AAA 3rd Person Controller – Basic Locomotion 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

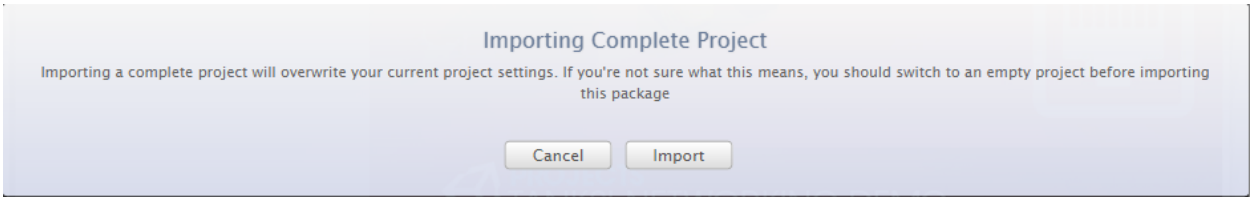
Summary

- First Run 3
- Creating a new Character 4
- Creating a new Character Template 7
- Creating a new Camera State 8
- CameraMode – Free Directional 9
- CameraMode – Fixed Angle 9
- CameraMode – Fixed Point 10
- Xbox 360 Controller Support 11
- Recommended Mobile Settings 11
- FootStep Audio System 12
- Creating a Ragdoll 14
- How it works? 16
- How to add new animations? 17
- RayCast Checkers (v1.1) 18
- Camera Culling Fade (v1.1a) 19
- Working with Standard Assets 21

First Run

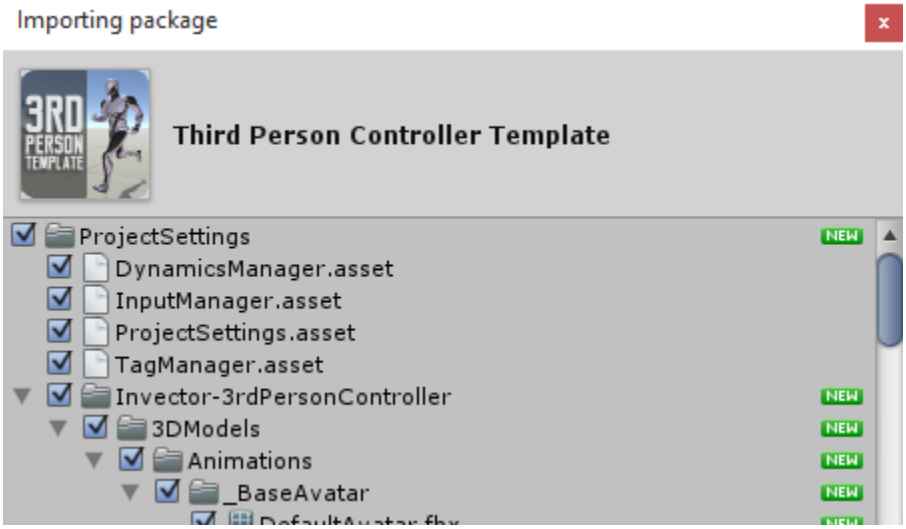
IMPORTANT

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*

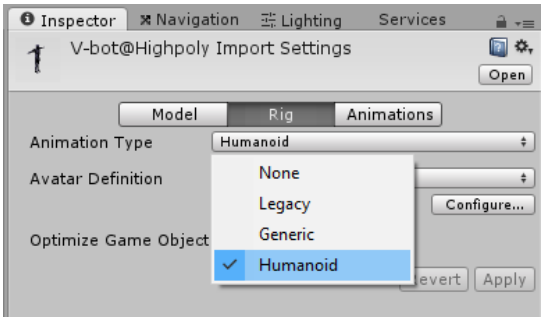
If you want to import into another project, you can UNCHECK some project settings to avoid conflicts or replace your project settings like the TagManager (wich includes the Layers), and add later the tags and layers that we use. We recommend to import the InputManager because it's kind of painful to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



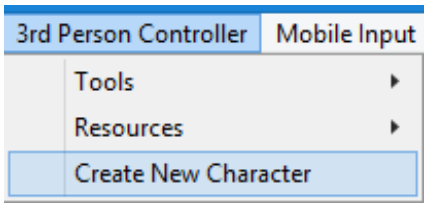
***Updates also need to be imported into a Clean Project, so MAKE SURE TO BACKUP your previous project and transfer the necessary files to your new project. ***

Creating a new Character

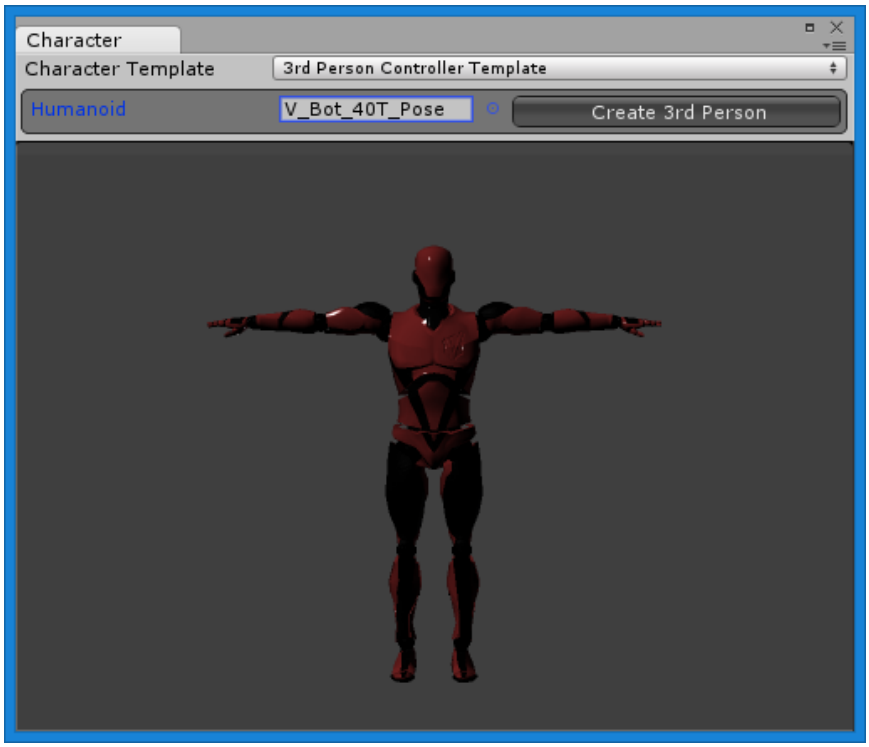
1- Make sure that your fbx character is set up as **Humanoid**



