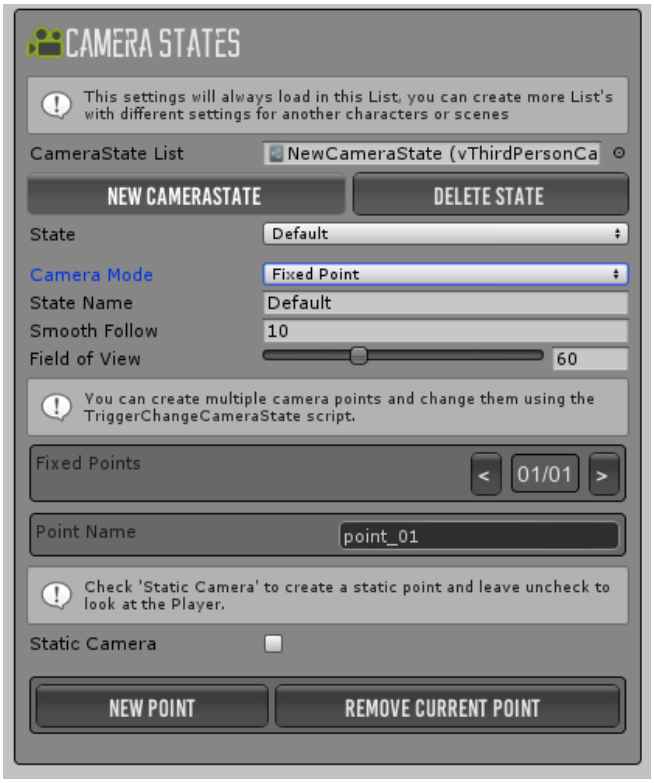
CameraMode - Fixed Angle

This is a feature to use for Isometric or Topdown games, you can set up a fixed rotation for the camera and make games like Diablo or MGS 1.

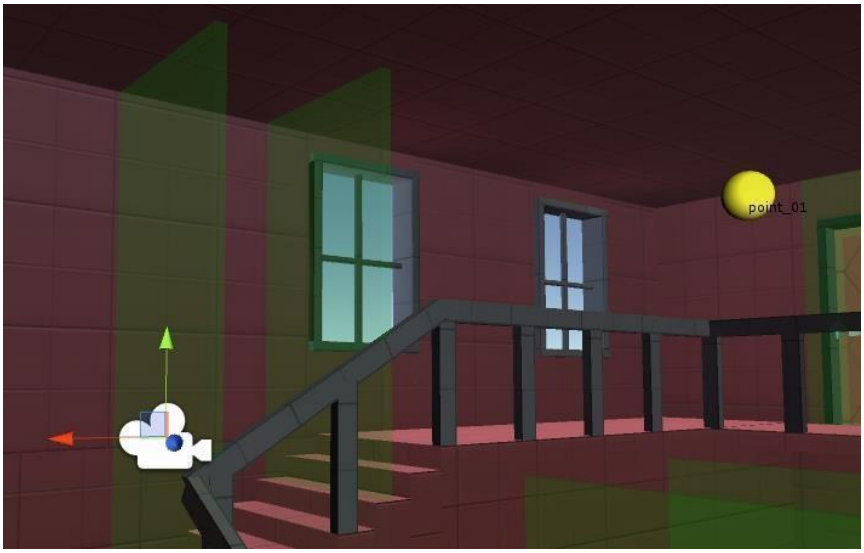


CameraMode - Fixed Point

Fixed Points are states that you can create to use the Camera as a CCTV mode (Oldschool Resident Evil series), this state will follow the character by default or you can check Static Camera to make it fixed.



You can also create multiple points and change with the **TriggerChangeCameraState** that has an option for smooth transition between points or not. *always leave a safe-space between triggers



XBOX CONTROLLER SUPPORT

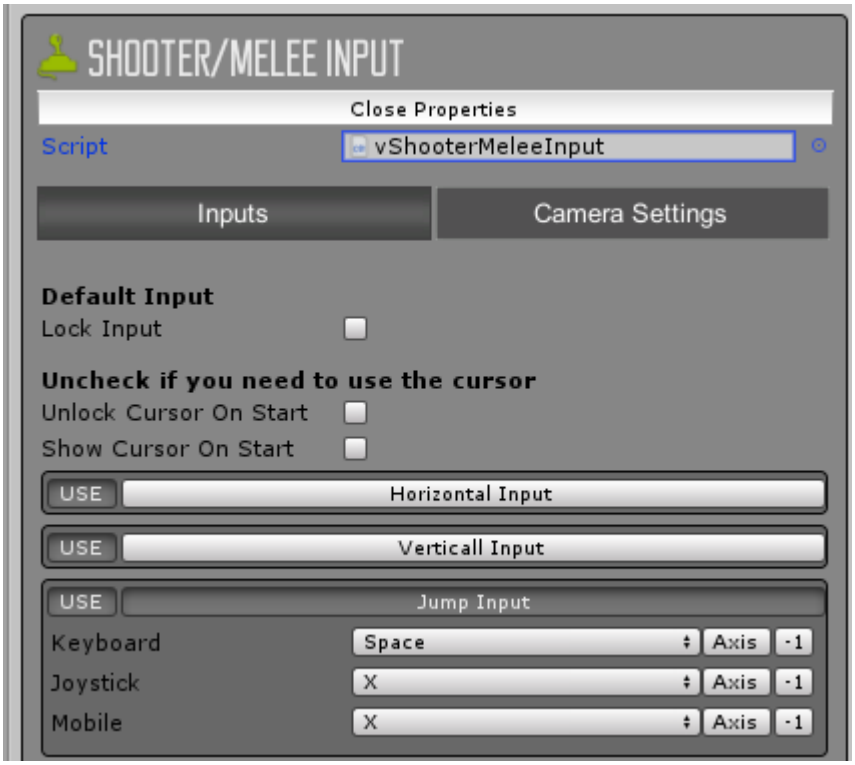
This package works great with the **360 controller and XboxOne controller**.
Make sure you imported our InputManager by going to Invector > Import ProjectSettings.

For other controllers like PS4, Logitech or Generic you need to remap the InputManager based on your controller inputs.

INPUT MANAGER

The InputManager manages input for the character actions and movement.

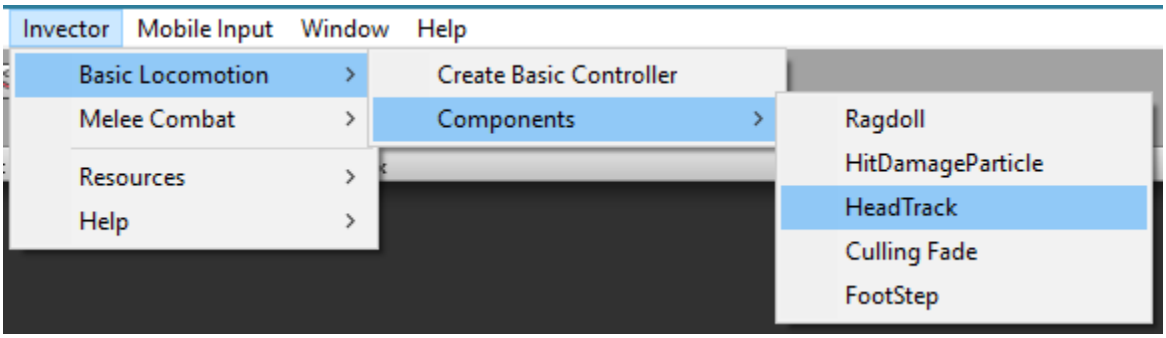
If you created a **Basic Locomotion** it will use the **vThirdPersonInput**, if it's a **Melee Combat** character it will use the **vMeleeCombatInput**, and the **Shooter** uses the **vShooterMeleeInput**.

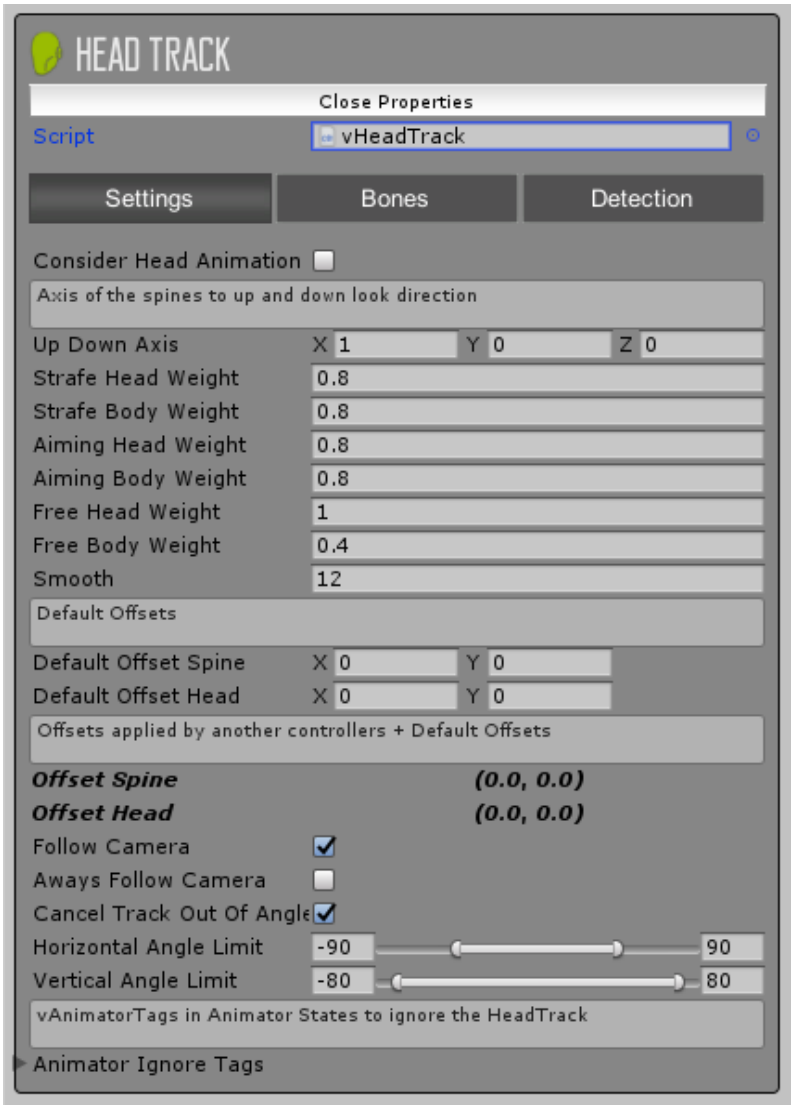


You can remap the input and disable it if you don't need to use it for your game.

HEAD TRACK

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





UpDownAxis: Depending on your character's rig, the orientation could be different so if you're looking up/down and the character is responding left/right instead, try changing the value 1 of the X, Y, Z until the character looks normal.

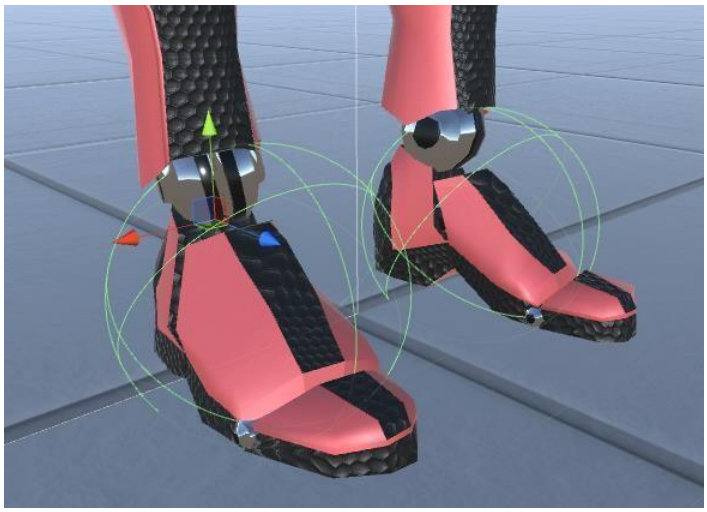
Weights: You can set different weights depending on the **Locomotion Type** (Free Movement or Strafe)
Angle X and Y: You can limit the max angle to look up/down and left/right
Smooth: the speed to look at something
AnimatorTags: Which animator tags will ignore the Headtrack, for example you have an animation that the character opens a door and look at the knoble, it's best to not use the headtrack while this animation is playing, making the animation play as it was designed without the headtrack interfering.
Offset Spine//Head: more offset options in case you want to fine tune your headtrack.
Follow Camera: tracks the camera forward, disable when using a topdown view
Always FollowCamera: force the controller to always follow the camera regardless of AnimatorTags

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.

FOOSTEP AUDIO SYSTEM

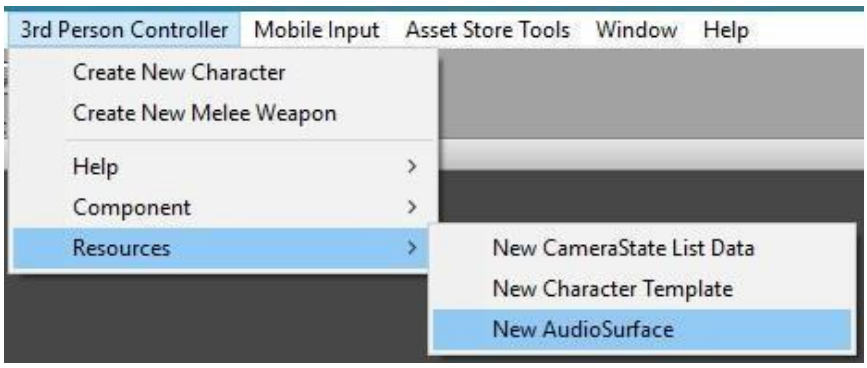
Video tutorial: *https://www.youtube.com/watch?v=gxesgNH0UBM*

When you create a new Character the FootStep component will be already attached, if you want to add a component into another Character go to the *3rdPersonController Menu* > *Component* > *FootStep*. The component will automatically create a **sphere collider** on the foot of your character, but you need to make sure that the Radius and Position of the sphere is **touching** the ground.



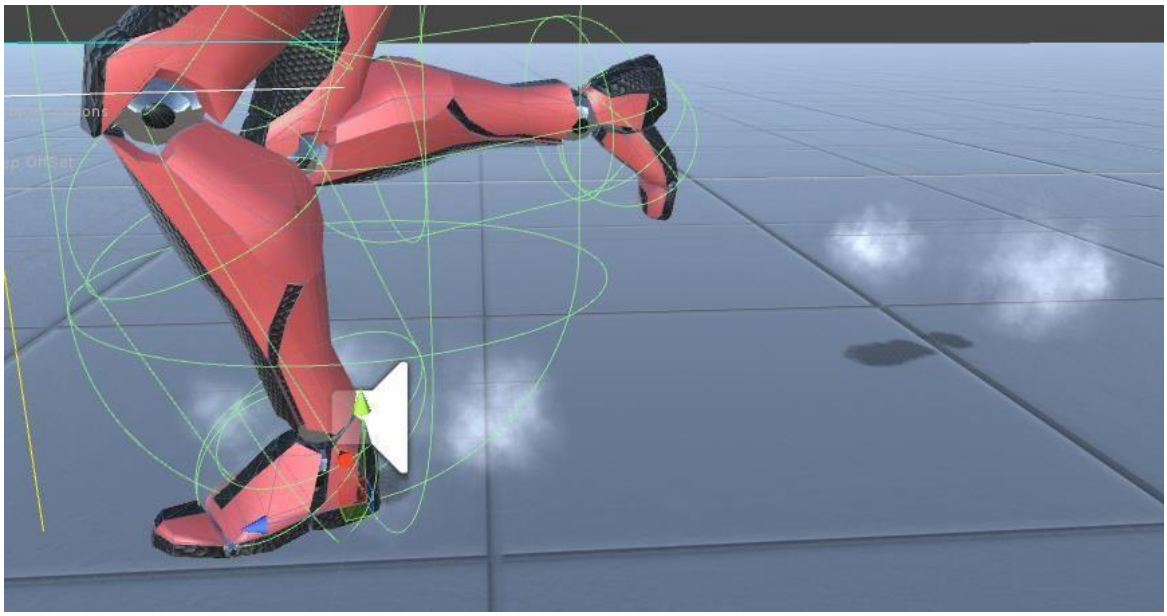
You can select the **LeftFoot** and **RightFoot** Sphere and manipulate the **Center XYZ** to position as you like, and change the **Collider Radius** too, the size of this sphere will depend on your Rig bone size. Assign the "***defaultSurface***" that comes with the package to have an example of how it works.

To create a new AudioSurface go to the 3rd Person Controller menu > Resources > New AudioSurface.



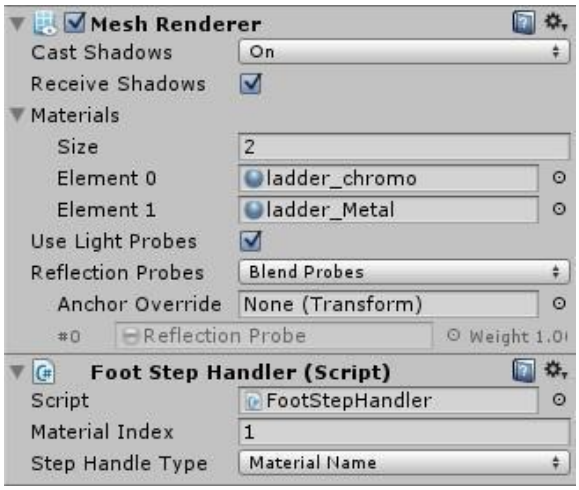
Now you can create **Custom Surfaces**, to play other audio clips based on the **material** that the sphere collider will hit. Assign the new CustomSurface to a new CustomSurface on the FootStep Inspector.

You can assign an AudioMixer for better control surfaces, and you can instantiate a **Particle** as well, see the example on the DefaultSurface call ‘smoke’ that also uses a **StepMark** sprite call SimpleStepMark.



V1.1 Using the FootStep system in objects with multiple Materials

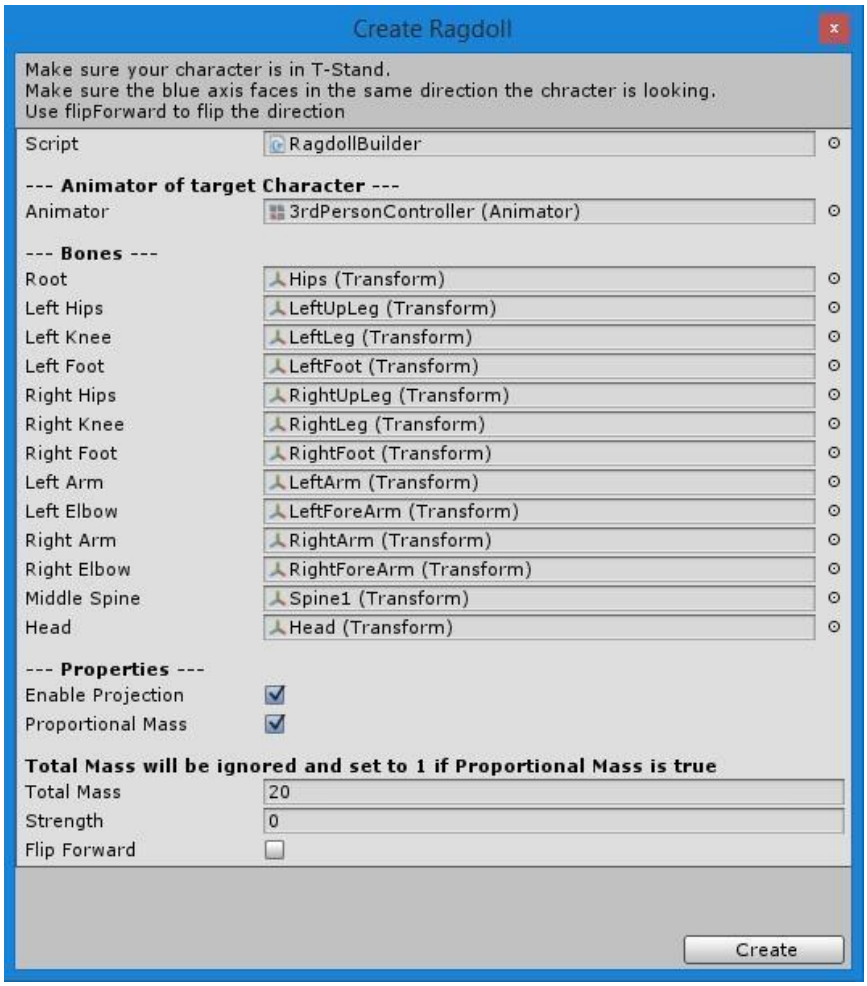
If your gameobject has multiple materials and you need to play a specific material, you can use the FootStepHandler script and set the correct Material Index of your object. (*See example on the Ladder prefab)



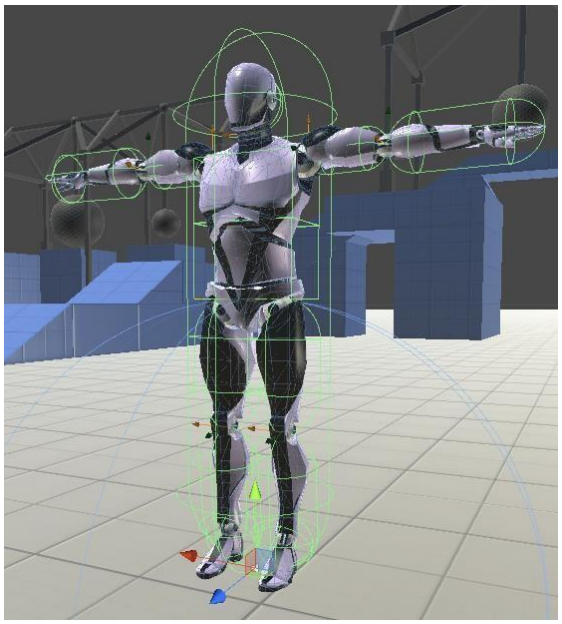
CREATING A RAGDOLL

Creating a Ragdoll is just as easy as creating your Character, just go to the tab *Invector* > *Basic Locomotion* > *Components* > *Ragdoll*.

If you have your character selected on the Hierarchy, all the fields will **autofill**, if not, just click on your character and it will autofill for you, this template was designed to **save time**, so you don't have to waste your time dragging and drop every bone, instead just hit the "Create" button and it's ready to go.

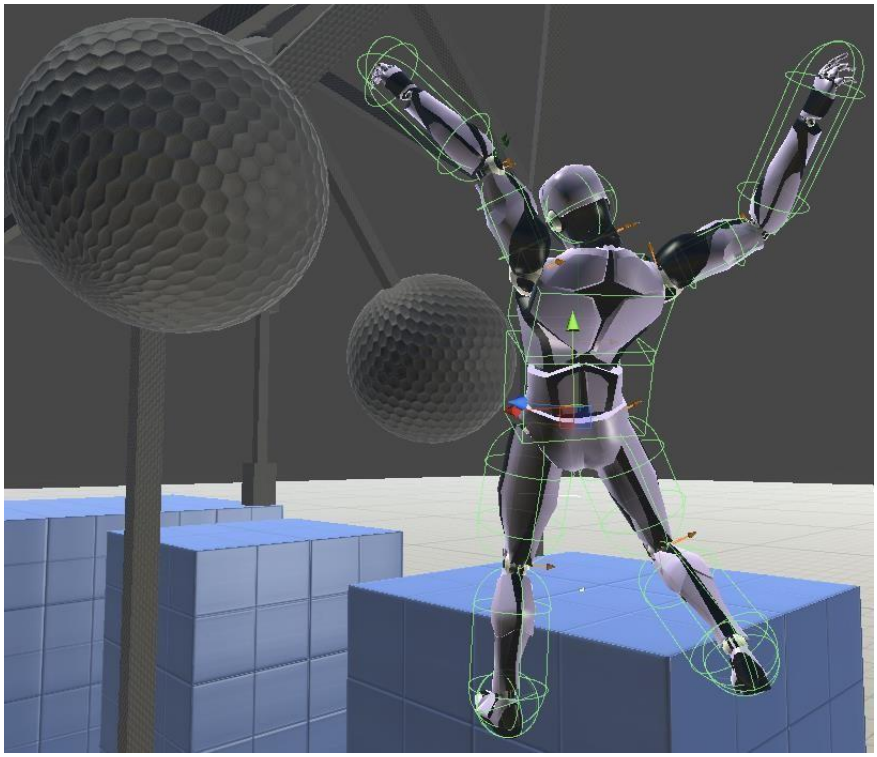


We strongly recommend keeping the **Enable Projection** and the **Proportional Mass** enabled, and do not forget to use **Scale Factor 1** on your **fbx** Model. This you provide better behavior of your ragdoll.

