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.

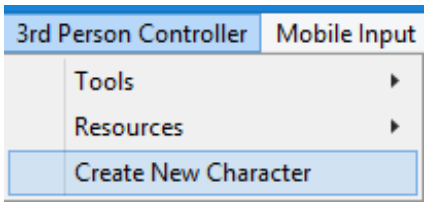
With this template, you can setup a 3D Model in just a few seconds, without the need of knowing hardcore code or wasting time dragging and drop gameobjects to the inspector, instead you can just focus on making your game.

Invector Team

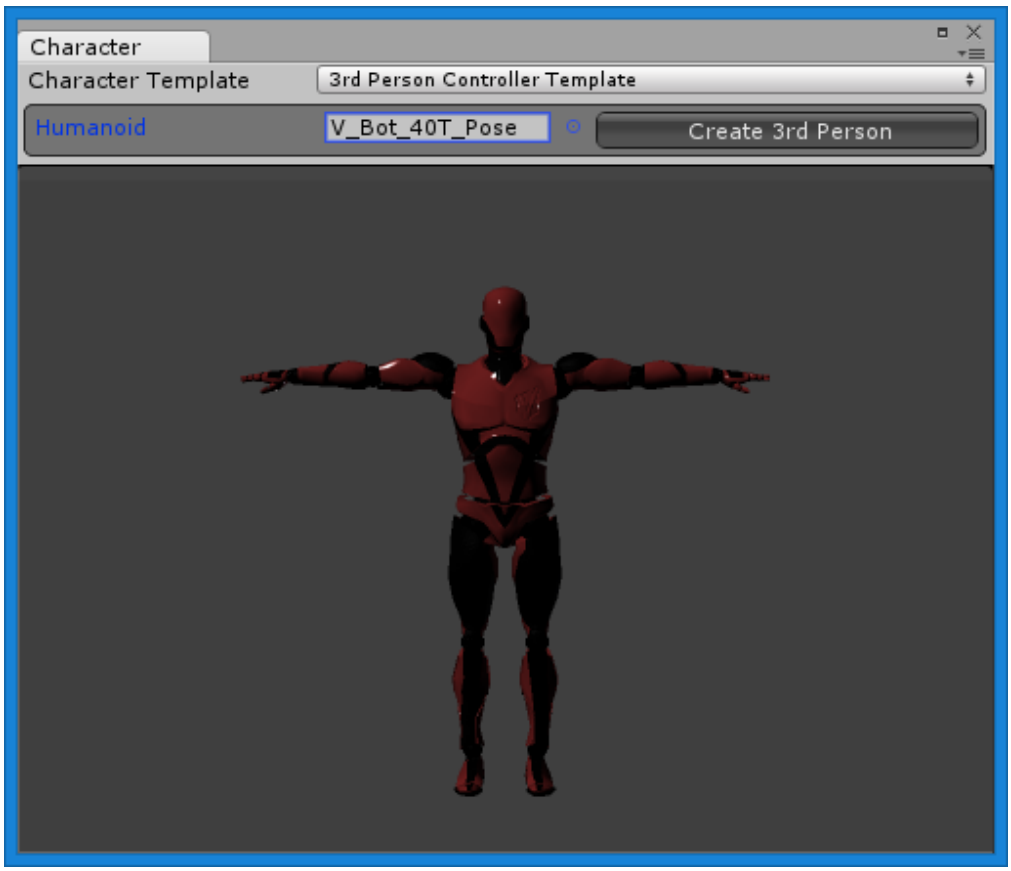
Summary

Creating a new Character

1- To setup a new character just go to the tab “3rd Person Controller” and click “Create New Character”



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



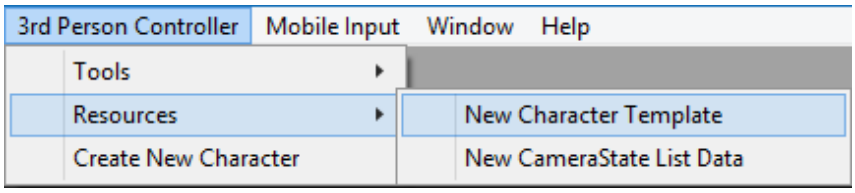
3- **Done**.
You do not have to do anything like dragging scripts, assign empty slots, nothing at all, the **Character Creator** will take care of all the hard work automatically and set up everything for you. It will create the **3rdPersonController**, **3rdPersonCamera** and a UI Canvas with a **HUD** to display health, stamina and others things.

