3rd Person Controller – Shooter Template

Thank you for support this asset, we develop this template because a lot of developers have good ideas for a 3rd Person Game, but build a Controller is really hard and takes too much time.

The goal on this project was always to deliver a top quality controller that can help those who wants 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 ---

Summary

FIRST RUN....3
CREATING A NEW SHOOTER CONTROLLER....4
TOPDOWN INPUT....6
HOW IT WORKS?....6
CREATING A NEW CAMERA STATE....7
XBOX CONTROLLER SUPPORT....10
INPUTMANAGER....10
TAGS & LAYERS....11
RECOMMENDED MOBILE SETTINGS....12
HEAD TRACK....13
FOOTSTEP AUDIO SYSTEM....15
CREATING A RAGDOLL....18
HOW TO ADD NEW ANIMATIONS?....20
RAYCAST CHECKERS (V1.1)....21
CAMERA CULLING FADE (V1.1A)....22
WORKING WITH STANDARD ASSETS....24
USING MELEE WITH A SHOOTER WEAPON....24
ITEM MANAGER....25
CREATING AN SHOOTER ENEMY AI....28
LOCK-ON TARGET....30

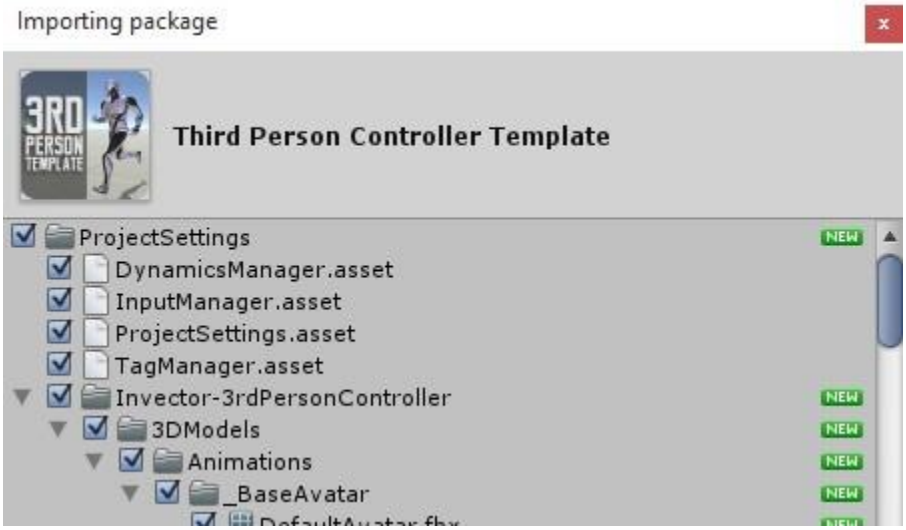
FIRST RUN

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*

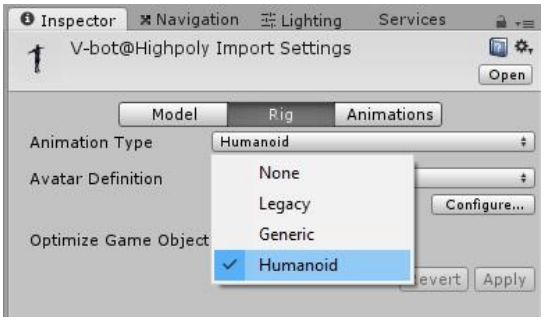
If you want to import into another project, you can UNCHECK some project settings to avoid conflicts or replace your project settings like the TagManager (which includes all the Layers), and add later the tags and layers that we use. We recommend to import the InputManager because it's kind of painful to add manually later (lots of inputs).



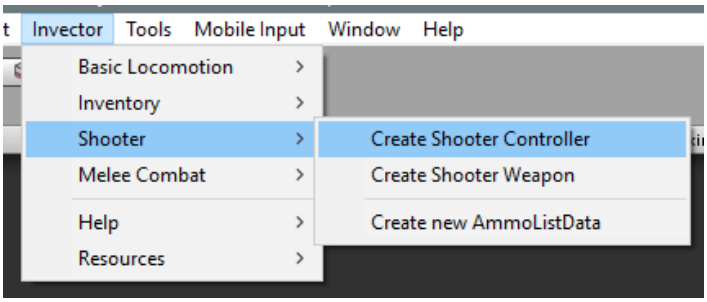
***Updates also need to be imported into a Clean Project, so MAKE SURE TO BACKUP your previous project and transfer the necessary files to your new project. ***

CREATING A NEW SHOOTER CONTROLLER

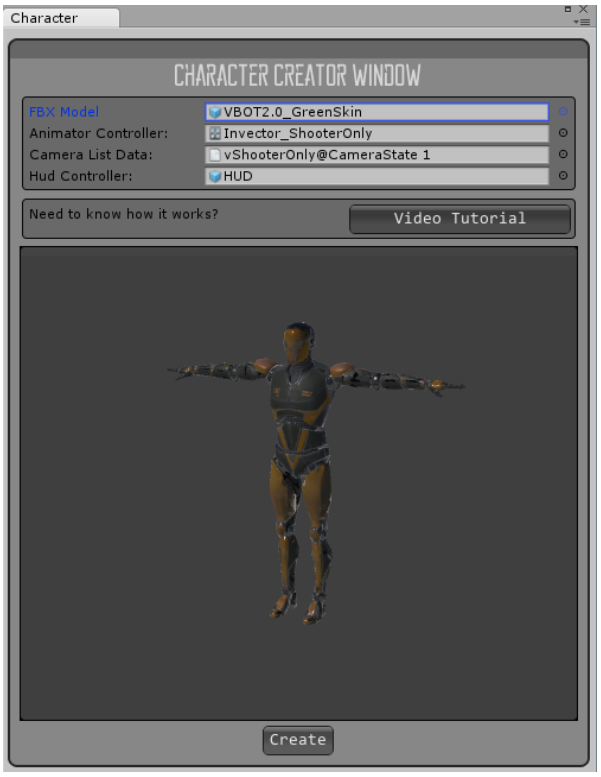
Make sure that your fbx character is set up as **Humanoid**



To setup a new character, go to the tab *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 field "Humanoid" and click on the button "Create 3rd Person Controller".



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

Done.

You don't have to do anything like dragging scripts, assign empty slots, etc… the **Character Creator** will take care of all the hard work automatically and set up everything for you. It will create the **3rdPersonController**, **3rdPersonCamera** and a UI Canvas with a **HUD** to display health, stamina and other information's.

The script will also adjust your Capsule Collider settings 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 SCALE 1** the same goes if the ragdoll **gets** weird.

Hit Play and enjoy ☺

Tips:

When creating a new ***ShooterController*** don't forget to:

- add a inventory from the menu **Invector > Inventory > ItemManager**
- select the Inventory Prefab from the **Project > ItemManager > Prefabs**
- and select the **ItemListData** > **vShooterMelee_ItemListData**
- use the **ItemFilter** to filter the items that you want to show (shooter and ammo)
- try add a weapon into the inventory, hit play, equip the weapon and aim (you will see that the character will aim down)
- go to the inspector and find the **ShooterManager**, hit the button **LockCamera** & **LockAiming**
- to fix the position of the weapon and arms, change the transform of the **defaultEquipPoint** and not the weapon itself or the renderer inside, start by rotating the defaultEquipPoint so the IK can work to aim forward, then position the weapon into the hand.
- copy the transform component of the defaultEquipPoint and paste the values after exiting PlayMode, now all the weapons should be align correctly.
- adjust the **IK Rotation and Position OffSet** on the **ShooterManager** to align the **LeftHandIK** of the weapons for your character.
- to create new **ShooterWeapons**, it's easier to just modify the prefab of one already setup by draging and drop into the scene and replace the model/values.

TOPDOWN INPUT

Currently this version does not support Topdown Shooter which is basically another controller and aim system so we will develop this features in the future.

HOW IT WORKS?

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

1- **vCharacter** takes care of Health/Stamina and has the method TakeDamage to apply damage.

2- **Third Person Motor** handles all the information of rigibody, colliders, verifications of ground distance, stepoffset, slope limit, etc…

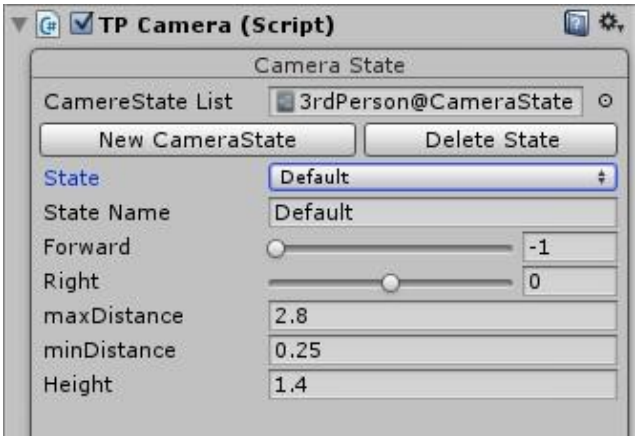
3- **Third Person Animator** is responsible to control the behavior of your animations, you can set bools, float, int and control the state of your animation.

