Third Person Controller – Basic Locomotion

(v2.4.2 - 10/08/2019)

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

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

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

--- Invector Team ---

Ps This Documentation is for the **Basic Locomotion**, there is another for the Melee Combat and Shooter in their respective folders.*

Summary

FIRST RUN

IMPORTANT

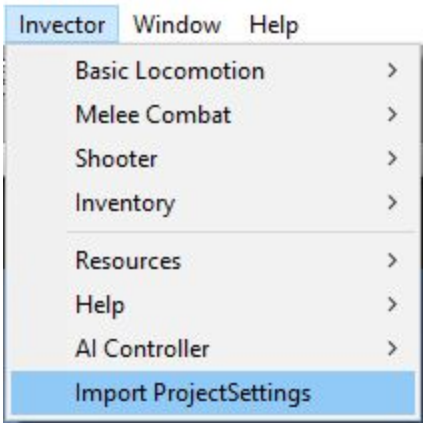
This is a **Complete Project**, and as every complete project it includes a custom **InputManager**, **Tags**, **Layers**, etc… **Make** sure that you import **on** a **Clean Project**.

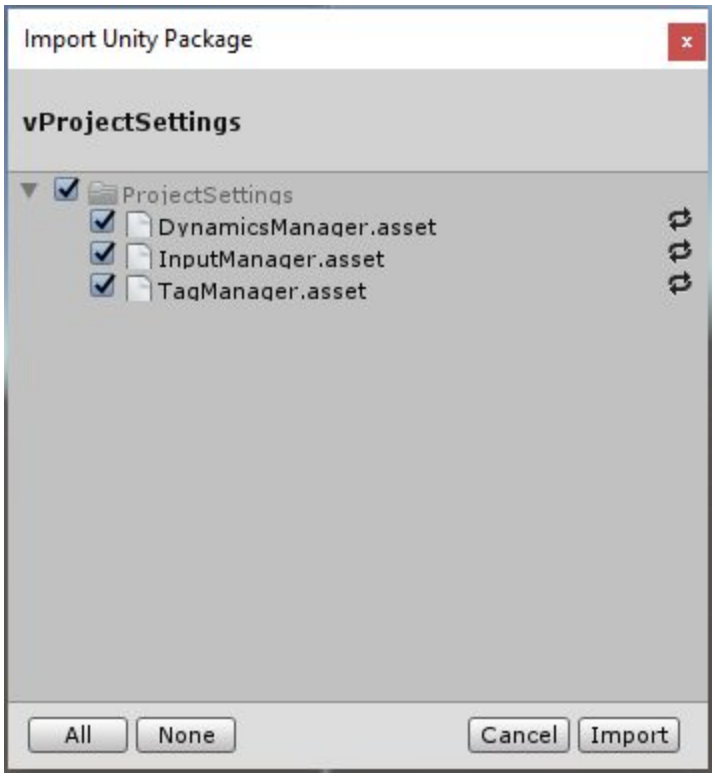
- *Importing on an existent project*

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 **Invector > Import ProjectSettings.**



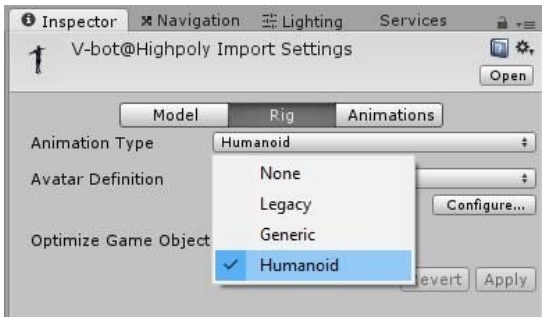


Now that you have imported the necessary files, you can explore the several demo scenes and figure it out what kind of Third Person Game you want to create.

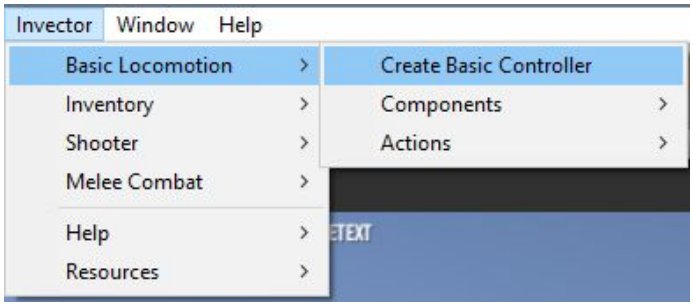
***Updates also need to be imported into a Clean Project, so MAKE SURE TO BACKUP your previous project and transfer the necessary files to your new project. ***

CREATING A CHARACTER CONTROLLER

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

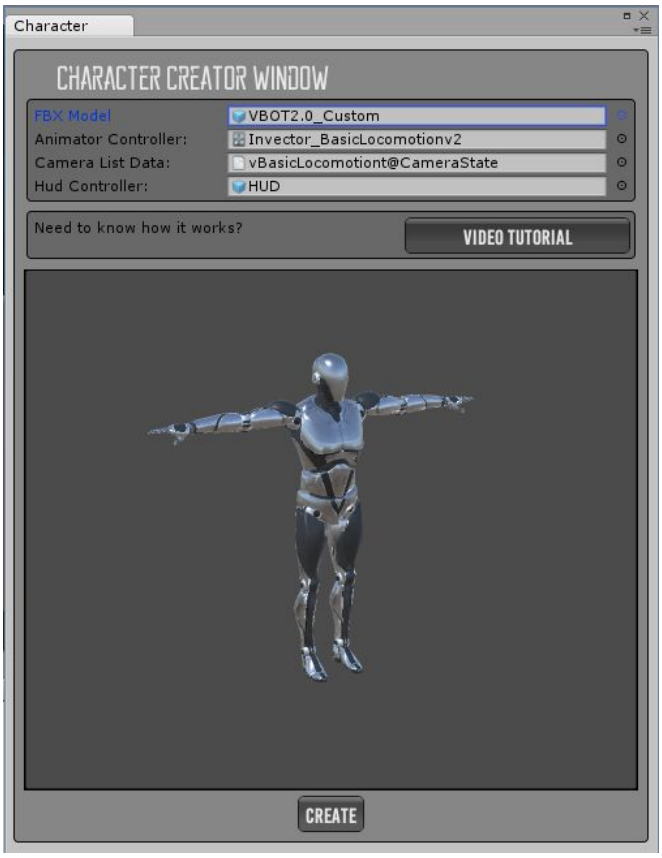


To setup a new character, go to the tab *Invector* > *Basic Locomotion* > *Create Basic Controller*



Make sure your Character is **Fully Rigged** and set up the FBX as a **Humanoid**, then assign the FBX to the field "FBX Model and the rest of the fields will be already assigned with our default AnimatorController and CameraState, you can of course replace to your own modified version of those files if you want.

Click on the button “Create” to finish the character.



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

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

Hit Play and enjoy ☺

TOPDOWN / 2.5D / CLICK TO MOVE CONTROLLER

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 folder *Basic Locomotion\Scripts\CharacterController\Examples* and replace your current **vThirdPersonController** component for the **vTopDownController** or **2_5Dcontroller** in the Player Inspector.

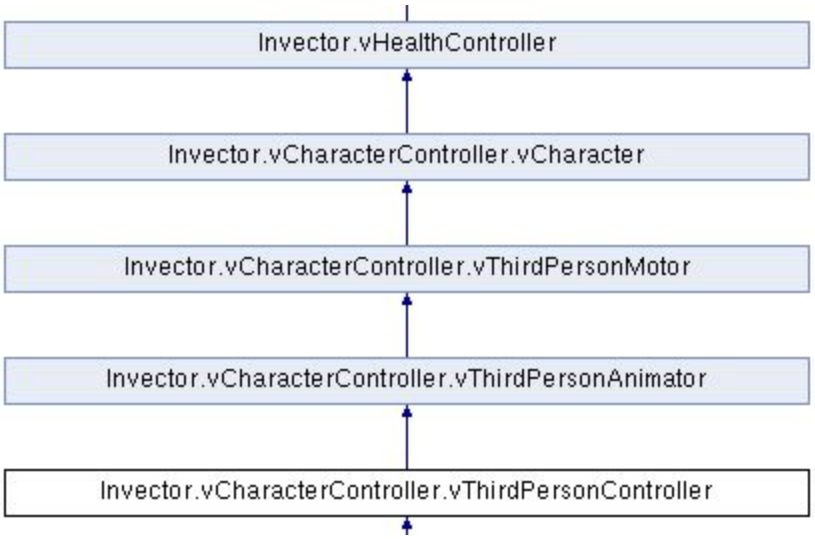
In order to the **TopDown Strafe** mode to work correct, don't forget to UNCHECK the "**Rotate to Camera While Strafe**" option in the **InputManager** Inspector.

To use the **ClickToMove** you can still use the vThirdPersonController, but you will need to replace the Input to **vClickToMoveInput**, for more information check the vClickToMove-Demo scene.

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

HOW IT WORKS?