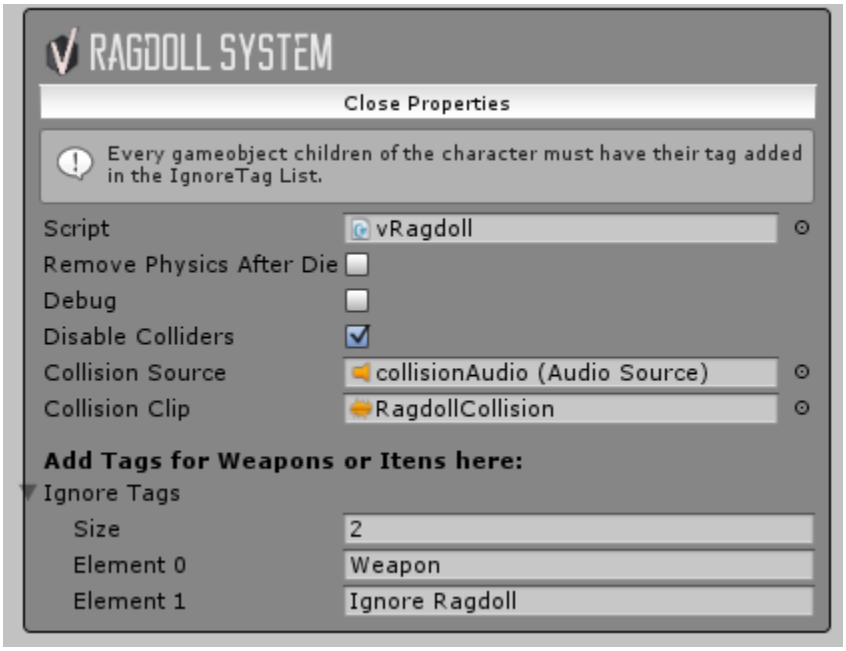


To enable the ragdoll, you can use the Script **ObjectDamage** or just call this line on the **OnCollisionEnter** method.

hit.transform.root.SendMessage ("ActivateRagdoll", SendMessageOptions.DontRequireReceiver);



v1.1b – Add "***Ignored Tags***" you can add a list of tags for objects that are children of the Player to keep the rotation correctly, otherwise it will mess up the rotation when the Ragdoll are on.



*** 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.

Ps* Don't forget to add the Layer "BodyPart" for each collider.

HOW TO ADD NEW ANIMATIONS/ACTIONS?

We have 2 excellent video tutorial showing examples on how to add simple and complex animations

Simple > https://www.youtube.com/watch?v=VVqkSlQ4x2M

Complex > https://www.youtube.com/watch?v=hlLWnsIQz-c

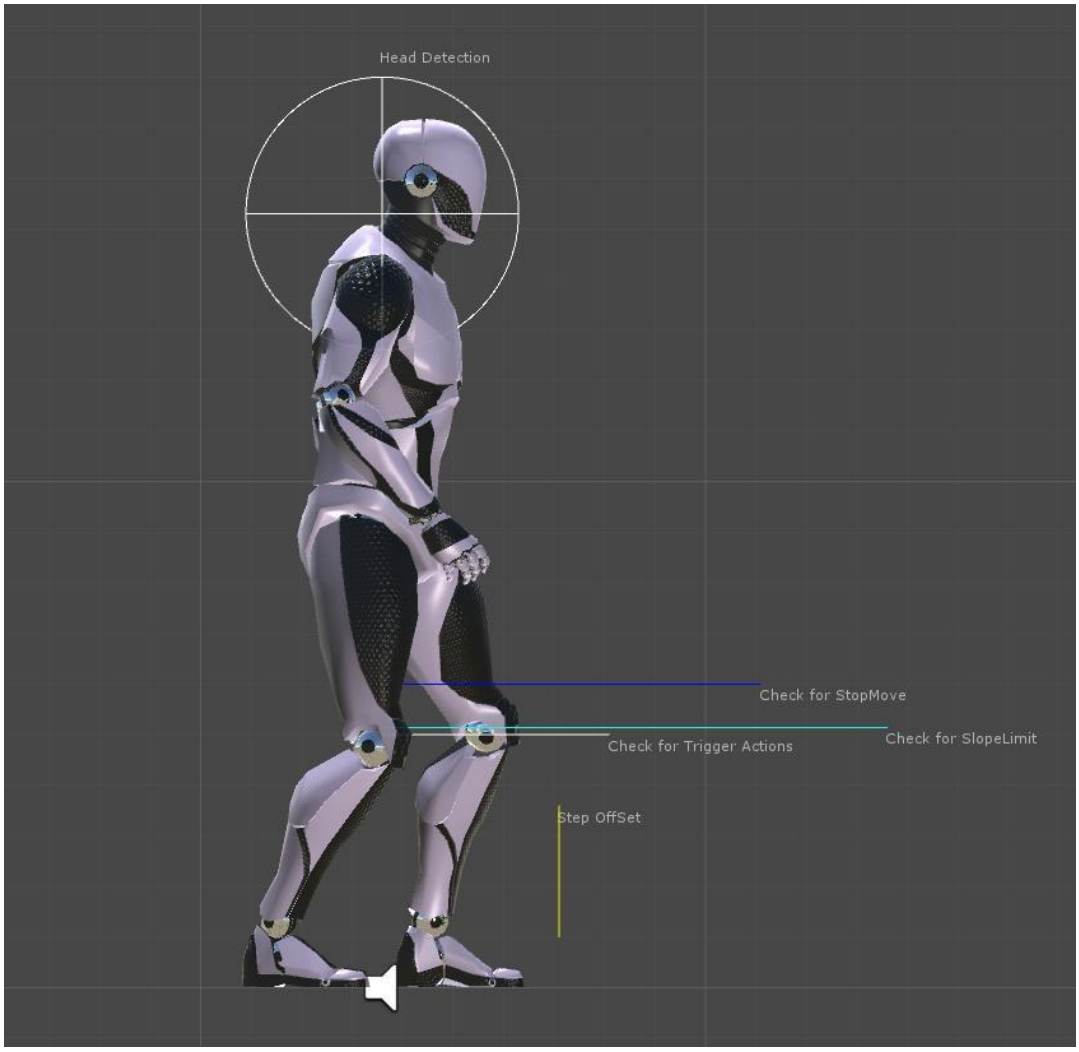
RAYCAST CHECKERS

Head Detection is a SphereCast that will detect if an object above, and keep the character crouched, use the same layer as the Ground Layer (Default). Just adjust to sync with the height of your capsule collider.

StopMove is a Raycast that detect any object with the layer (Default, StopMove) to prevent the character to walk in place, you can use a StopMove in an invisible wall for example, and the camera will not clip, because the culling layer is set to “Default”.

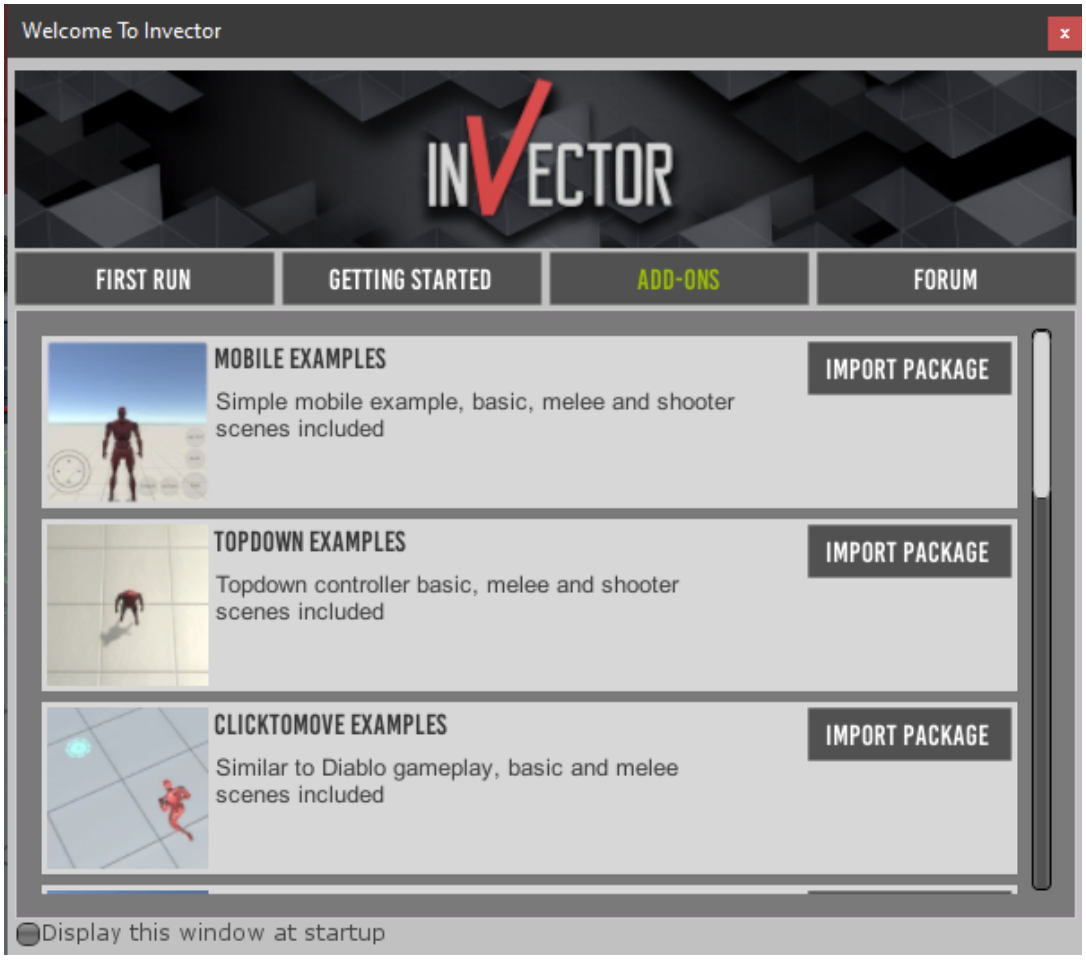
SlopeLimit will prevent the character from walking in absurd angle heights, float customizable on the Player Inspector.

StepOffset is to help the character walk in custom height steps, adjust the values on the Player Inspector.

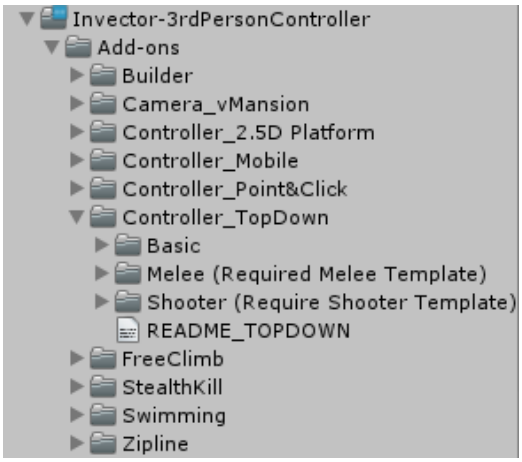


TOPDOWN / 2.5D / POINT & CLICK CONTROLLER / MOBILE

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



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



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.

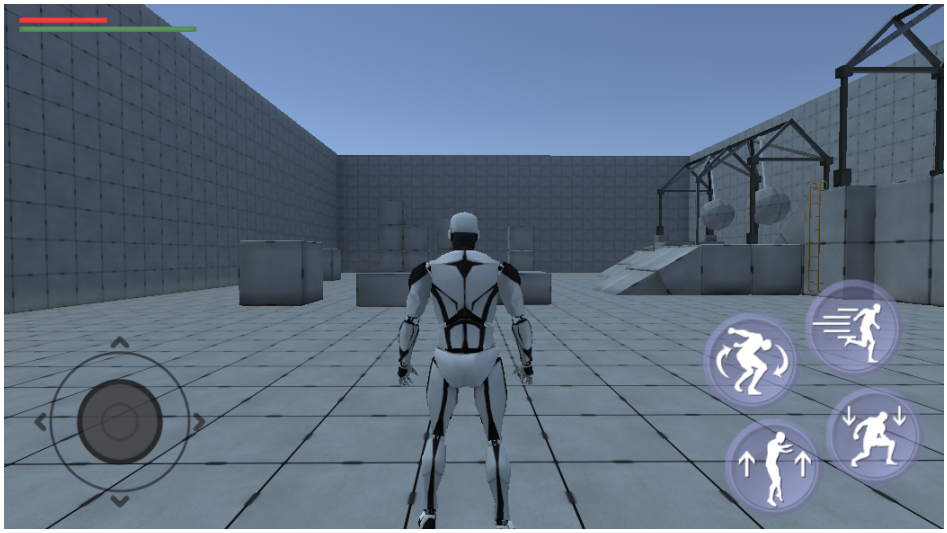
Go to the add-on folder you want and replace your current **vThirdPersonController** component for the **vTopDownController** or **2_5Dcontroller** in the Player Inspector.

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

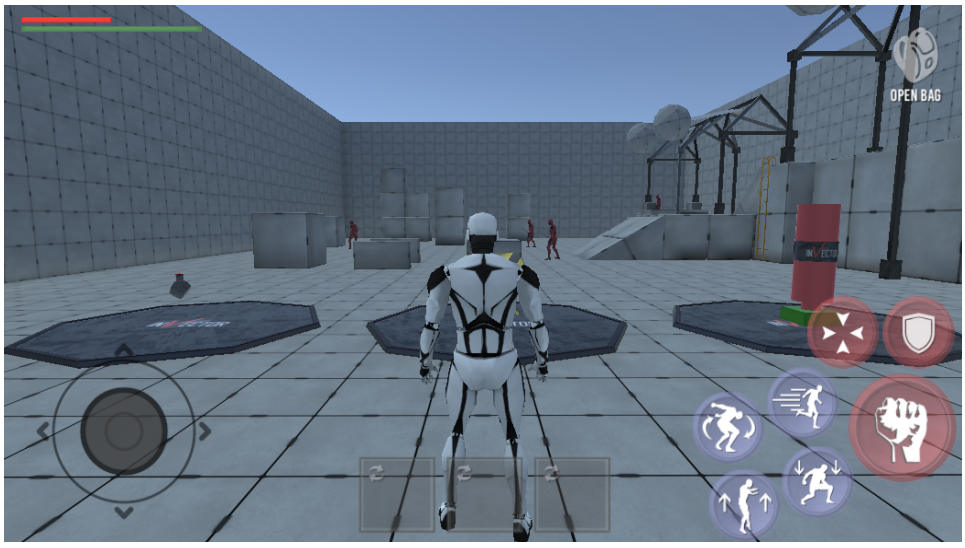
You can import the Mobile demo scenes by going to the menu *Invector > Addons*.



After importing the mobile package, you can create a regular ThirdPersonController using the Character Creator Window and drag and drop a Mobile UI prefab inside your character.

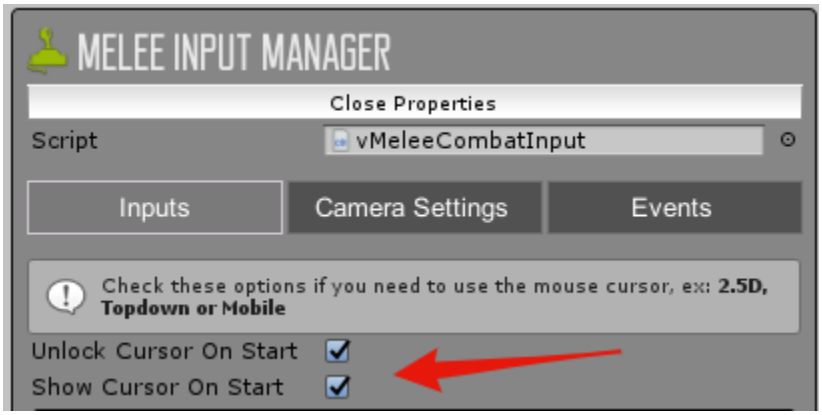
You can find the MobileUI prefab examples inside each folder Basic, Melee or Shooter

ps You must own the MeleeCombat or Shooter Templates in order to use their prefabs, otherwise you will receive warnings about missing content.*



Make sure to check the options “Unlock Cursor and Show Cursor” when using the Mobile UI.

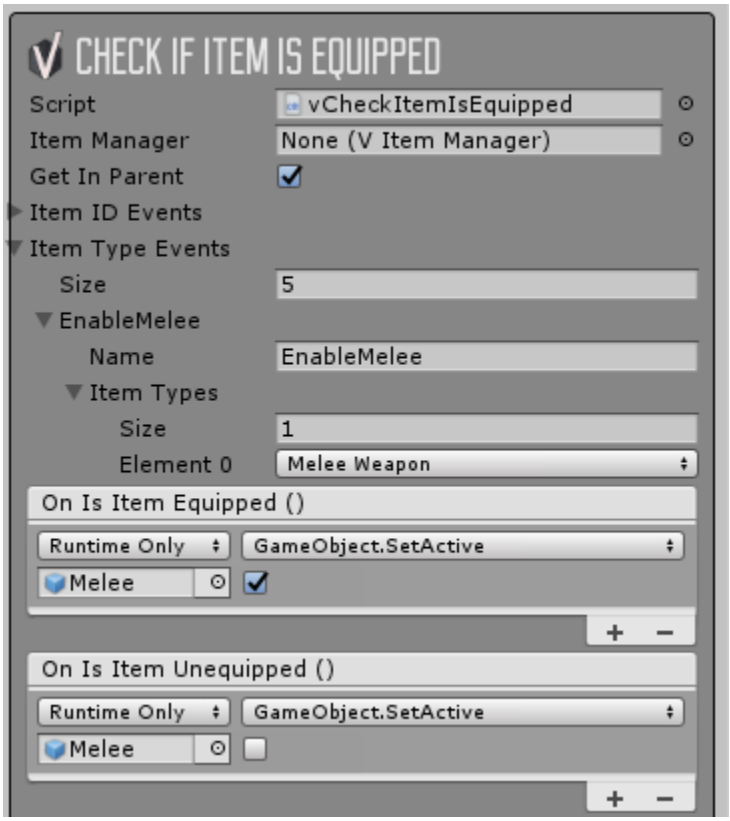
vThirdPersonInput > Basic Controller
vMeleeCombatInput > Melee Controller
vShooterMeleeInput > Shooter or Shooter w/ Melee Controller



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.

You can use the component **vCheckItemIsEquipped** if you’re using the Inventoryto turn on/off mobile buttons:

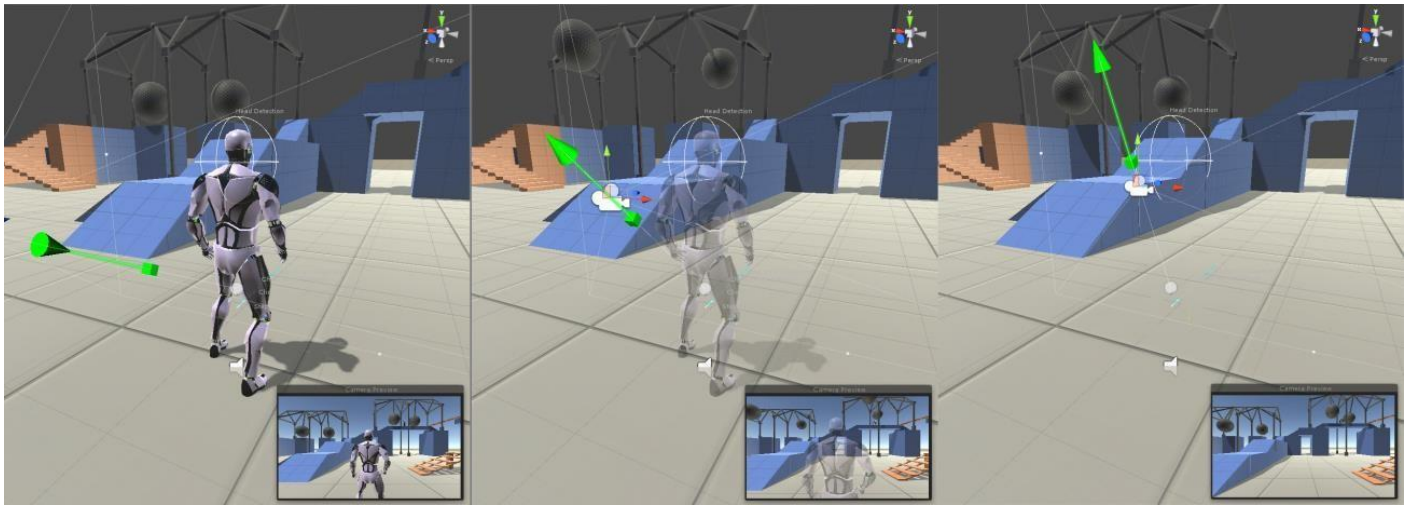


If you’re not using the Inventory, you can use the **vCollectShooterMeleeControl** Events to manage the buttons, it identifies if you’re using a shooter weapon or a melee weapon only.

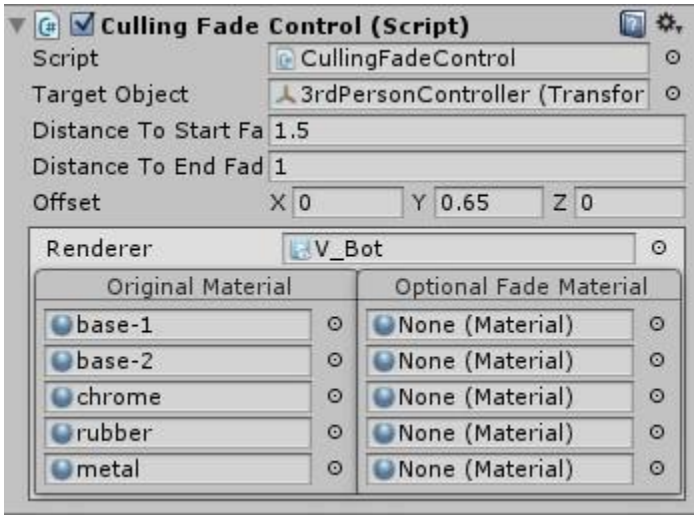
CAMERA CULLING FADE

We add a Culling Fade script for the camera to avoid seeing through the character's mesh, you can set up the distance to start fading and an offset.