4- **Third Person Controller** manage methods like sprint, crouch, roll, jump, etc…

5- **Third Person Input** receives all the input and call every method of the other scripts on Updates.

CREATING A NEW CAMERA STATE

On your 3rd 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.



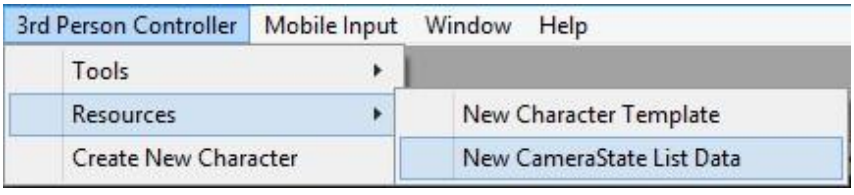
Then just change the CameraState on the method ControlCameraState() on the script TP_Motor.

Example:

if(aiming) tpCamera.ChangeState ("Aim", true);

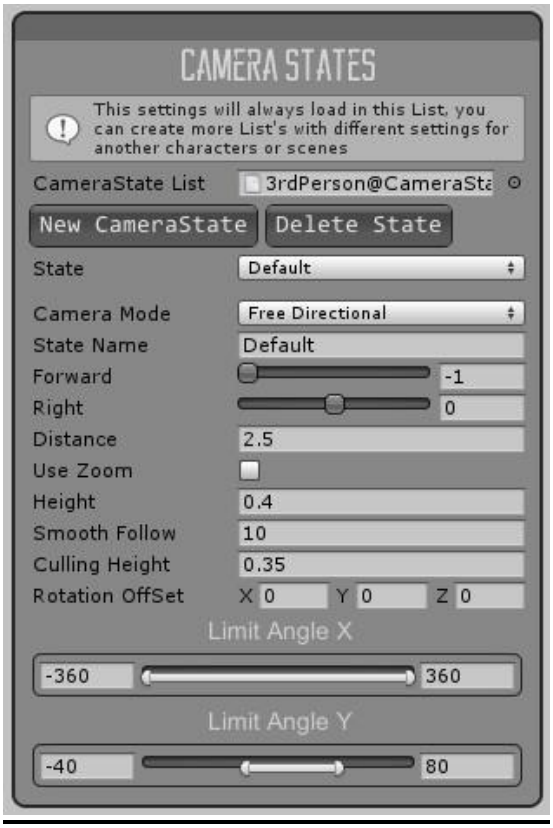
The first string value is the State Name that you created on the Camera Inspector, the second value is a bool, leave it true if you want a smooth transition to this state or false if not.

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.



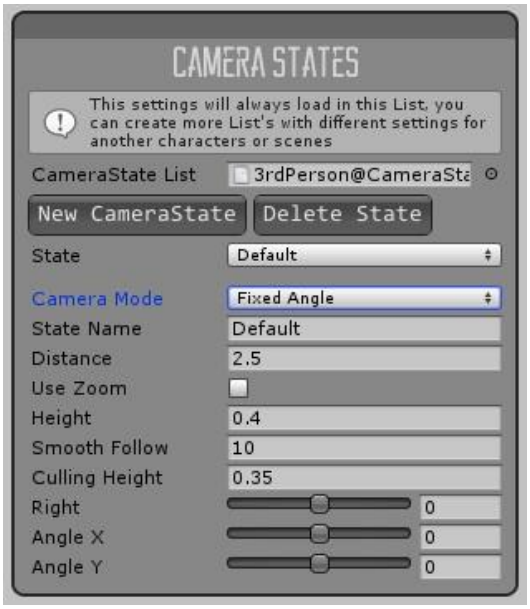
CameraMode - Free Directional

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…



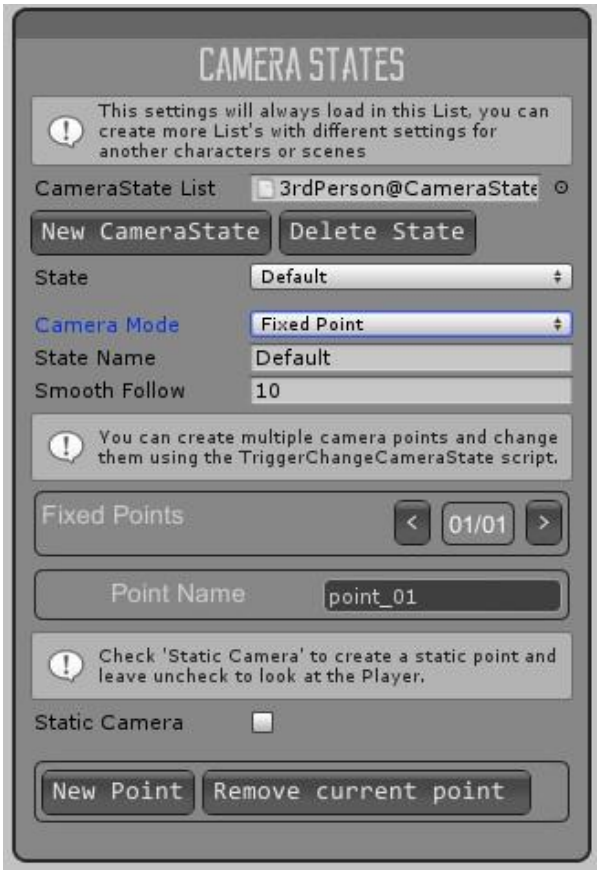
CameraMode - Fixed Angle

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.

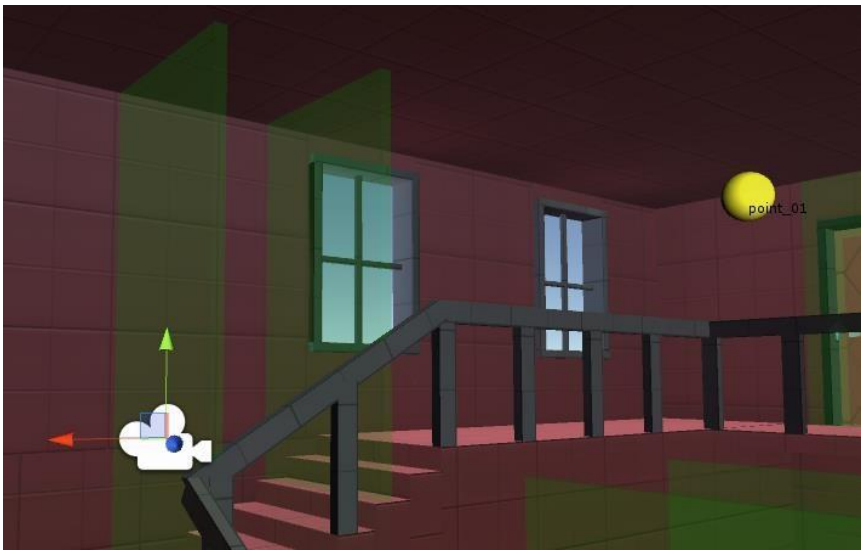


CameraMode - Fixed Point

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.

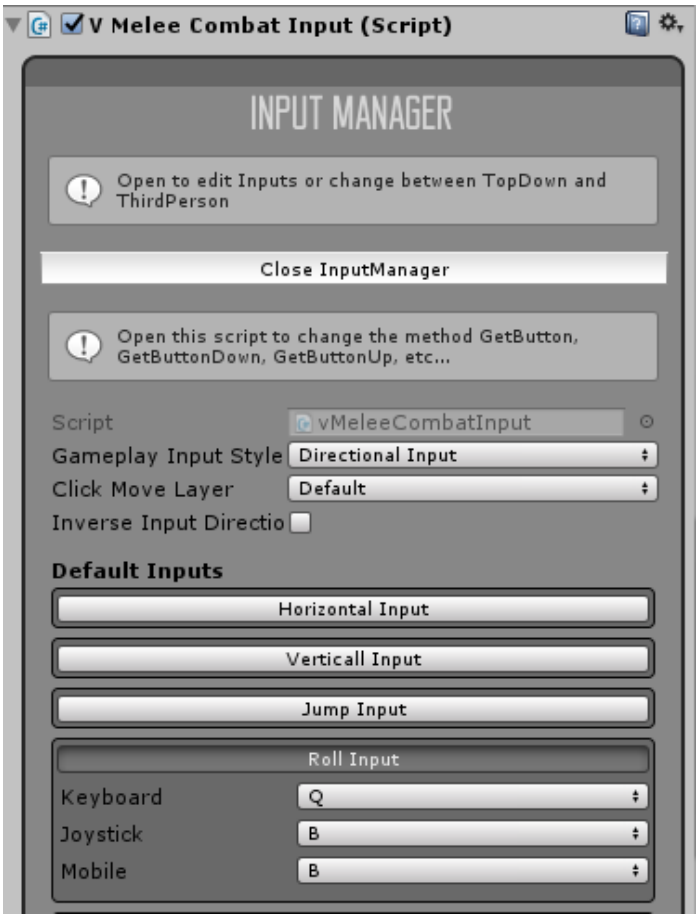
To apply the vibration, you can call the method by SendMessage to the player, for example:

target.SendMessage("GamepadVibration",0.25f,SendMessageOptions.DontRequireReceiver); The float value is the duration that you want for the vibration to last.