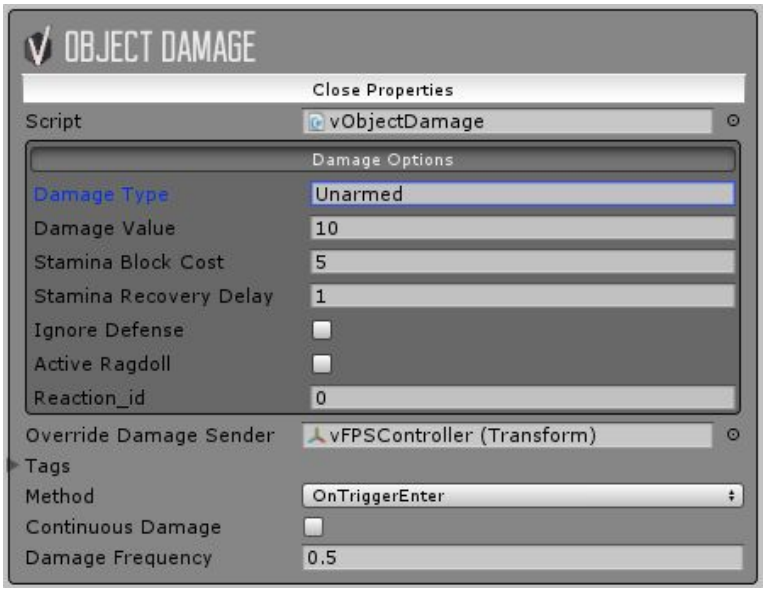
The Controller works with **six main scripts**:



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** manage methods like sprint, crouch, roll, jump, etc…
6- **vThirdPersonInput** receives all the input and call every method of the other scripts on Updates.

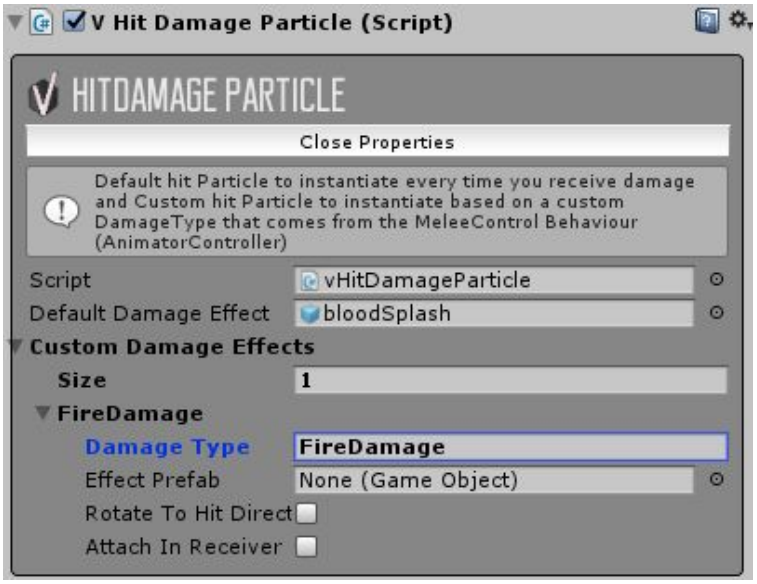
HOW TO APPLY DAMAGE TO THE PLAYER

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.



- **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.

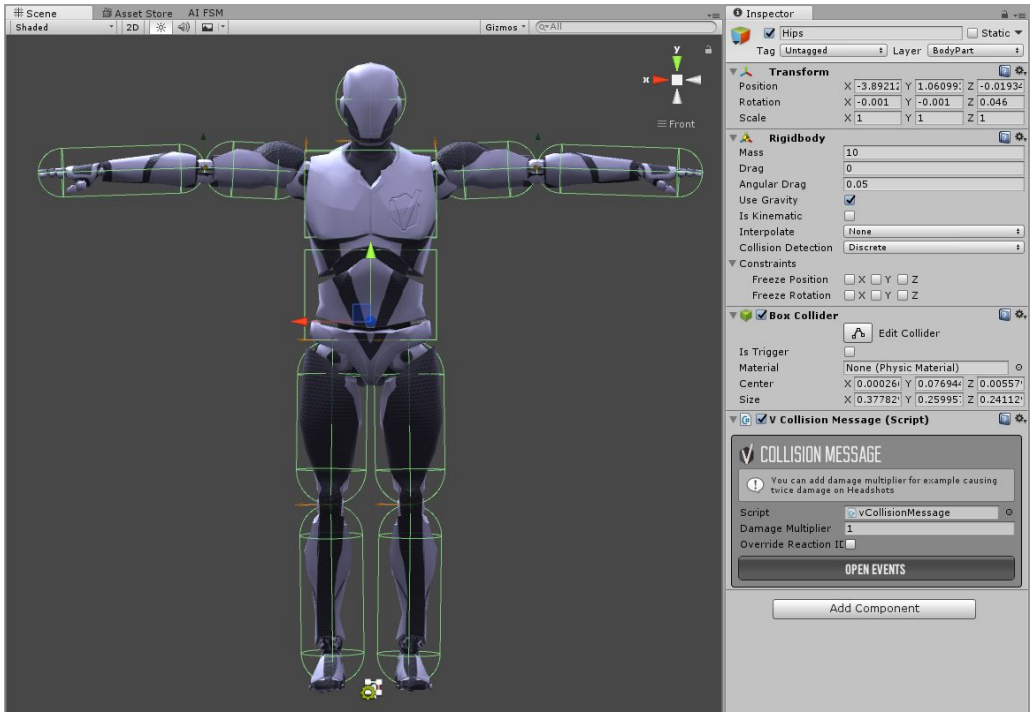


*This component is attached to the Controller or any object that contains the HealthController

- **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.

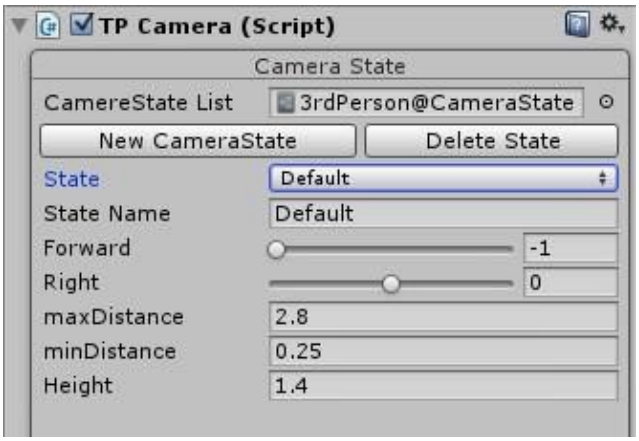
A DamageReceiver is attached to each ragdoll collider, this will allow to Player or AI to apply damage to each bodypart instead of the CapsuleCollider.

- **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.

CREATING A NEW CAMERA STATE

In 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.



Then just change the CameraState on the method ControlCameraState() on the script TP_Motor.

Example:

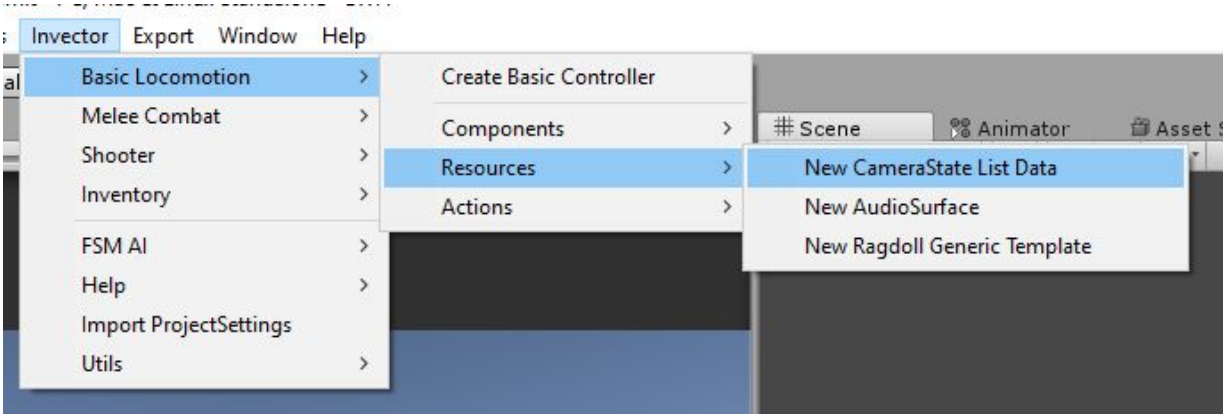
tpInput.ChangeCameraState(string cameraState, bool useLerp = true)

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();

If you have more than one character and want to use different States, you can create a new **CameraState List Data** here (pic below) and assign on the CameraState List field on TP Camera Inspector.

