AAA 3rd Person Controller – Melee Combat 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.

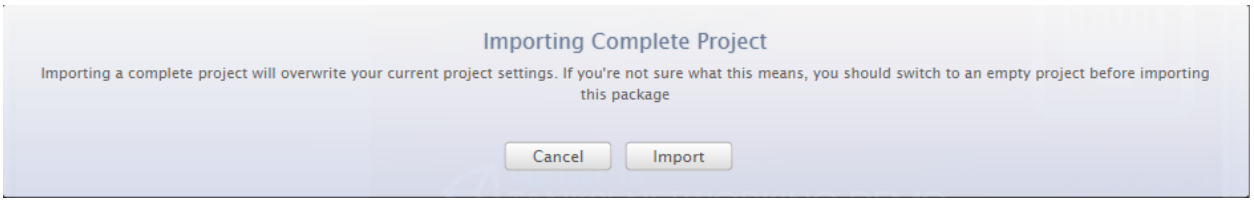
Invector Team

Summary

First Run

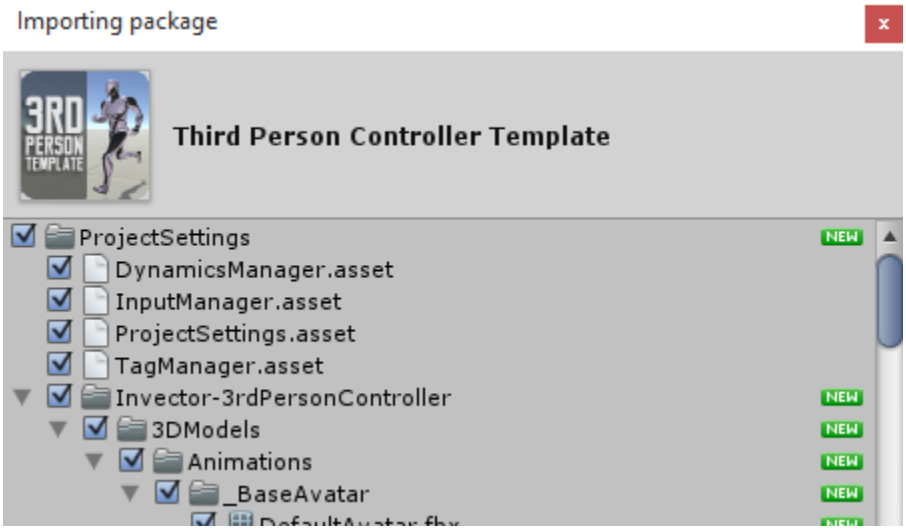
IMPORTANT

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 already existent project*

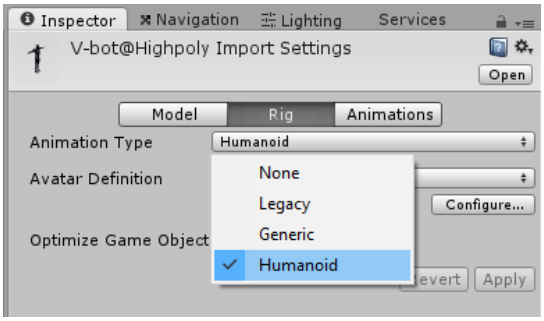
If you want to import into another project, you can UNCHECK some project settings to avoid conflicts or replace your project settings like the TagManager (which includes all the Layers), and add later the tags and layers that we use. We recommend to import the InputManager because it's kind of painful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



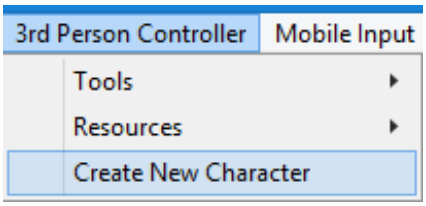
***Updates also need to be imported into a Clean Project, so MAKE SURE TO BACKUP your previous project and transfer the necessary files to your new project. ***

Creating a new Character

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

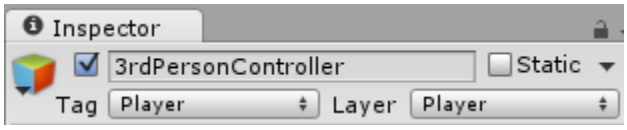


To setup a new character, go to the tab "3rd Person Controller" and click "Create New Character"

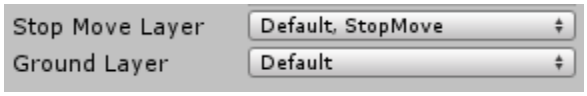


Make sure your Character is **Fully Rigged** and set up the FBX as a **Humanoid**, then assign the FBX to field "Humanoid" and click on the button "Create 3rd Person Controller".

1- ***ADD V1.1 -*** Make sure to set up a different **Layer** for the **Player** and the **Ground Layer**, if your project does not contain this Layers, create then on Edit > Project Settings > Tags and Layers.



When create a new character, this 2 layers will be set as “Nothing”, just replace like this on the Player Inspector:

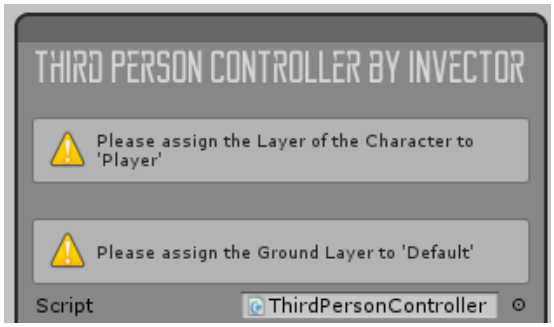


2- **Done**.

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.

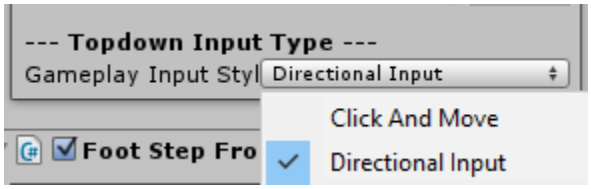
ADD V1.2 - The character inspector will show warnings accusing if the Layers are not correct



3- Hit Play and enjoy ☺

V1.1 Additional Info:
3rd Person and **2.5D** bot use the **3rd Person Controller Template**, just change the CameraStateList.

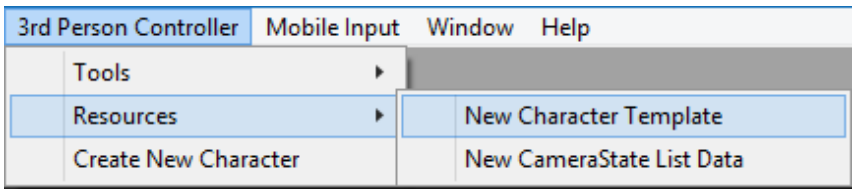
Topdown and **Isometric** both use the **Topdown Controller Template**, you can change to input style to Mouse and Click or control with inputs on the Player Inspector.



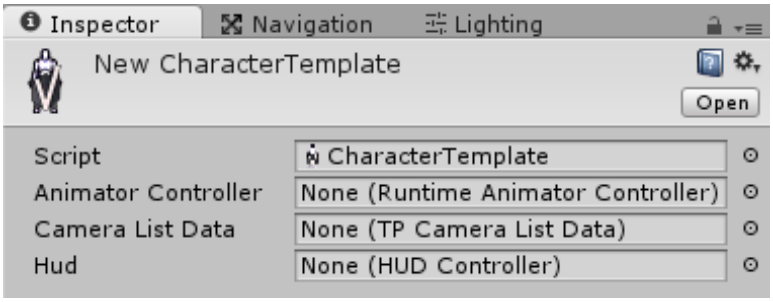
Also, remember to **Uncheck** the **HeadTrack** option at the Player Inspector if you are using **Topdown** or **Isometric** mode.

Creating a new Character Template

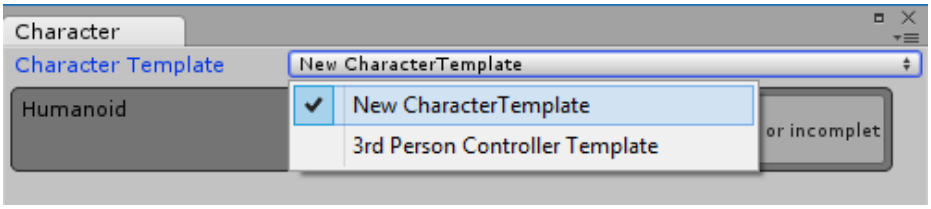
You can set up the Character Creator to create **custom controllers** that you have modified, to create a new template go to "3rd Person Controller" tab, "Resources" and click on "New Character Template".



Assign your **modified prefab** of an **Animator Controller**, **Camera List Data** and the **HUD Controller**.



The next time you create a new controller, choose the new template.