V1.1 add support for MFi iOS gamepad.

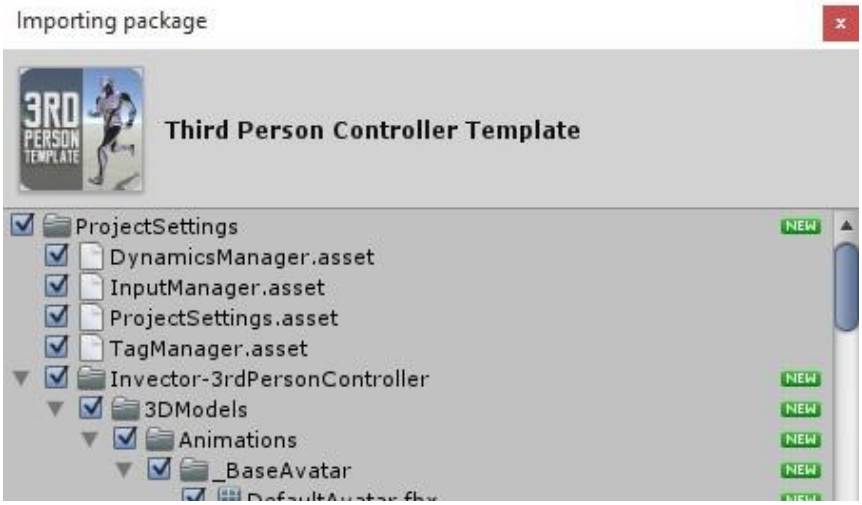
INPUTMANAGER

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.

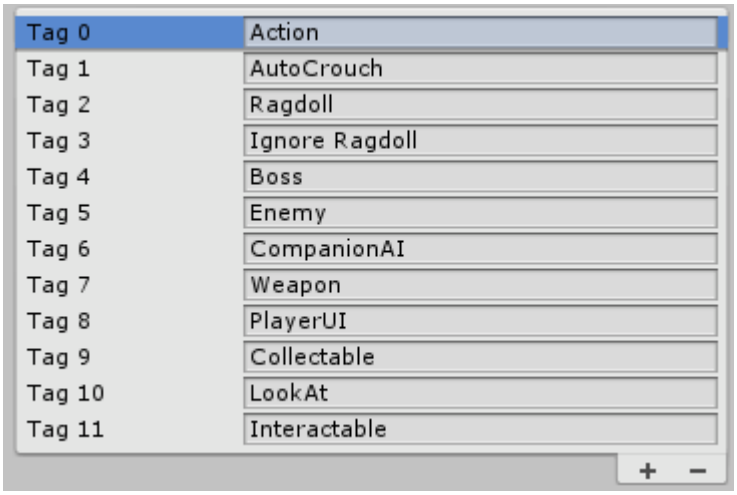


TAGS & LAYERS

V1.3 – If you will import the package into an existing project with your own tags and layers, you can uncheck the TagManager.asset and the system will automatically add our tags and layers without replace yours.

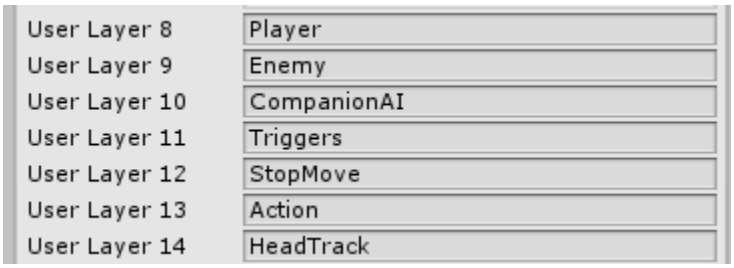


This is all the Tags we need to for the template work properly:



*Shooter uses the BodyPart layer, and new Tags for specific objects like Metal, Glass, Woods

Layers:



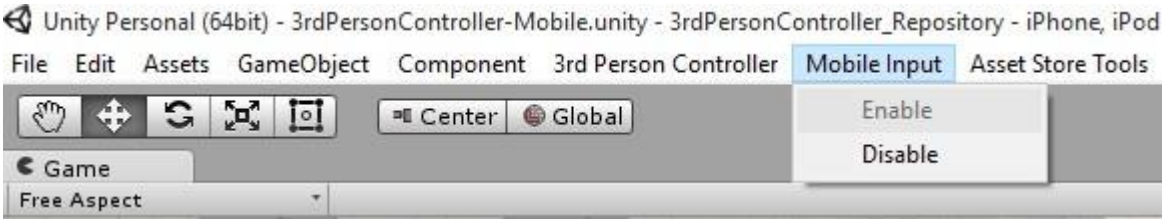
RECOMMENDED MOBILE SETTINGS

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.**

Change your platform to **Android** or **iOS** on the **Build Settings** and make sure you have the **SDK** installed.

Export the build with **ETC1** selected on Texture Compression and change your Shaders materials to **Mobile Diffuse** or **Legacy Diffuse** (this will improve performance a Lot in lower devices)

Don't forget to **Enable** the Mobile Input after change the platform, it should work right on the Editor.

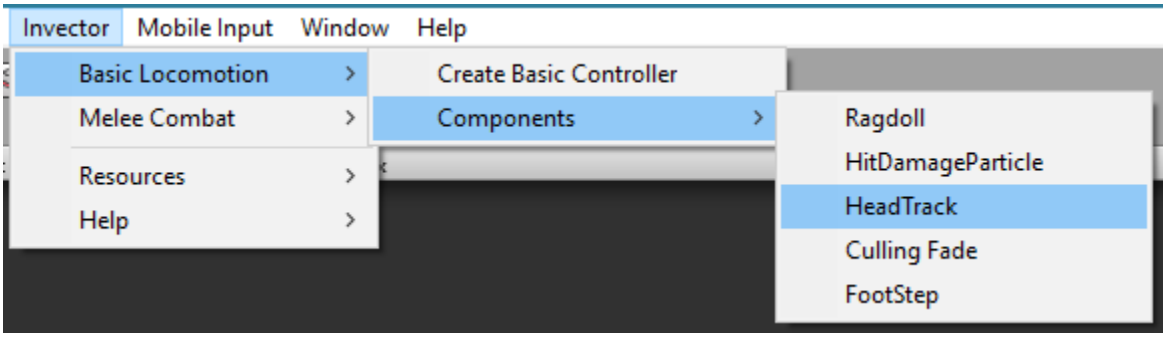


With these settings, we manage to get **stable 60fps** on several Android smartphones

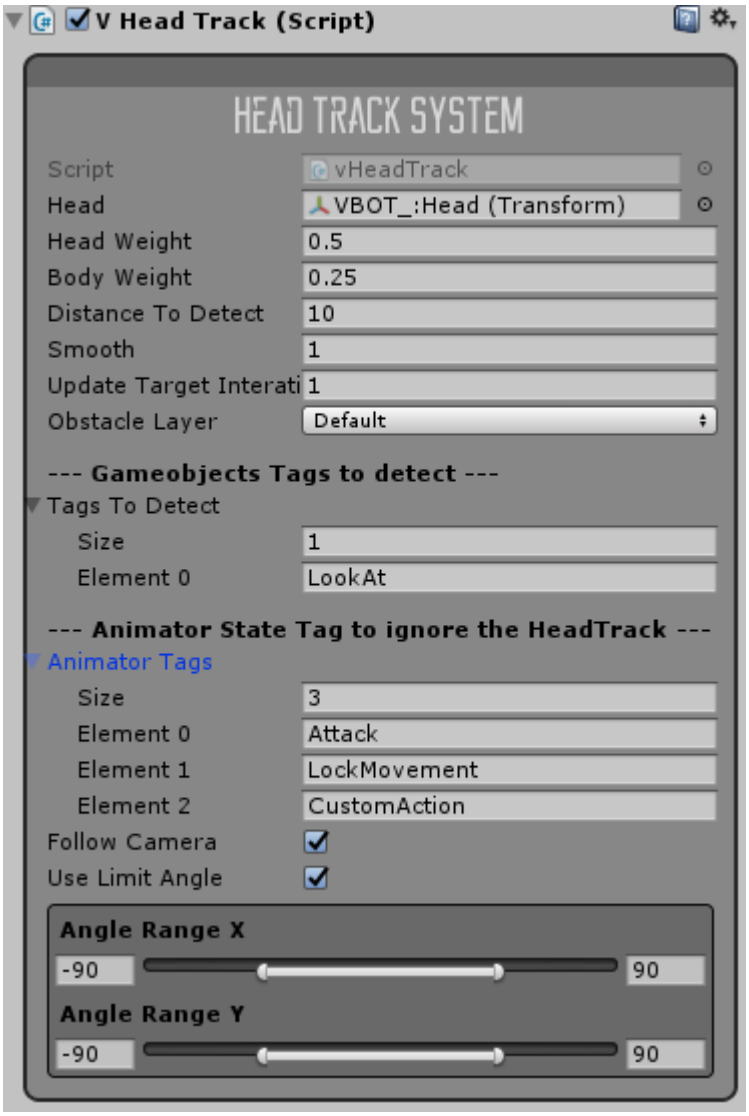
**Unity does no longer supports Tegra devices*