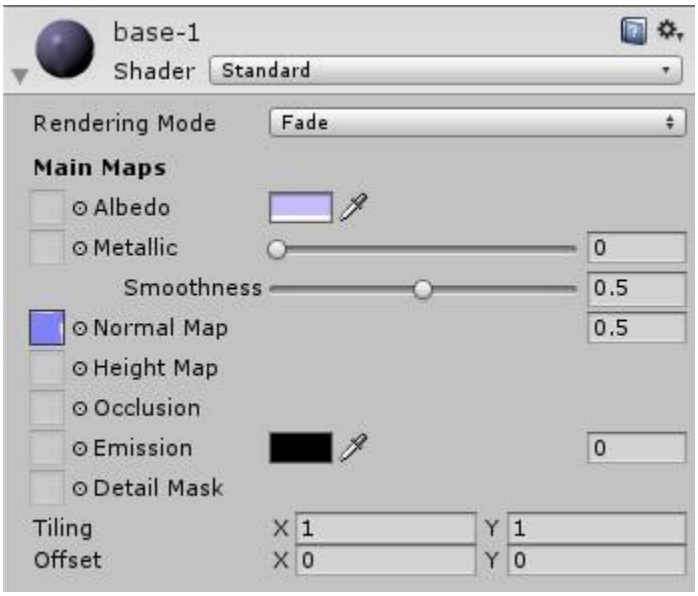
Example:



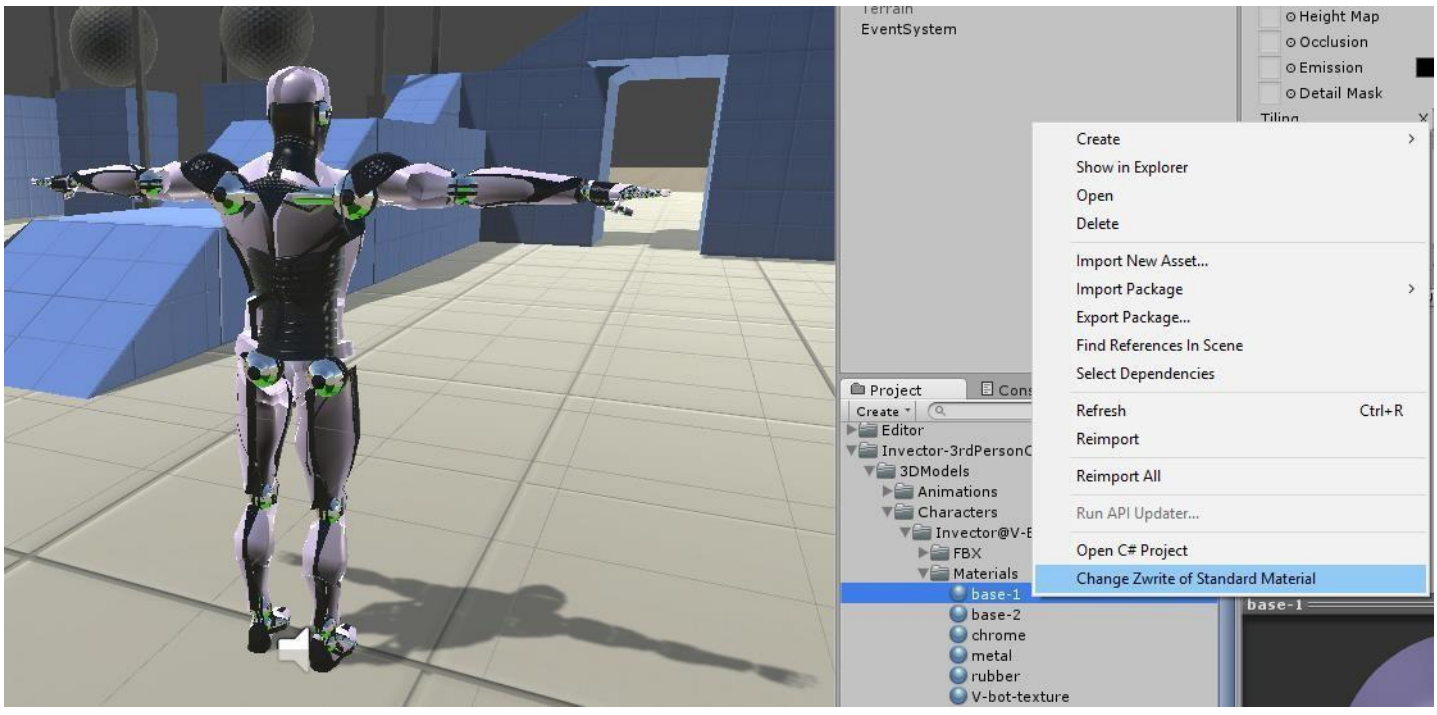
Our Culling Fade will set up automatically for the default Standard Shader of Unity's, but you also can use custom shaders, just make an additional copy with the fade material and assign in the "Optional Fade Material" field.



If you are using the Standard Shader, just select the Rendering Mode "Fade" on the Material.



The character will look like this (picture below) but you can fix it by right clicking at the material and “Change Zwrite of Standard Material”.

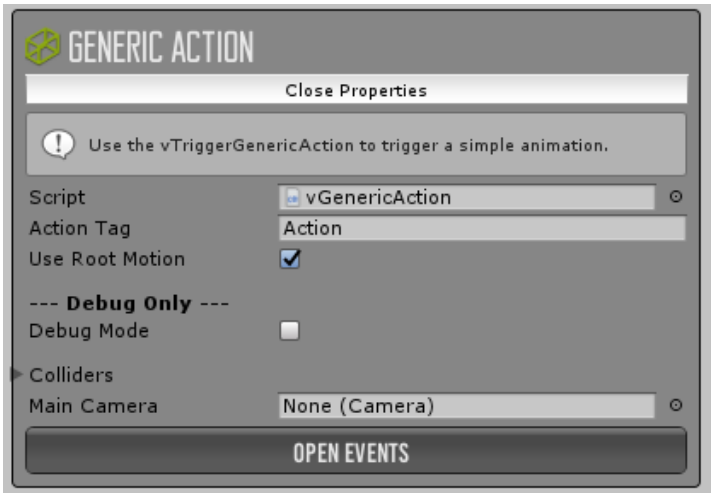


UPDATE V1.1B – now the script will be attached into the Controller just like the Ragdoll and the Footstep, It’s a modular feature.

GENERIC ACTION (HOW TO INTERACT WITH OBJECTS)

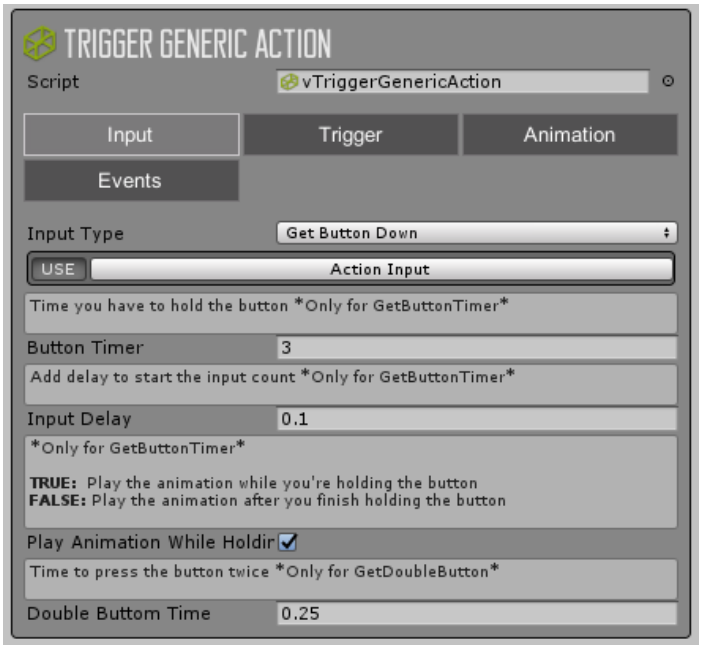
Video tutorial is also available: https://youtu.be/hlLWnsIQz-c

The GenericAction is a system designed to trigger simple interaction animations such as opening a door, jumping over an obstacle, pushing a lever, etc...

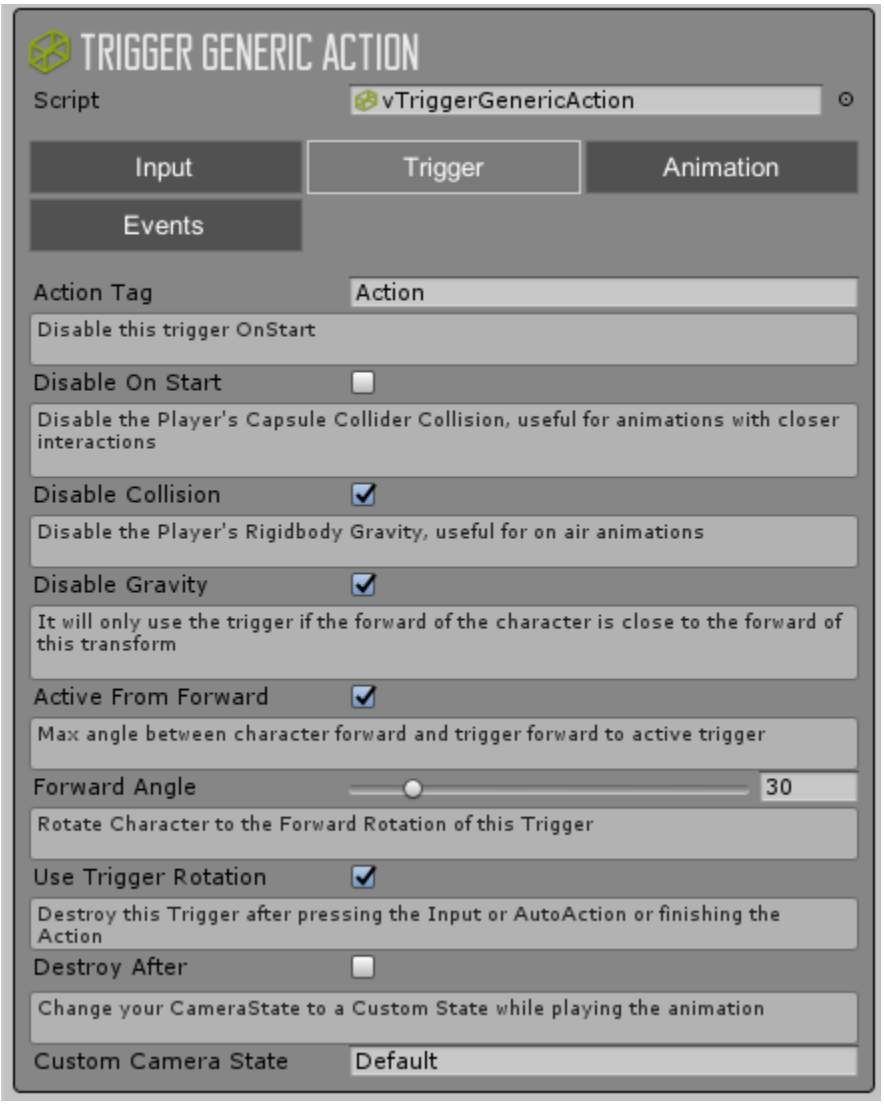


(*This component is already added to a Player when creating a new controller*)

You need to add a **vTriggerGenericAction** component on a collider IsTrigger to perform the action.
In the **Input** tab you can assign an ActionInput to trigger the action and select between the InputType GetButtonDown, GetDoubleButton, GetButtonTimer or perform an AutoAction as soon as the player enters the trigger.

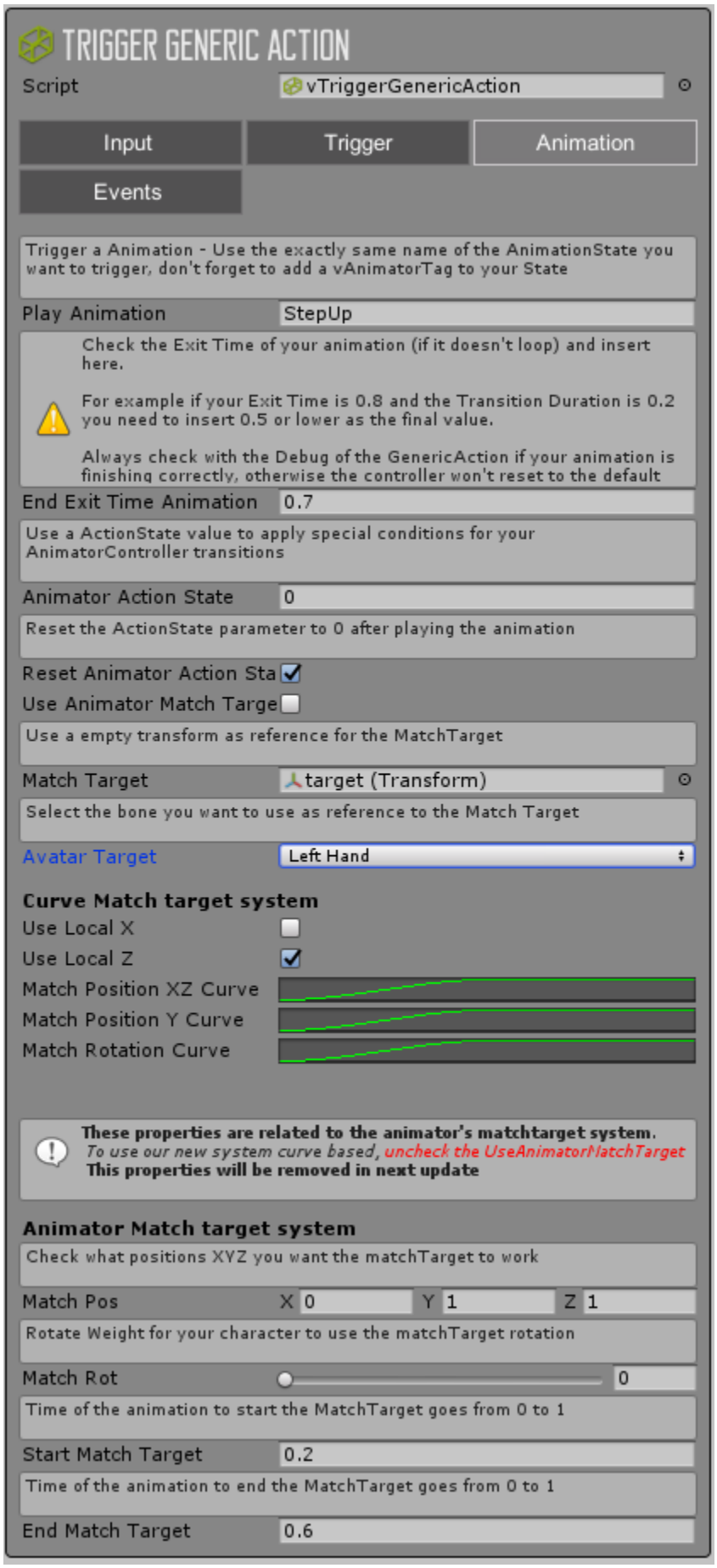


There are several options in the **Trigger** tab such as disabling the player's gravity and collision, only performing the action if the player is facing the trigger forward, destroying the trigger after the interaction, etc… every option has a description of what it does:



In the **Animation** tab, you must assign the name of the animation clip you want to play it
(*not required, you can perform an action without playing an animation as well*)

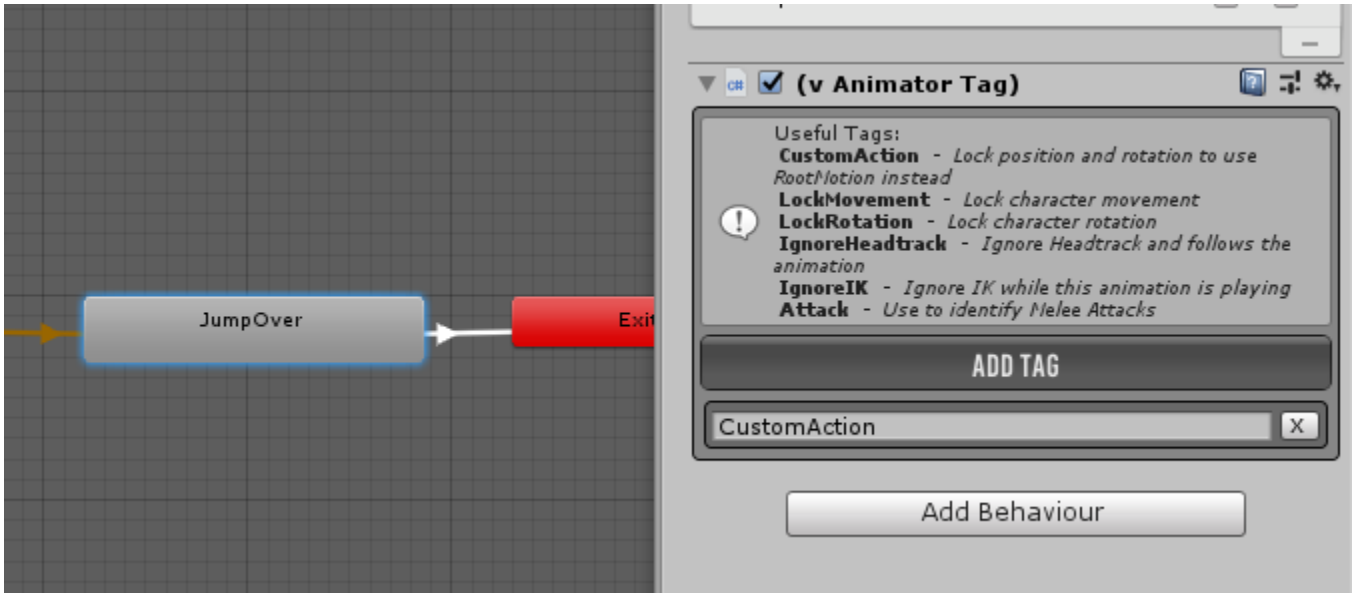
If you need to align the player into a specific position you can use the MatchTarget option or our Curve system to place the character into a custom transform while playing the animation.


TRIGGER GENERIC ACTION
Script
vTriggerGenericAction
Input
Trigger
Animation
Events
Trigger a Animation - Use the exactly same name of the AnimationState you want to trigger, don't forget to add a vAnimatorTag to your State
Play Animation
StepUp
Check the Exit Time of your animation (if it doesn't loop) and insert here.
For example if your Exit Time is 0.8 and the Transition Duration is 0.2 you need to insert 0.5 or lower as the final value.
Always check with the Debug of the GenericAction if your animation is finishing correctly, otherwise the controller won't reset to the default
End Exit Time Animation
0.7
Use a ActionState value to apply special conditions for your AnimatorController transitions
Animator Action State
0
Reset the ActionState parameter to 0 after playing the animation
Reset Animator Action Sta
Use Animator Match Targe
Use a empty transform as reference for the MatchTarget
Match Target
target (Transform)
Select the bone you want to use as reference to the Match Target
Avatar Target
Left Hand
Curve Match target system
Use Local X
Use Local Z
Match Position XZ Curve
Match Position Y Curve
Match Rotation Curve
These properties are related to the animator's matchtarget system.
To use our new system curve based, uncheck the UseAnimatorMatchTarget
This properties will be removed in next update
Animator Match target system
Check what positions XYZ you want the matchTarget to work
Match Pos
X 0
Y 1
Z 1
Rotate Weight for your character to use the matchTarget rotation
Match Rot
0
Time of the animation to start the MatchTarget goes from 0 to 1
Start Match Target
0.2
Time of the animation to end the MatchTarget goes from 0 to 1
End Match Target
0.6

And in the Events tab you can trigger things like Playing a sound or particle effect once you PressTheActionInput, StarAnimation, EndAnimation, FinishTimerInput, etc…

The GenericAction is a very powerful tool for the template, and there are several generic actions included in the project, open the Basic Locomotion demo scene to see those examples in action.

When adding a new Animation to your Action StateMachine, make sure to use the vAnimatorTag "*CustomAction*" to it, so that the Controller knows you're doing an Action.



ANIMATOR TAG

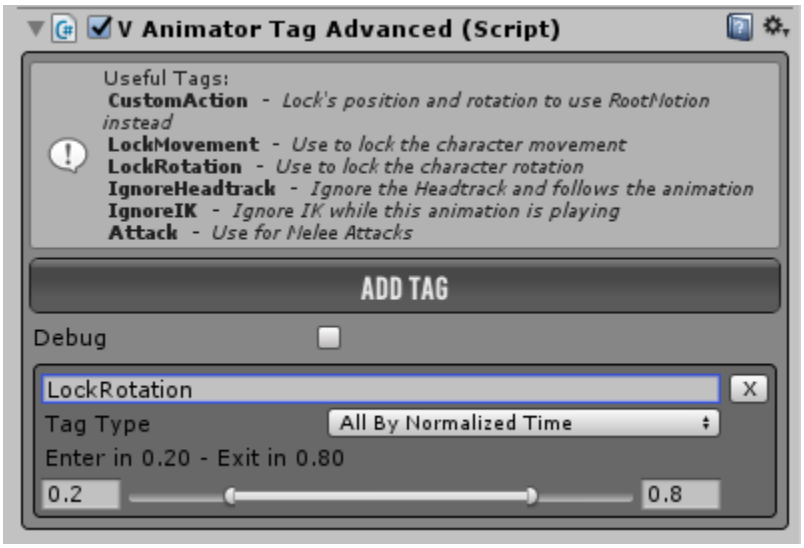
This is an Animator Behavior that you can attach directly on Animation State inside the AnimatorController, it's useful to know what animation is being played and what you can do while this animation is playing.

We have a few useful built in tags such as the image below in the infobox, but you can also create your own tag and verify in code, for example:

First access the **vThirdPersonController** via script so you can call the **method**

```
if(tpController.IsAnimatorTag("MyCustomTag")
{
        //do stuff
}
```

We also have an **AnimatorTagAdvanced** in case you want to check a tag but only during a certain period of the animation, every animation goes from 0 to 1 and you can filter the tag to only run your method during this time.



ANIMATOR EVENT MESSAGE/RECEIVER

Send a message from a AnimationState to any GameObject:
https://www.youtube.com/watch?v=uZn53kKsI0I

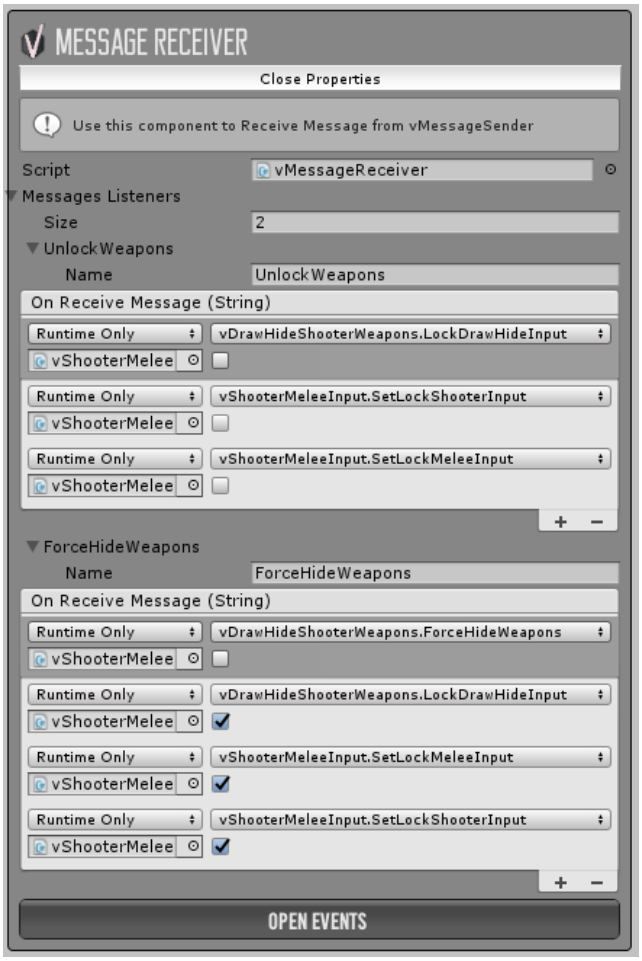
MESSAGE SENDER/RECEIVER

With this component you can create Custom Messages and call several public methods using Events and trigger them at any time by using a trigger for example.