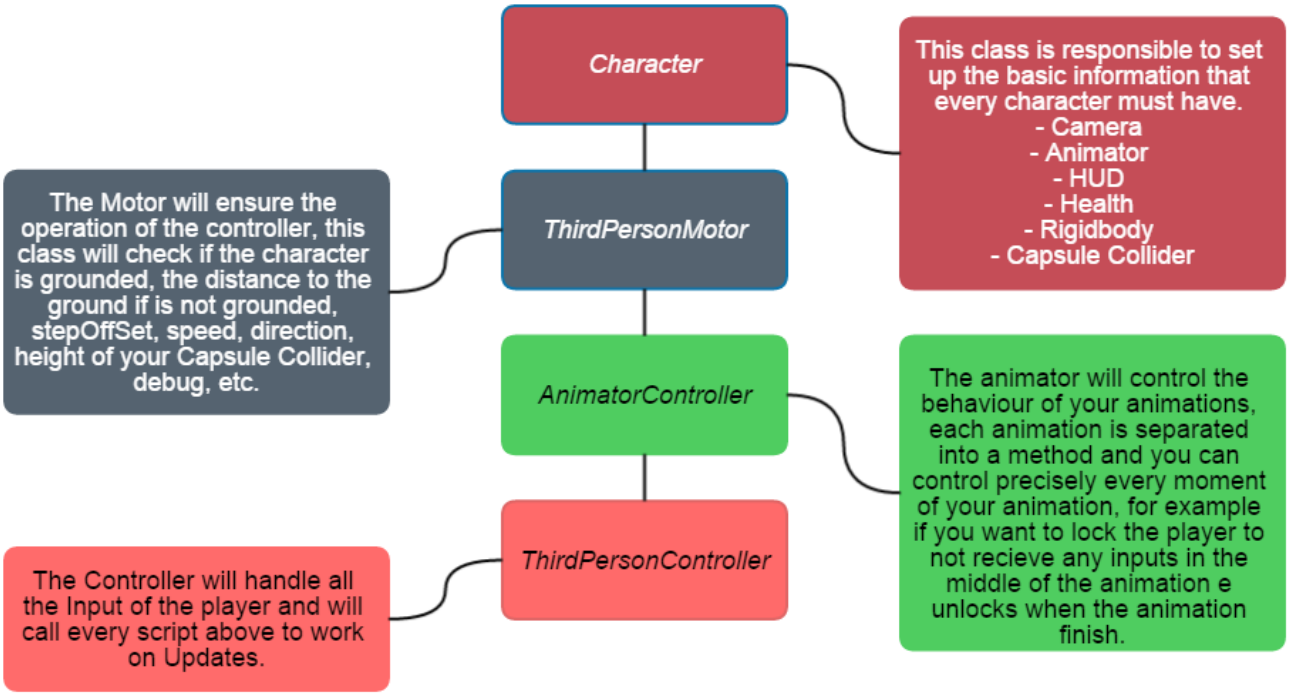
How it works?

The Controller works with **four main scripts**, Character, Motor, Animator and the Controller.

Character will take care of the basic components in order to make the controller works, things like a Camera, Animator, Rigidbody, Capsule Collider, etc..

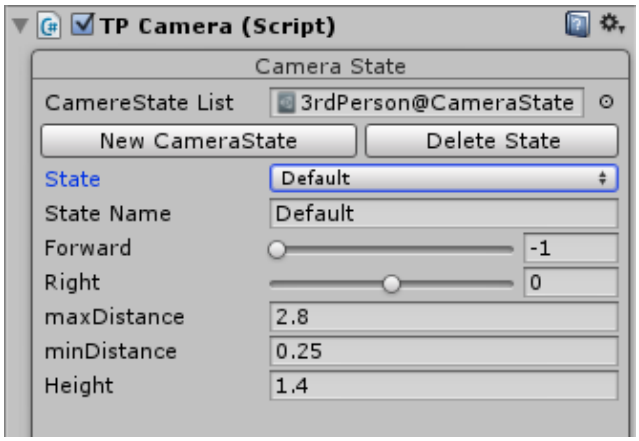
Third Person Motor handles all the verifications of ground distance, stepoffset, slope limit, etc..

Animator Controller is responsible to control the behavior of your animations, you can set set bools, float, int and control the state of your animation. **Third Person Controller** receives all the input and call every method of the other scripts.



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.

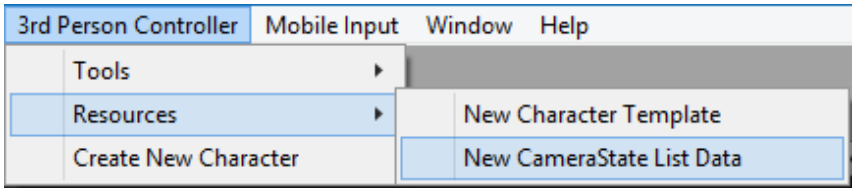


Then just change the CameraState on the method ControlCameraState() on the script TP_Motor.

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

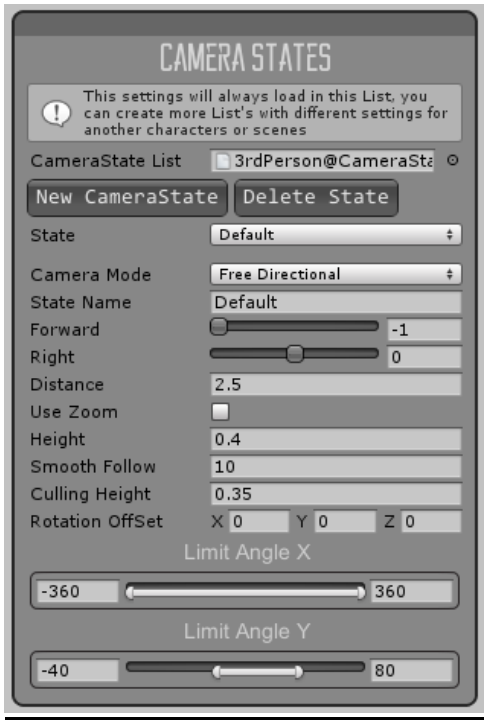
The first string value is the State Name that you created on the Camera Inspector, the second value is a bool, leave it true if you want a smooth transition to this state or false if not.

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.



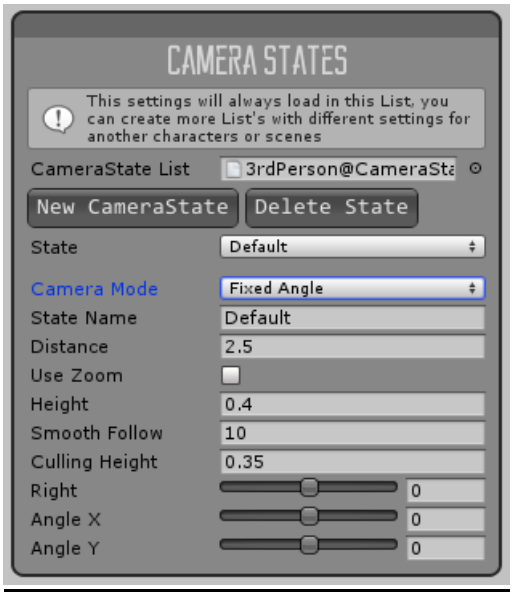
CameraMode – Free Directional

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

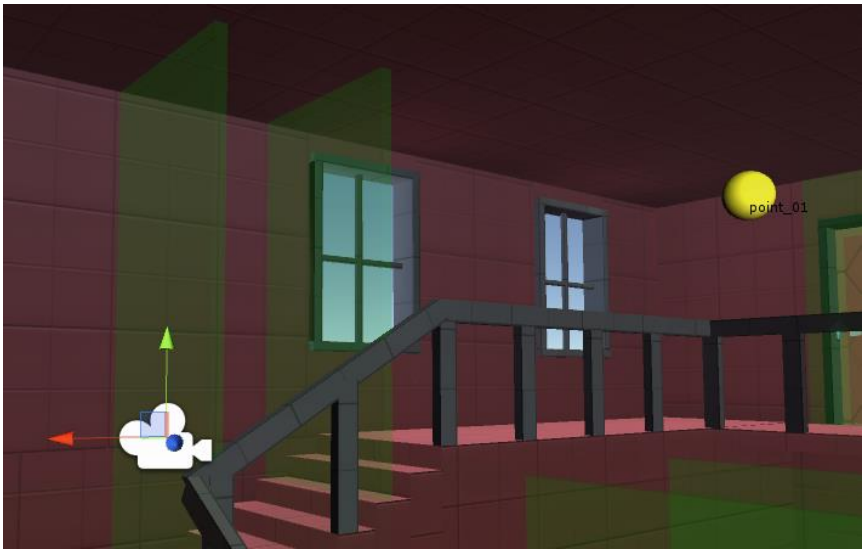
This is a feature to use for Isometric or Topdown games, you can set up a fixed rotation for the camera and make games like Diablo or MGS 1.



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 360 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.

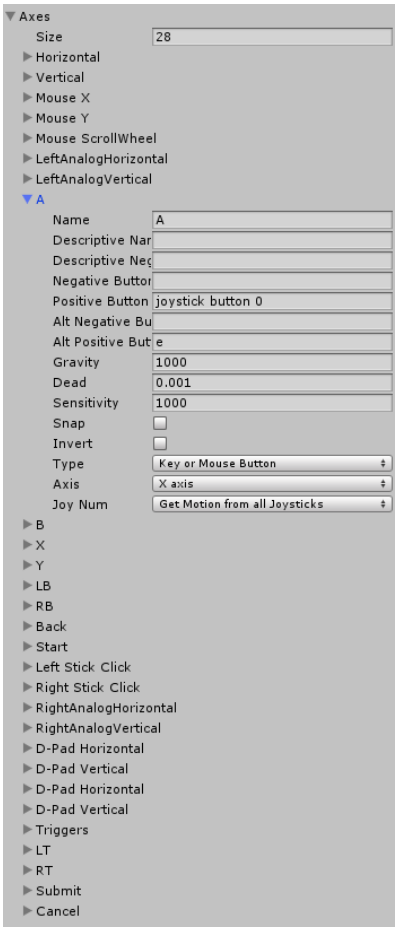
To apply the vibration, you can call the method by SendMessage to the player, for example:

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**.**

Input Manager

Our input manager is mapped with the 360 controller and use alternative buttons for the Keyboard, you can modify the inputs on the Input Manager of Unity. All the inputs are control by the ***ThirdPersonController*** script.