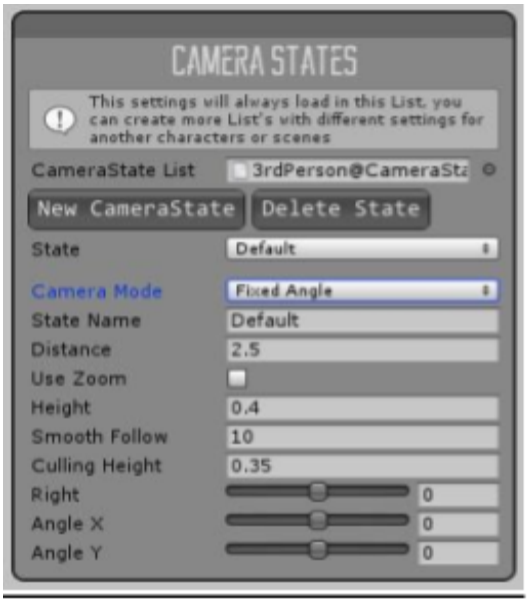


CameraMode – Free Directional

This CameraMode offer 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…

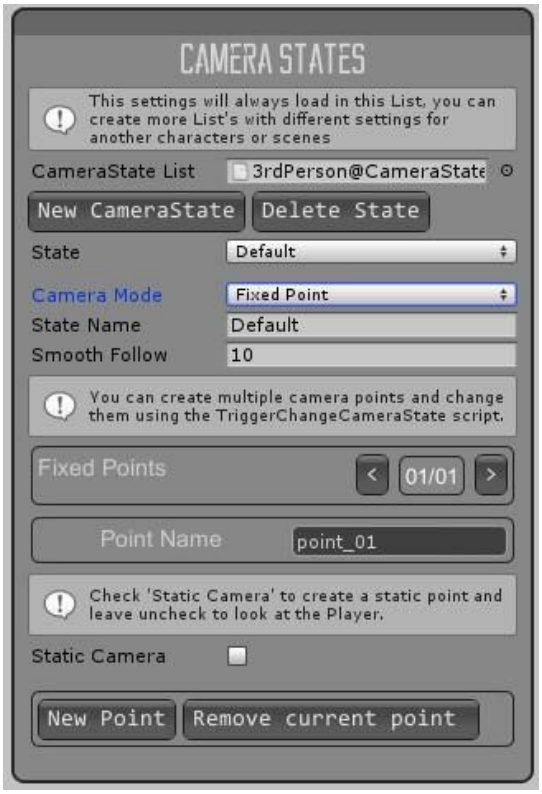
CameraMode - Fixed Angle

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

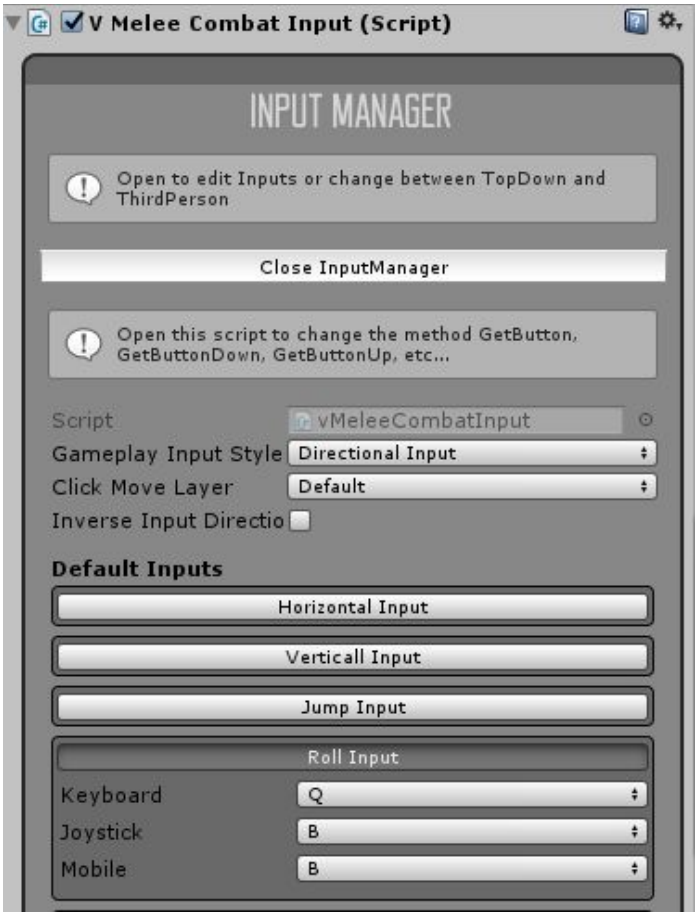
XBOX CONTROLLER SUPPORT

This package works great with the **360 controller** and supports **vibration** (Windows only), make sure you compile your build according to your system. If you are using Windows 32bits make sure the build settings are set to x86 or if you are using Windows 64bits make sure the build settings are set to x86_x64.

V1.1 add support for MFi iOS gamepad.

INPUT MANAGER

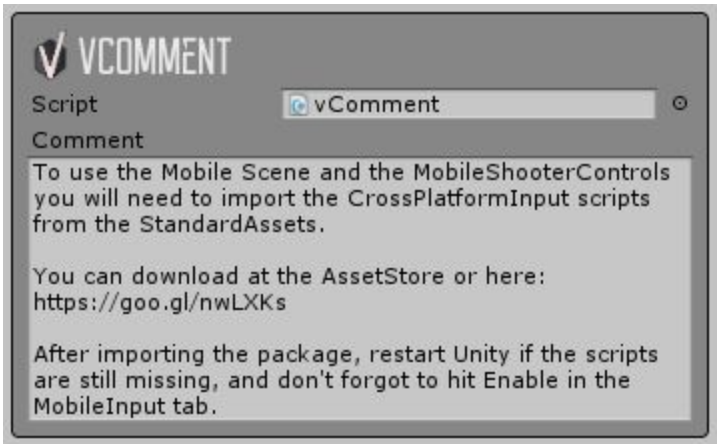
We have a InputManager so you can change the input of the character actions and movement.
If you created a Basic Locomotion it will use the vThirdPersonInput, if it's a Melee Combat character it will use the vMeleeCombatInput, and the Shooter uses the vShooterMeleeInput.



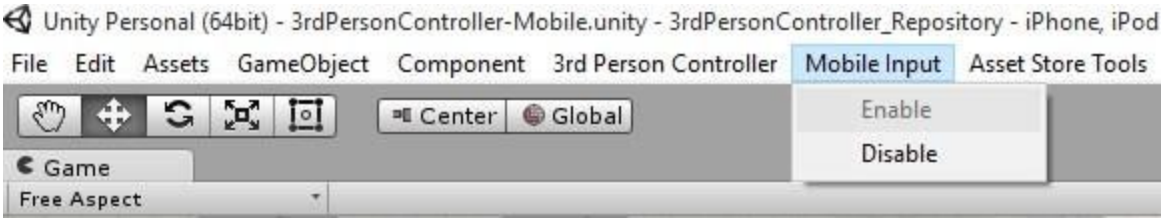
MOBILE CONTROLS

Since the release of the Shooter Template, we have to remove all content of the **StandardAssets** from our project, and since we need some files from the **CrossPlatformInput** in order to the Mobile Controls work, we have to separate those files into a package, you can [DOWNLOAD HERE]

This information is also available in the Mobile Demo Scene, in the hierarchy we add the gameObject "**___README FIRST!!!**"



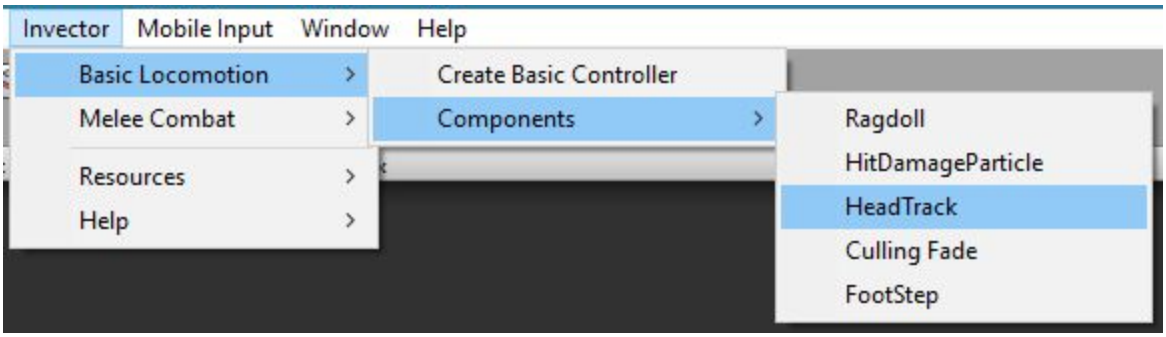
After importing the package, change your platform to **Android** or **iOS** on the **Build Settings** and make sure you have the **SDK** installed and don't forget to **Enable** the Mobile Input after changing the platform, it should work right on the Editor.