HEAD TRACK

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

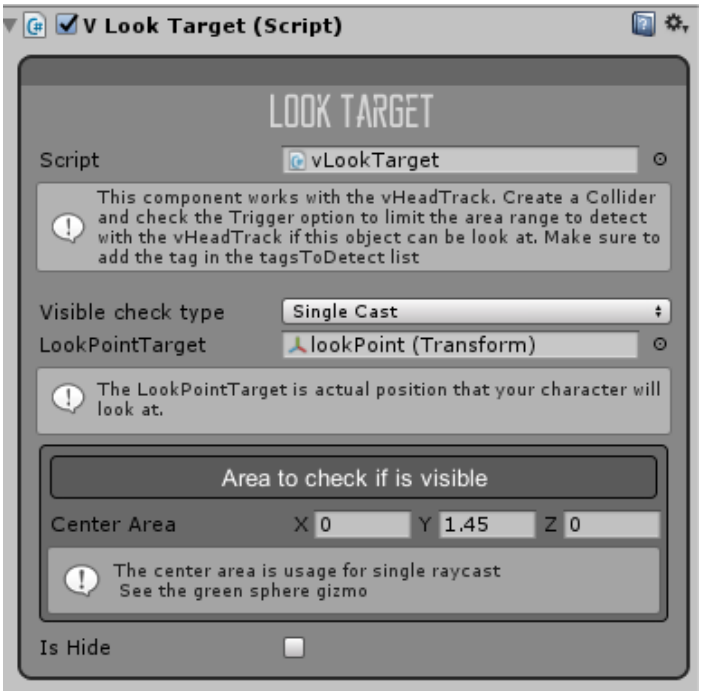


Now we have a lot of more options and we can use the LookAt feature as well.



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

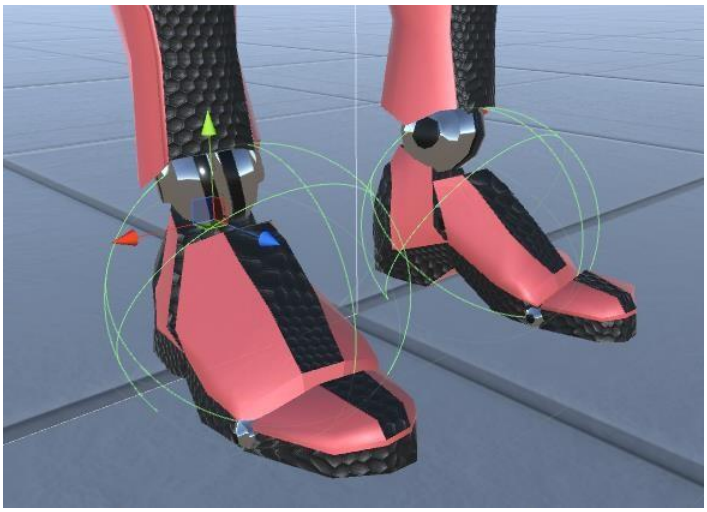
To make the character look at an object, you need to add the component vLookTarget into the object, you can take a look at several examples in the DemoScenes.



FOOTSTEP AUDIO SYSTEM

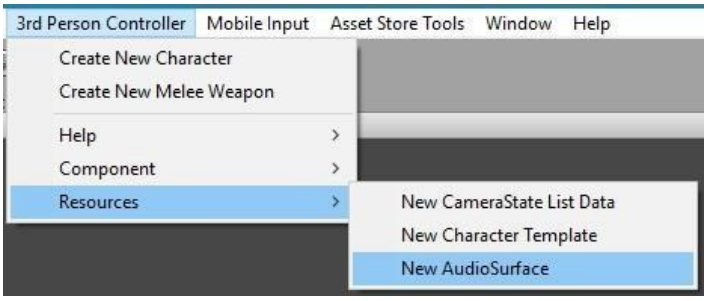
V1.2.1 New FootStep System.

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.



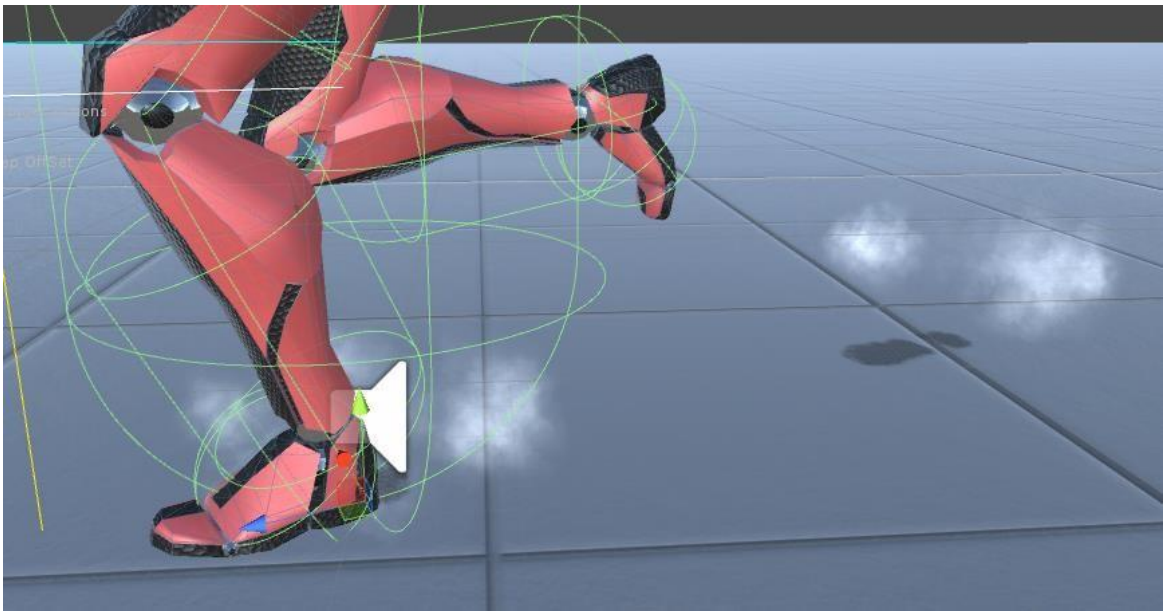
You can select the **LeftFoot** and **RightFoot** Sphere and manipulate the **Center XYZ** to position as you like, and change the **Collider Radius** too, the size of this sphere will depend on your Rig bone size. Assign the "***defaultSurface***" that comes with the package to have an example of how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

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



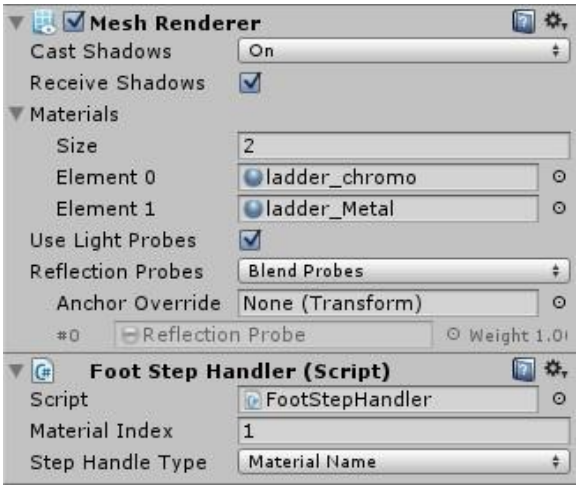
Now you can create **Custom Surfaces**, to play other audioclips based on the **material** that the sphere collider will hit. Assign the new CustomSurface to a new CustomSurface on the FootStep Inspector.