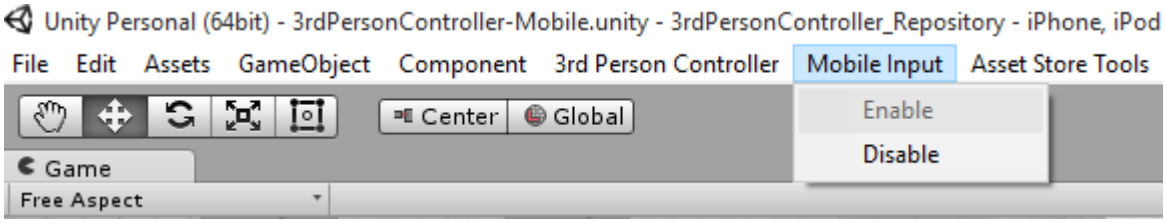
Recommended Mobile Settings

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 a Lot in lower devices

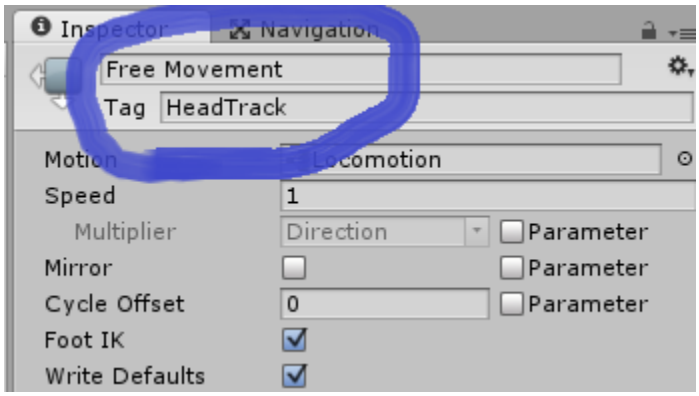
Don't forget to **Enable** the Mobile Input after change the platform, it should work right on the Editor.



With these settings, we manage to get **stable 60fps** on several Android smartphones
**Unity does no longer supports Tegra devices*

Head Track

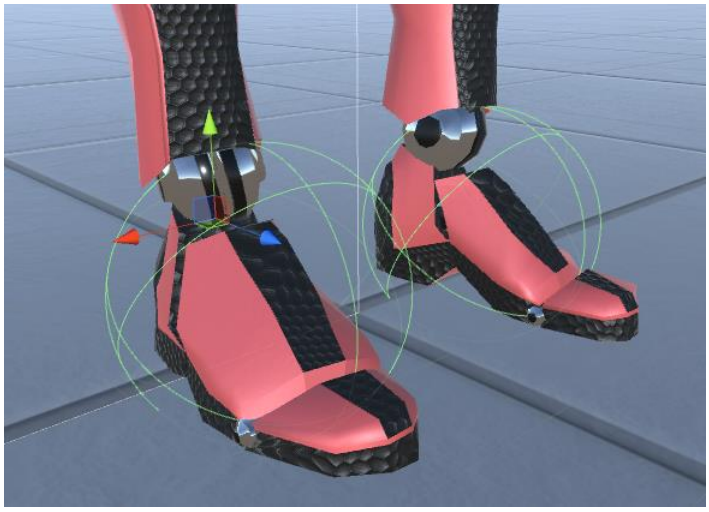
To enable the head track, you need to apply the Tag "HeadTrack" into the state clip on the Animator Controller, usually the main locomotion states. It's best to disable the headtrack on games like topdown/isometric for obvious reasons it will not work properly.



FootStep Audio System

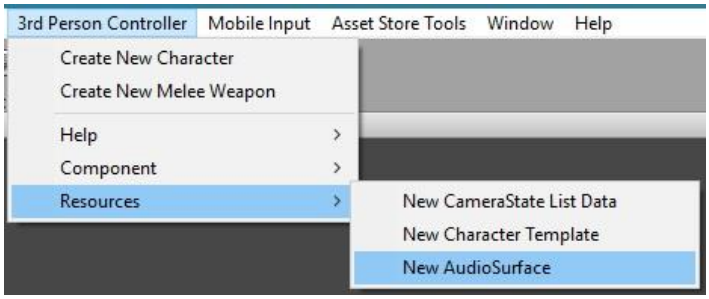
V1.2.1 New FootStep System.

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.



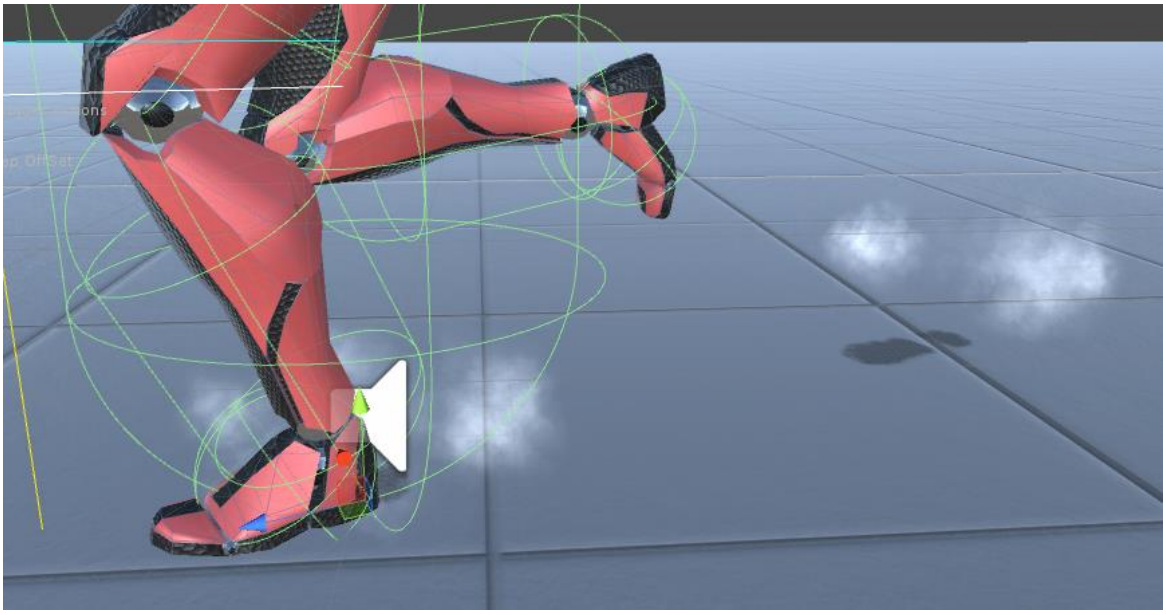
You can select the **LeftFoot** and **RightFoot** Sphere and manipulate the **Center XYZ** to position as you like, and change the **Collider Radius** too, the size of this sphere will depend on your Rig bone size. Assign the "***defaultSurface***" that comes with the package to have an example of how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

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



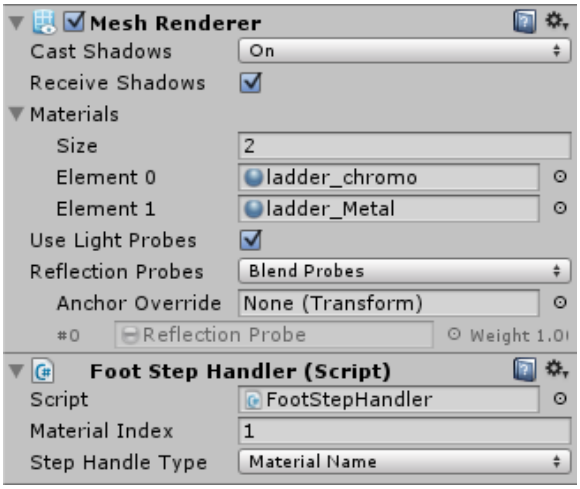
Now you can create **Custom Surfaces**, to play other audioclips based on the **material** that the sphere collider will hit. Assign the new CustomSurface to a new CustomSurface on the FootStep Inspector.

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.



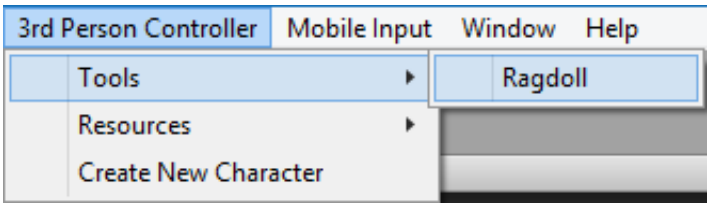
V1.1 Using the FootStep system in objects with multiple Materials

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)