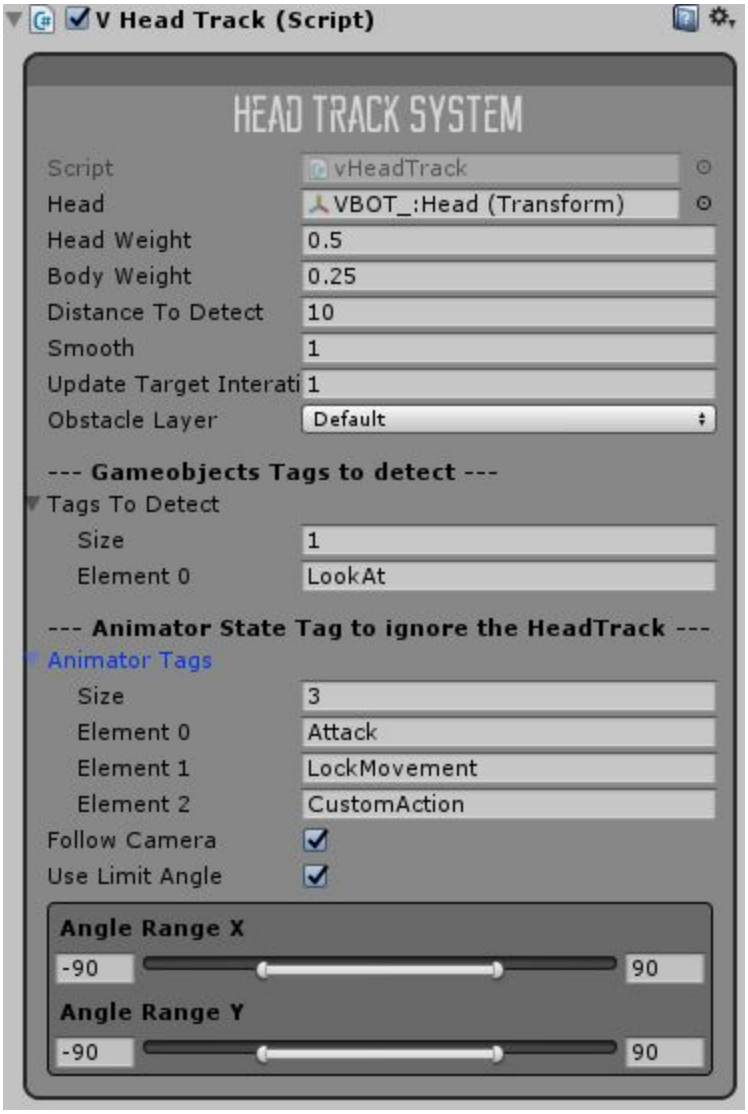
In order to have a **stable performance** on mobile devices, we recommend **compress all your textures**, set the **Quality Settings to Good or Simple**, and remove any **Camera Effects.**

HEAD TRACK

ADD V2.0 - Now the Headtrack is a separate component and you need to add manually:
*__Shooter__ - automatically add the headtrack in order to aim up/down

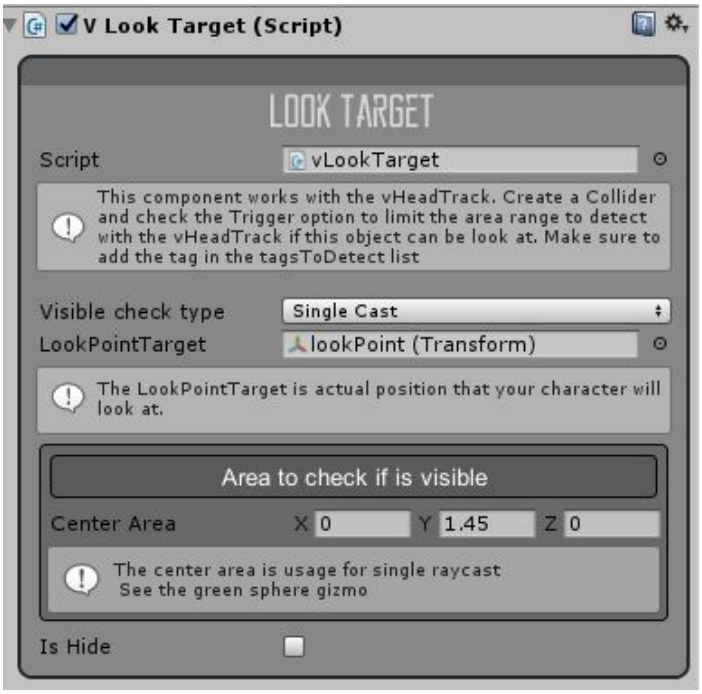


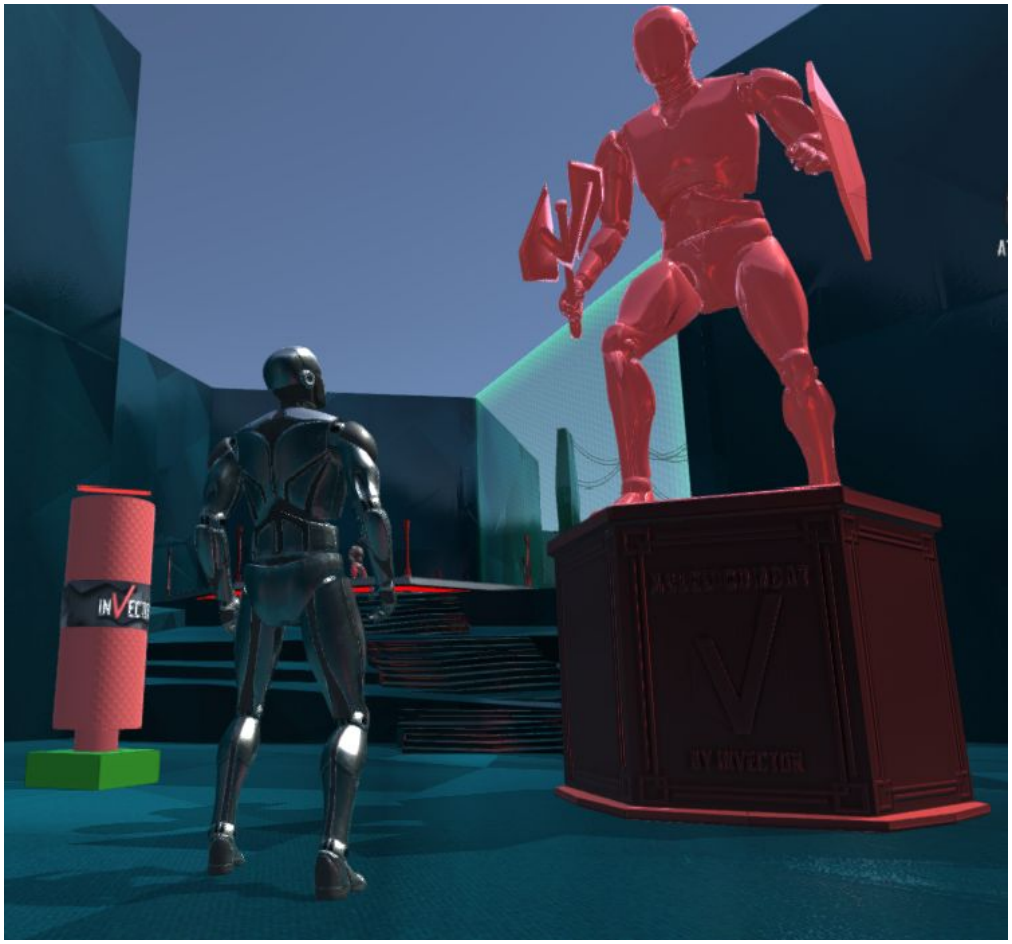
Now we have a lot more options and we can use the LookAt feature as well.



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

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

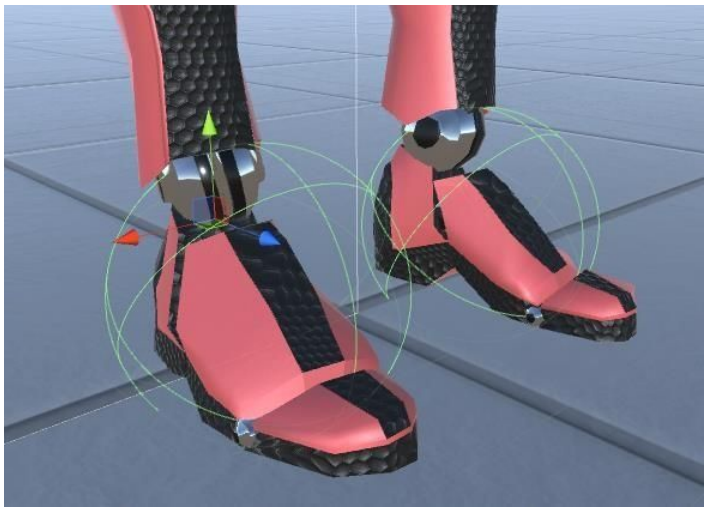

V Look Target (Script)
LOOK TARGET
Script
vLookTarget
This component works with the vHeadTrack. Create a Collider and check the Trigger option to limit the area range to detect with the vHeadTrack if this object can be look at. Make sure to add the tag in the tagsToDetect list
Visible check type
Single Cast
LookPointTarget
lookPoint (Transform)
The LookPointTarget is actual position that your character will look at.
Area to check if is visible
Center Area
X 0
Y 1.45
Z 0
The center area is usage for single raycast
See the green sphere gizmo
Is Hide



FOOSTEP AUDIO SYSTEM

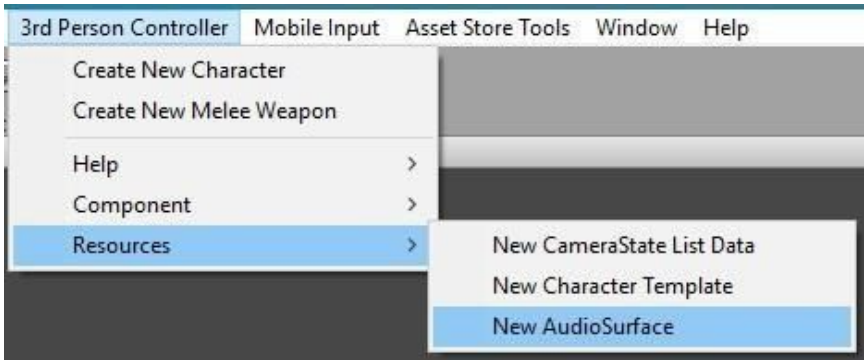
Video tutorial: *https://www.youtube.com/watch?v=gxesgNH0UBM*

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.