The script will also adjust your Capsule Collider settings based on your model proportions, we recommend adjust the values to be rounded numbers after created the controller.

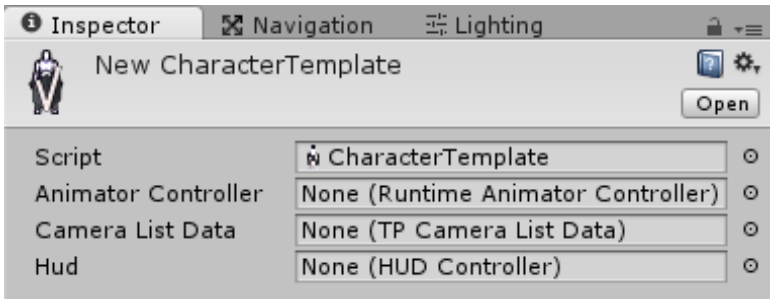
4- Hit Play and enjoy ☺

Creating a new Character Template

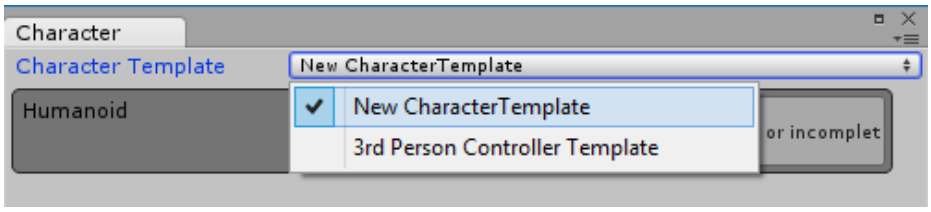
You can set up the Character Creator to create **custom controllers** that you have modified, to create a new template go to "3rd Person Controller" tab, "Resources" and click on "New Character Template".



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

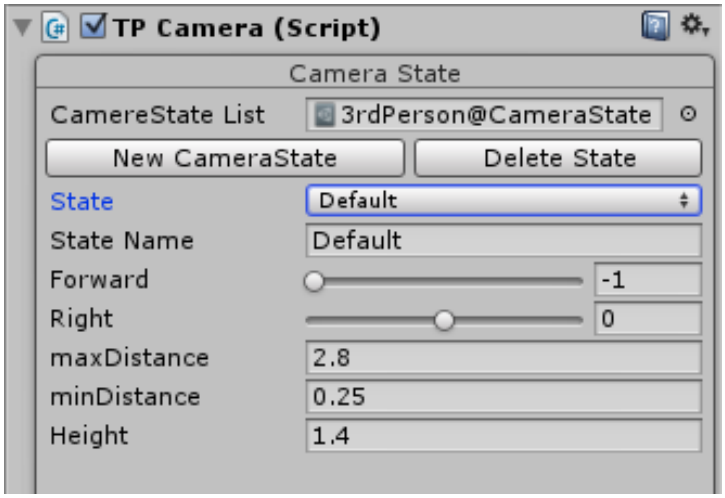


The next time you create a new controller, choose the new template.



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.

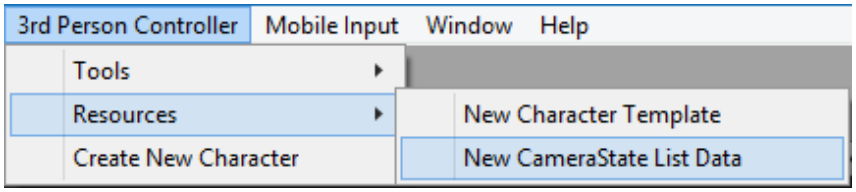


Then just change the CameraState on the method ControlCameraState() on the script TP_Motor.