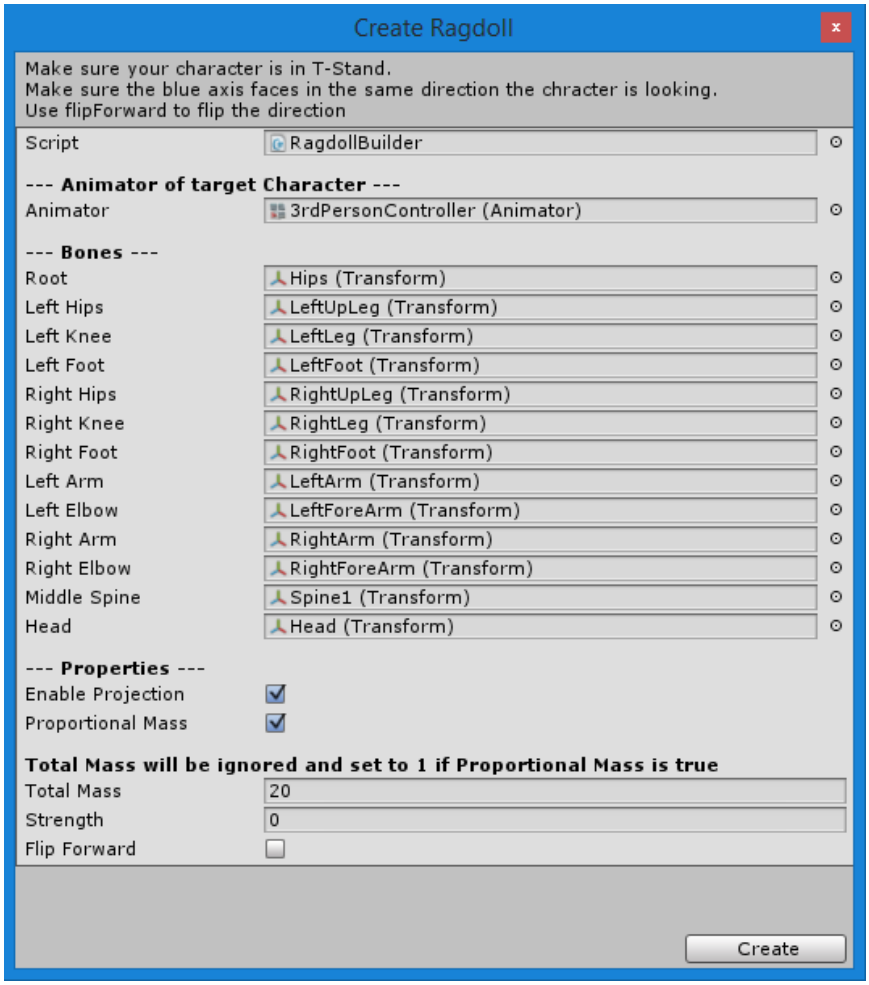


Creating a Ragdoll

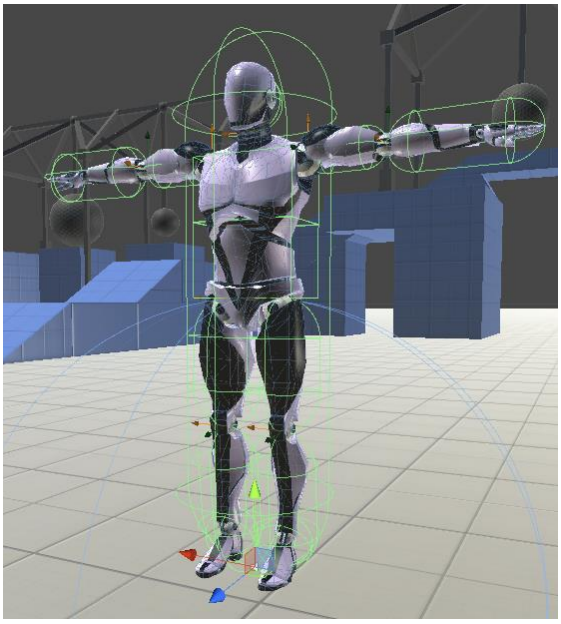
Creating a Ragdoll is just easy as creating your Character, just go to the tab “3rd Person Controller” > “Tools” > “Ragdoll”.



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.

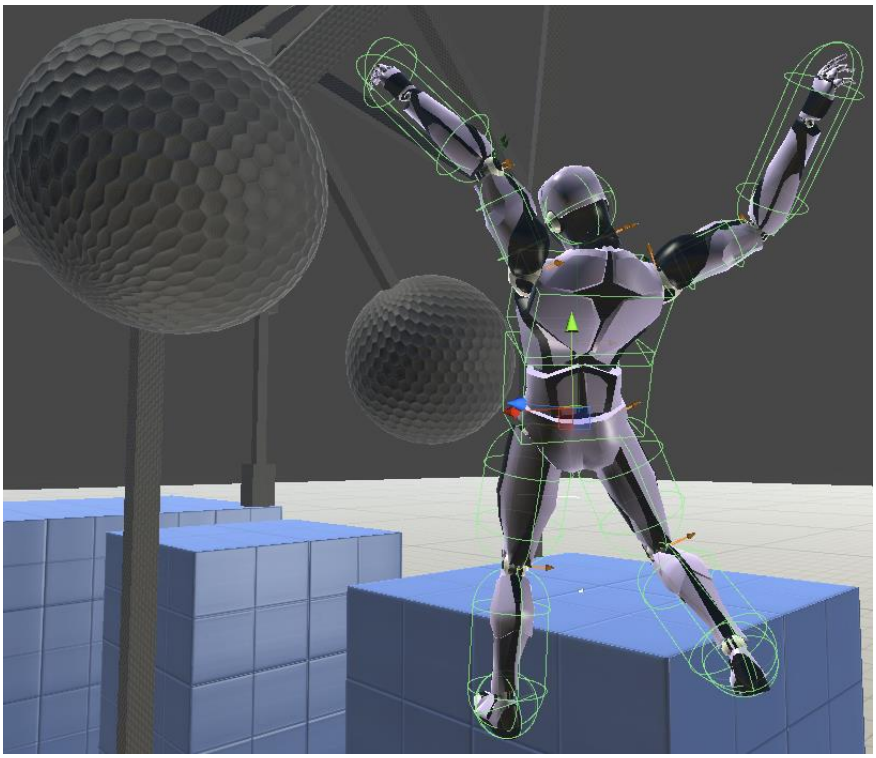


We strongly recommend keep the **Enable Projection** and the **Proportional Mass** enabled, and do not forget to use **Scale Factor 1** on your **fbx** Model. This you provide better behavior of your ragdoll.

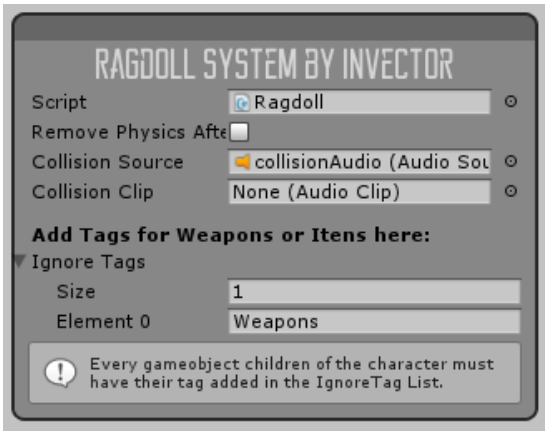


To enable the ragdoll, you can use the Script **ObjectDamage** or just call this line on the **OnCollisionEnter** method.

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:

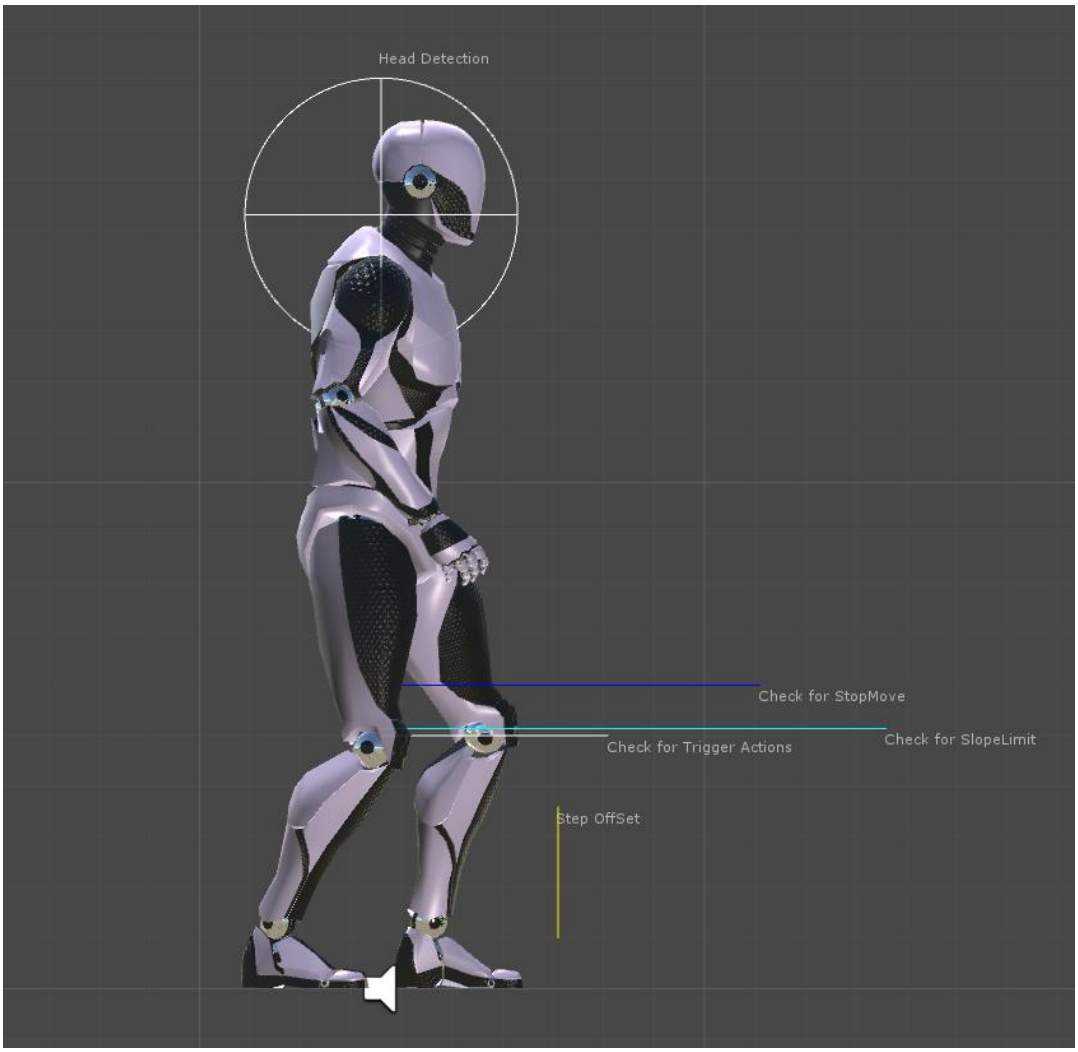
[How to add new animations]