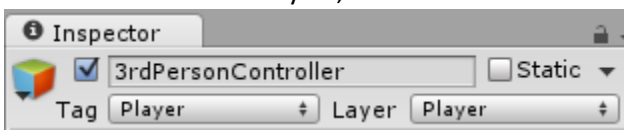
2- To setup a new character, go to the tab "3rd Person Controller" and click "Create New Character"



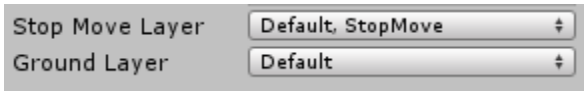
3- Make sure your Character is **Fully Rigged** and set up the FBX as a **Humanoid**, then assign the FBX to field "Humanoid" and click on the button "Create 3rd Person Controller".



4- ***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:



5- **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.

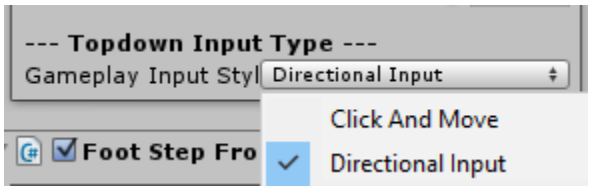
The script will also adjust your Capsule Collider settings based on your model proportions, if the capsule gets the wrong size, make sure that you rig is correct, and that your **model is using SCALE 1** the same goes if the ragdoll **gets** weird.

6- Hit Play and enjoy ☺

V1.1 Additional Info:

3rd Person and **2.5D** bot use the **3rd Person Controller Template**, just change the CameraStateList.

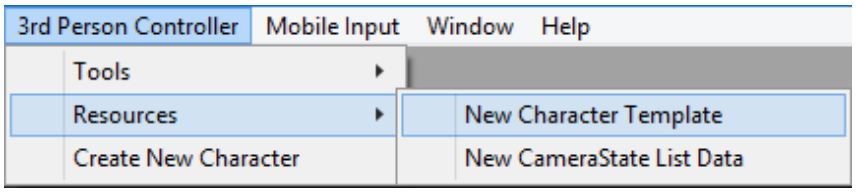
Topdown and **Isometric** both use the **Topdown Controller Template**, you can change to input style to Mouse and Click or control with inputs on the Player Inspector.



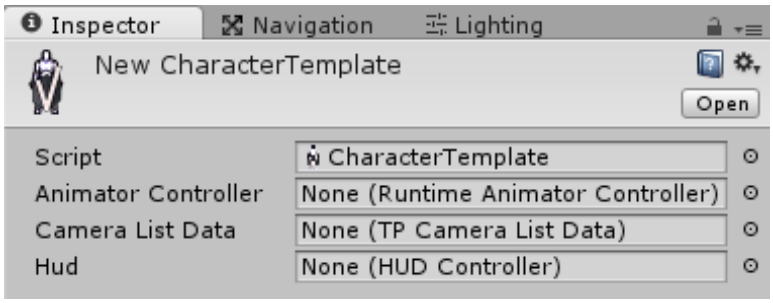
Also, remember to Uncheck the **HeadTrack** option at the Player Inspector if you are using topdown or isometric mode.

Creating a new Character Template

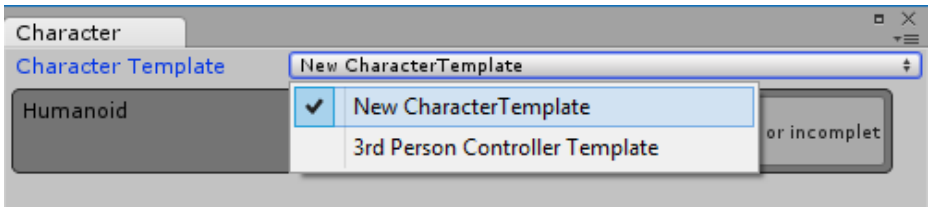
You can set up the Character Creator to create **custom controllers** that you have modified, to create a new template go to "3rd Person Controller" tab, "Resources" and click on "New Character Template".



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

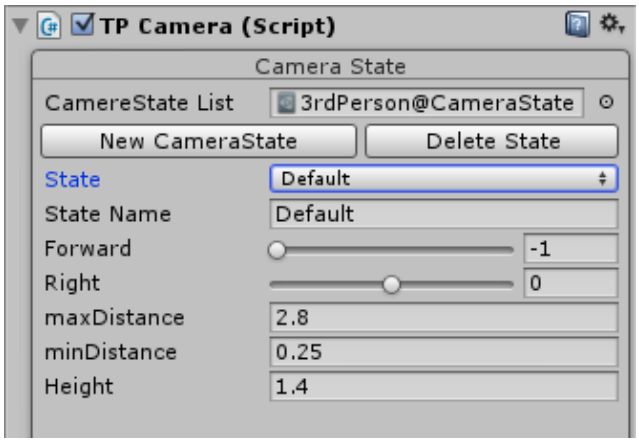


The next time you create a new controller, choose the new template.



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.

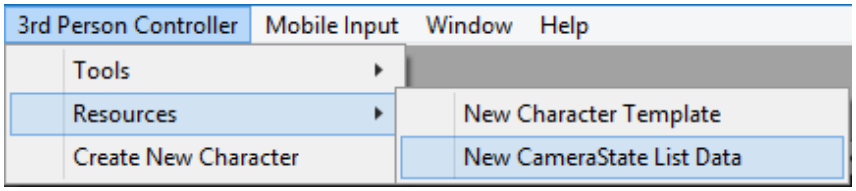


Then just change the CameraState on the method ControlCameraState() on the script TP_Motor.