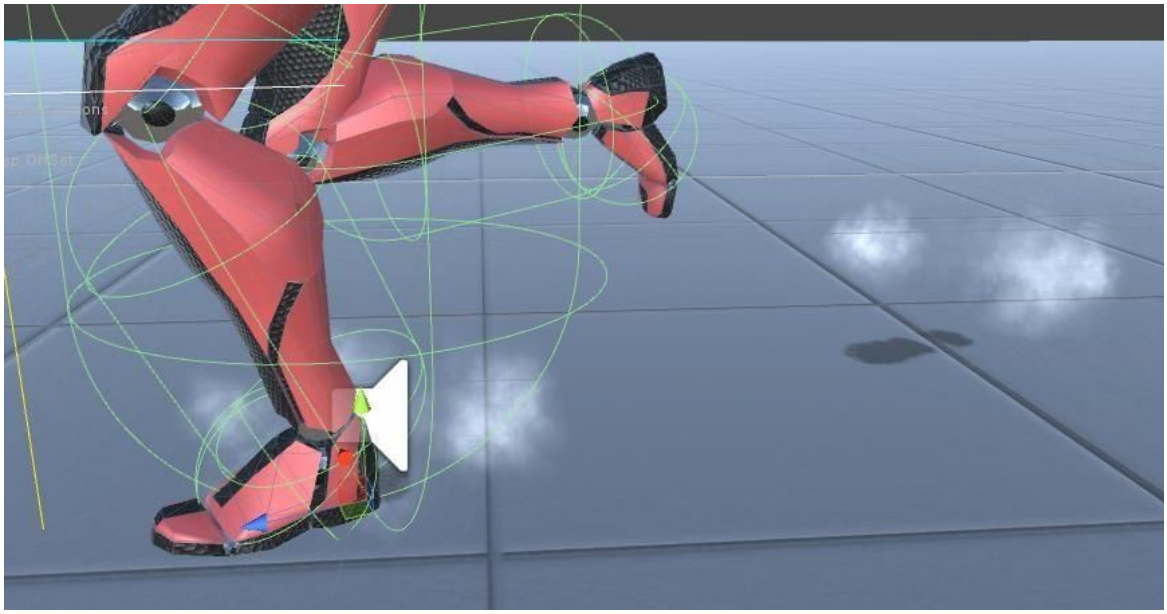
You can select the **LeftFoot** and **RightFoot** Sphere and manipulate the **Center XYZ** to position as you like, and change the **Collider Radius** too, the size of this sphere will depend on your Rig bone size. Assign the "***defaultSurface***" that comes with the package to have an example of how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

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



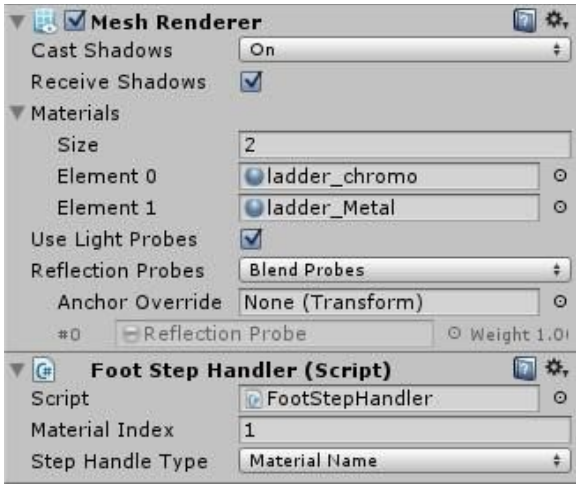
Now you can create **Custom Surfaces**, to play other audio clips based on the **material** that the sphere collider will hit. Assign the new CustomSurface to a new CustomSurface on the FootStep Inspector.

You can assign a **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)

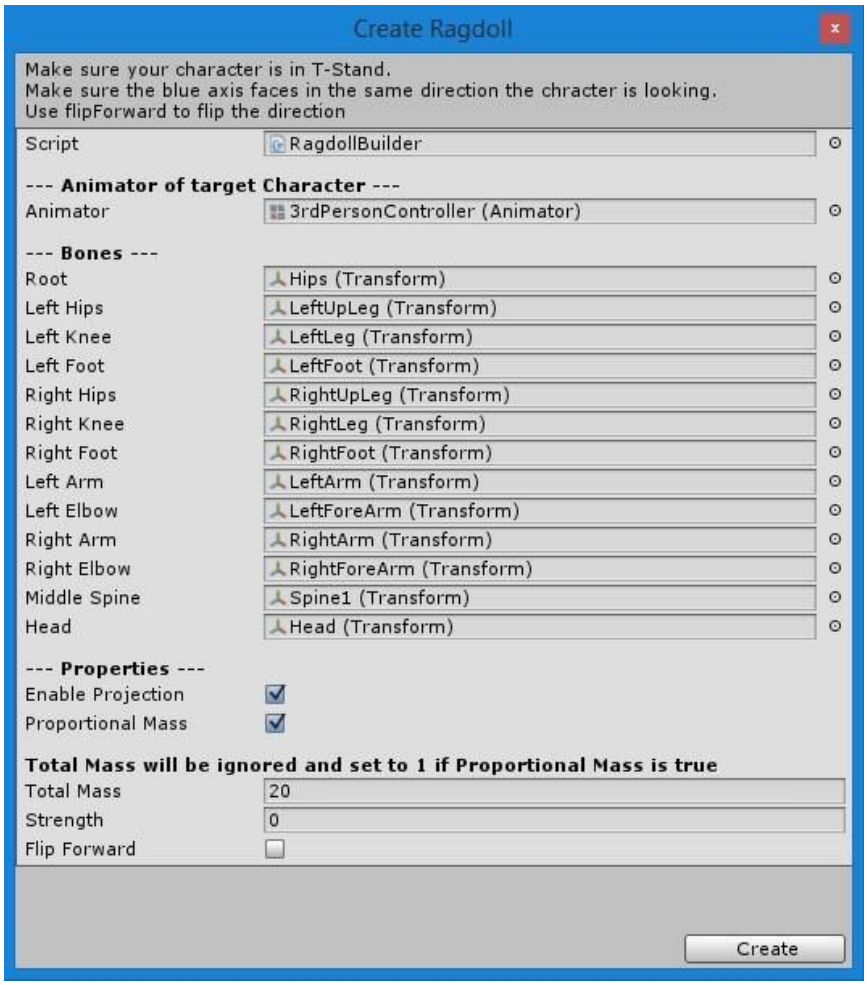


CREATING A RAGDOLL

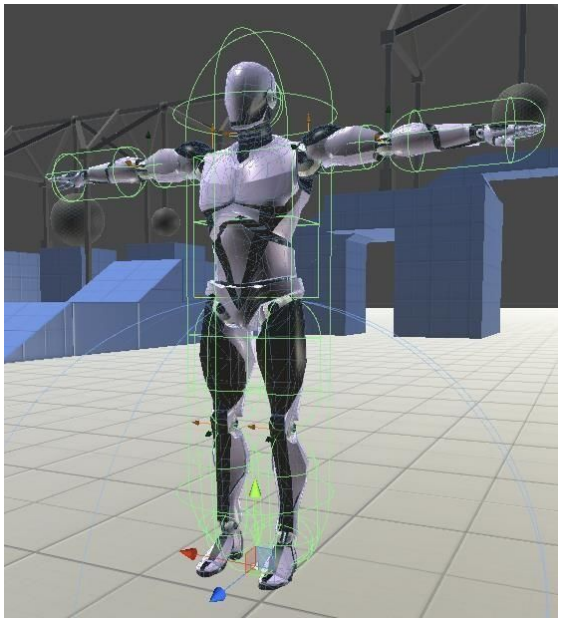
Ps* Make sure to add the Ragdoll First and then equip the character with the MeleeManager and Weapons!