In this example we have 2 Message Listeners:

- ForceHideWeapons will lock the input of drawing weapons and call the methods to lock the Shooter and Melee input, it's useful for example when entering a npc area where you cannot attack anyone.

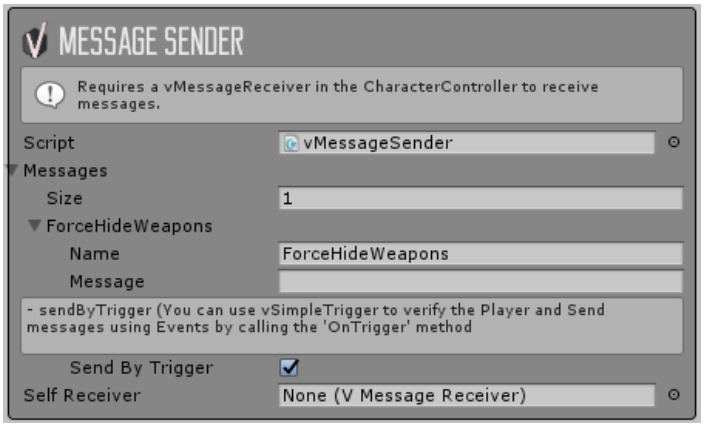
- UnlockWeapons were we call the method to unlock the hide weapons input and unlock the shooter and melee input.

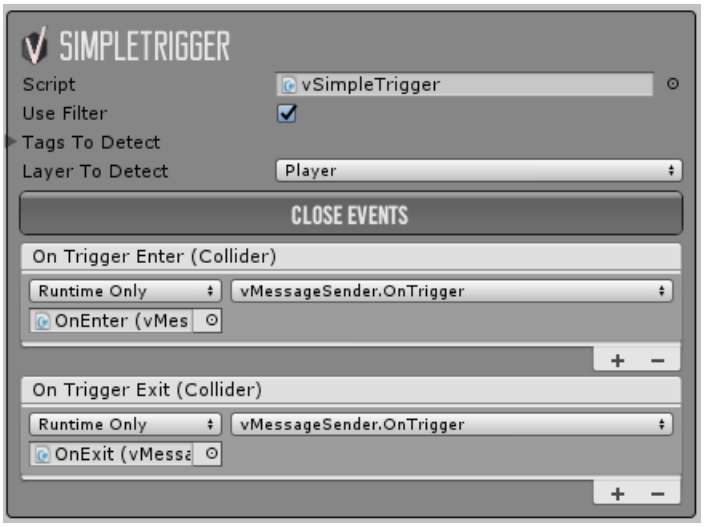


This is just one example, you can create any listener and call any public method you need.

Now to trigger those combined events of the MessageReceiver, we will use the vMessageSender to send the message and the vSimpleTrigger to detect collision with the Player.

We created 2 empty gameObject called OnEnter and OnExit, add the vMessageSender and create the Message ForceHideWeapons and UnlockWeapons, we also checked the option SendByTrigger since we're going to use the vSimpleTrigger to detect the player.





Now in the vSimpleTrigger we can use the option Use Filter to only detect the Tag and Layer "Player", create a box collider or mesh collider with the option IsTrigger checked and the layer "Triggers" that matches the size you want and call the method "vMessageSender.OnTrigger" for each Event OnTriggerEnter and OnTriggerExit.

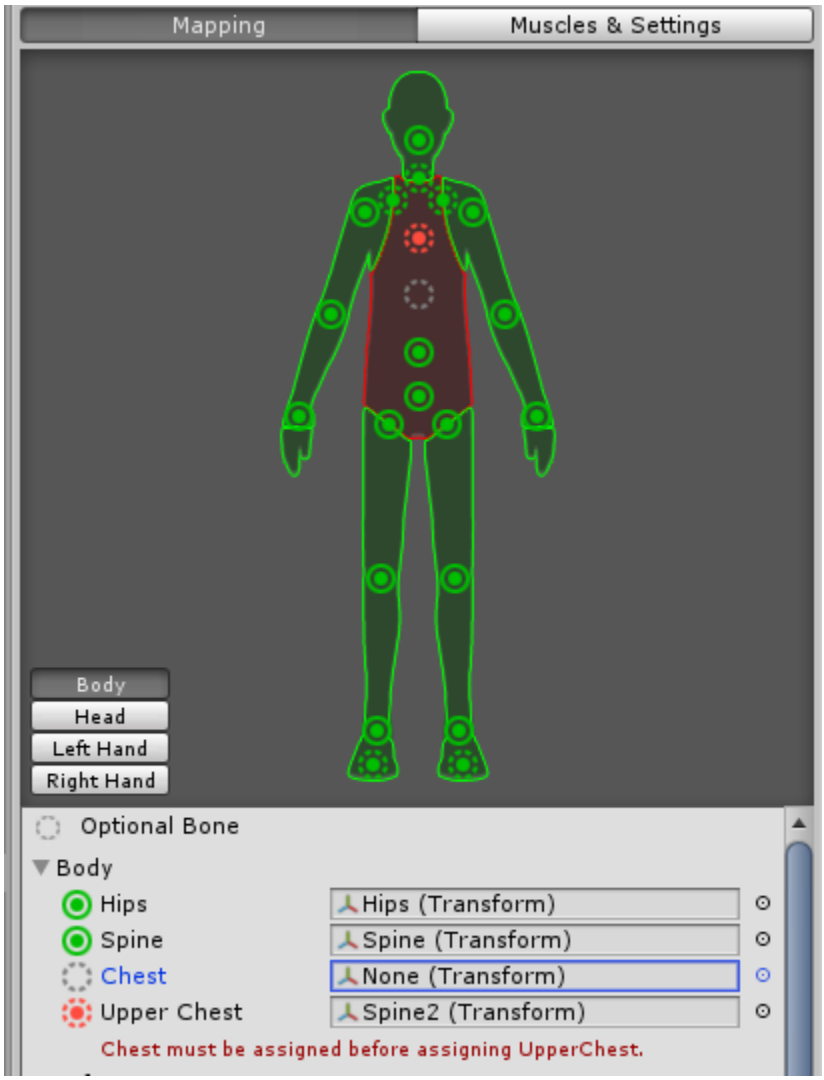
BODY SNAPPING ATTACHMENTS

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

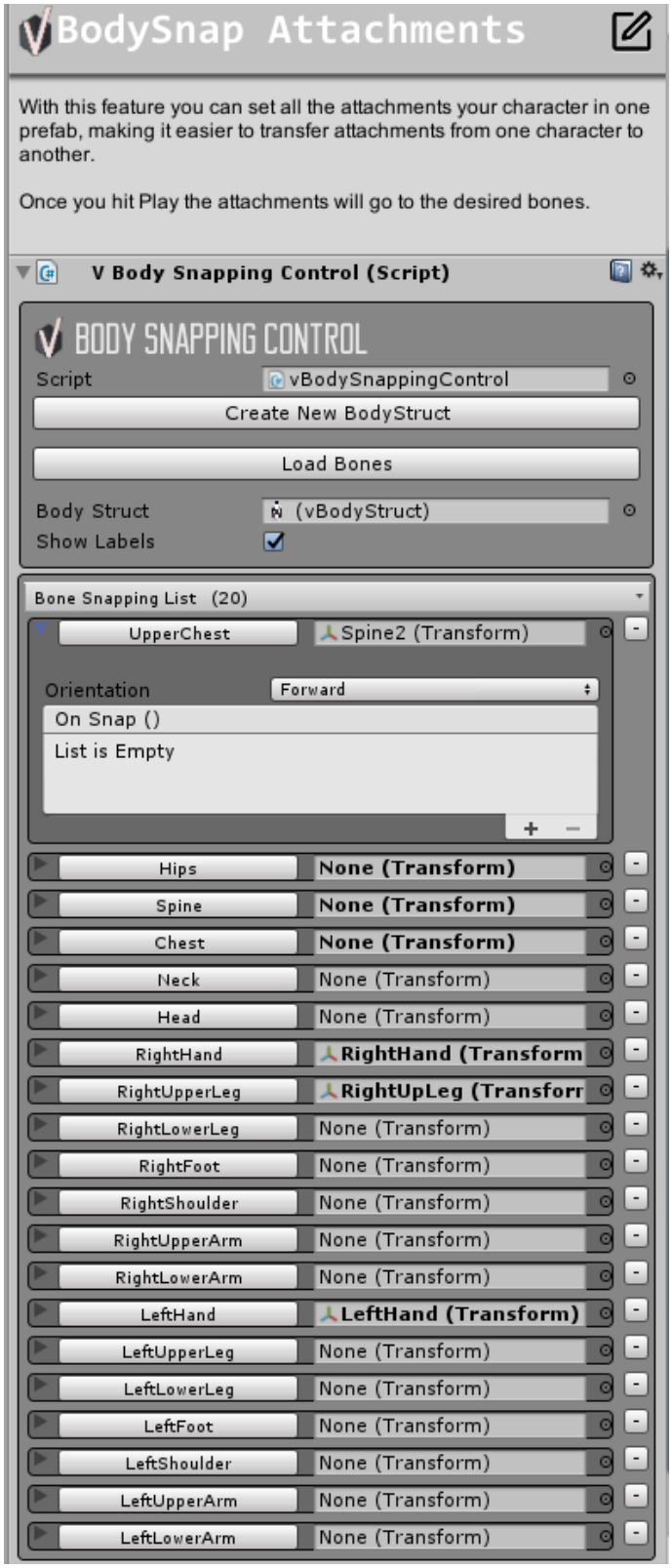
This means that you can create a Prefab of a Character Attachments and quickly add to another character, without the need of adding attachments one by one on each bone.

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:


Mapping
Muscles & Settings
Body
Head
Left Hand
Right Hand
Optional Bone
Body
Hips
Hips (Transform)
Spine
Spine (Transform)
Chest
None (Transform)
Upper Chest
Spine2 (Transform)
Chest must be assigned before assigning UpperChest.

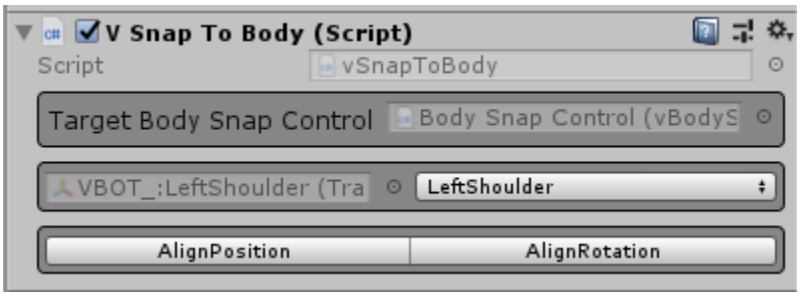
Now going back to our Character BodySnap Control, you can add all your character attachments such as particles that activated on a specific bone, itemManager Handles, anything that you may use and assigned to a specific bone, once you hit Play that GameObject will be attached to the bone you assigned.



SNAP TO BODY

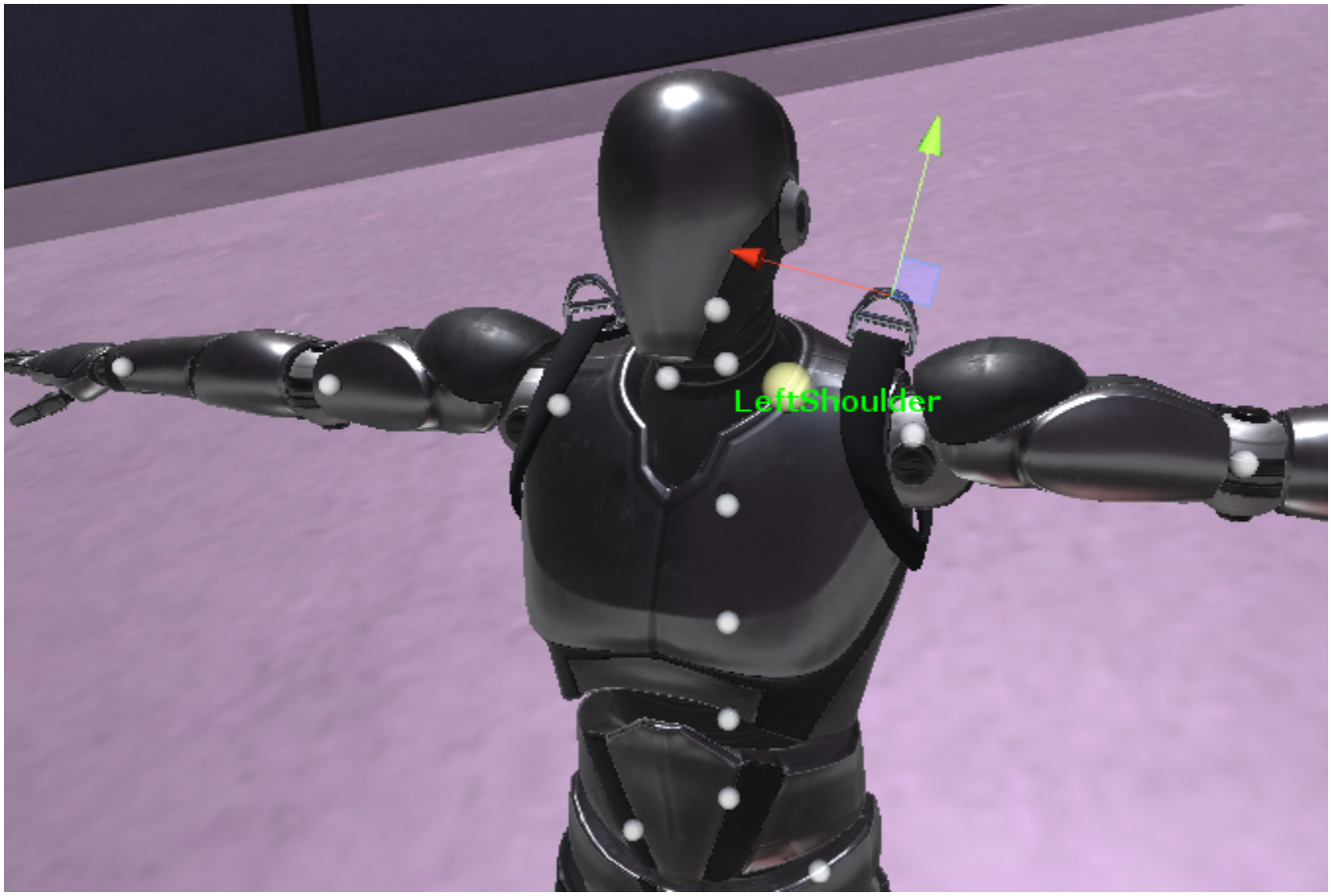
**Use this component with a vBodySnappingControl*

Attach this component to any object to make it snap to a specific bone when you hit Start.



For example, instead of manually drag and drop objects like Armour, Helmet, Capes, etc… to each bone of your character, you can create a Prefab called “Attachments” and add ‘SnapToBody’ on each object and assign where this object must be attached.

This makes the workflow much easier when replacing the character model for example.

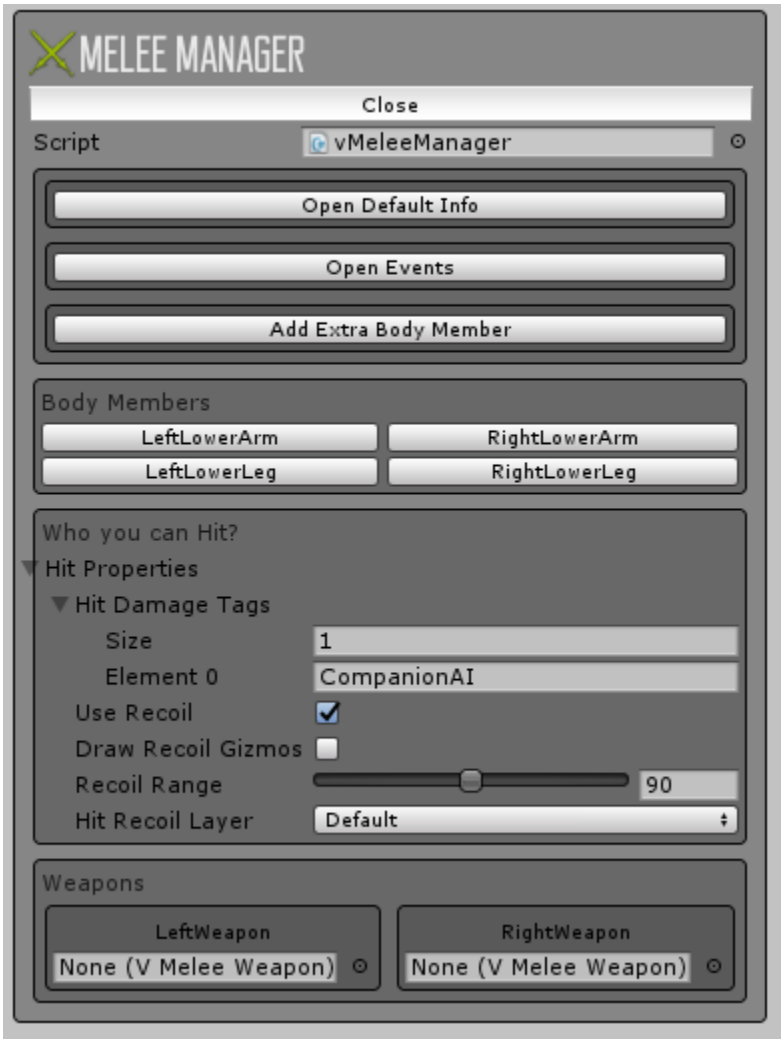


-MELEE FEATURES-

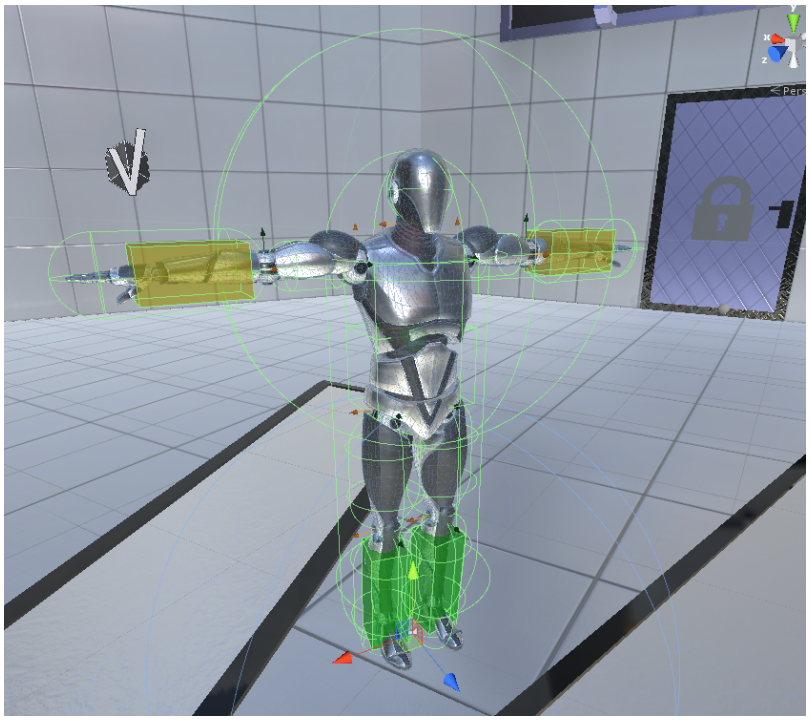
MELEE MANAGER

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.

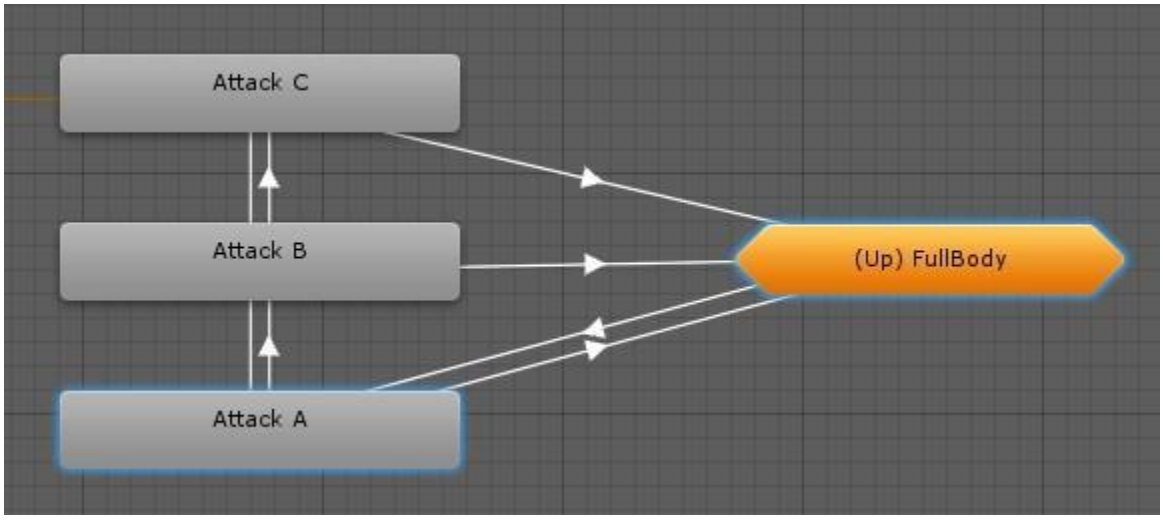


The animations for the hand to hand combat can be set up in the **Unarmed** state machine, triggered by the **ATK_ID 0** and the defense **DEF_ID 1** on the UpperBody Layer**, Default Defense.**

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 needs to have a **vMeleeAttackBehaviour** script attached.

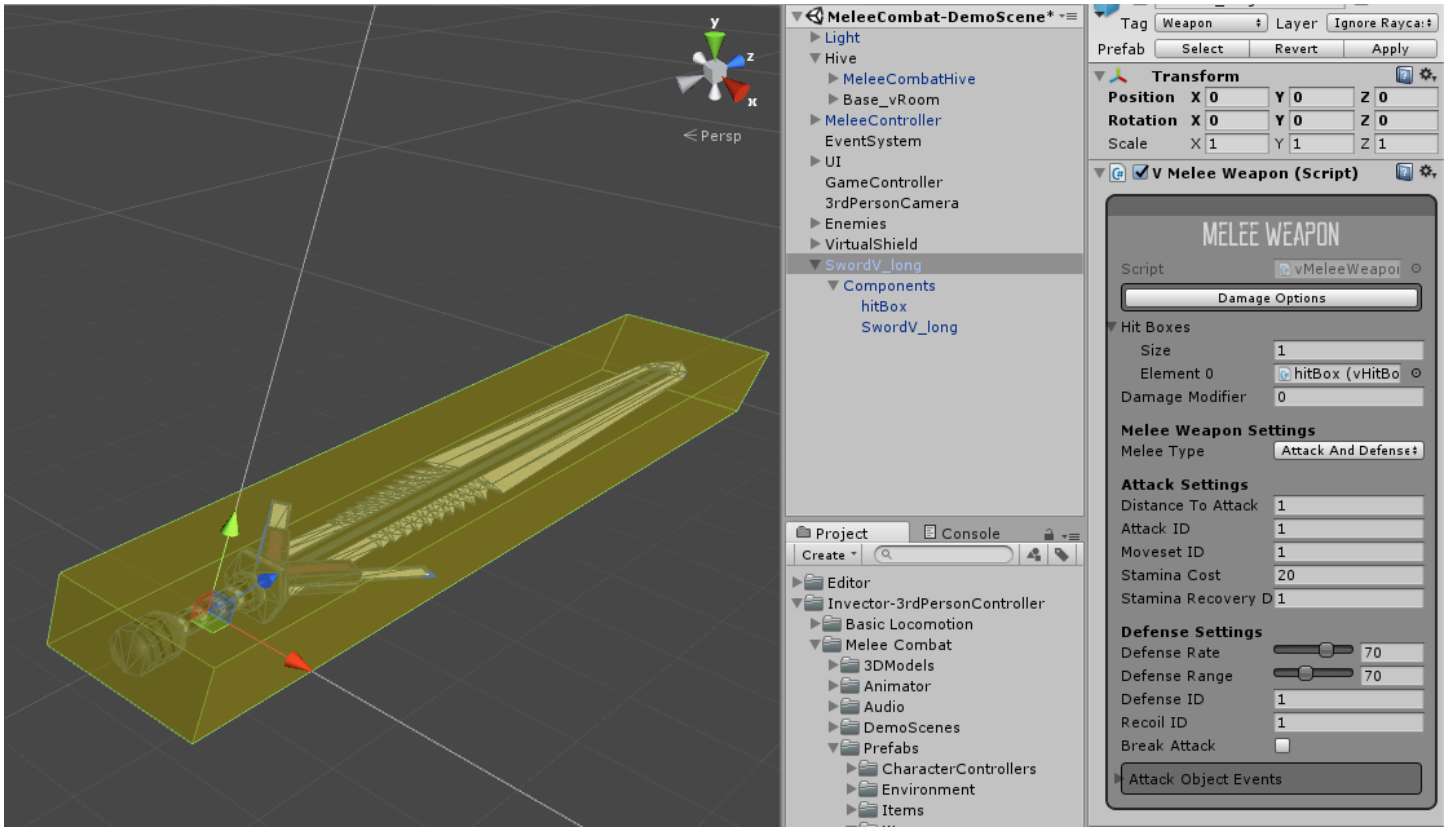
StartDamage > Time of the animation that you will apply damage
End Damage > Time of the animation that will stop trying to apply damage
~~**Allow Movement At:** free your character rotation during the attack animation~~
*Removed on update 2.4.2 use the **AnimatorTagAdvanced** *LockMovement* and *LockRotation* 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

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 Invector > 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.