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

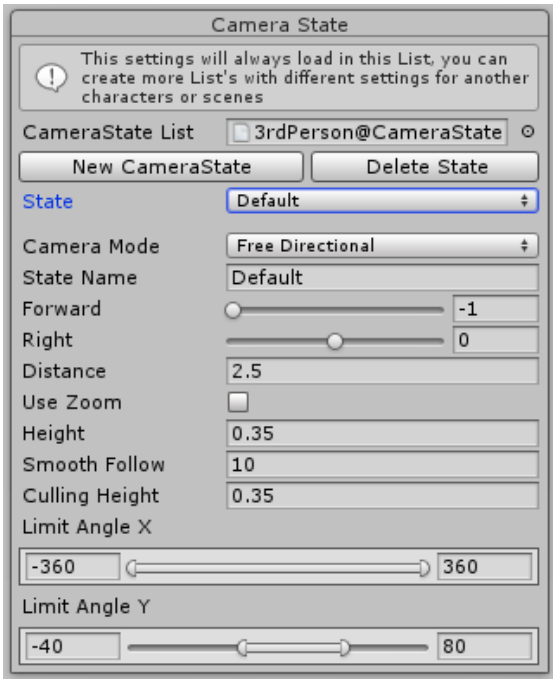
The first string value is the State Name that you created on the Camera Inspector, the second value is a bool, leave it true if you want a smooth transition to this state or false if not.

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.



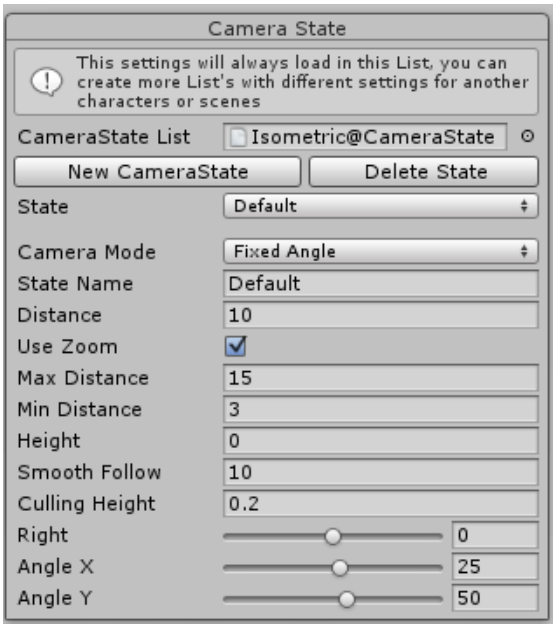
CameraMode – Free Directional

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, etc…



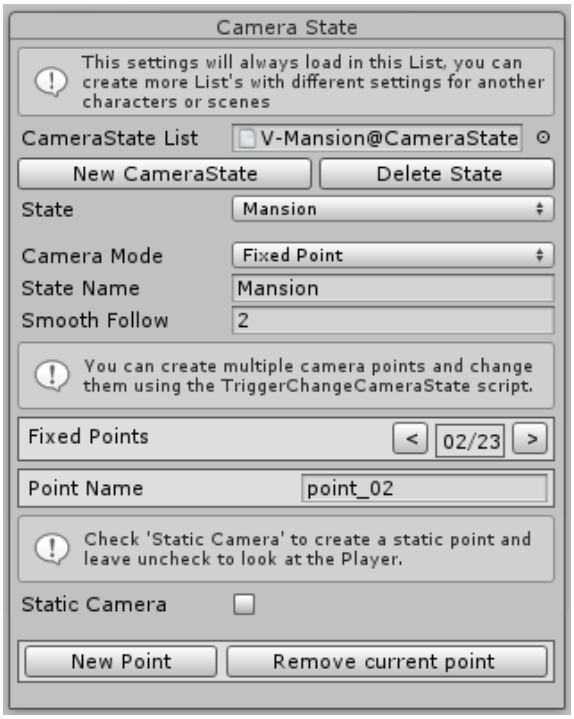
CameraMode – Fixed Angle

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.

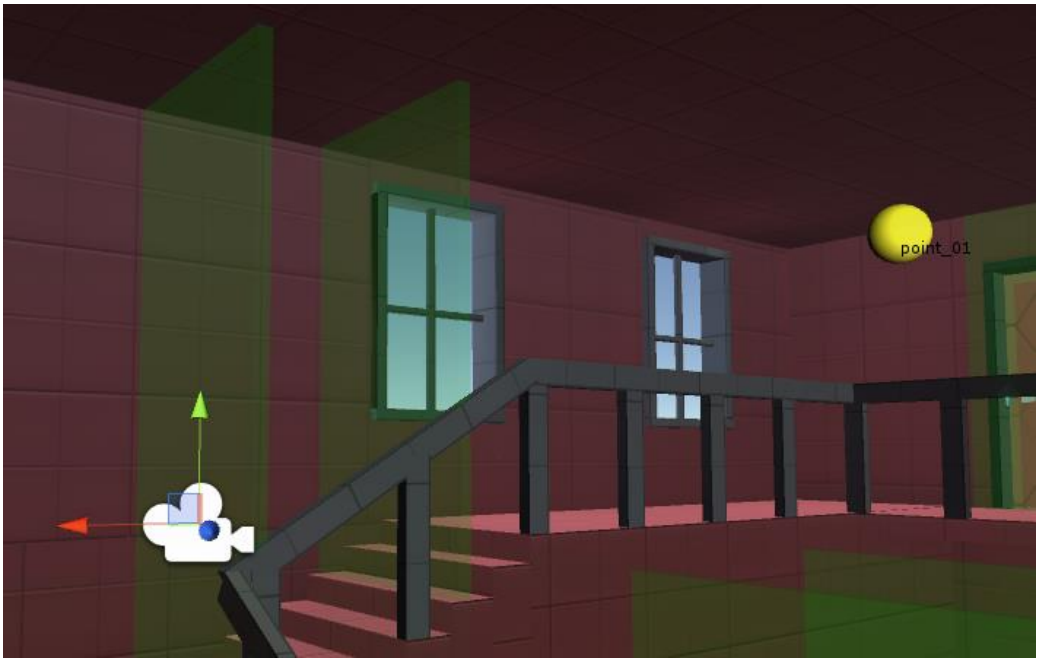


CameraMode – Fixed Point

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 are compiling 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**.**