Creating a Ragdoll is just 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 design 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.

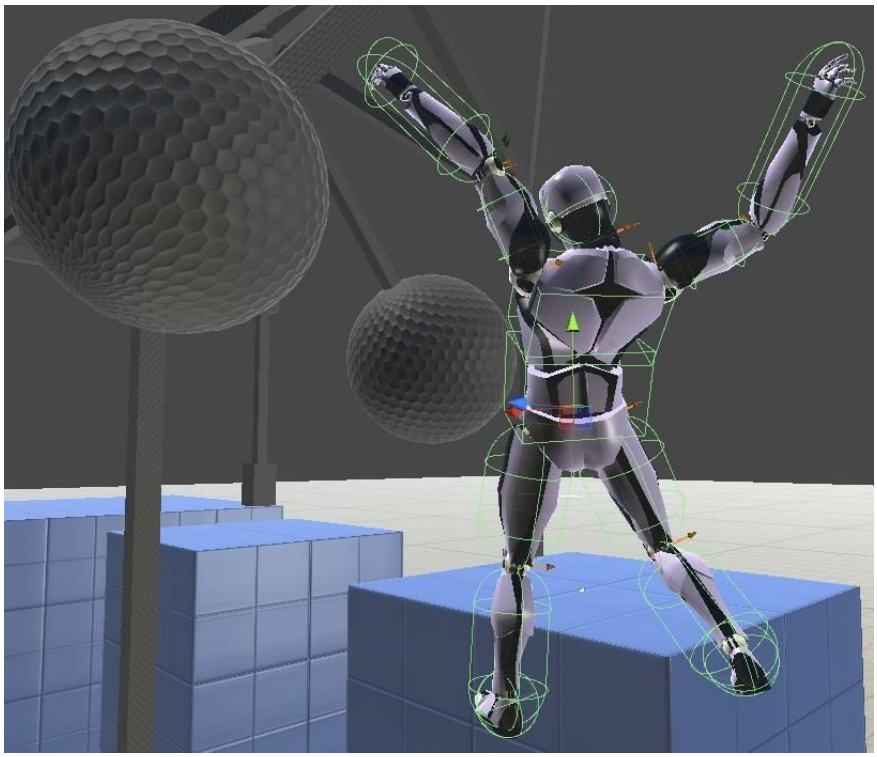


We strongly recommend keeping the **Enable Projection** and the **Proportional Mass** enabled, and do not forget to use **Scale Factor 1** on your **fbx** Model. This you provide better behavior of your ragdoll.

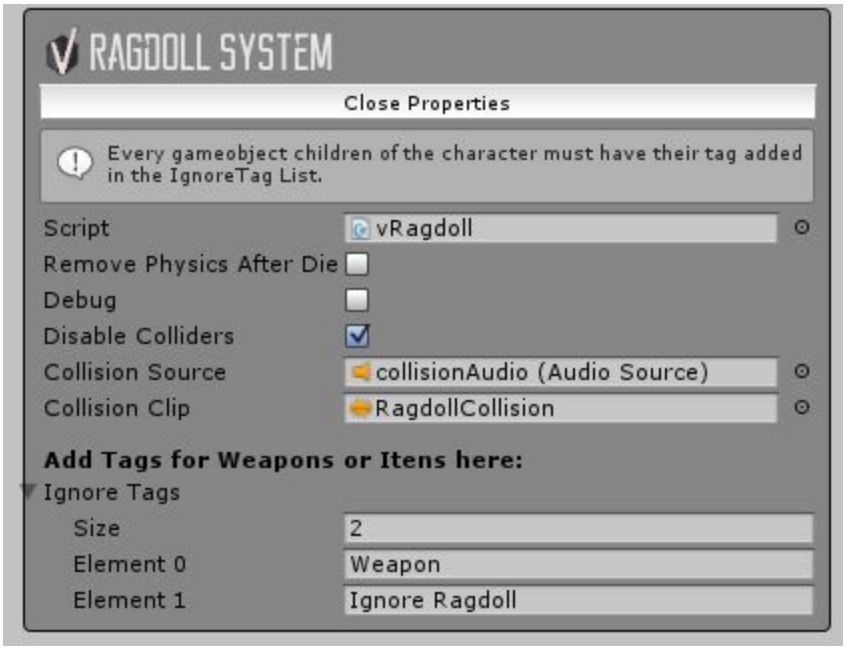


To enable the ragdoll, you can use the Script **ObjectDamage** or just call this line on the **OnCollisionEnter** method.

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

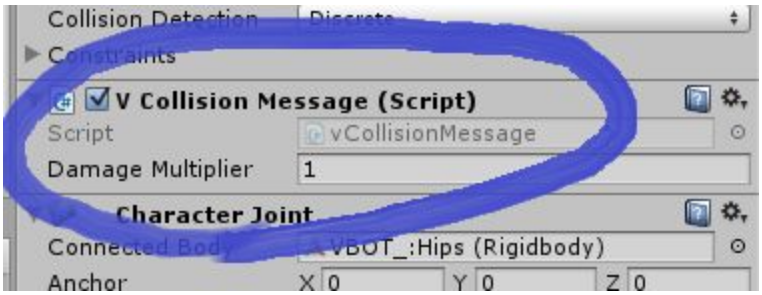


v1.1b – Add "***Ignored Tags***" you can add a list of tags for objects that are children of the Player to keep the rotation correctly, otherwise it will mess up the rotation when the Ragdoll are on.



*** 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

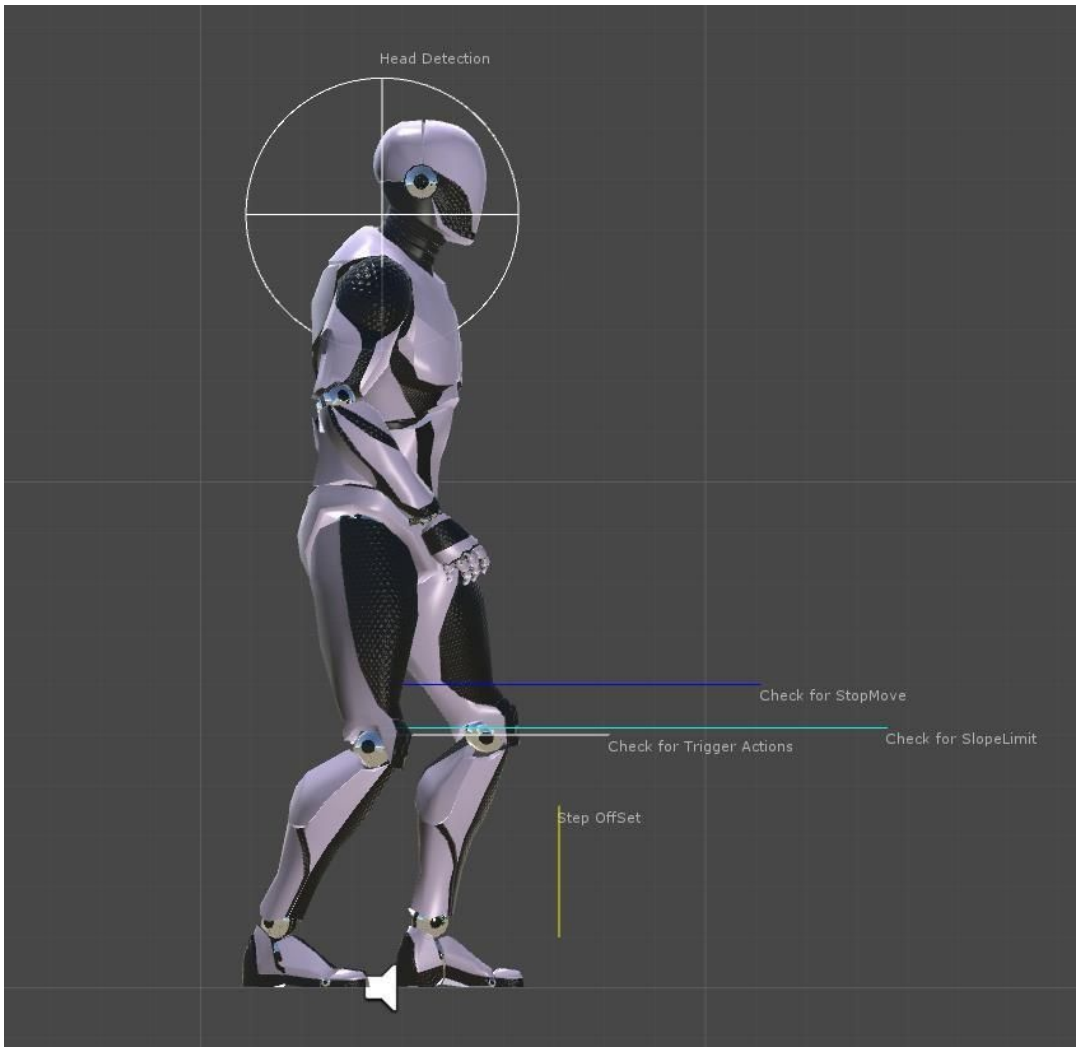
RAYCAST CHECKERS

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 of walking in absurd angle heights, float customizable on the Player Inspector.