[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)

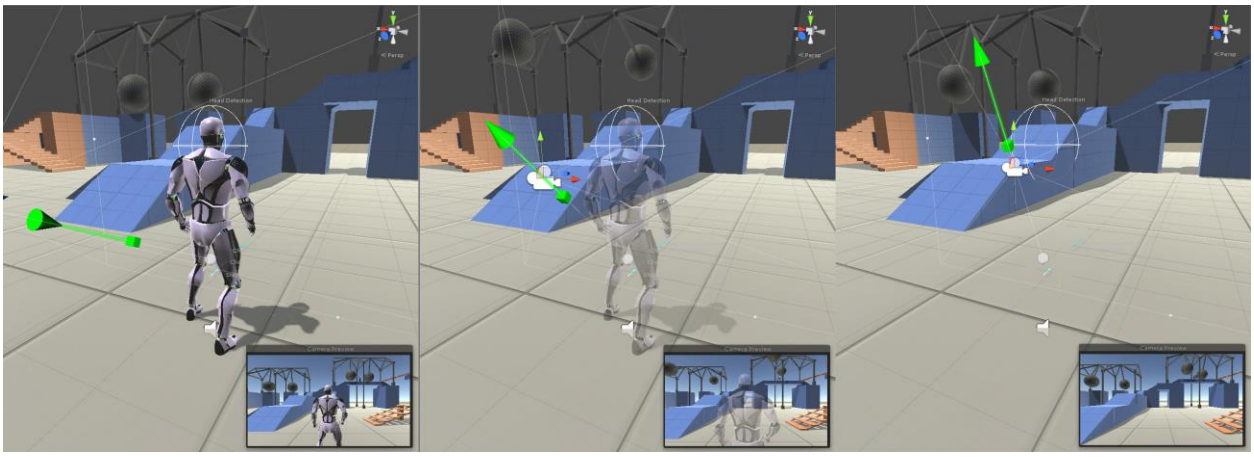
1- **Head Detection** is a SphereCast that will detect if has an object above, and keep the character crouched, use the same layer as the Ground Layer (Default). Just adjust to sync with the height of your capsule collider.
2- **StopMove** is a Raycast that detect any object with the layer (Default, 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”.
3- **SlopeLimit** will prevent the character of walking in absurd angle heights, float customizable on the Player Inspector.
4- **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.
5- **StepOffset** is to help the character walk in custom height steps, adjust the values on the Player Inspector.



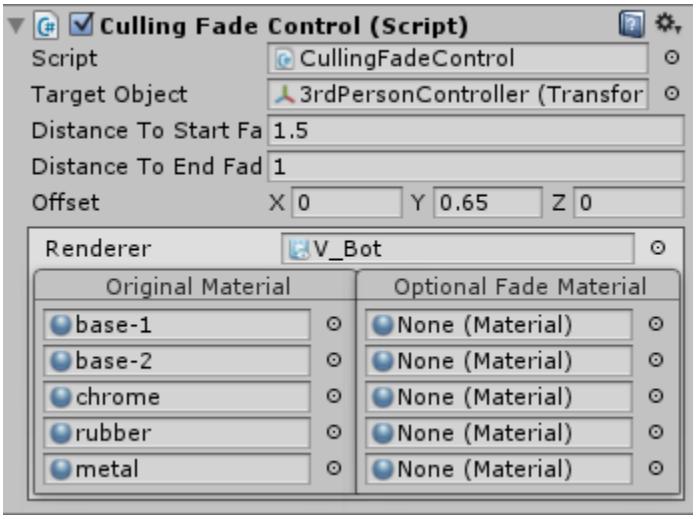
Camera Culling Fade (v1.1a)

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.

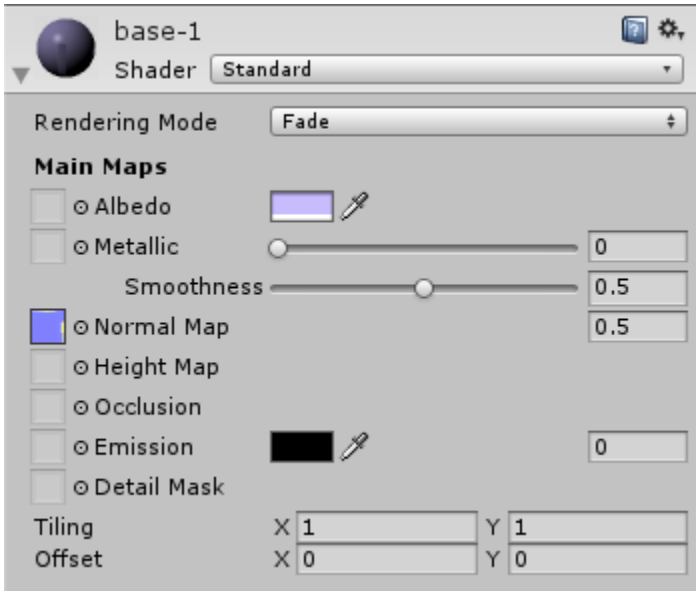
Example:



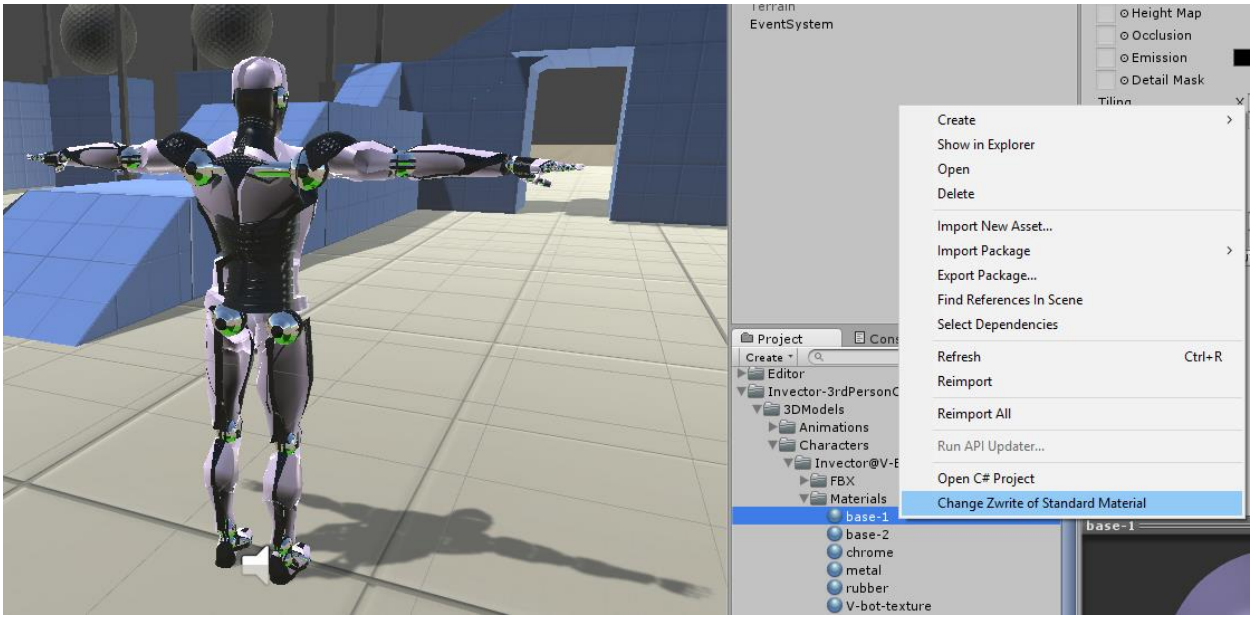
Our Culling Fade will set up automatically for the default Stardard Shader of Unity's, but you also can use custom shaders, just make an additional copy with the fade material and assign in the "Optional Fade Material" field.



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



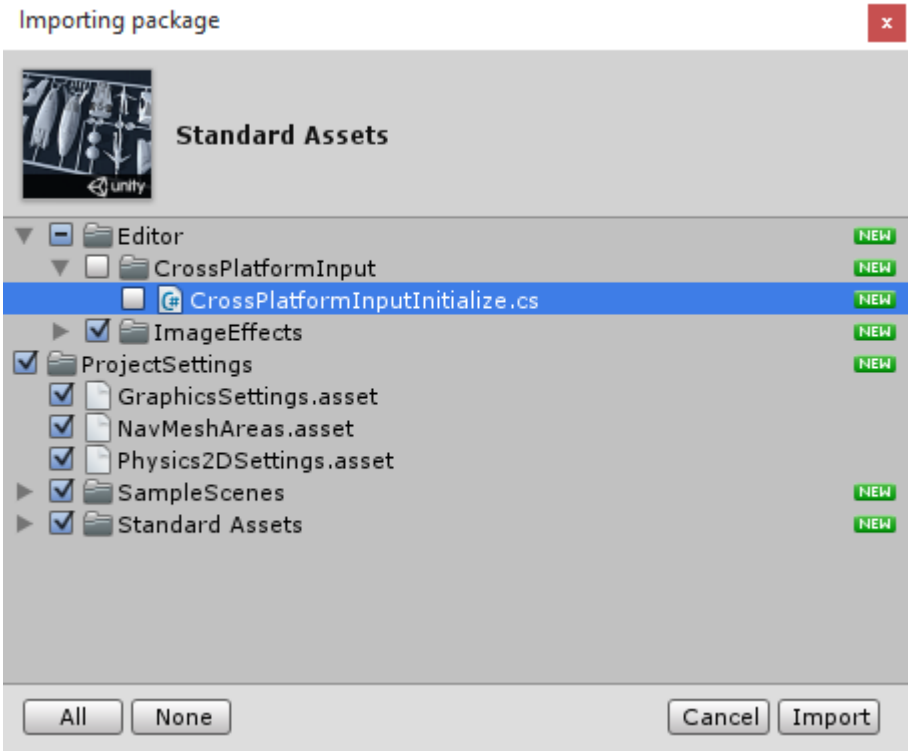
The character will look like this (picture below) but you can fix by right clicking at the material and “Change Zwrite of Standard Material”.