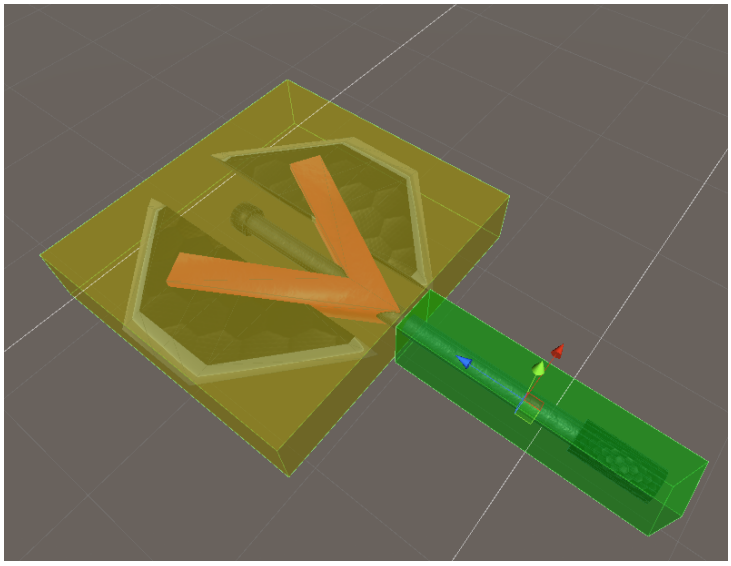
IMPORTANT - don't forget to set your **Weapon Layer to Ignore Raycast and the Tag to Weapon**, if you put a weapon into an Enemy or Player and change the Layer and children's, you need to set the weapon layer to Triggers again

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 an ItemManager assigned 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

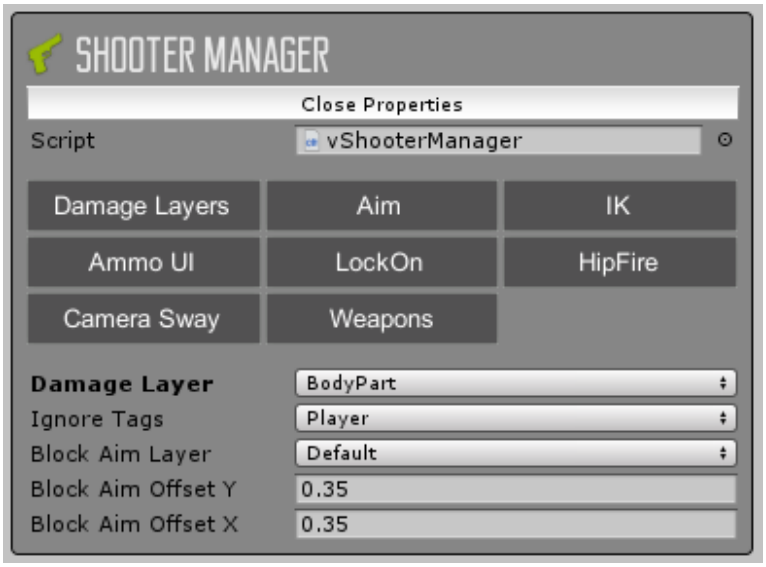
HOW TO APPLY DAMAGE TO A TARGET

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.

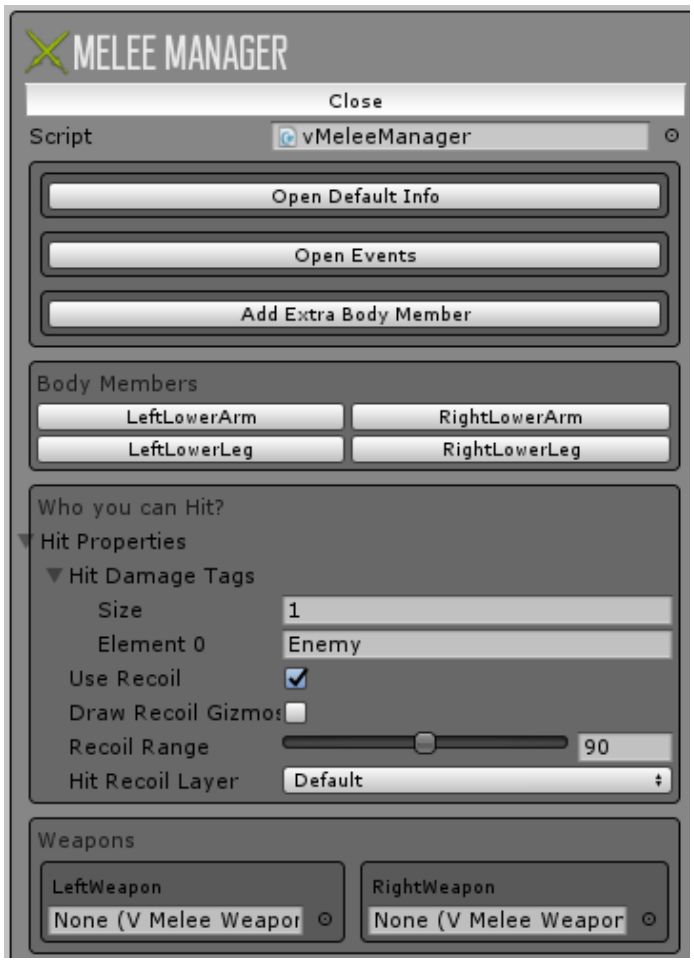
Shooter: You need to set the DamageLayer of your target in the ShooterManager.

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.

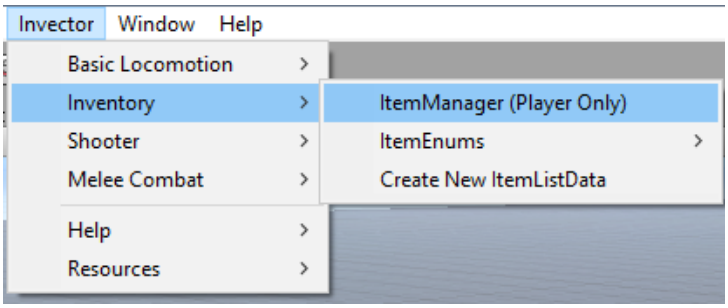


MeleeCombat: You need to set the Tag of your Target in the MeleeManager / HitDamageTags field, you can assign several tags to hit different targets.



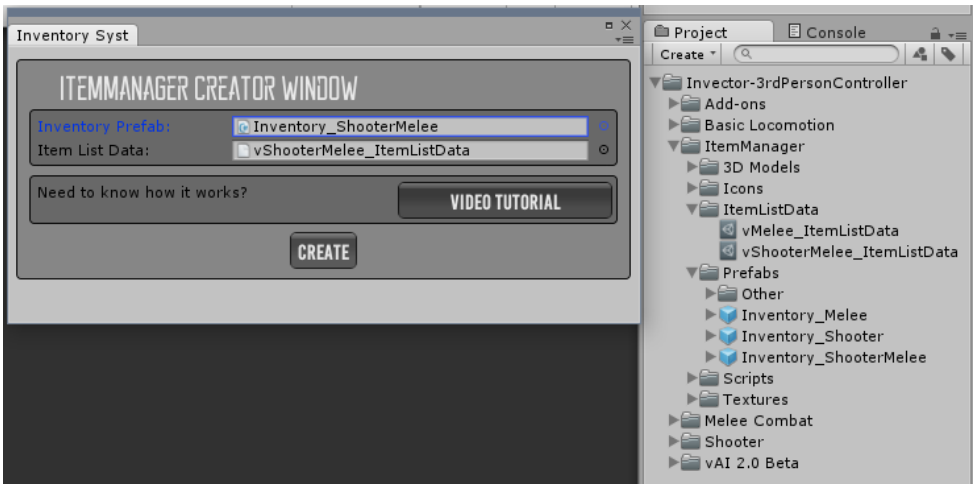
ITEM MANAGER

- Add the ItemManager into your Player from the menu **Invector > 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**

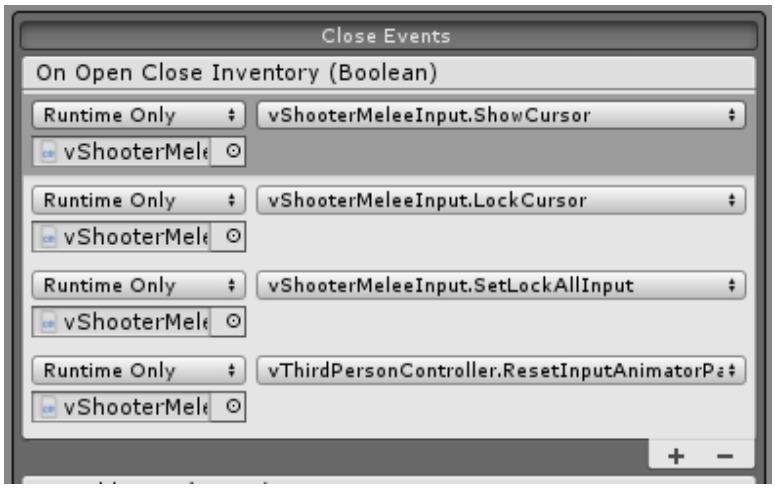


- You can also add the ItemManager manually using the Add Component button via inspector, but don't forget to assign an ItemListData.

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:

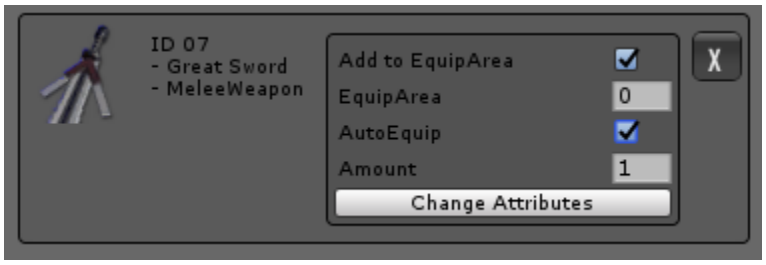
- Hide/Show the mouse cursor
- Lock/Unlock the cursor at the screen center
- SetLockAllInput to lock all the input from the controller (basic, melee and shooter)
- ResetInputAnimatorParameters to reset back to zero the animator parameters, useful when using the CharacterCameraPreview inside the Inventory,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



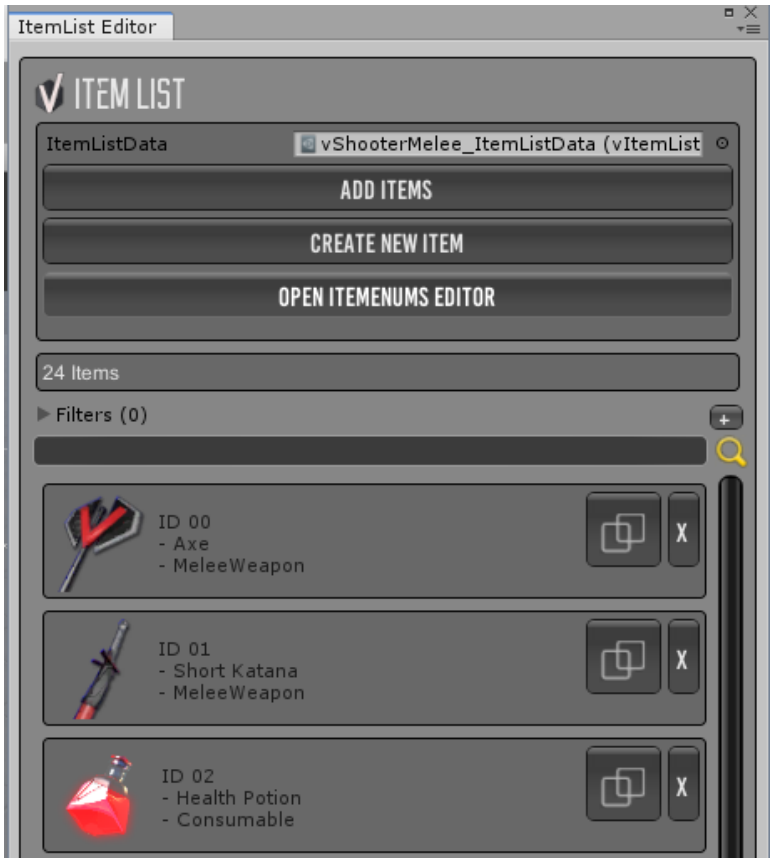
Click at the Add Item button to open the Filter, so you can search or filter exactly what ItemType you need: You can add multiple ItemTypes to filter multiple items.

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.

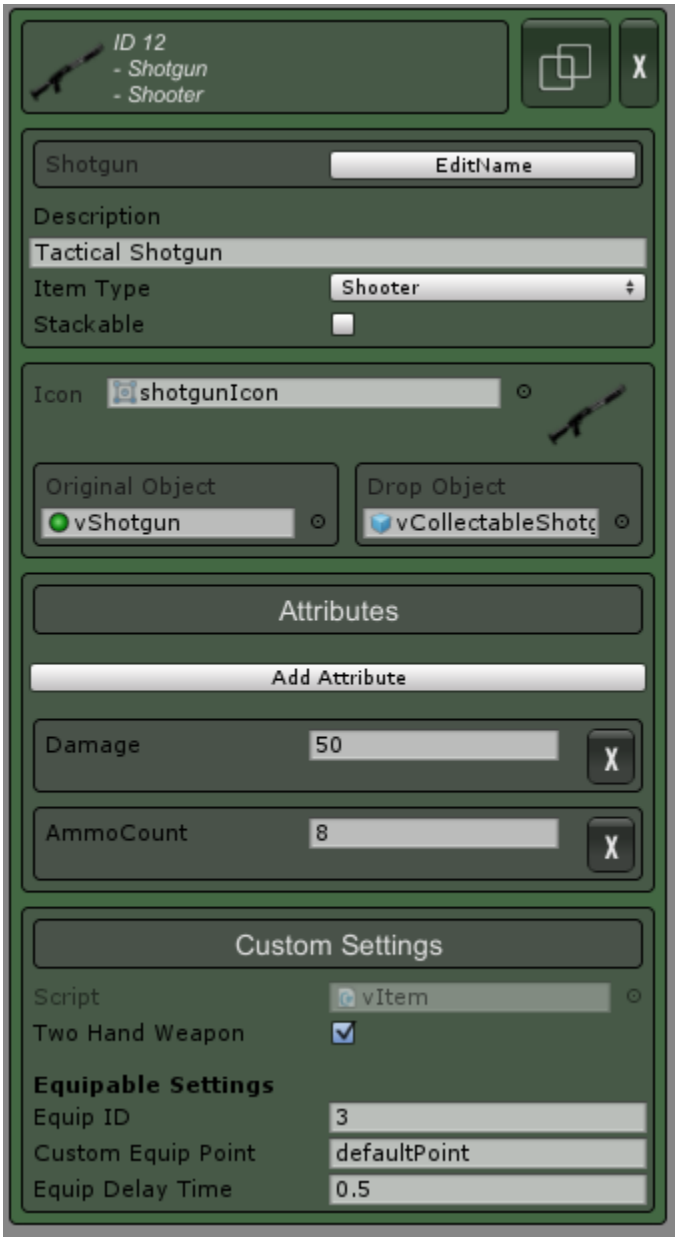


Click in the **Open Item List** button, to manage, search or create new items



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.



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.

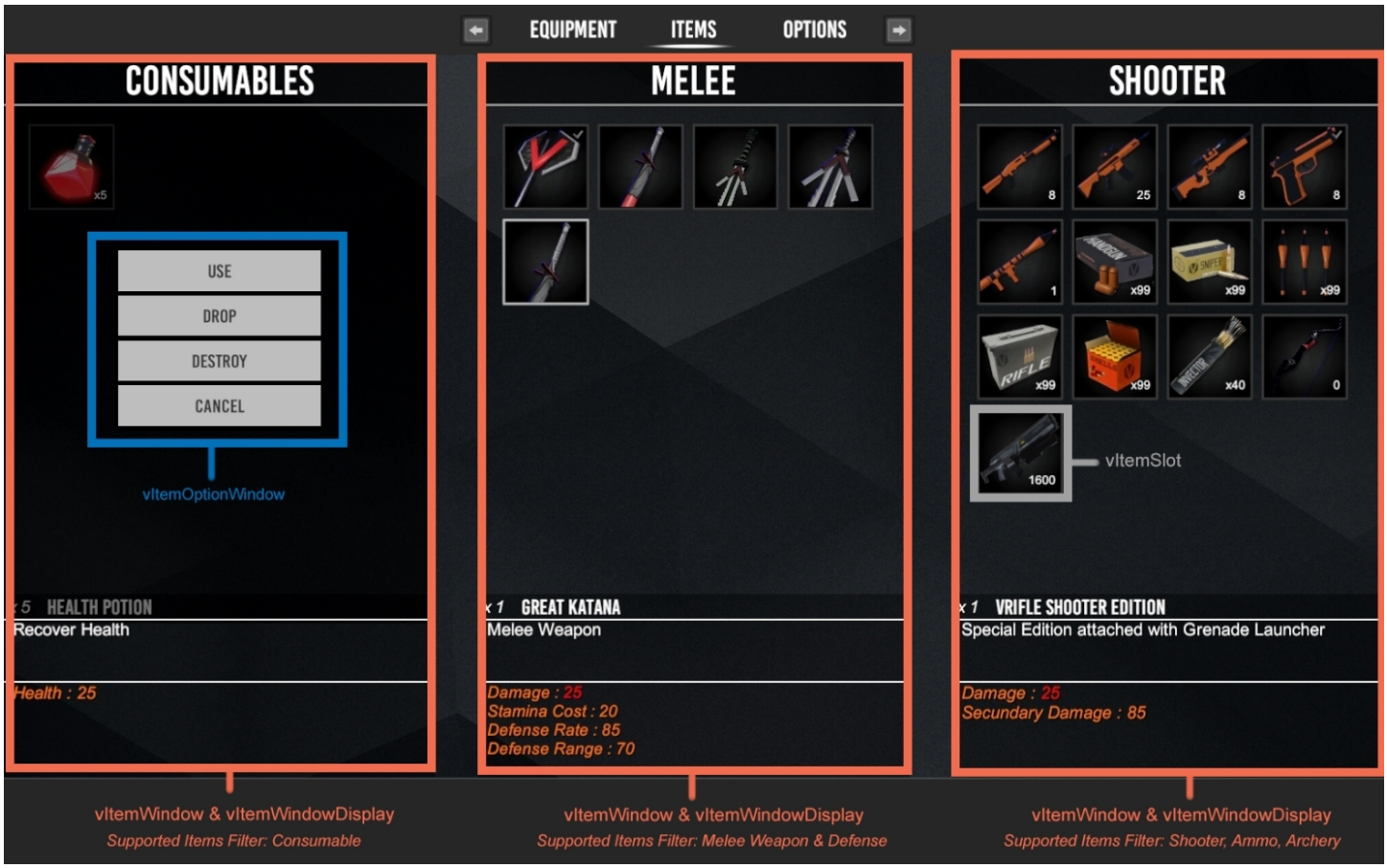
INVENTORY

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'

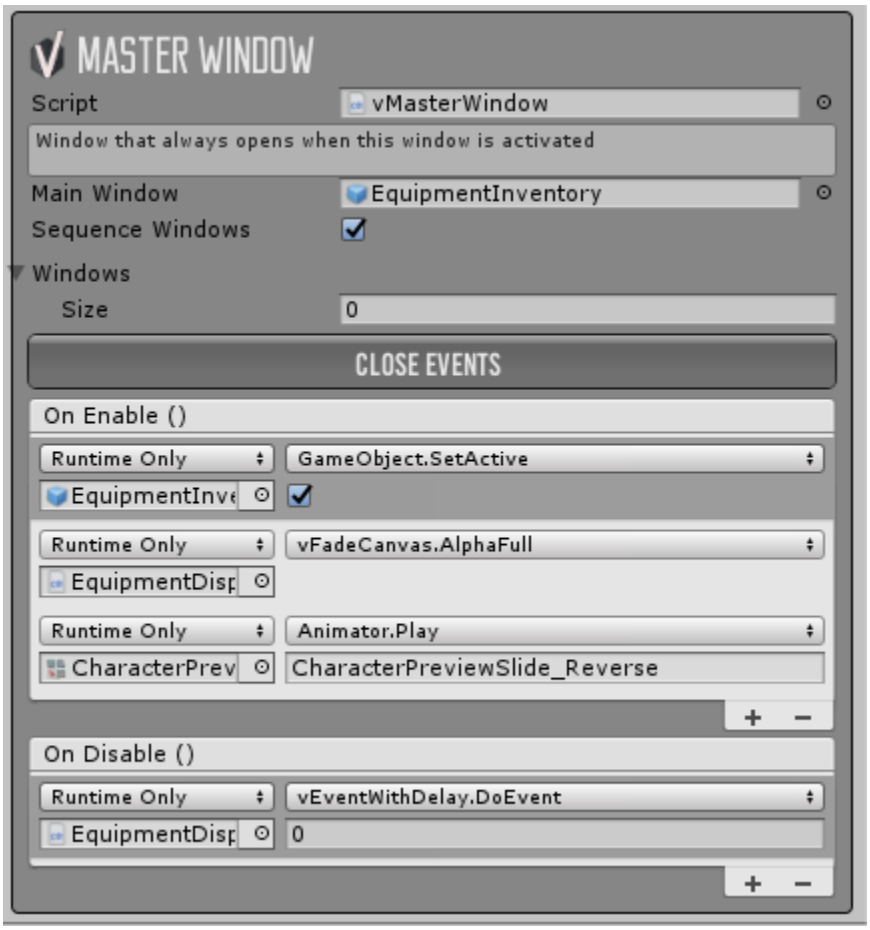
This is an example of a window that displays several different Items based on their Types.