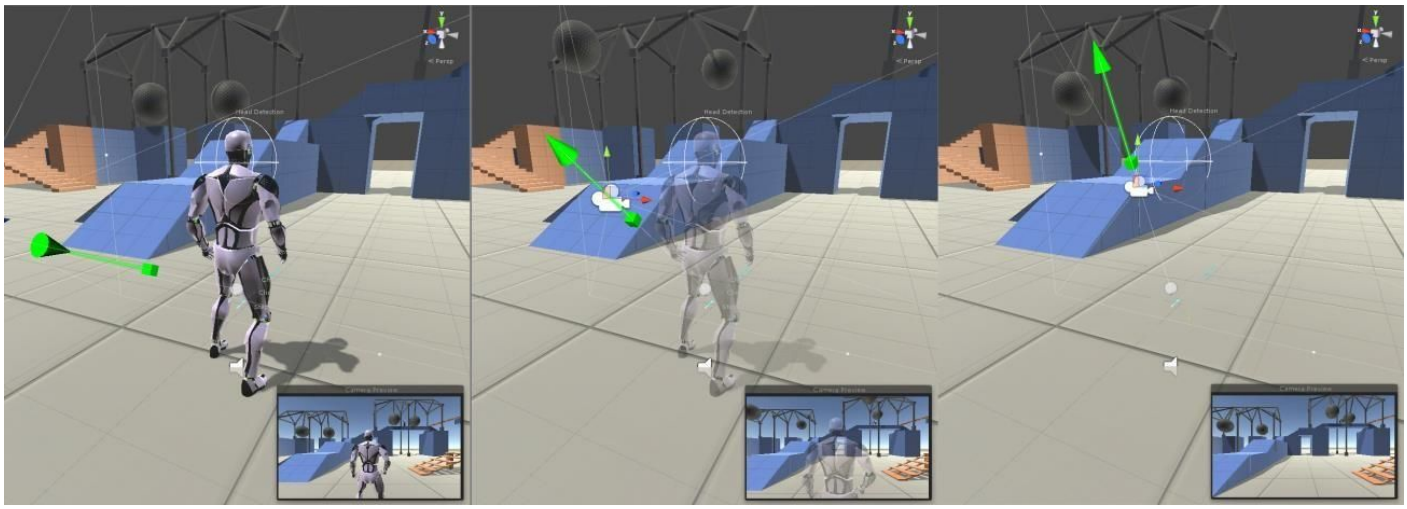
StepOffset is to help the character walk in custom height steps, adjust the values on the Player Inspector.



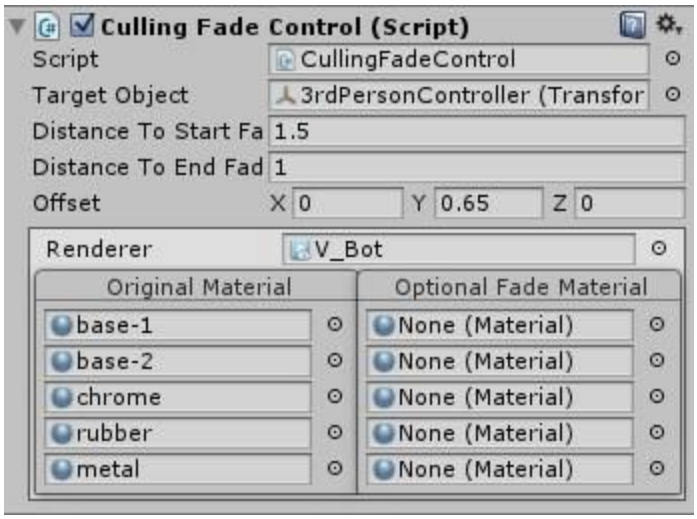
CAMERA CULLING FADE

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

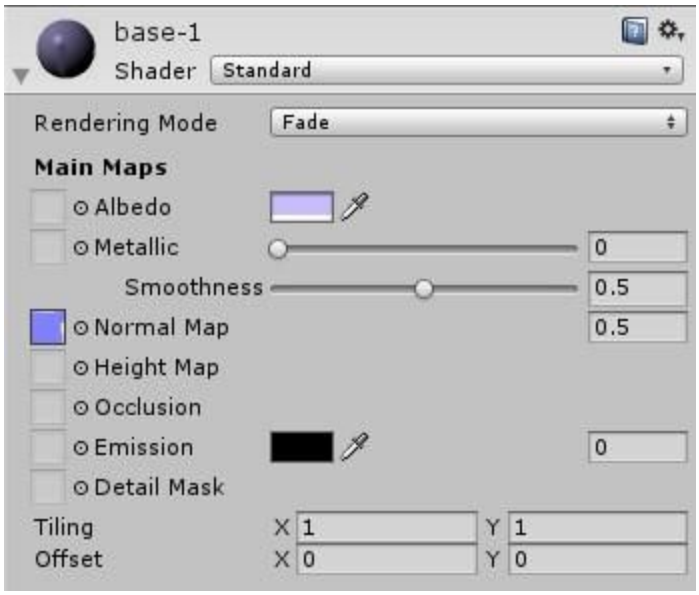
Example:



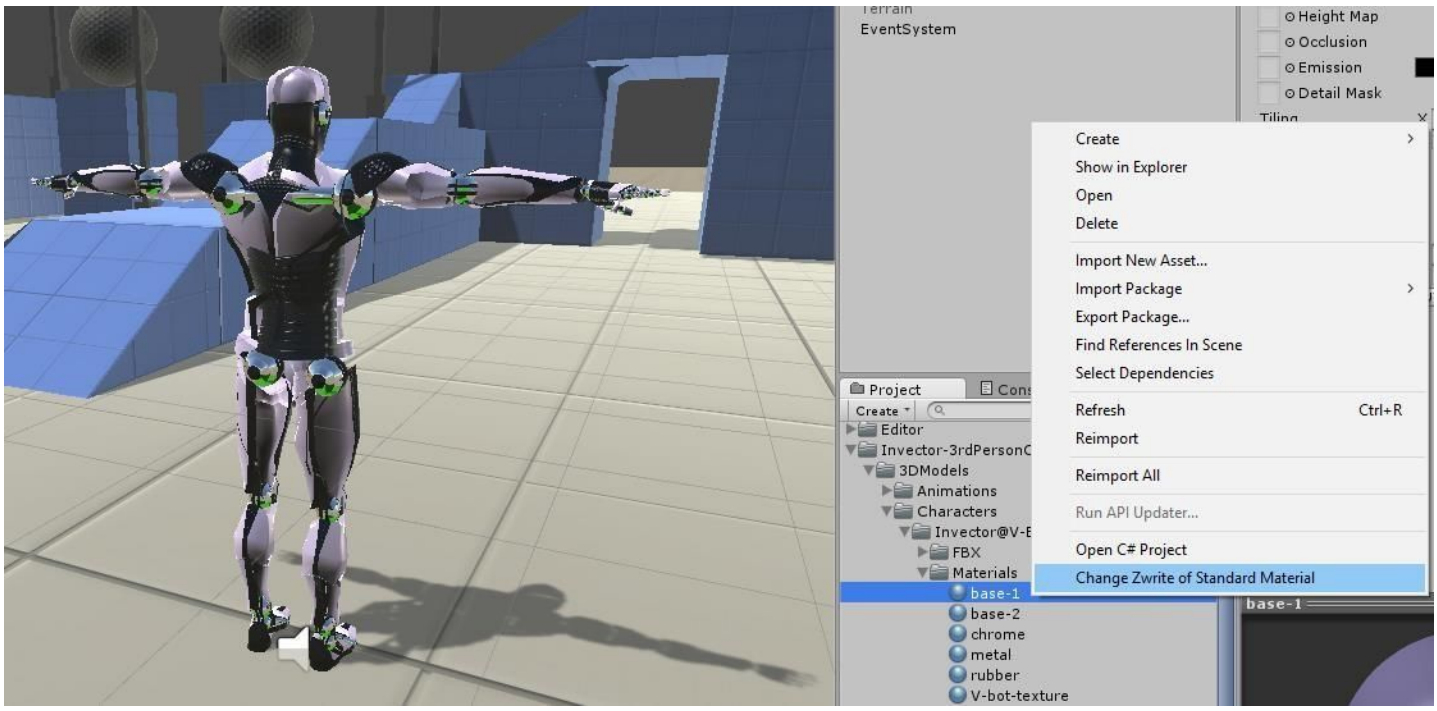
Our Culling Fade will set up automatically for the default Stardard Shader of Unity's, but you also can use custom shaders, just make an additional copy with the fade material and assign in the "Optional Fade Material" field.



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 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.