UPDATE V1.1B – now the script will be attach 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]

Melee Weapon Manager

V1.2 - You can add a **Melee Manager** Component by opening the 3rd Person Controller tab > Add Component.

After add, you can set up the following features:

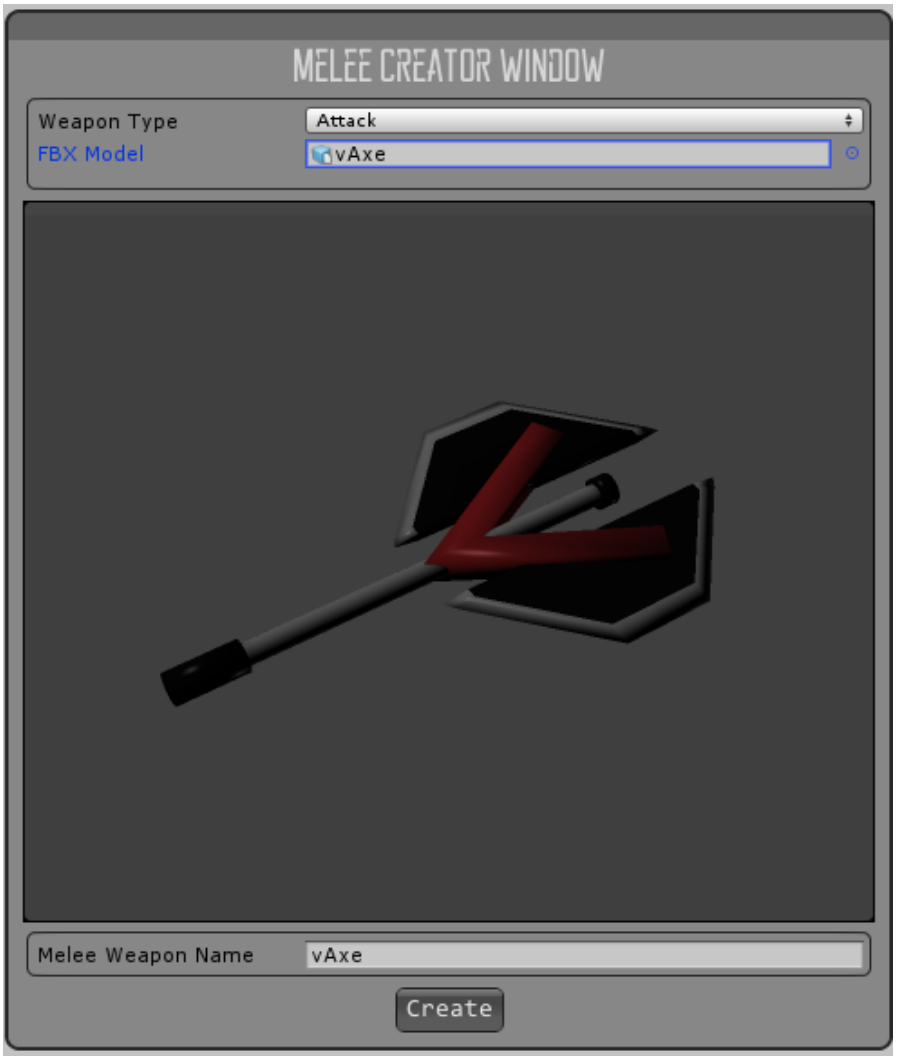
- **Use Recoil** > if you want the character to kick's back a recoil animation when hit a wall
- **Recoil Max Distance** > increase the area of hit recoil
- **Hit Recoil Layer** > the layer that will affect the recoil (usually the Default)
- **Hit Damage Tag** > **Important** this is the tag that will receive Damage, so if you are using this component into the Player, assign the correct Tags. If you are using on an Enemy AI, assign to Player.
- **Weapon Handlers** > each weapon will have a handler, this way you can position and rotate the weapon for each character, when you pick up a weapon they will find and parent to the corresponding handler.

You can add handlers automatically by clicking on the "+ "button next to the corresponding bone.

Creating a Melee Weapon

V1.2 - You can create a new weapon by opening the 3rd Person Controller tab > Create new Melee Weapon.

This window will bring the option to create an **Attack** or **Defense** weapon and the weapon's **name**.
*Make sure to assign only the original FBX of the model without any other components like Rigidbody or Colliders, you can set up this later and make prefabs.

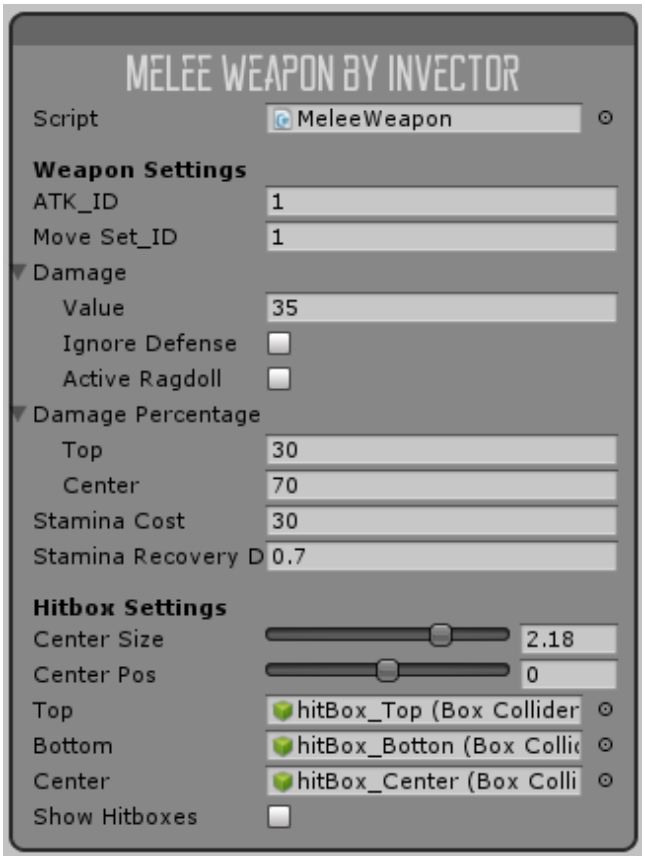


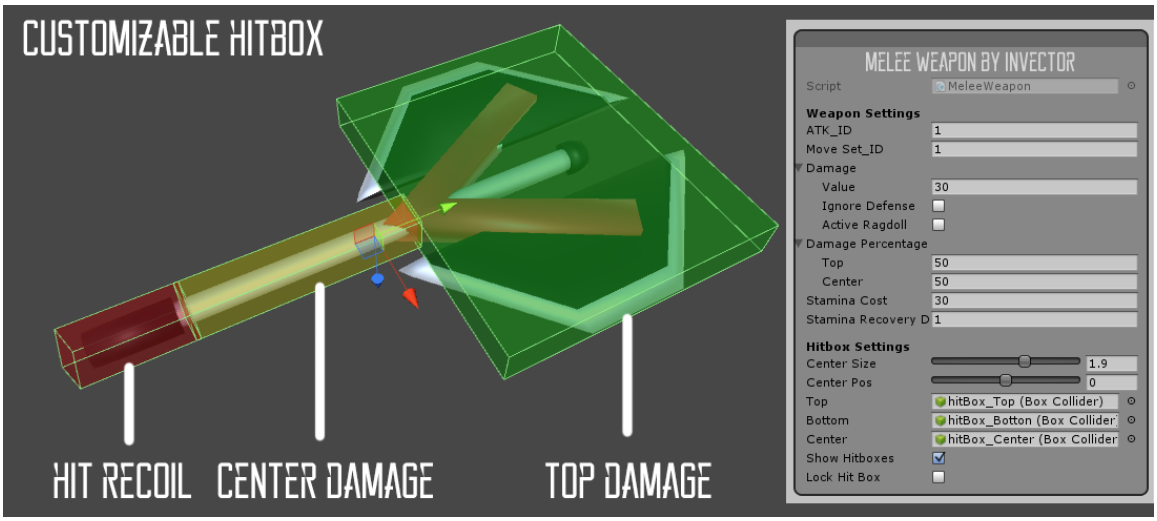
After hit the Create button, your weapon will be automatically assign with **Rigidbody**, **Box collider** **Sphere Collider** and the components **Collectable Melee** and **HitBox**.

IMPORTANT – don't forget to set your **Weapon Layer to Triggers**, if you put a weapon into an Enemy or Player and change the Layer and childrens, you need to set the weapon layer to Triggers again.

You can replace the box collider to a mesh collider later, but is important to keep this order for everything works.