You can assign a **AudioMixer** for better control the 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

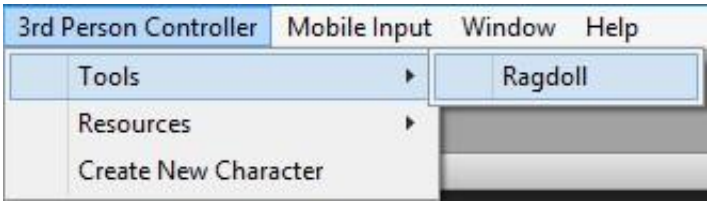
If your gameobject has multiple materials and you need to play a specific material, you can use the FootStepHandler script and set the correct Material Index of your object. (*See example on the Ladder prefab)



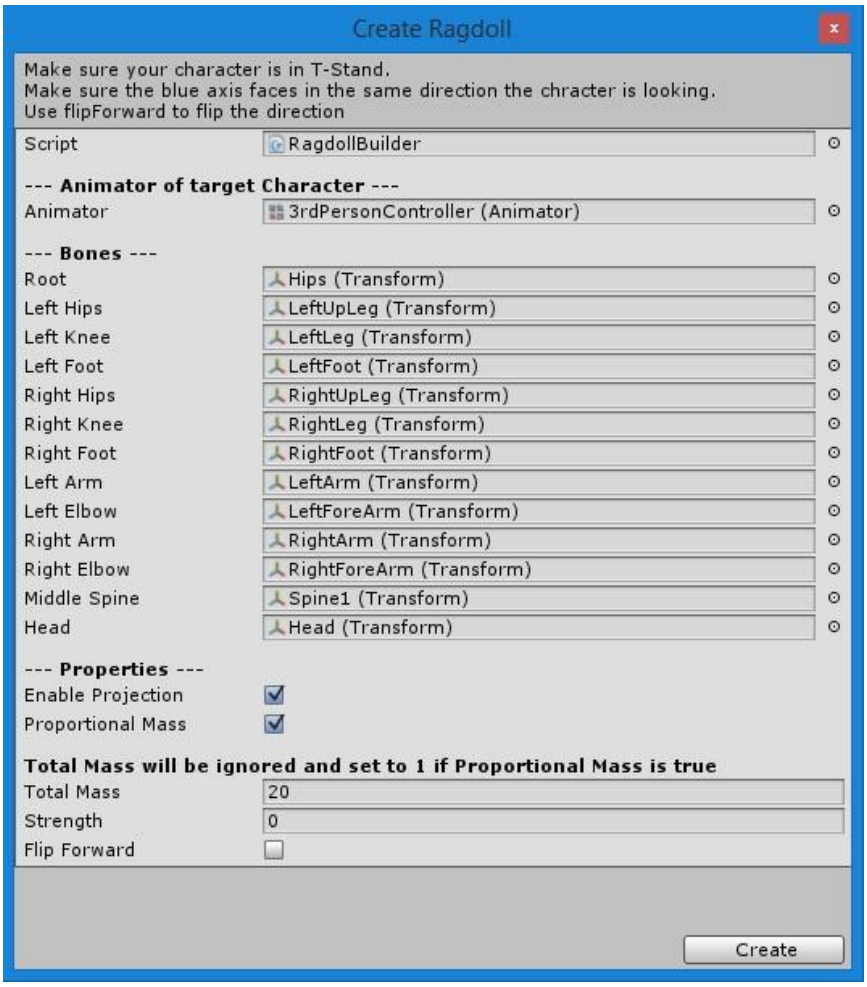
CREATING A RAGDOLL

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

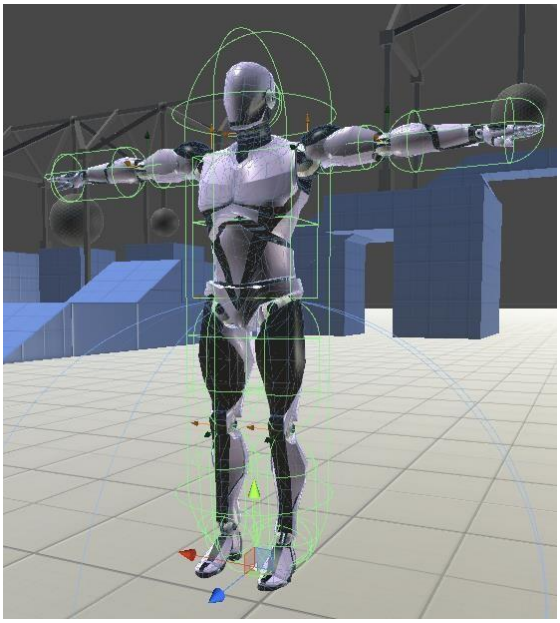
Creating a Ragdoll is just easy as creating your Character, just go to the tab "3rd Person Controller" > "Tools" > "Ragdoll".



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.

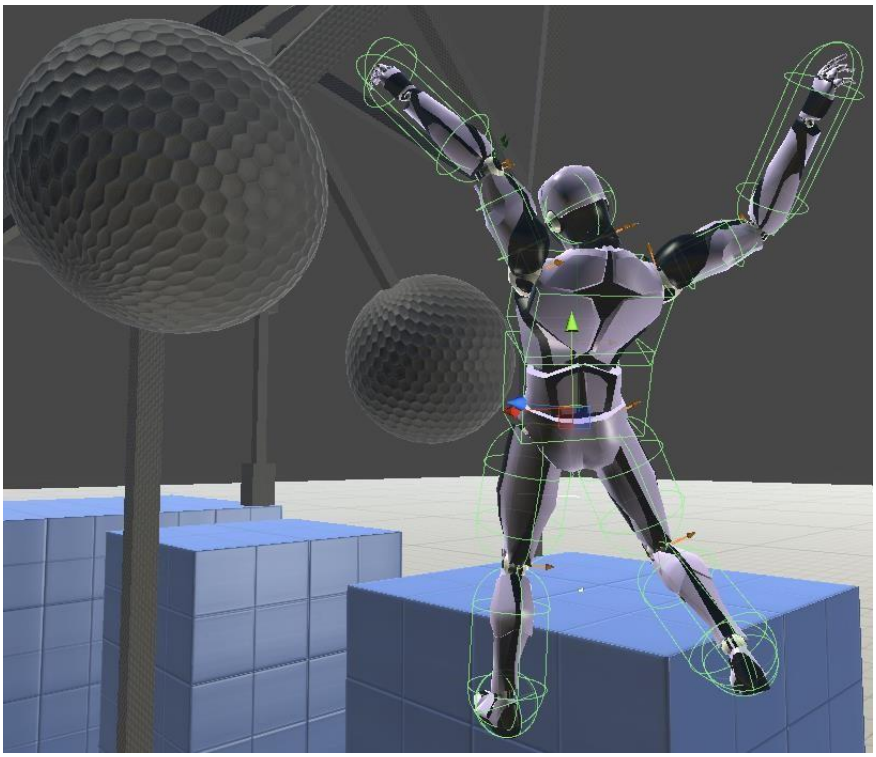


We strongly recommend keep the **Enable Projection** and the **Proportional Mass** enabled, and do not forget to use **Scale Factor 1** on your **fbx** Model. This you provide better behavior of your ragdoll.



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.

HOW TO ADD NEW ANIMATIONS?

The process is:

Set your animation clip as Humanoid and retarget to your T-Pose character

If it is an action like open a door, put the animation on the Action State of the Animator.

At the Motor script, create a variable like a bool to control the animation

At the AnimationControl script, tell what variable controls what animation

At the Controller script, run the method to trigger the animation

Here is a Video Tutorial showing the process to apply a Jump Animation:

[How to add new animations]

[Replacing Animations]

This is just an example, but of course, you may have to prepare the script to what your new actions are going to do, just as we prepare for the jump animation example.