ANIMATOR TAG

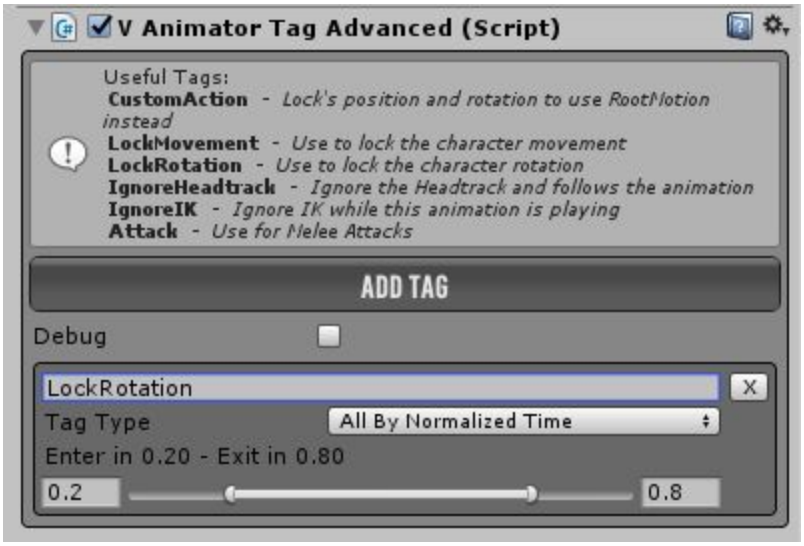
This is an Animator Behavior that you can attach directly on Animation State inside the AnimatorController, it's useful to know what animation is being played and what you can do while this animation is playing.

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 a **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 messages from a AnimationState to any GameObject:
https://www.youtube.com/watch?v=uZn53kKsl0I

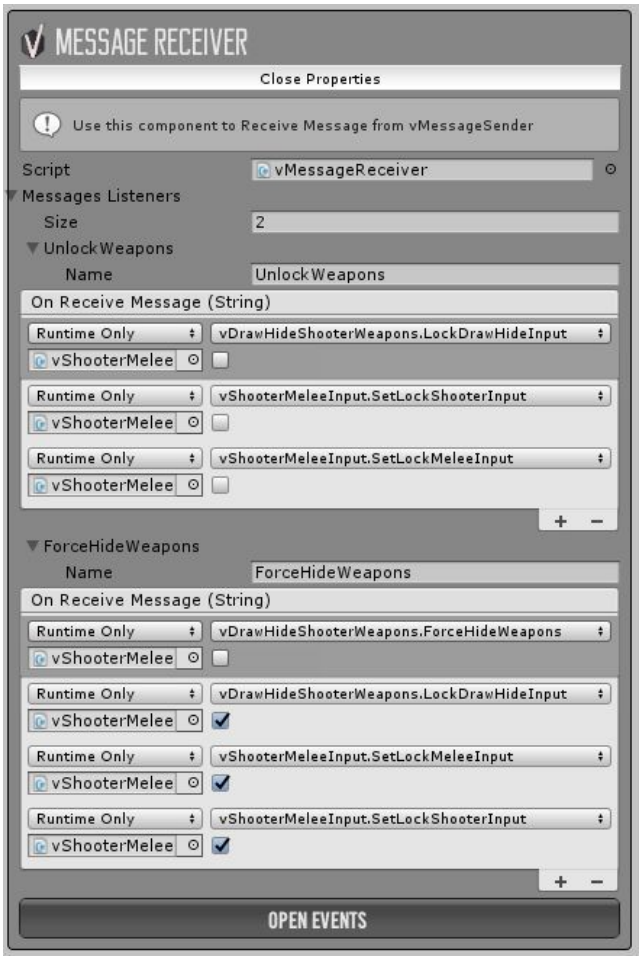
MESSAGE SENDER/RECEIVER

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 were you cannot attack anyone.

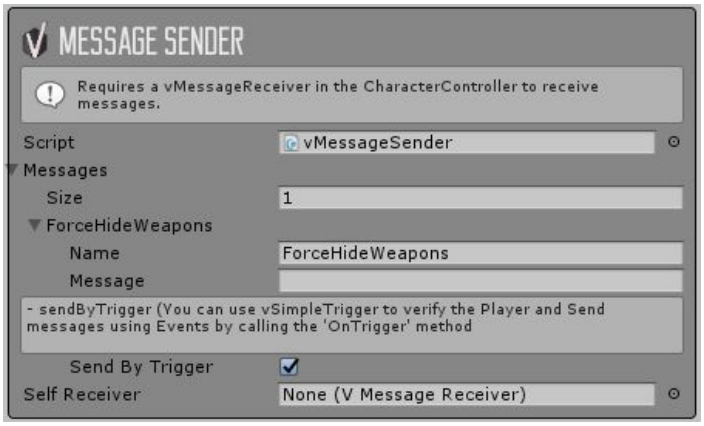
- UnlockWeapons were we call the method to unlock the hide weapons input and unlock the shooter and melee input.

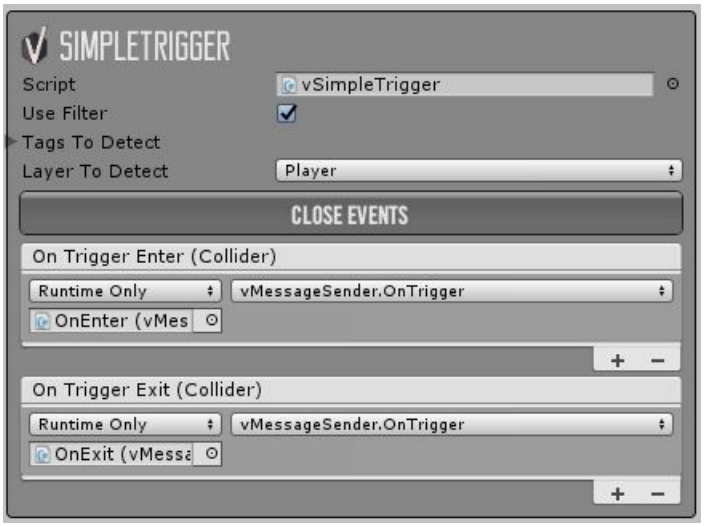


This is just one example, you can create any listener and call any public method you need.

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.

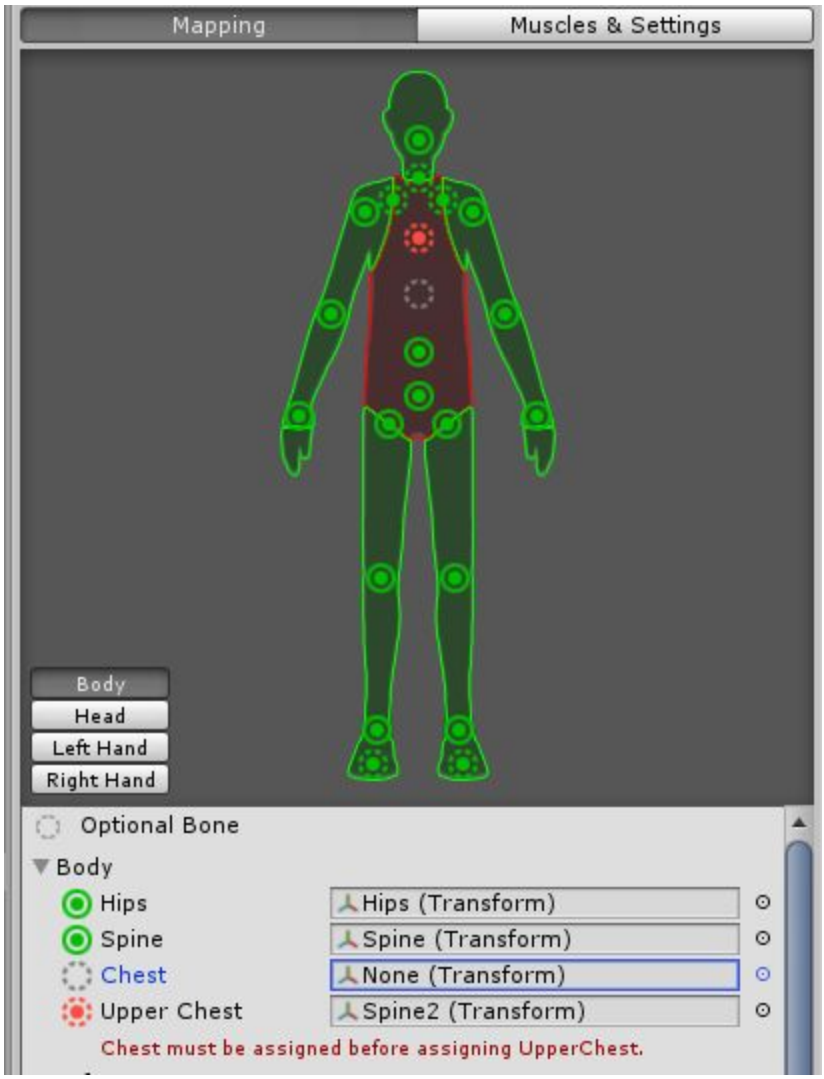
BODY SNAPPING ATTACHMENTS

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:



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.