- 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

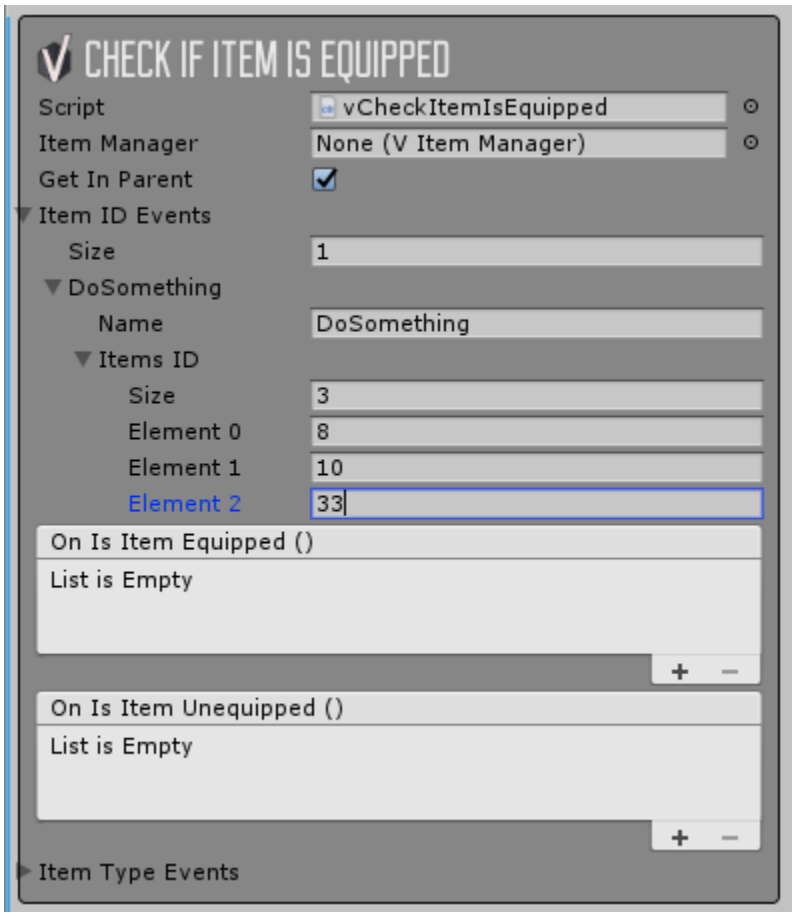
You can see a **MasterWindow** component on each main window of the Inventory, you can use the MasterWindow to manage your windows for example if you open a window and open another, you can press the back button to close the current window and get back to the previous, it's basically a window manager.

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 *vCheckItemIsEquipped* to check if a specific Item or ItemType is currently equipped.



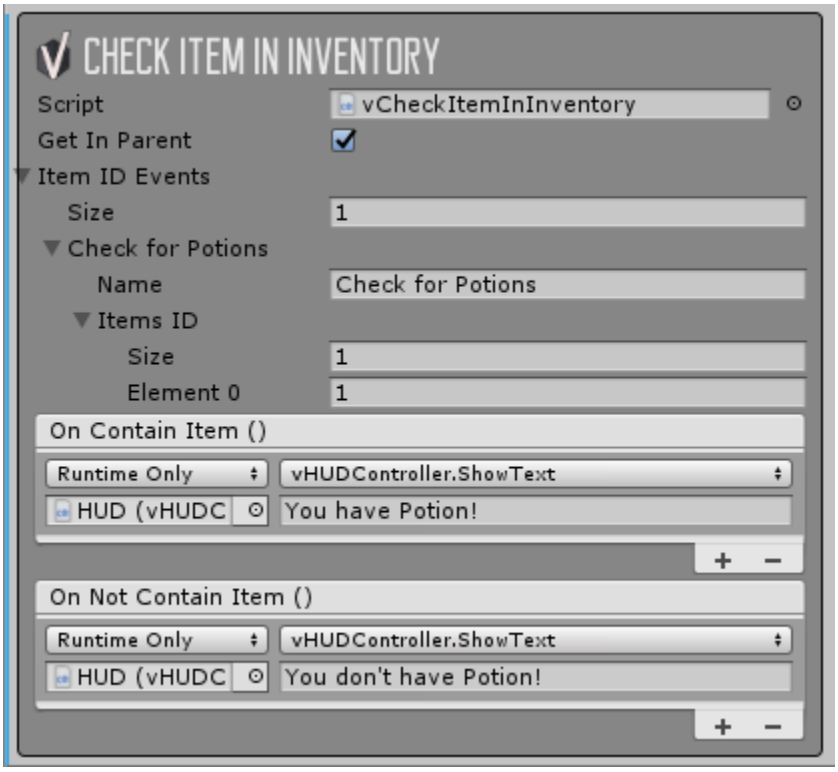
You can attach this component directly to the character or inside the character (check the option “GetInParent” if it’s inside the Player).

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.

CHECK ITEM IN INVENTORY

With this component you can check if you currently have or not a specific item(s) in your Inventory and use Events to do something if true or false.



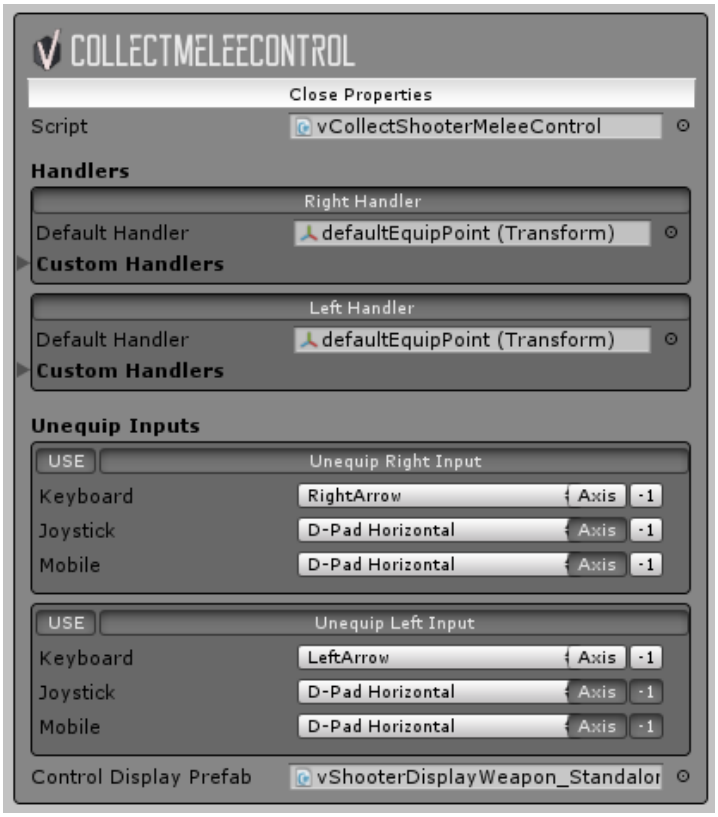
**Element 0 = Item ID*

COLLECTABLE STANDALONE (NO INVENTORY)

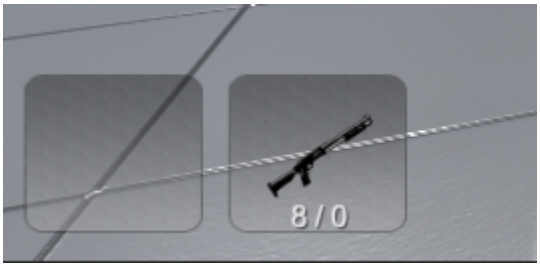
If you don't want to use the ItemManager to manage your items, we have another solution for 'on demand' collectibles, notice that you can only equip 1 item, once you try to equip another the current item will drop.

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

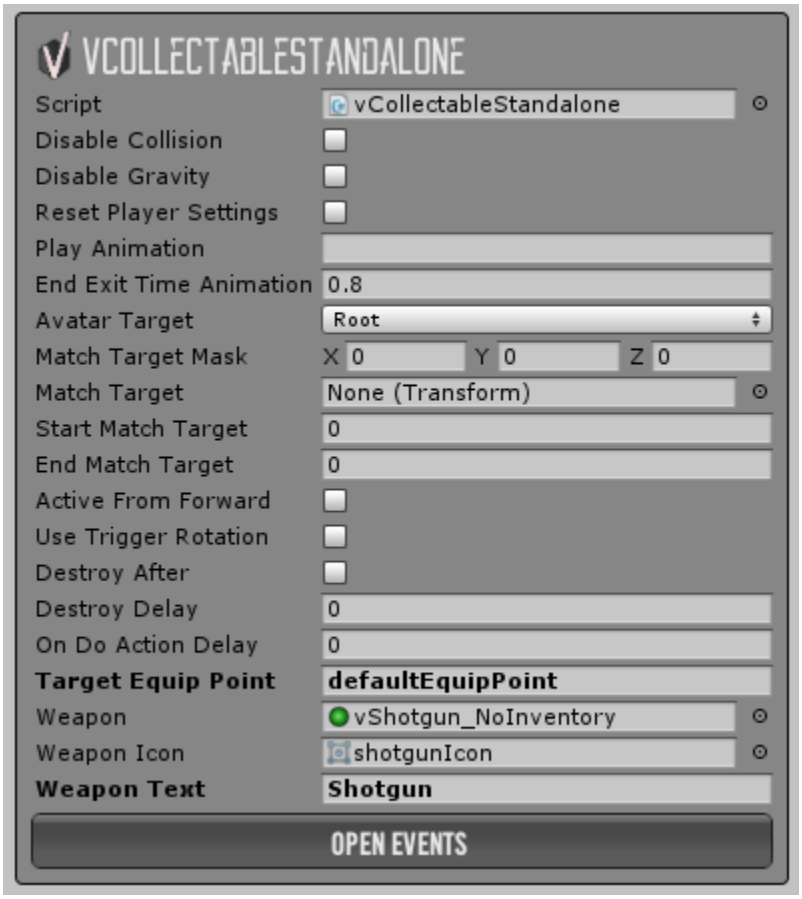
You need to create the defaultEquipPoint to equip weapons and assign inputs to drop them.



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.



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.

CREATING A ENEMY AI

Invector > 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.

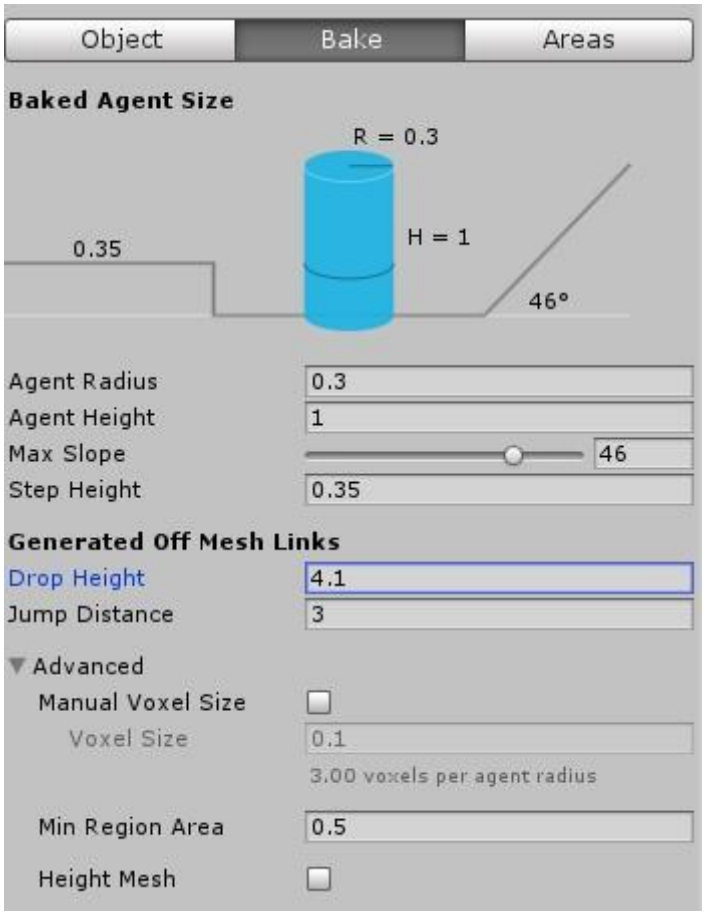
Locomotion - It works pretty much the same as the Character Controller, you still need to set up the

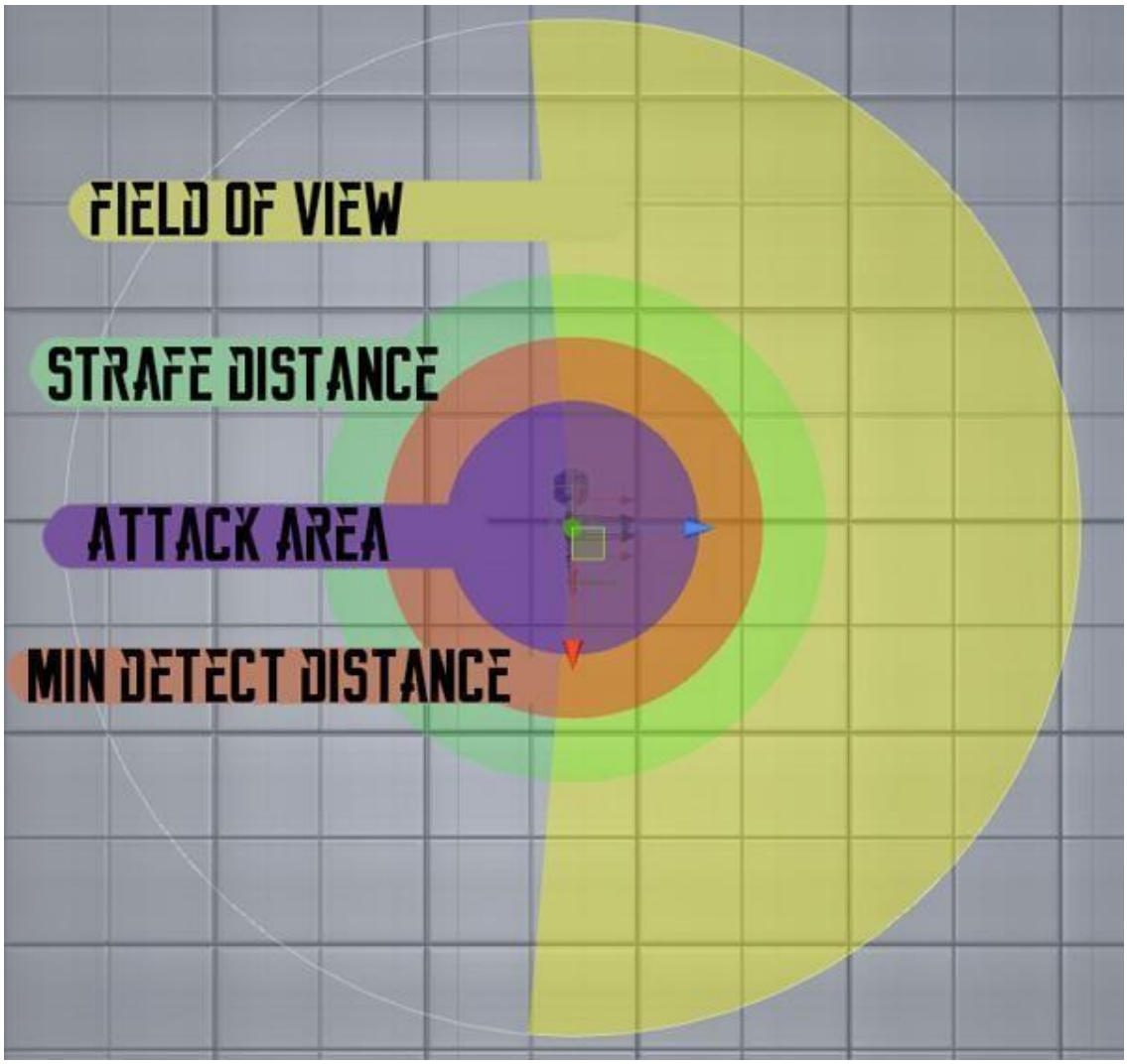
Layers – just like the Player, you need to set up a Layer for the AI (Enemy) and a layer for the Ground (Default). If you equip a Weapon on the character, don't forget to assign the Layer Triggers for this weapon.

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 setup, but of course this will depend on your scene, terrain, meshs, etc…





Field of View > total range to detect the Player

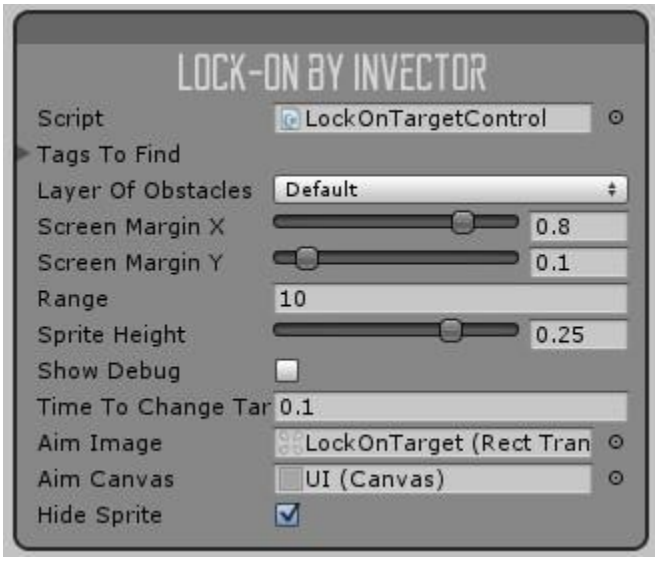
Strafe Distance > once in combat, the character will move Strafing

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 activates 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**.

You can use the LockOn with **any target** as long as you add a **vHealthController** to this target, so that the LockOn can identify whether this target is alive or dead.

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 as many AI as you want, and limit the area / limit of AI that will access.

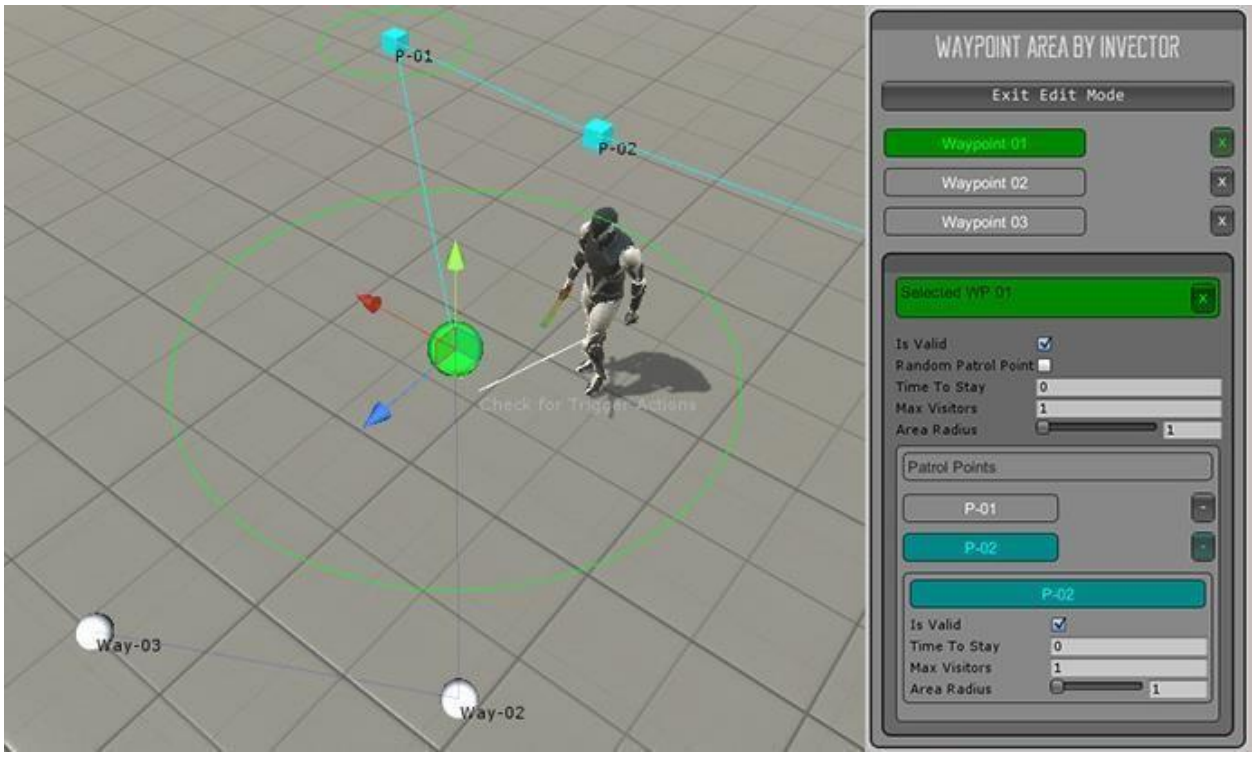
Patrol Points are points of interest that one waypoint has, for example if you have a corridor with 3 rooms, you can create 1 waypoint in the middle of the corridor and 3 patrol points with **Max Visitors of 1**, this means that if an AI is already on a room, the other AI will not come to the same room,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

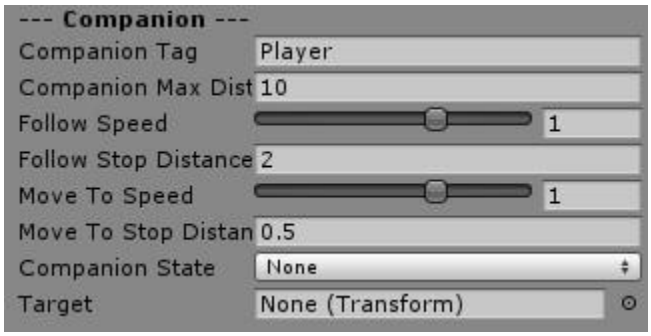
Waypoints are represented by Spheres and Patrol Points are represented by Cubes.



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:

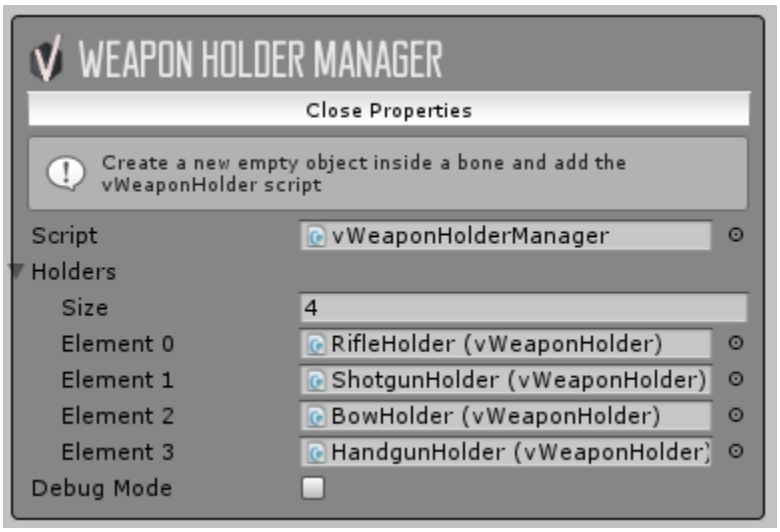
- Stay
- Follow
- Aggressive/Passive
- Move to (moveToTarget)

*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.