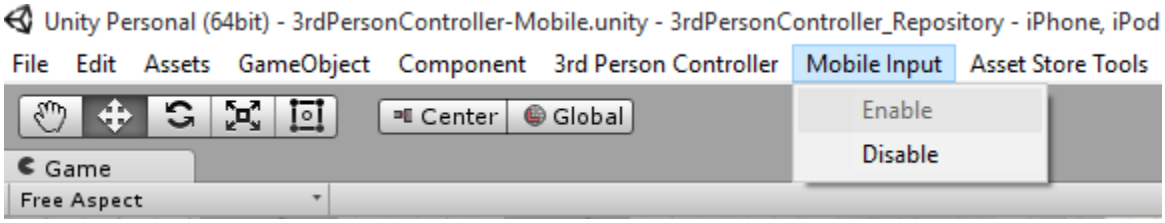
Recommended Mobile Settings

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*

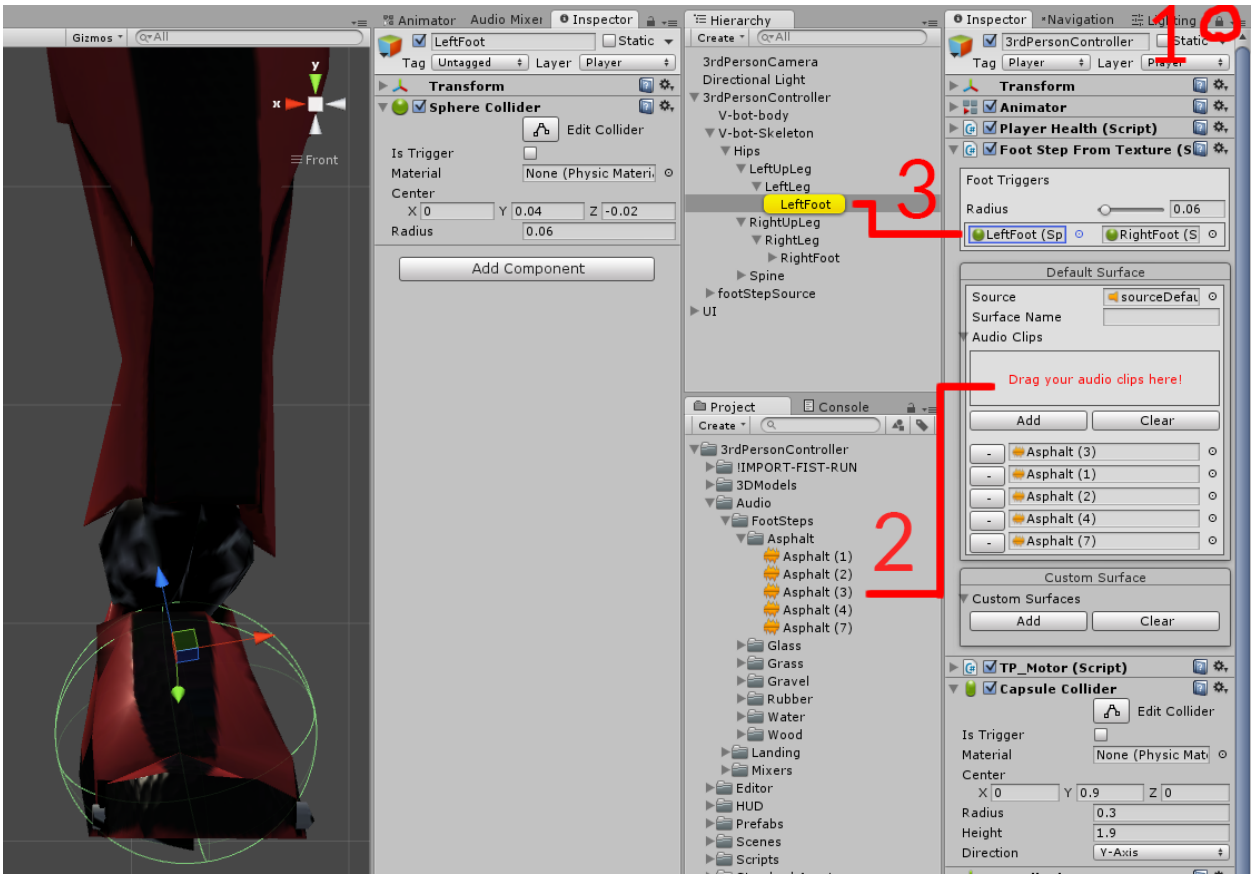
FootStep Audio System

We created a footstep system that work by texture's name, first you need to assign the **default footstep** that will play in case you have no texture at all.

1 - Lock the inspector

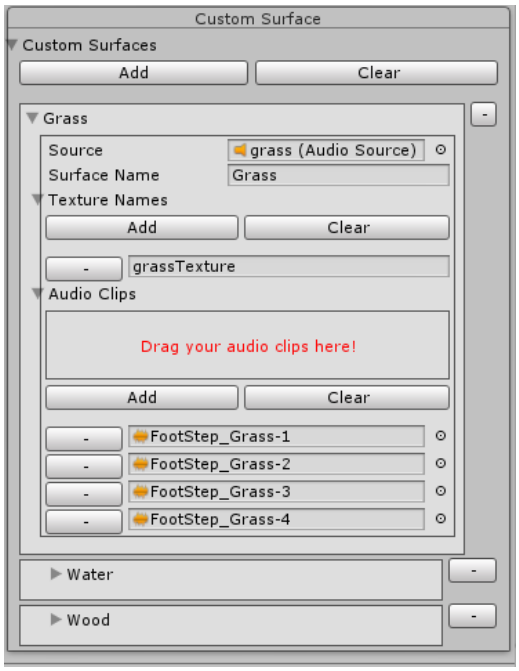
2- Drag and drop the AudioClips into the area "**Drag your audio clips here!"** or just add one by one. Click on the "–" button to remove the audio clip.