RAYCAST CHECKERS (V1.1)

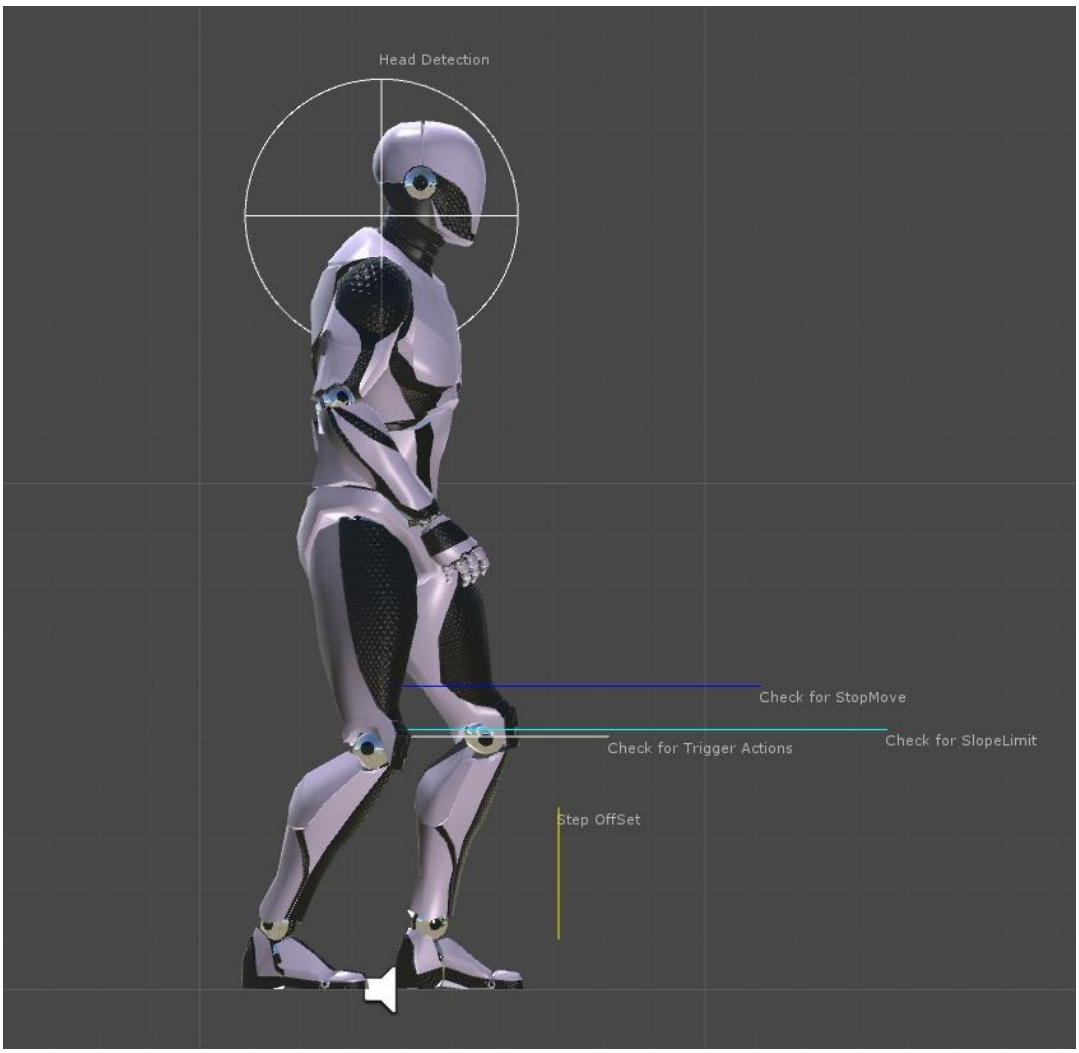
Head Detection is a SphereCast that will detect if has an object above, and keep the character crouched, use the same layer as the Ground Layer (Default). Just adjust to sync with the height of your capsule collider.

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.

Trigger Actions is the raycast that check for objects with the component TriggerAction, you can trigger a specific action based on the tag, display information and pass a specific transform position and rotation using matchTarget.

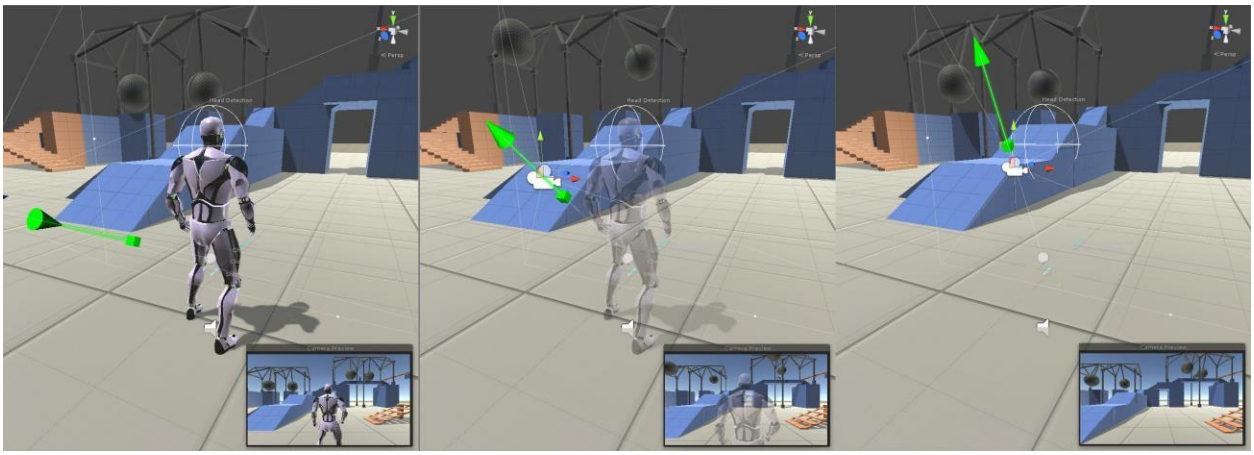
StepOffset is to help the character walk in custom height steps, adjust the values on the Player Inspector.



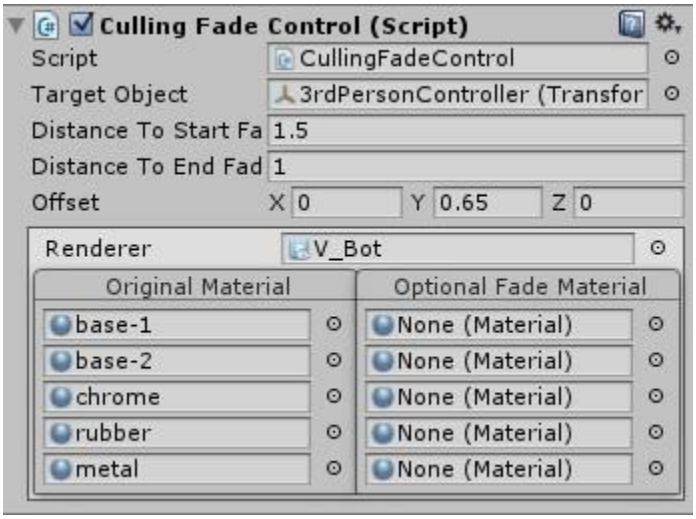
CAMERA CULLING FADE (V1.1A)

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.

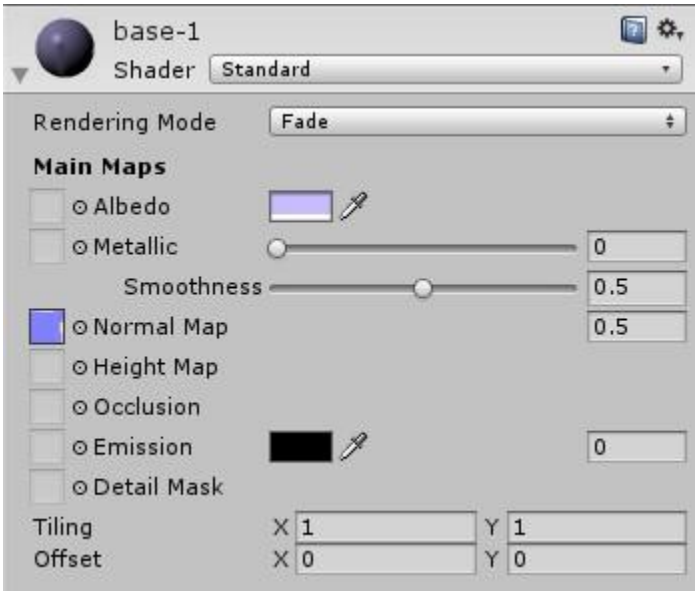
Example:



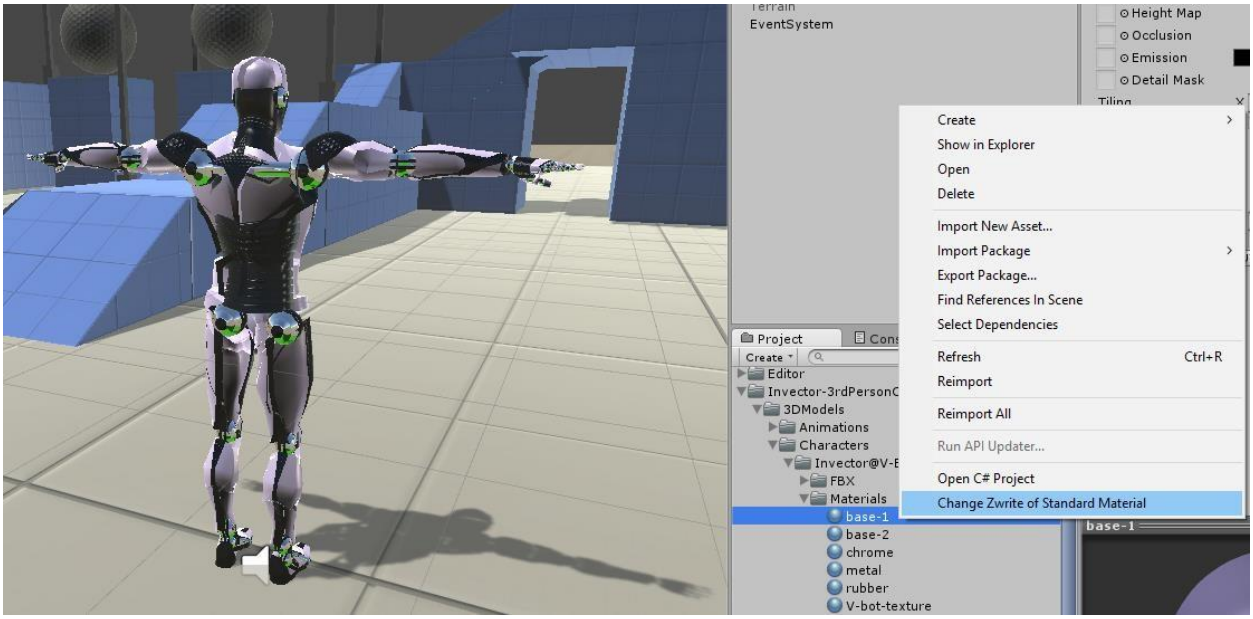
Our Culling Fade will set up automatically for the default Stardard Shader of Unity's, but you also can use custom shaders, just make an additional copy with the fade material and assign in the "Optional Fade Material" field.



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



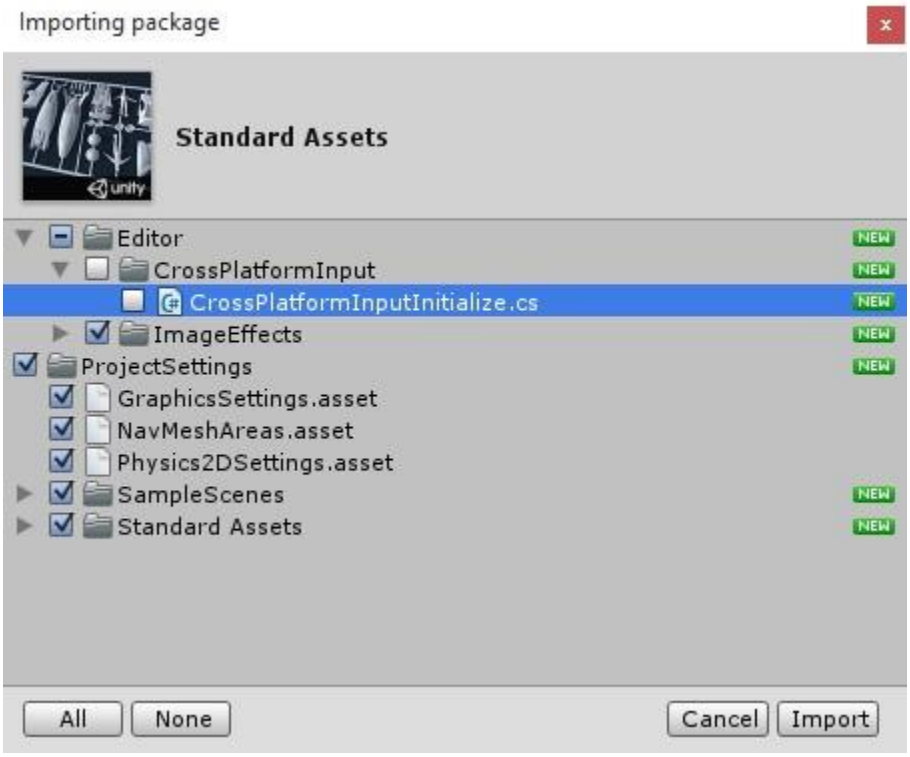
The character will look like this (picture below) but you can fix by right clicking at the material and “Change Zwrite of Standard Material”.



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

WORKING WITH STANDARD ASSETS

Our template already comes with the **CrossPlataformInput** and the **ImageEffect AntiAliasing** imported, so if you want to import the **Standard Assets** package into the project, just make sure to ***UNCHECK*** the following item:



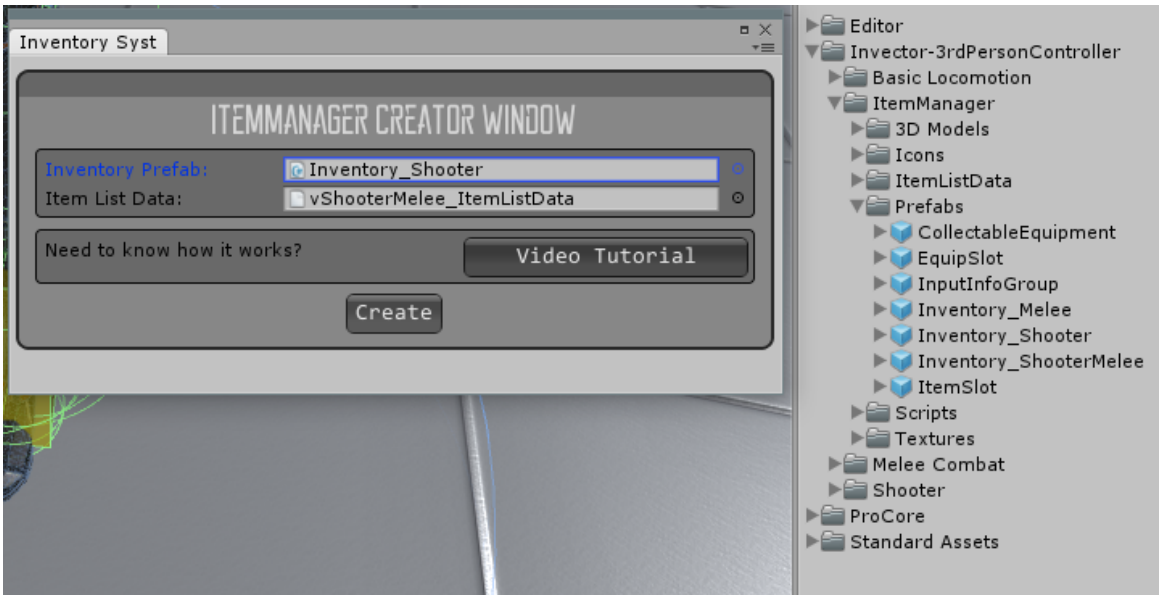
If you imported by mistake and are getting some errors, try deleting the folder **Editor** and the folder **Standard Assets** and ***reimport*** the Standard Assets package again, the errors will be gone. [Video Tutorial]

USING MELEE WITH A SHOOTER WEAPON

To use your Shoter weapon as a Melee, take a look into the documentation of the Melee Combat Template, it's the same process and you can also look into the ShooterMelee demo scene to see how it works.

ITEM MANAGER

To add an Item Manager, select your Player and go to Invector > Inventory > Item Manager.
Select the prefab of the inventory you want to use and the ListData of your items.



After that the Item Manager Component will be attach into your Character.

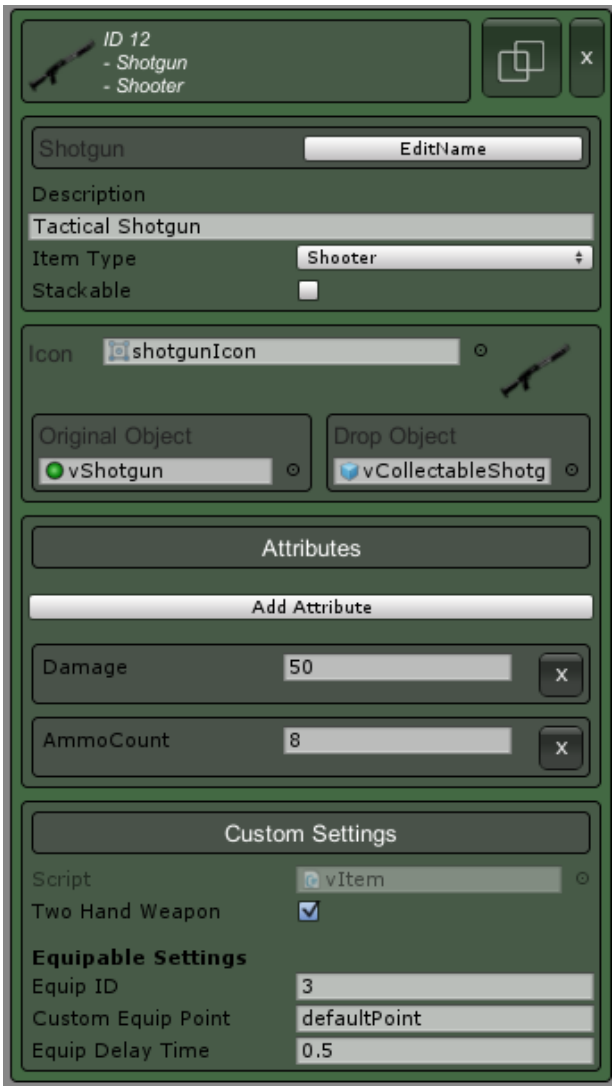
The **ItemListData** is a resource file, and you can create a new one by going into Invector >Inventory > NewI temListData, just like the CameraState ListData or the Audio Surface.