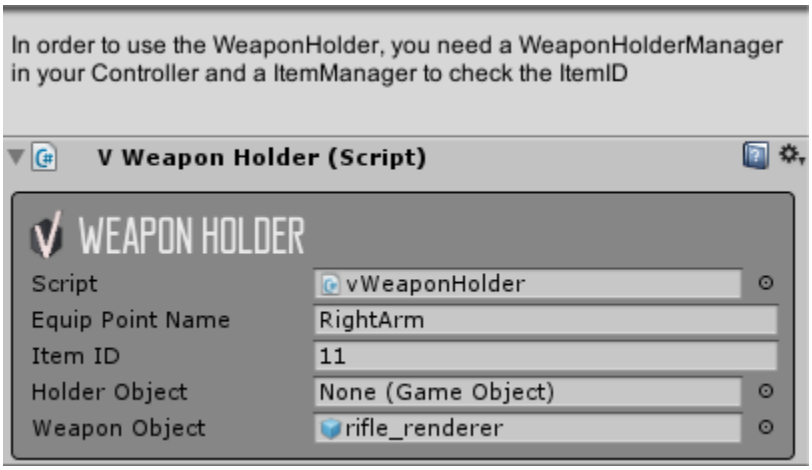
WEAPON HOLDER MANAGER

*This feature requires a ItemManager

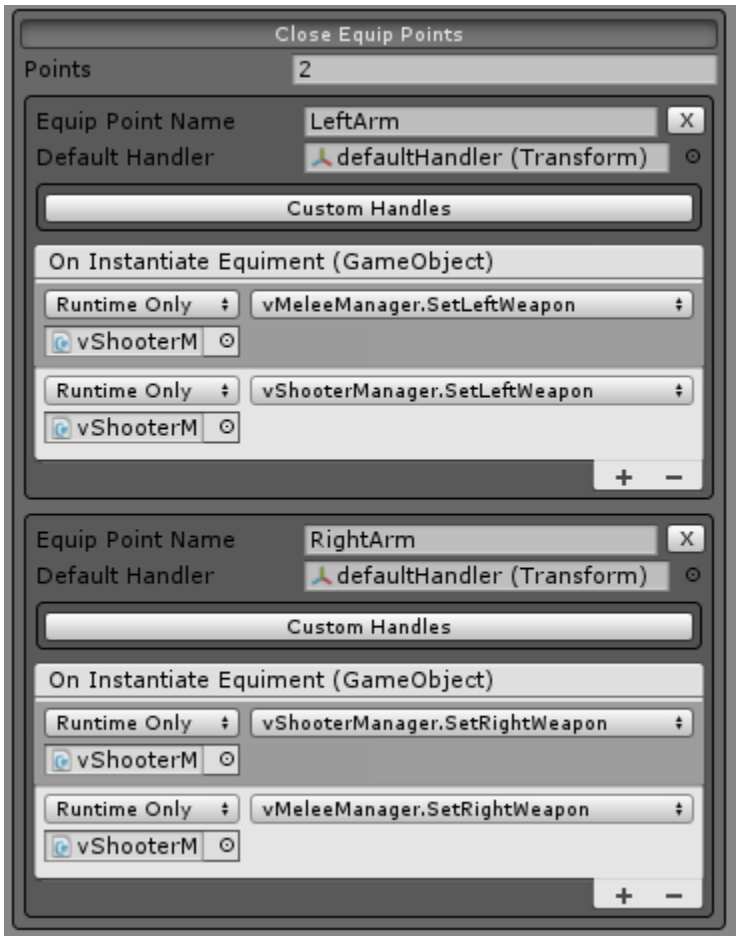
Add this component to your Shooter Controller and there is no need to setup anything here:



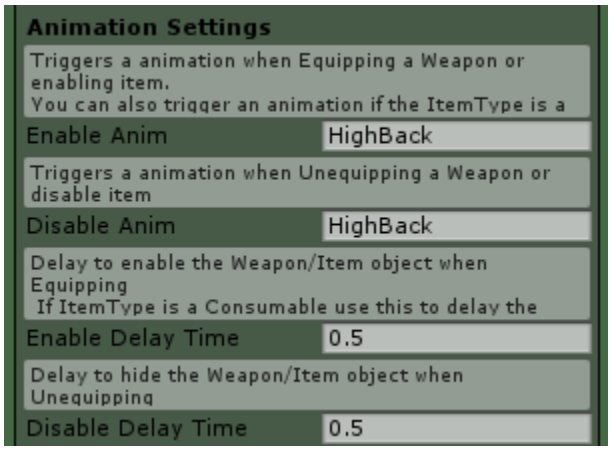
Now go inside your controller, create an empty gameObject and add the vWeaponHolder component.



The **EquipPointName** can be found in the **ItemManager** > EquipPoints.
By default it comes with 2 EquipPoins, LeftArm and RightArm but you also create your own custom equipPoints.



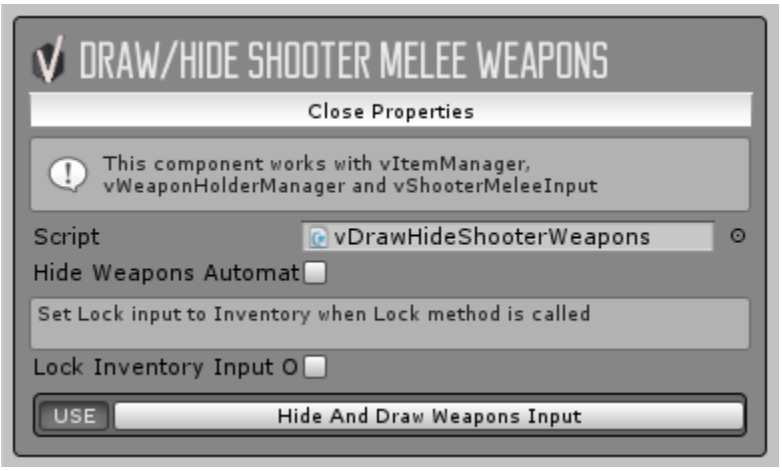
The **ItemID** can be found in the ItemManager **ItemListData**, there you can also set what animation will be played when you equip/unequip your weapon.



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 an 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 it will hide or you can leave it unchecked and use an Input to draw/hide the weapon.

You can also call the methods HideWeapons and DrawWeapons on events, for example, in the if you want to hide your weapons when performing a GenericAction, go to the GenericAction component and call the method to HideWeapons in the OnStartAction Event.

- SHOOTER FEATURES -

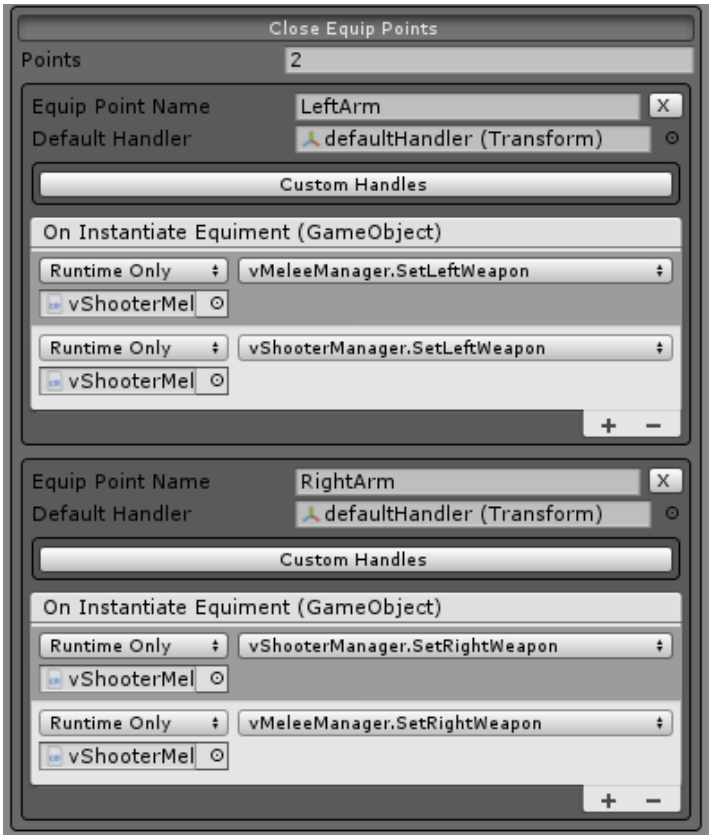
HOW TO ALIGN A SHOOTER WEAPON (RIGHT HAND)

Also available in video tutorial: https://youtu.be/XdEzxhblqS8

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

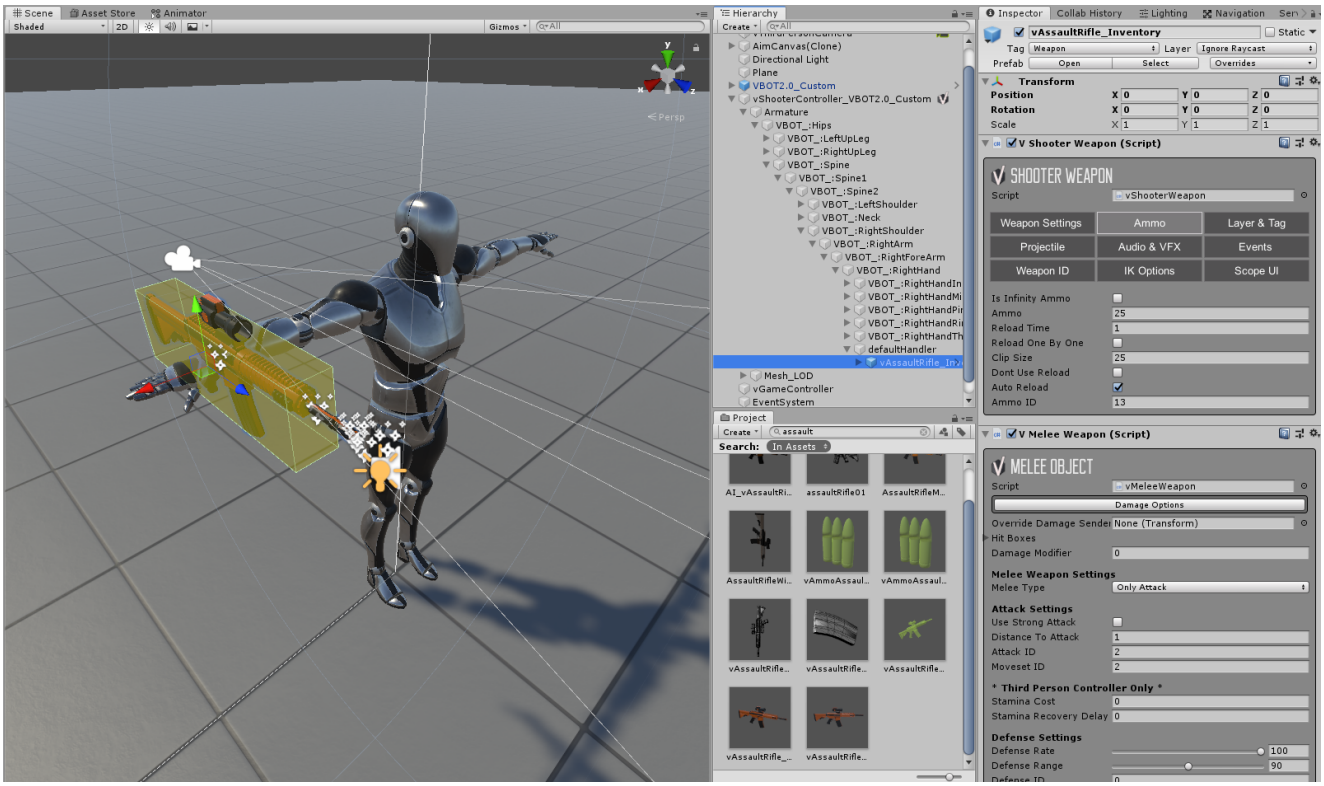
A **Handler** is basically an empty transform located inside your character's left and right hands, you will first align this handler into a position where a weapon looks correct in the character's hand and all weapons will be instantiated there when you pick up the collectible.

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 Equip Points"** tab.

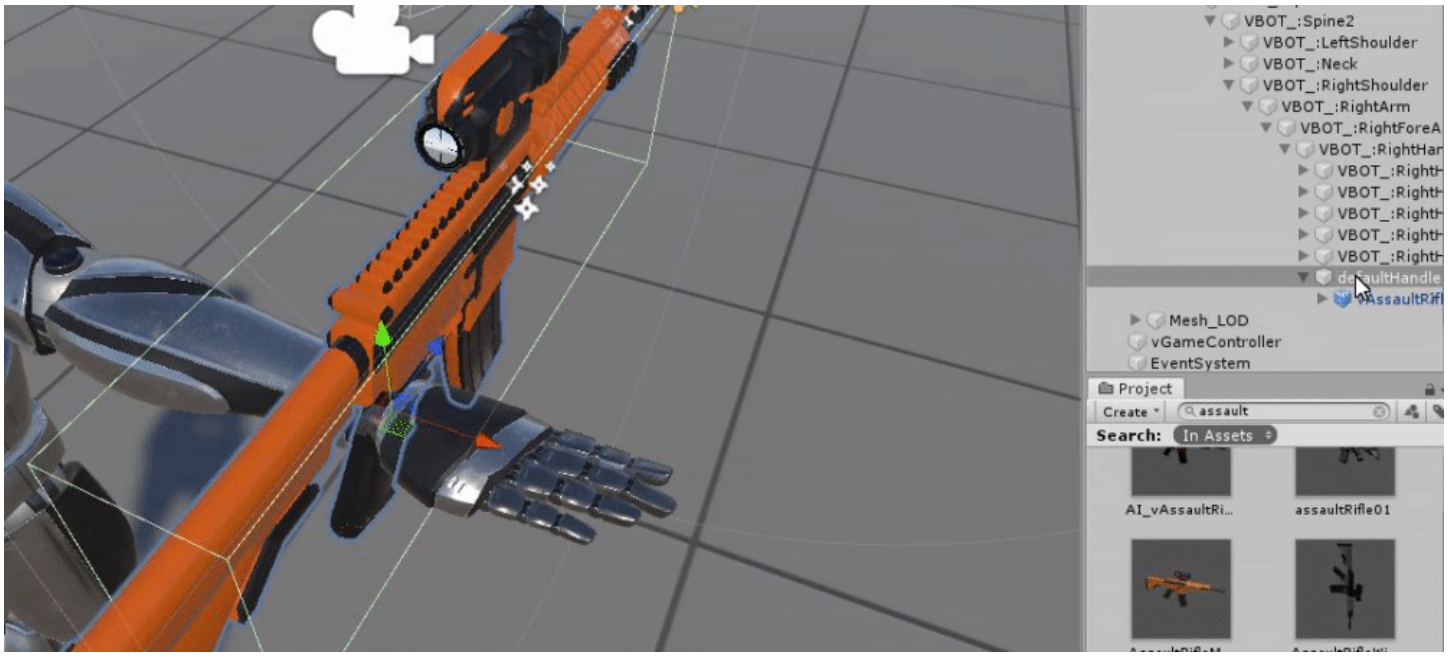


Shooter Weapons will be equipped in the **RightArm defaultHandler**, so select this transform in the hierarchy and drag and drop any Weapon Prefab inside the defaultHandler.

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.



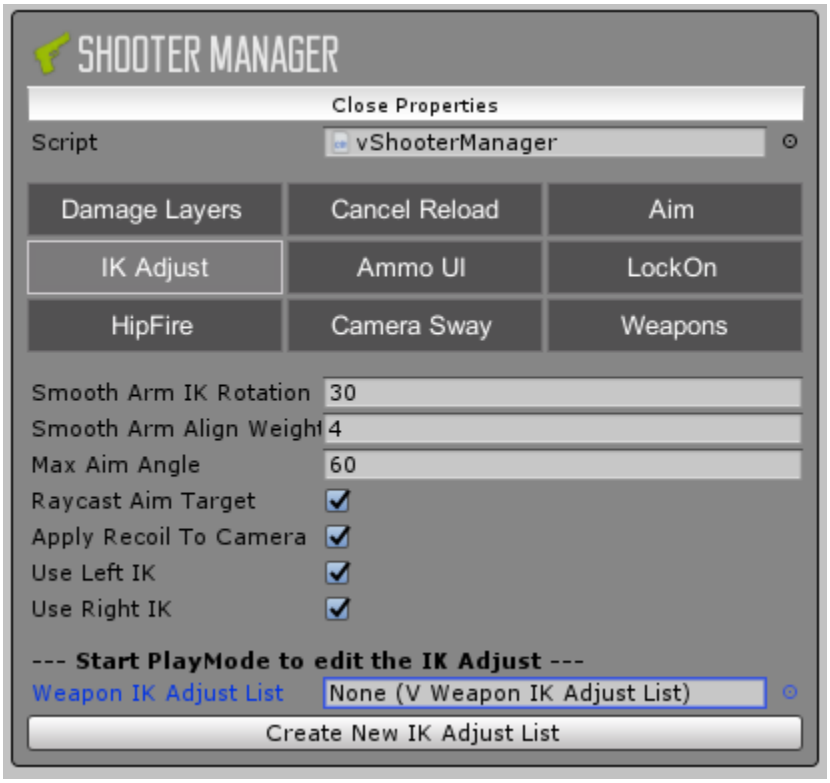
You can now hit **Play** to see if the **weapon position in the RightHand looks good**, it doesn't need to look perfect because we're still going to the **IK Adjustments** where we will **align the LeftHandIK.**

HOW TO ALIGN THE LEFT HAND IK FOR ALL WEAPONS

Also available in video tutorial: https://youtu.be/XdEzxhblqS8

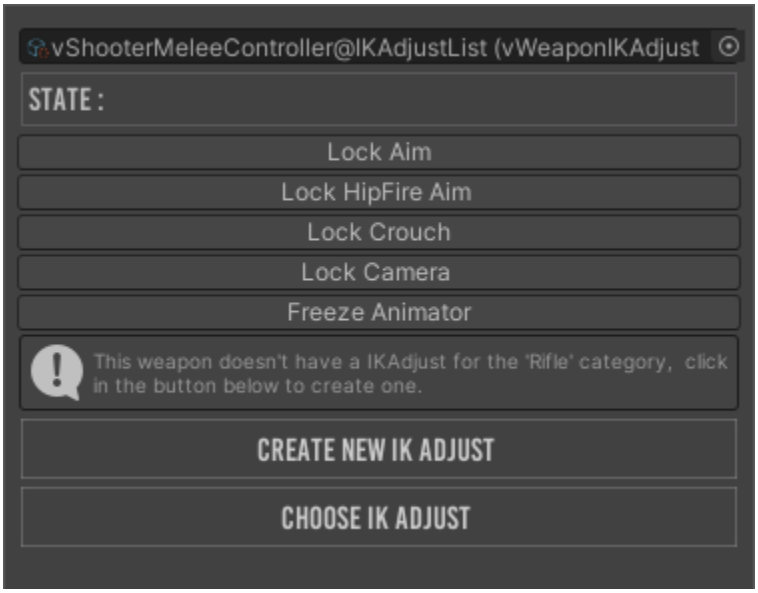
After you create a new character you must properly align the **Handlers** into your character hand to have a first base alignment which can be improved further by using the IK Adjust feature.

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

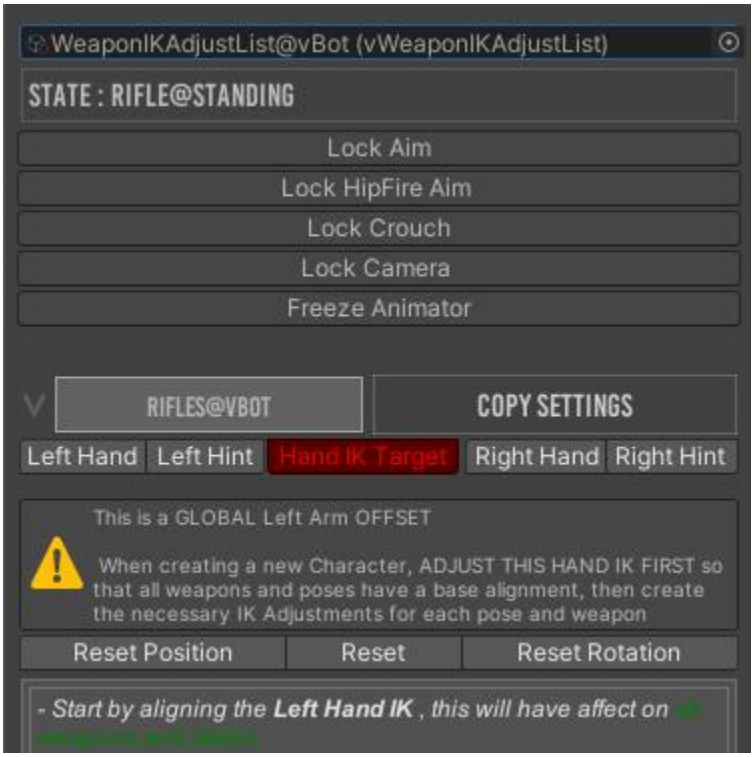


A new IK Adjustment List will be created and automatically assigned for you, a new button called “**Edit IK Adjust List**” will appear, click on it and a new window will open, you can dock this window anywhere you like.

Add any weapon to your Character, Hit **Play** and select the **ShooterController** to see this menu:



Hit **“Create New IK Adjust”** and before we start playing around with the many cool IK options, first you need to **align the Suport Hand IK**, Click on the Red Button “Hand IK Target” and align the red sphere by moving and rotating the transform to be in the position of a suport hand of your weapon



You only need to adjust the **Hand IK Target** once in one weapon, this base initial position will be applied for all weapons and it doesn't need to be perfect because now you can do the fine tuning by selecting the LeftHand or RightHand button depending on what side your weapon is equipped and move/rotate to fit exactly where you need.

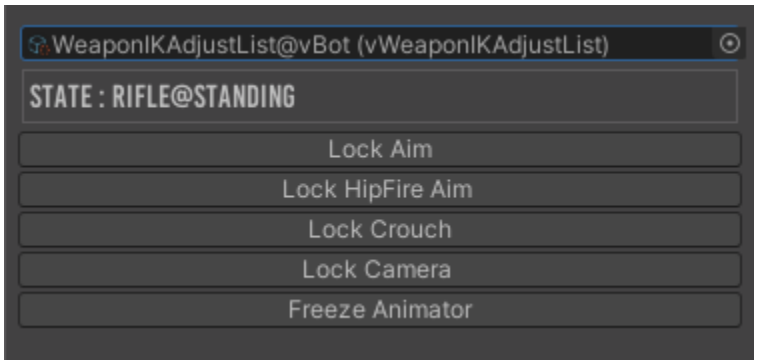
IK ADJUST FOR FINE TUNING OR CREATE NEW POSES

Also available in video tutorial: https://youtu.be/XdEzxhblqS8

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

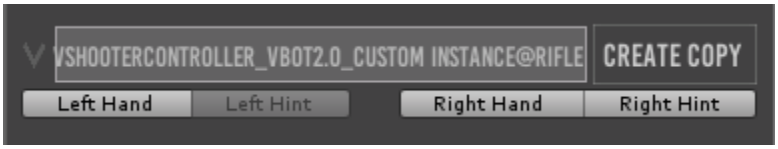
You can switch the state by pressing the buttons to LockAim, LockCrouch and LockCamera.
Start by Locking the Camera so that the Headtrack doesn’t influence your pose.