3- FooStep will automatically create a **sphere collider** on the foot of your character, but you need to make *sure* the Radius and Position of the sphere is **touching** the ground.



Now you can create **Custom Surfaces**, to play other audioclips based on the **texture** that the sphere collider will hit.

1- The **Character Creator** will create a default Audio Source as a child; assign this audio source to the first field Source. You can create a **Mixer** and select an Output to control every surface **independently**.
2- Click the Add button to create a new **Custom Surface**, you can add as much Custom Surfaces as you need. Open the tab and type the Surface Name
3- Open the tab Textures Names and you can add as much textures you need (texture name, **not material**)
4- Drag and Drop the audio clips.

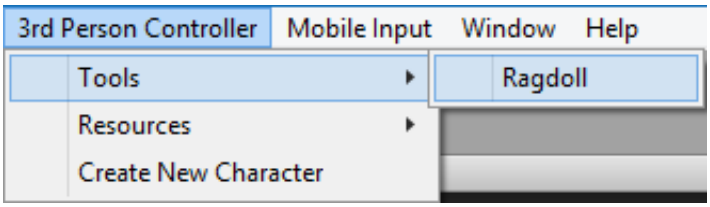


- *V1.1 Using the FootStep system in objects with multiple Materials*

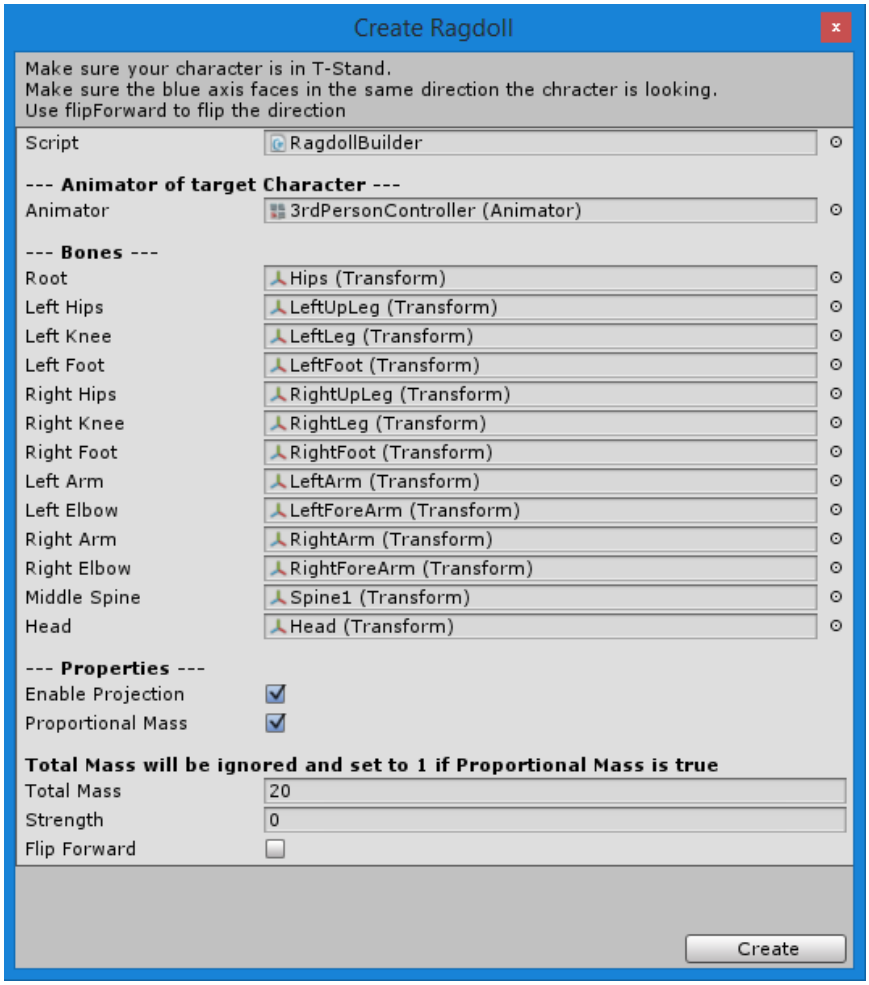
If your gameobject has multiple materials and you need to play a specific material, you can use the FootStepHandler script and set the correct Material Index of your object. (*See example on the Ladder prefab)

Creating a Ragdoll

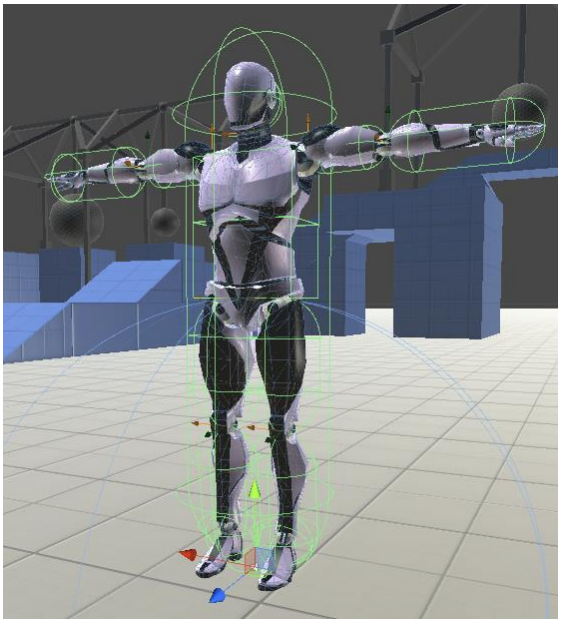
Creating a Ragdoll is just easy as creating your Character, just go to the tab "3rd Person Controller" > "Tools" > "Ragdoll".



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.