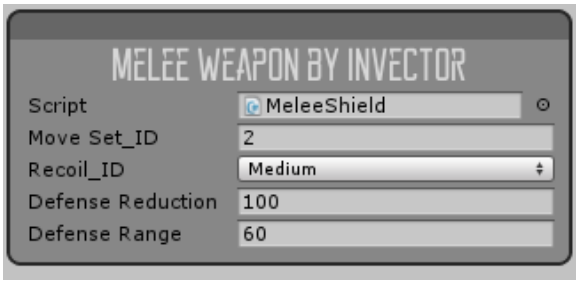
- **Start Using** > if the weapon **already** is with the character, leave this check True
- **Message** > the text that will display when you enter the Sphere Trigger collider
- **Handler** > the corresponding handler to position/rotate the weapon
  If you create an **Attack** weapon, you need to set up the **HitBox** inside the weapon:



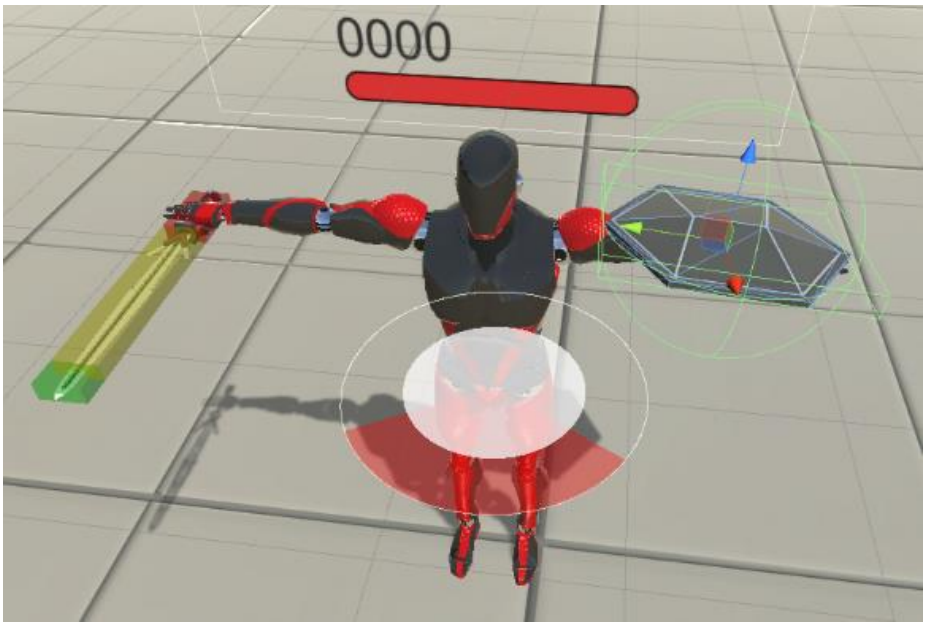


- **ATK_ID** > correspond to the Attack Animation State that will trigger
- **MoveSet_ID** > it's the correct move set that the character will move when using this weapon
- **Damage Value** > how much damage this weapon cause
- **Ignore Defense** > True if this weapon can pass through shield
- **Active Ragdoll** > True if this weapon can activate the enemy Ragdoll
- **Damage Percentage** > the HitBox has 2 damage areas, Top and Center and you can slip the percentage on these areas.
- **Stamina Cost** > how much stamina the attack will cost
- **Stamina Recovery Delay** > how much time will take to the stamina start recovery
- **Center Size >** you can adjust the size of the hitbox center area
- **Center Pos >** you can position the hitbox center
- **Lock HitBox >** uncheck to create custom hitbox like this Axe:

To create a **Defense** weapon, change the Weapon Type to Defense on the Melee Weapon Creator Window and you will have a **MeleeShield** script attach into the gameObject and you can set up the following variables:



- **MoveSet_ID** > When you hold the Defense Input the character will blend to this moveSet
- **Recoil_ID** > Trigger a recoil animation
- **Defense Reduction** > how much is the Defense Rate
- **Defense Range** > When you select the shield, a Gizmos will appear to help you see how much of Defense Range you want. If you want your character to block all the attack from all sides, leave to 180.



Creating an Enemy AI

V1.2 – Follow the same steps as creating a new character but change the **Character Type** to **Character AI**

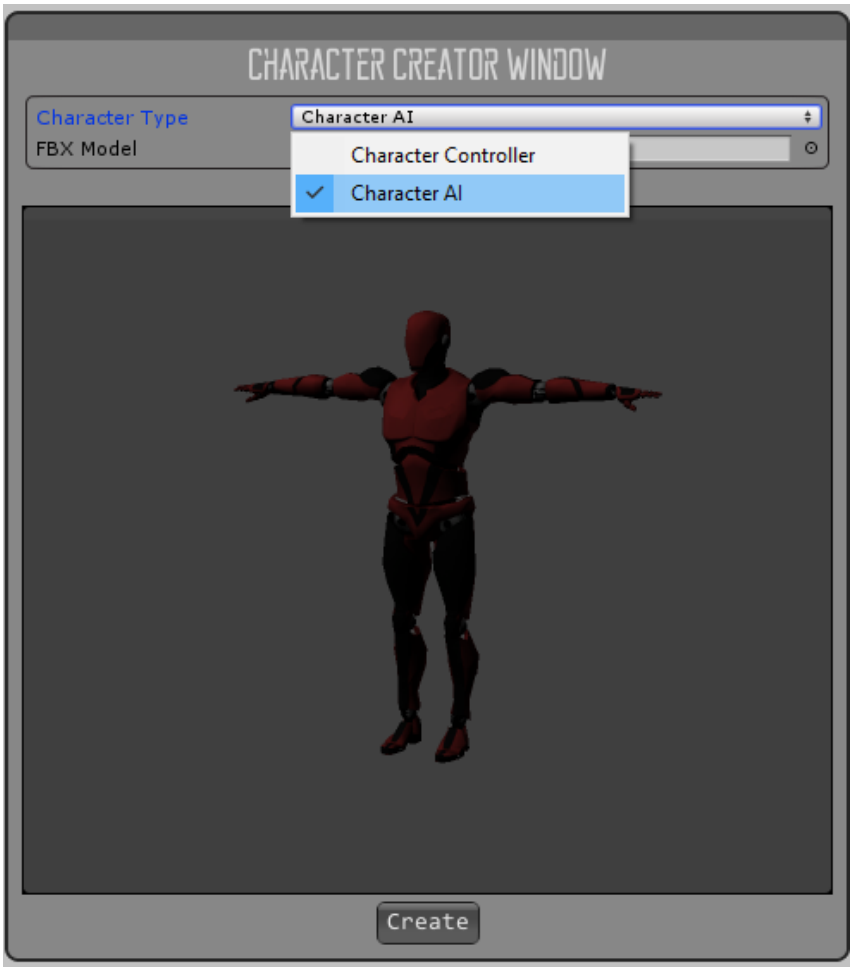
After hit the Create button, our scripts will handle all the most time consuming stuff and make the AI almost done to hit Play, you just need to create a NavMesh on the Scene.

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

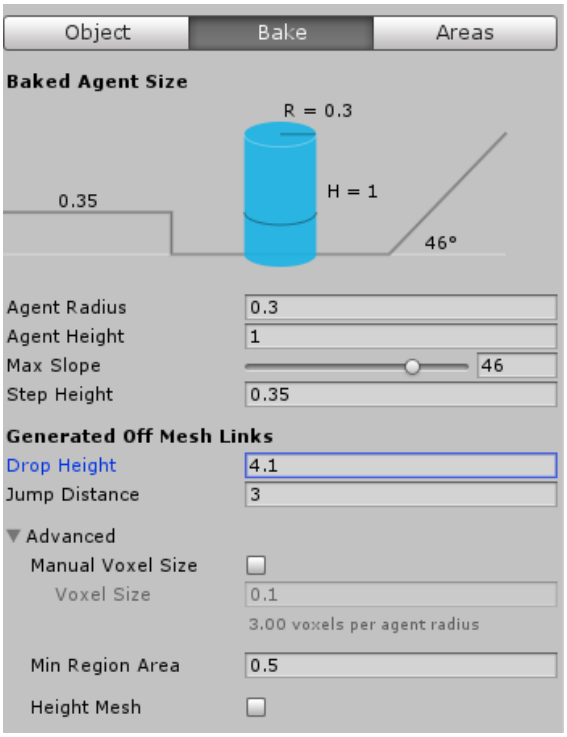
Layers – just like the Player, you need to set up a Layer for the AI (Enemy) and a layer for the Ground (Default). If you equip a Weapon on the character, don't forget to assign the Layer Triggers for this weapon.

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 shield), chance to roll, chance to defend an attack, change the attack frequency, etc…

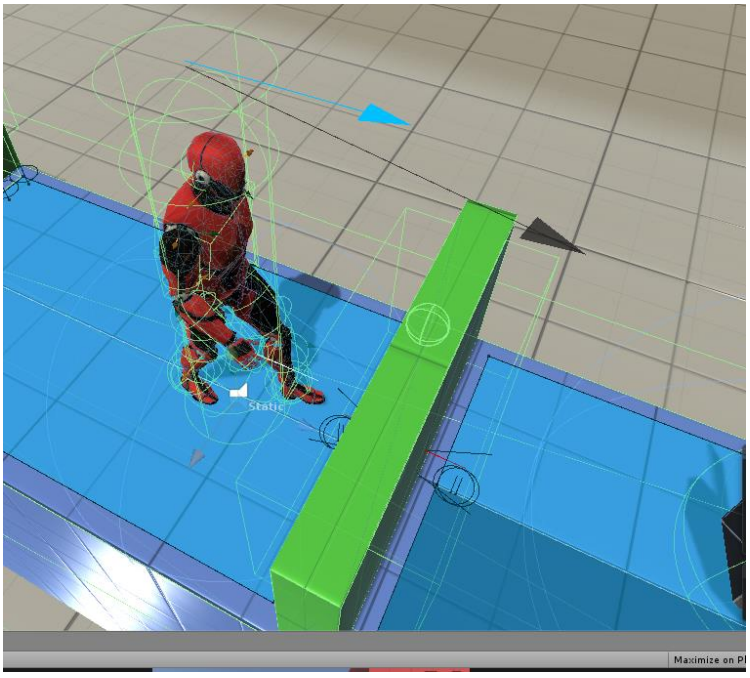
Waypoints – You can add waypoints for the AI to follow in sequence of activate the option to Random.