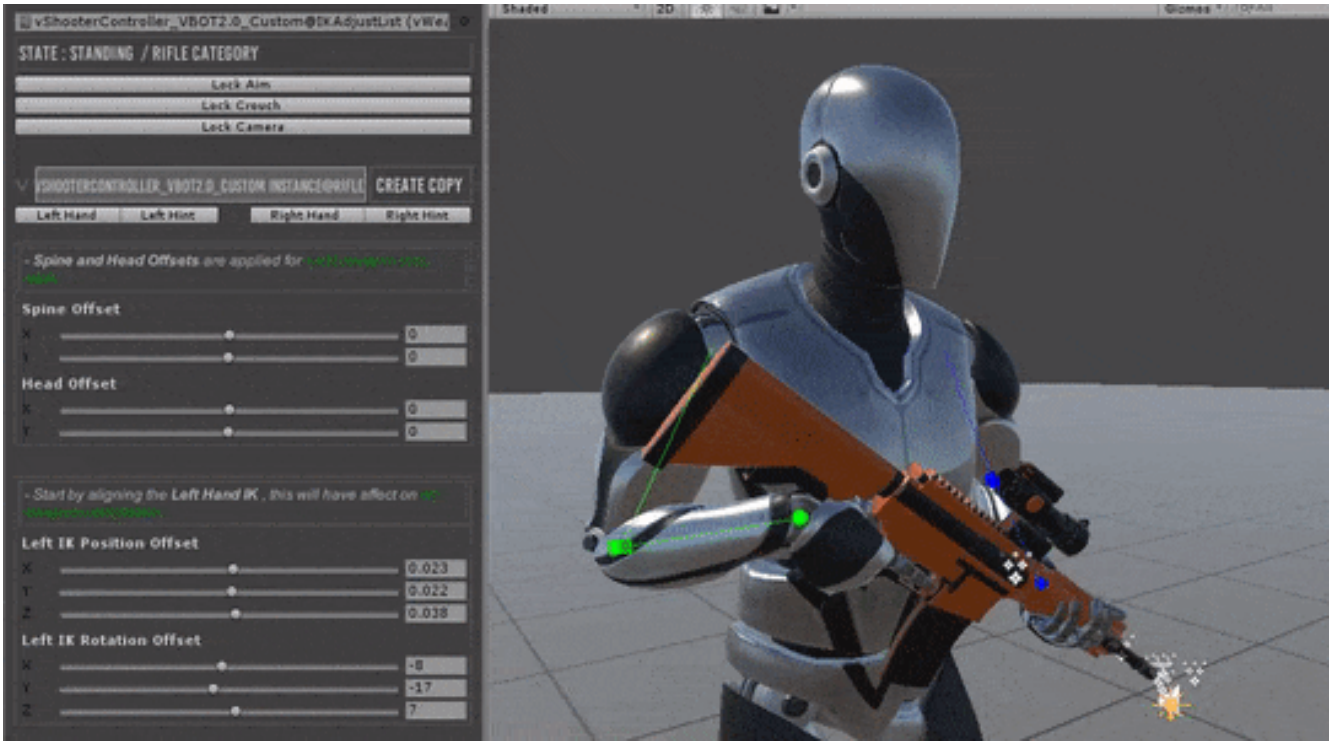
You can create different IKAdjusts for each weapon or use the same adjust if the weapon share the same **Category**, for example, if you have a large number of Pistols in your game that use the same pose, you don’t need to create an IKAdjust for each weapon, just set the same Category for all the ShooterWeapon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

CUSTOM IK ADJUST

You can create a custom IK Adjust for a specific case with the method ***SetCustomIKAdjustState****(myCustomIK)*, for example, if the character is falling and you want to have a Custom IK just for that specific case.

```
public virtual void SetCustomIKAdjustState(string value)
{
        if (!string.IsNullOrEmpty(value)) CustomIKAdjustState = value;
}

public virtual void ResetCustomIKAdjustState()
{
        if (!string.IsNullOrEmpty(CustomIKAdjustState)) CustomIKAdjustState = string.Empty;
}
```

When you call the method, the IK Adjust Window will detect that this method was called and inform you that this IK Adjust doesn't exist yet, just click on Create New Custom and adjust your new pose.

Once you finish falling, call the method ***ResetCustomIKAdjustState***() to go back to the default states.

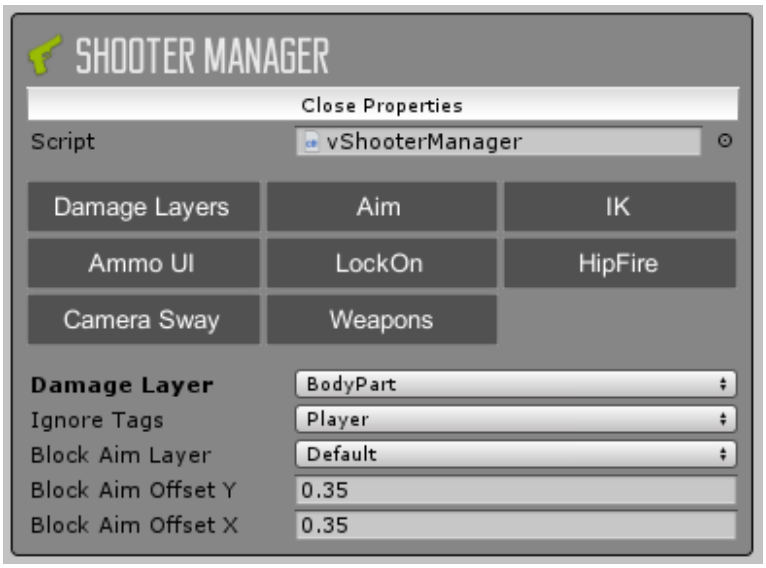
HOW TO APPLY DAMAGE TO A BODY PART

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.

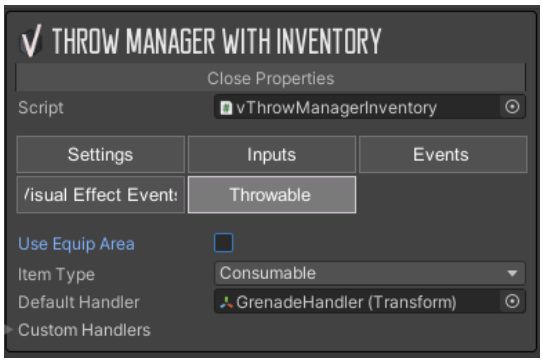
OR if you're creating a simple top down game for example, you can save performance leaving the enemies without a ragdoll and applying damage directly to the main capsule collider, in this case you can set the Damage Layer to Enemy.



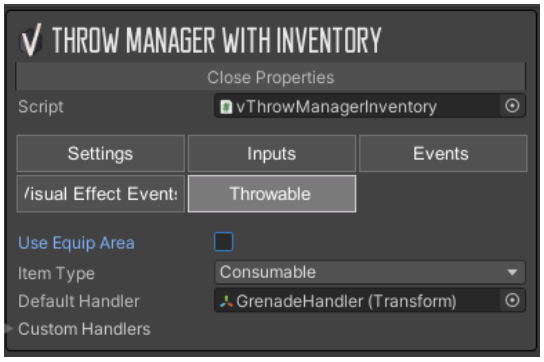
THROW OBJECT SYSTEM

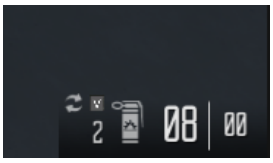
The Throw System is mostly **Plug & Play** but there are 3 options you can choose from:

1- **Throw with Inventory EquipArea**: Uses Invector Inventory system so the Throwable is a actual Item that uses vItemCollection trigger and is a item listed in the ItemListData, it will be equipped on a **EquipArea** from the Inventory (*consumable area by default but can be customized*)

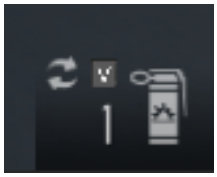


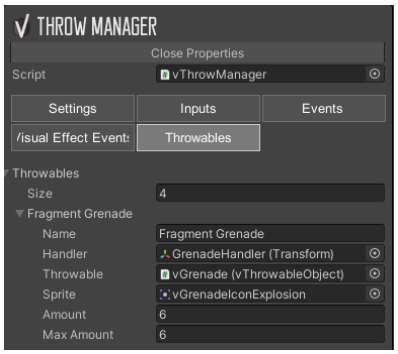
2- **Throw with Inventory but Standalone UI/Input:** Like the example above it uses a actual Items from the Inventory so you can use, drop, destroy (1) but if you uncheck the option “Use EquipArea” (2) it doesn’t need to be equipped in the EquipArea to be used neither uses that EquipAreaInput to switch between slots, instead, it will use the Standalone Throw UI (3) and separated input to switch between throwable types.



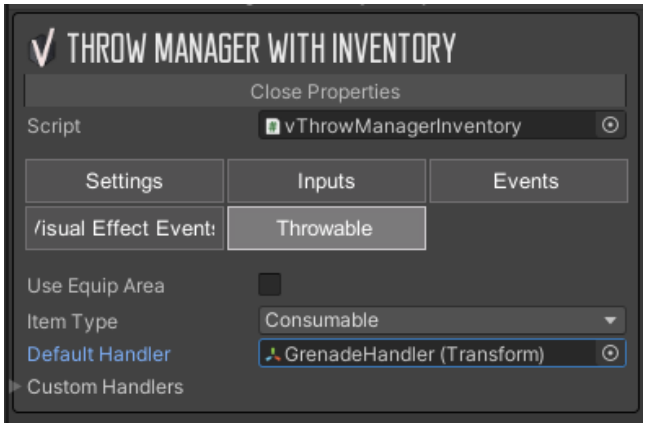


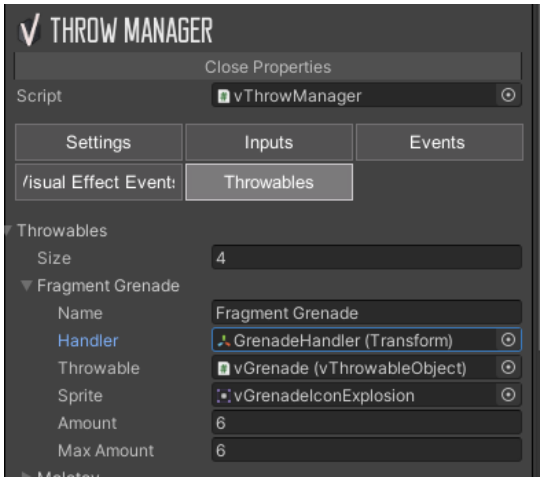
3- **Throw Standalone**: Doesn’t require the Invector Inventory System and can be used as a standalone feature, uses a Throw UI and a SimpleInput to switch between different throw types.





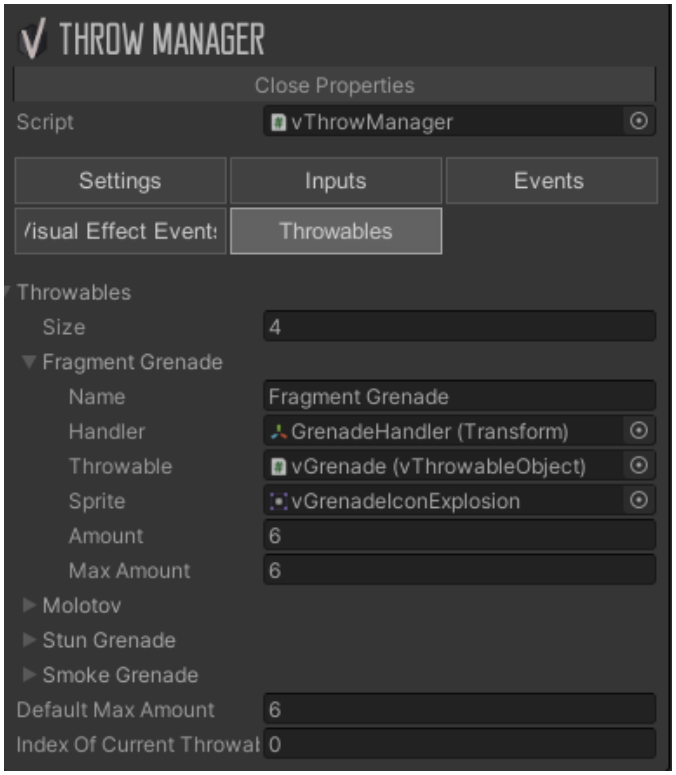
On either one of the 3 ThrowManager prefabs you end up choosing from, you need to assign a Handler for the Throwable Object to be instantiated when you enter ThrowMode, you can follow the example included in the demo scene.



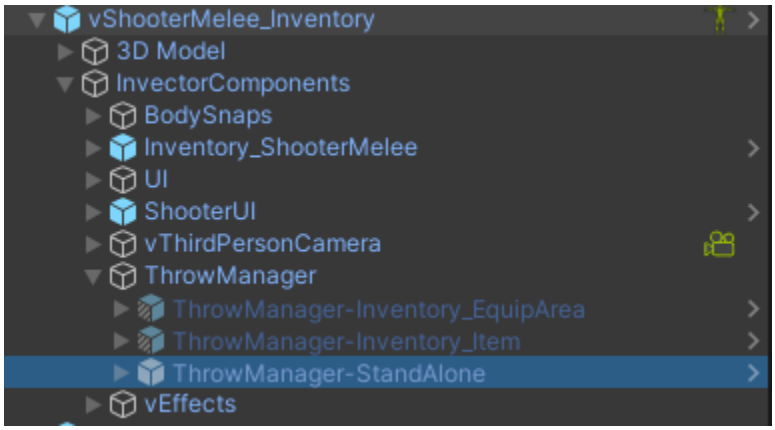


* Remember that when using the *ThrowManager With Inventory*, the option “Use EquipArea” will determine if it’s possible to equip the throwable item on a EquipArea from the Inventory UI or it will use the Standalone Throw UI.

To use the ThrowManager Standalone version you need to manually set up your throwable types



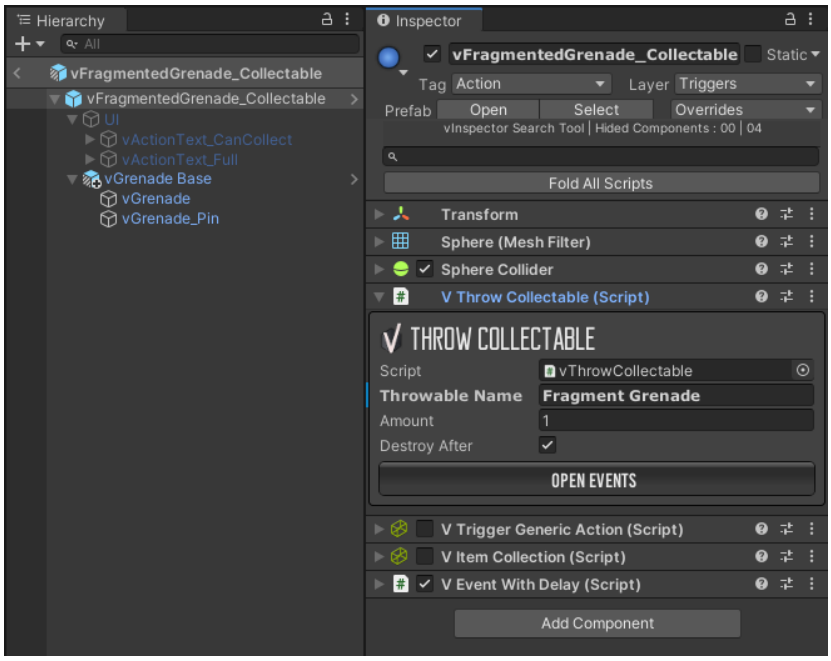
You can play the ShooterInventory demo scene and switch between the 3 different types to figure out what style fits your game better, then, after creating a New Character, drag and drop the Throw prefab you want inside your character.



Throw Collectables

If you take a look at one of our collectables it may seem confusing but in reality you only need to change the **Throwable Name** to match the name you've added in the ThrowManage, the amount and the **3D Model inside** the hierarchy.

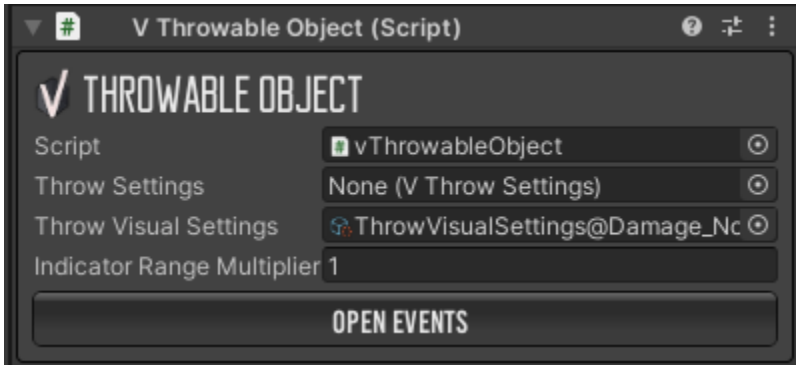
The rest is all pre-setup to work with Standalone or Inventory.



- **Throwable Object**

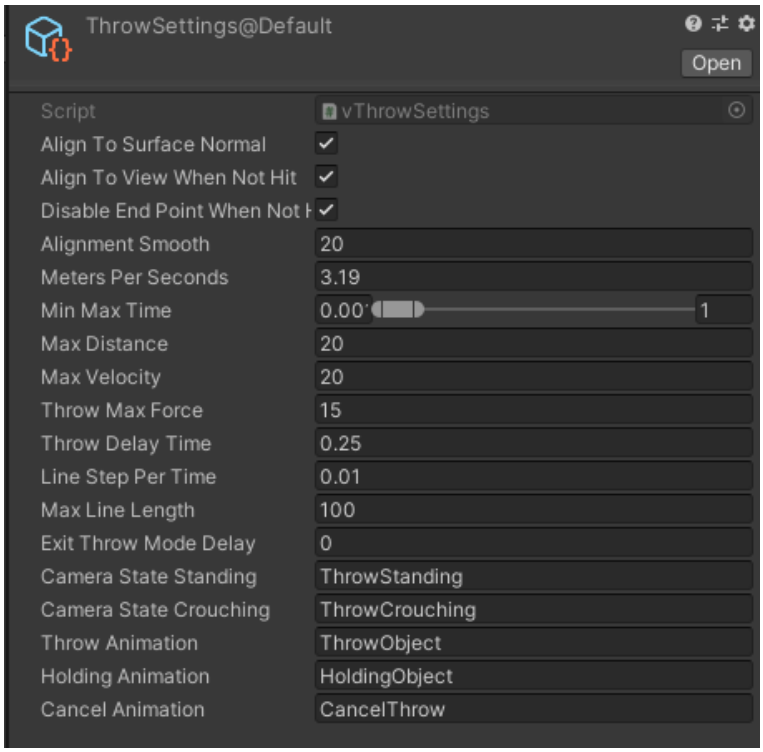
This prefab is the actual object your character will throw, it needs this component and the rest is up to you, the object can explode, can trigger sounds, effects, events to do pretty much whatever you want.

We have several different examples included, Fragmented Explosion, Molotov, Stun and Smoke.



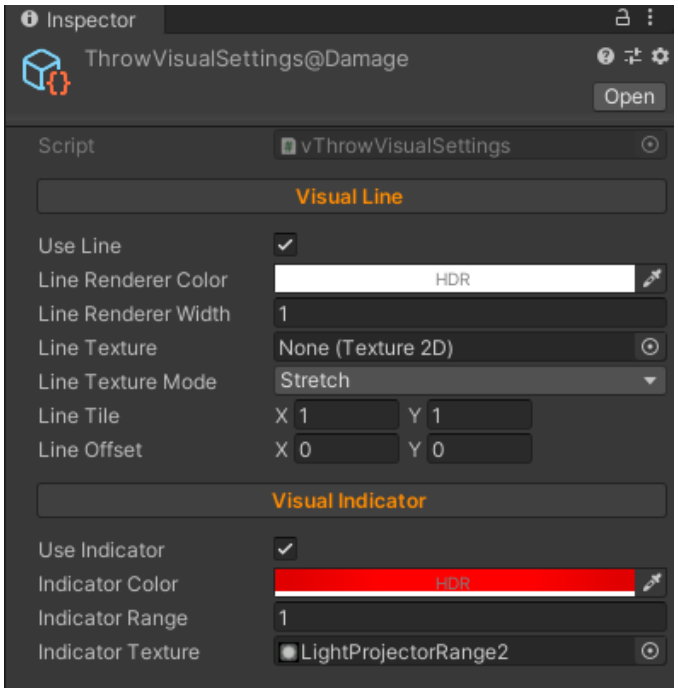
- **Throw Settings**

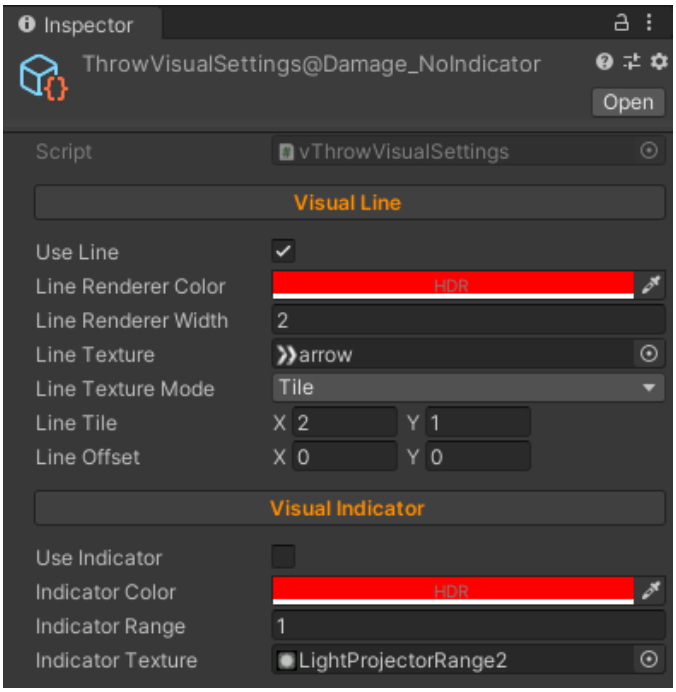
These settings can be used for all throwable objects (*assign in the ThrowManager*) or each individual throwable (*assign in the ThrowableObject*) and you can customize things like unique animations, camera state, distance/force, etc…



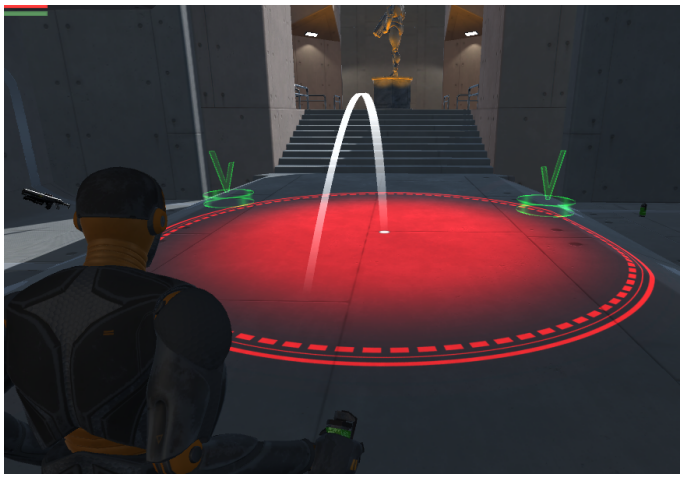
- **Throw Visual Settings**

Similar to the Throw Settings but this one only affects the Line Renderer and Visual Indicator when you're Aiming, again you can set one default for all throwables or have each throwable with a different line/visual indicator.



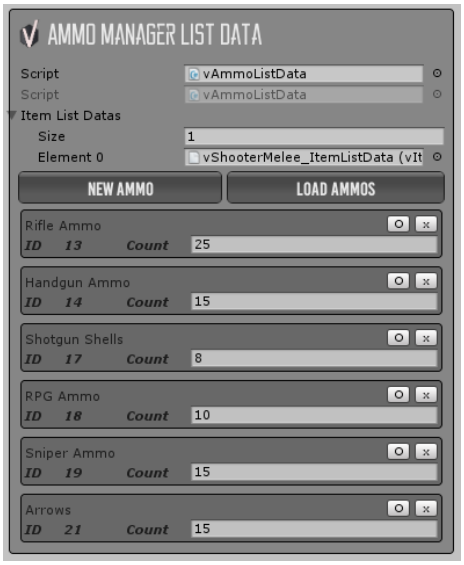


Here is a example of a custom Visual Indicator (left) and LineRenderer Texture (right)



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



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

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