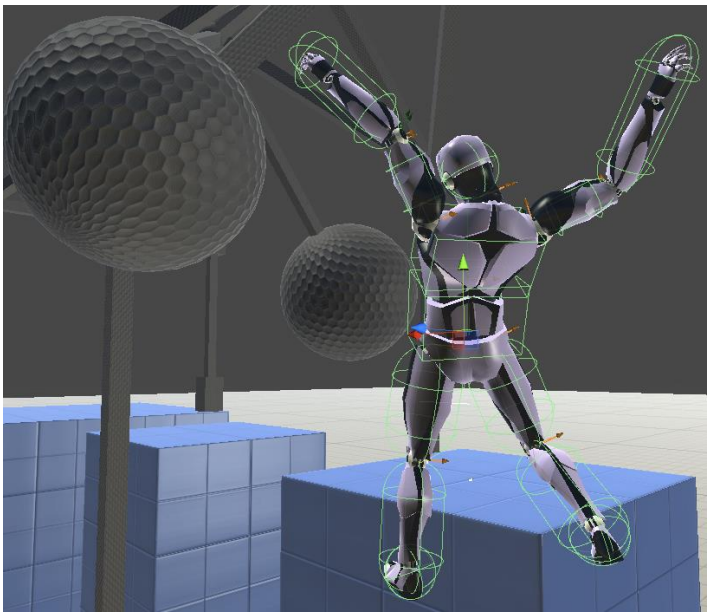


We strongly recommend keep the **Enable Projection** and the **Proportional Mass** enabled, and do not forget to use **Scale Factor 1** on your **fbx** Model. This you provide better behavior of your ragdoll.



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);

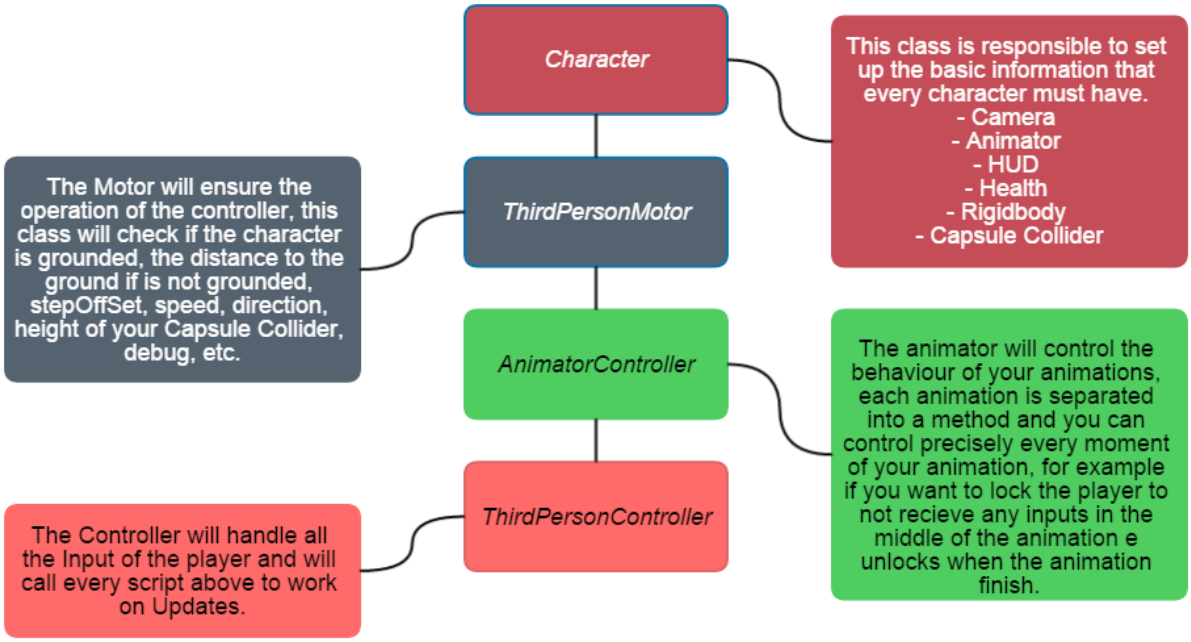


v1.1b – Add "***Ignored Tags***" you can add a list of tags for objects that are children of the Player to keep the rotation correctly, otherwise it will mess up the rotation when the Ragdoll are on.

How 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 Controler** 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.



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:

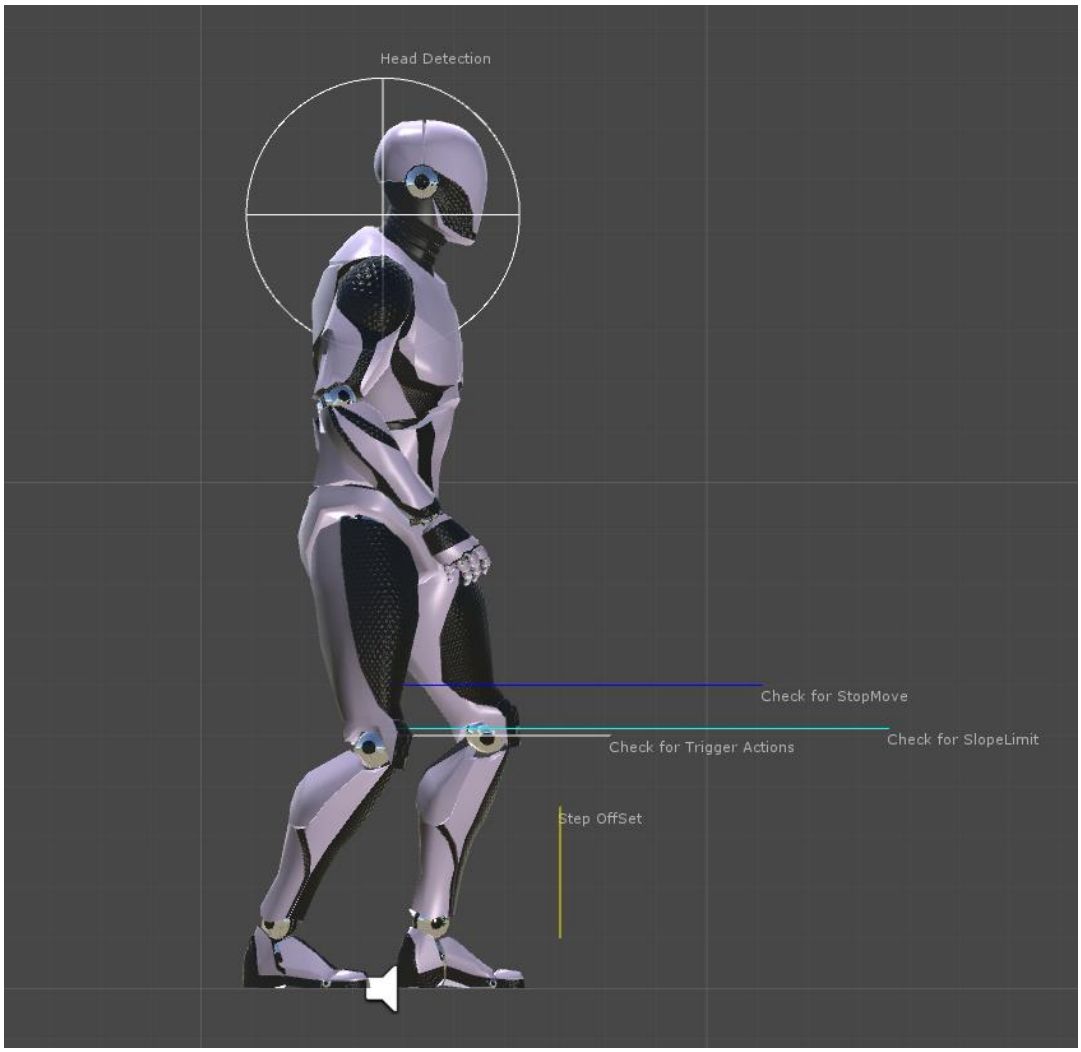
[How to add new animations]

[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)

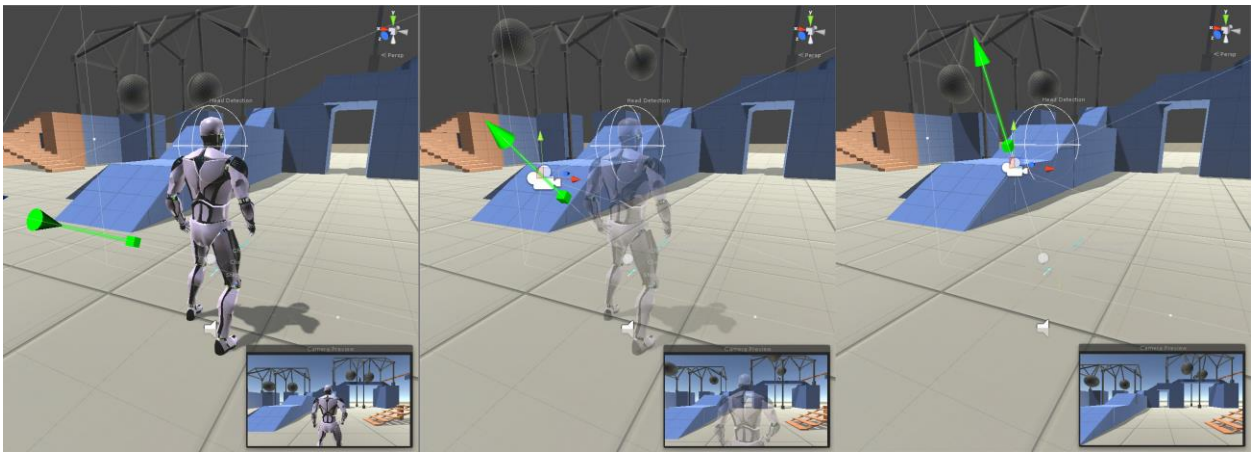
1- **Head Detection** is a SphereCast that will detect if has an object above, and keep the character crouched, use the same layer as the Ground Layer (Default). Just adjust to sync with the height of your capsule collider.
2- **StopMove** is a Raycast that detect any object with the layer (Default,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



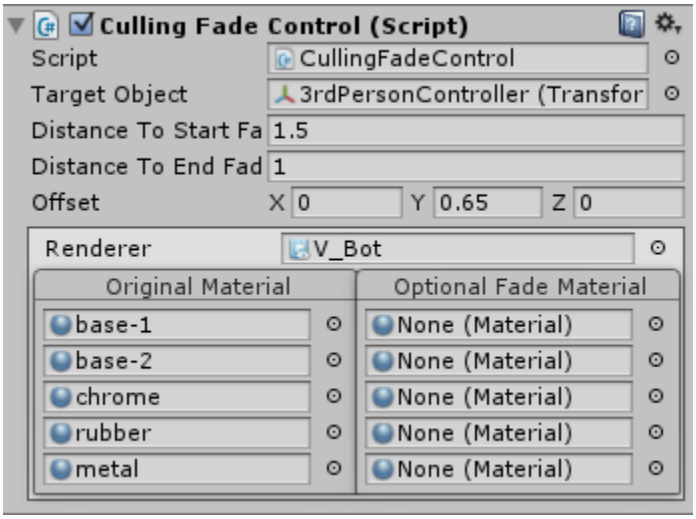
Camera Culling Fade (v1.1a)

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.