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

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

By clicking in **Open Item List**, a new window will open and you can create new items there.

When creating a Weapon Item, you need to assign the **Original Object** (that instantiate into the Player with a vMeleeWeapon) and a **DropObject** which we have a prefab called "**CollectableEquipment**" that you can use.



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 ammount of weapon, making it into a unique weapon.

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

CREATING AN SHOOTER ENEMY AI

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

SHOOTER - Make sure to create a ragdoll for your enemy and change the layer to BodyPart so you can apply damage into the members colliders and not the main capsule collider.

After hit 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.

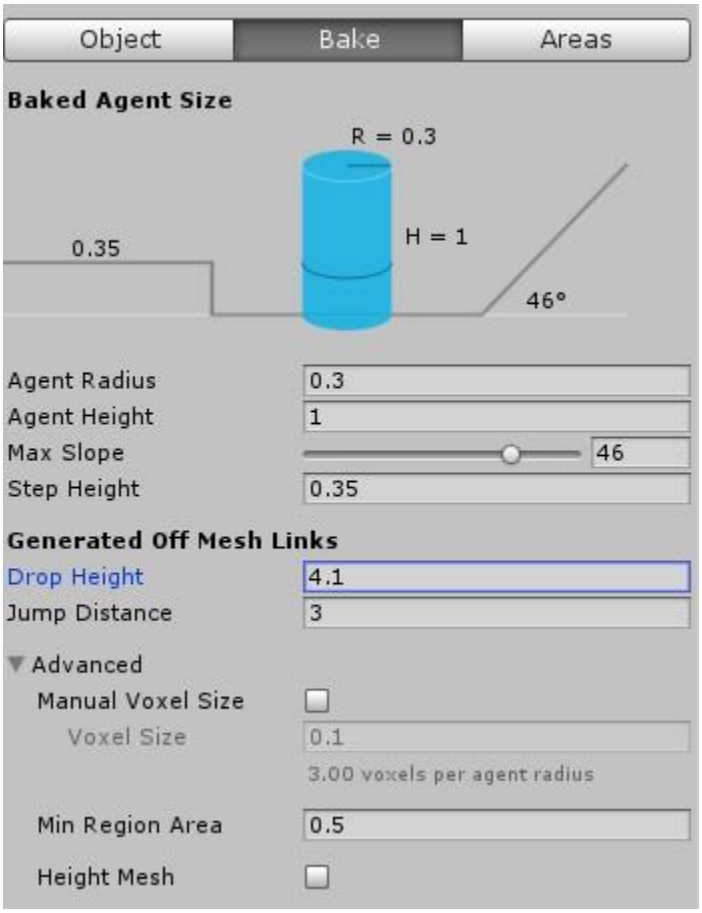
Locomotion - It works pretty much the same as the Character Controller, you still need to set up the

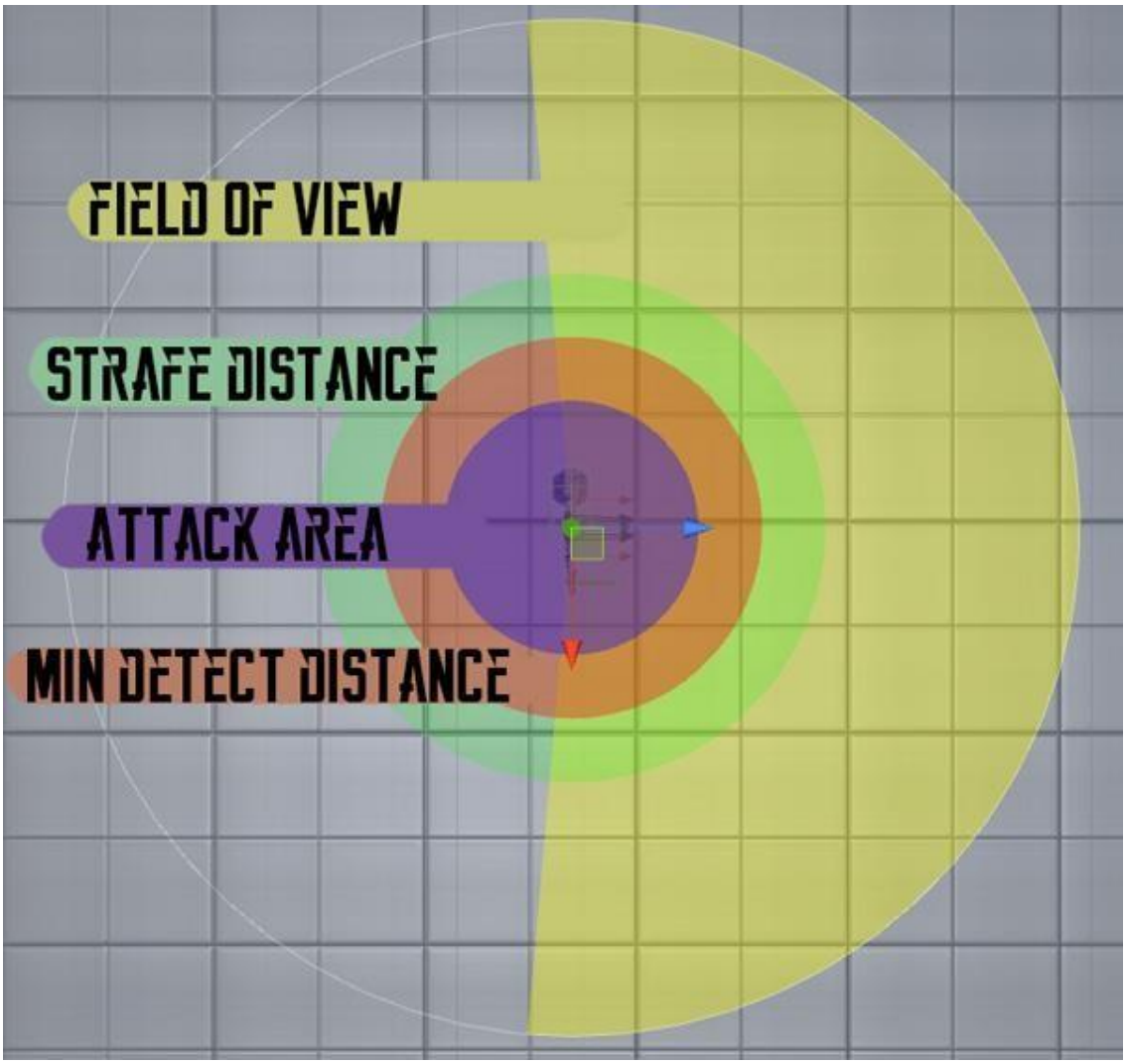
Layers – just like the Player, you need to set up a Layer for the AI (Enemy) and a layer for the Ground (Default). If you equip a Weapon on the character, don't forget to assign the Layer Triggers for this weapon.

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 of activate the option to Random.

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





Field of View > total range to detect the Player

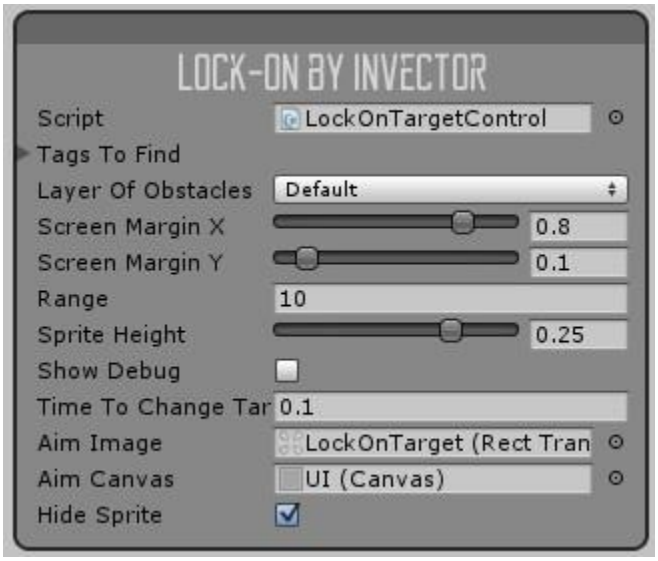
Strafe Distance > once in combat, the character will move Strafing

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 activate the Lock-on in the **ThirdPersonController** script, at the method **LockOnInput.**



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

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

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

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