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…

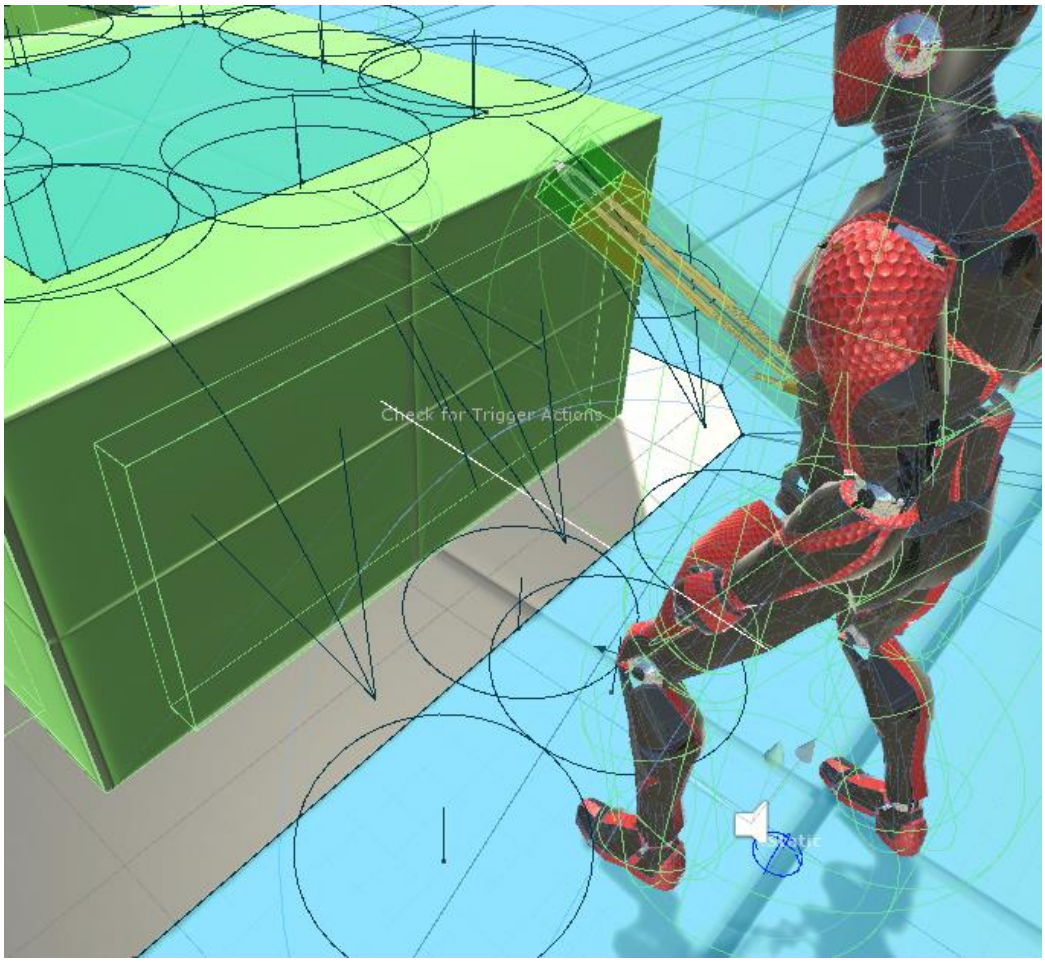


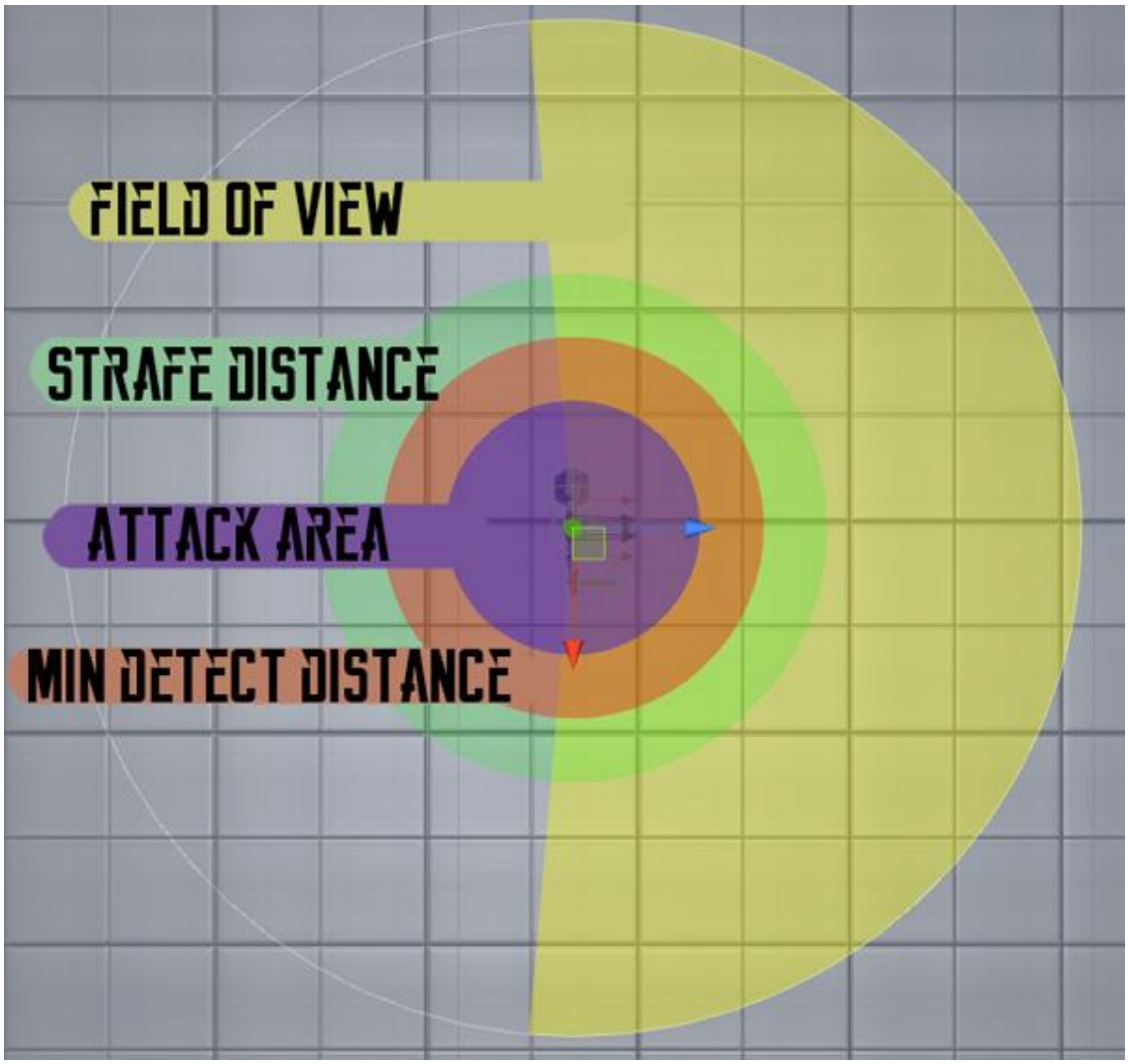
The AI use **CheckForwardAction** to do **Actions** just like the Player does, but the main difference is that the Player does the action by using an **Input**, and with the AI we are using **OffMesh Links**. There are examples of Climbup, setup and jumpOver that you can use to set up.



Make sure to always keep the trigger outside the NavMesh just like the image, and if your AI agent is stuck instead of doing the action, if because the Raycast is not reaching the ActionTrigger or because the Action Trigger is too large and the Raycast is inside the trigger.

You can change the distance of the Raycast on the variable **Check Trigger Distance** at the AI Inspector.

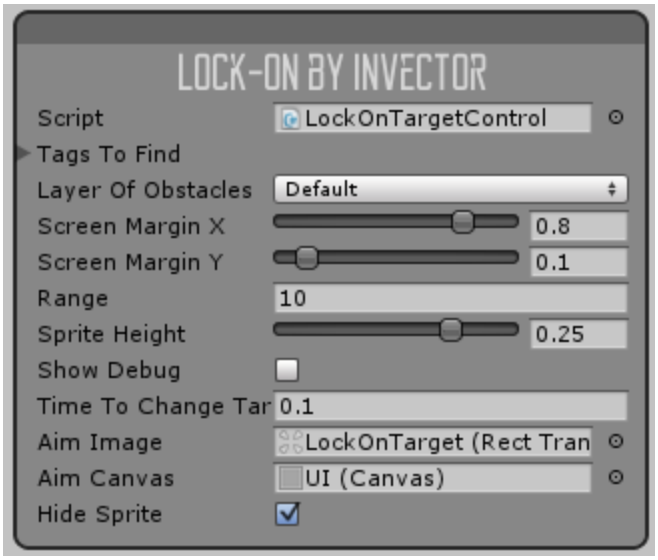




Field of View > total range to detect the Player
Strafe Distance > once in combat, the character will move Strafing
Attack Area > total range to attack
Min Detect Distance > Min distance to detect the player, even if the player is outside the Field of View range.

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-AI objets because it need's the **iCharacter** interface to know if the target is alive.

Waypoint System

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

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

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

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

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

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