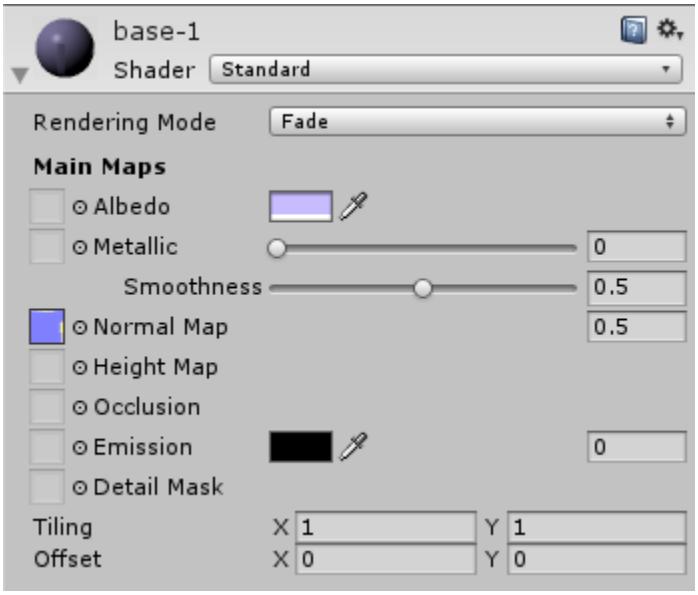
Example:



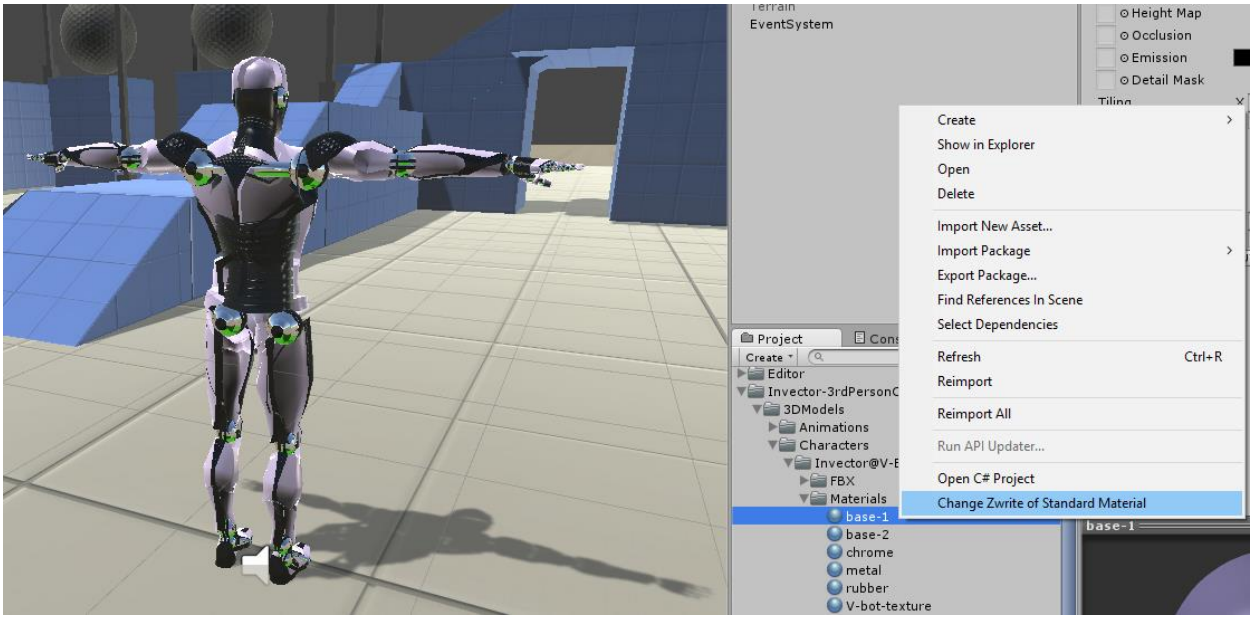
Our Culling Fade will set up automatically for the default Stardard Shader of Unity's, but you also can use custom shaders, just make an additional copy with the fade material and assign in the "Optional Fade Material" field.



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



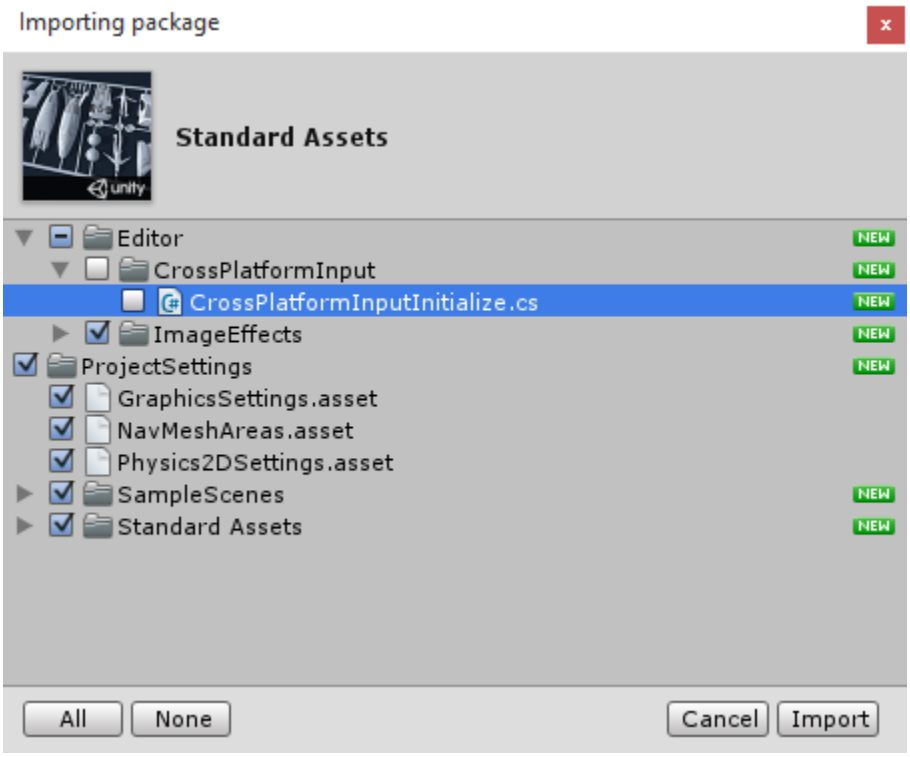
The character will look like this (picture below) but you can fix by right clicking at the material and “Change Zwrite of Standard Material”.



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, just delete the folder Editor that is INSIDE the folder Invector-3rdPersonController, and everything should work. In any case of errors after importing the Standard Assets, probably is a script that are duplicated, so just make sure that it has none.

[Video Tutorial]