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.



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.

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 plataform to **Android** 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)

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

**Unity does no longer supports Tegra devices*

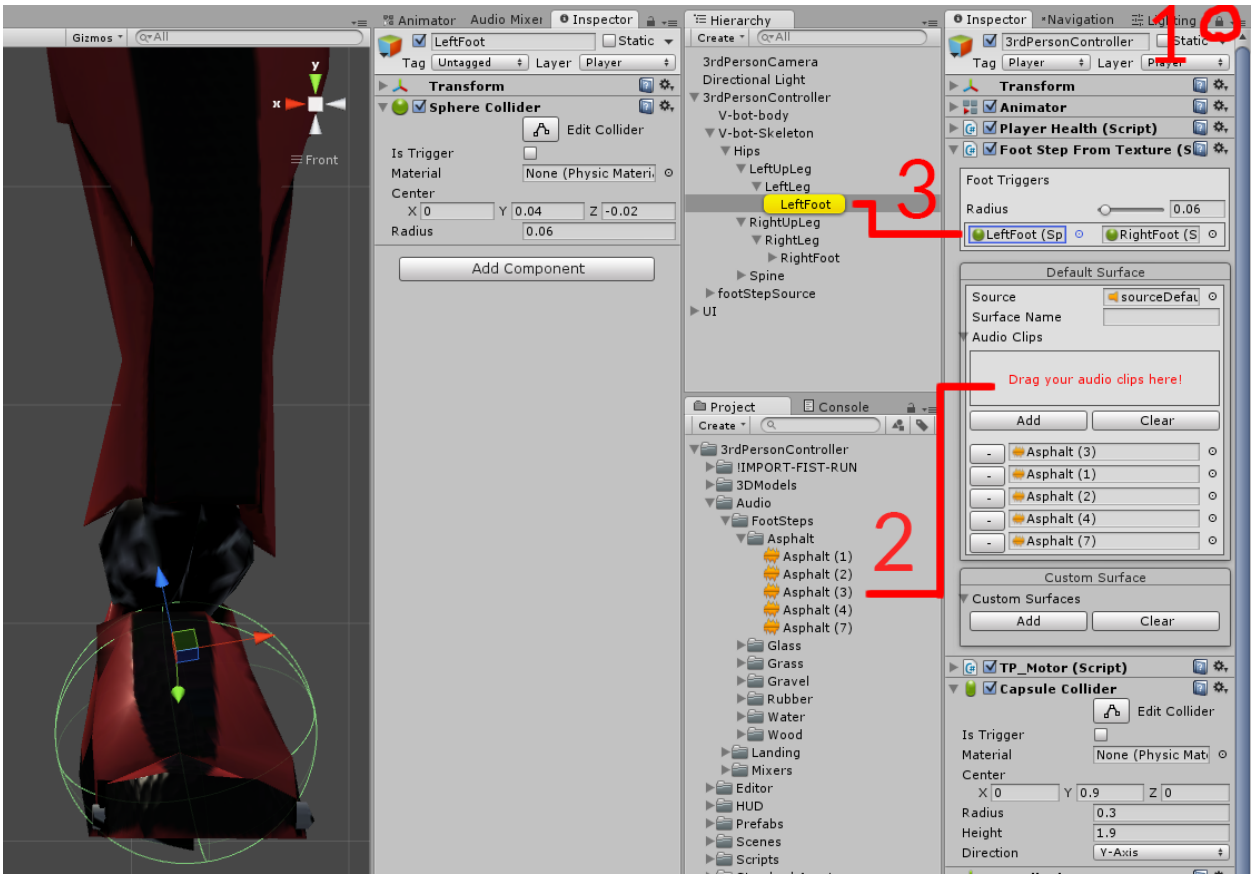
FootStep Audio System

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

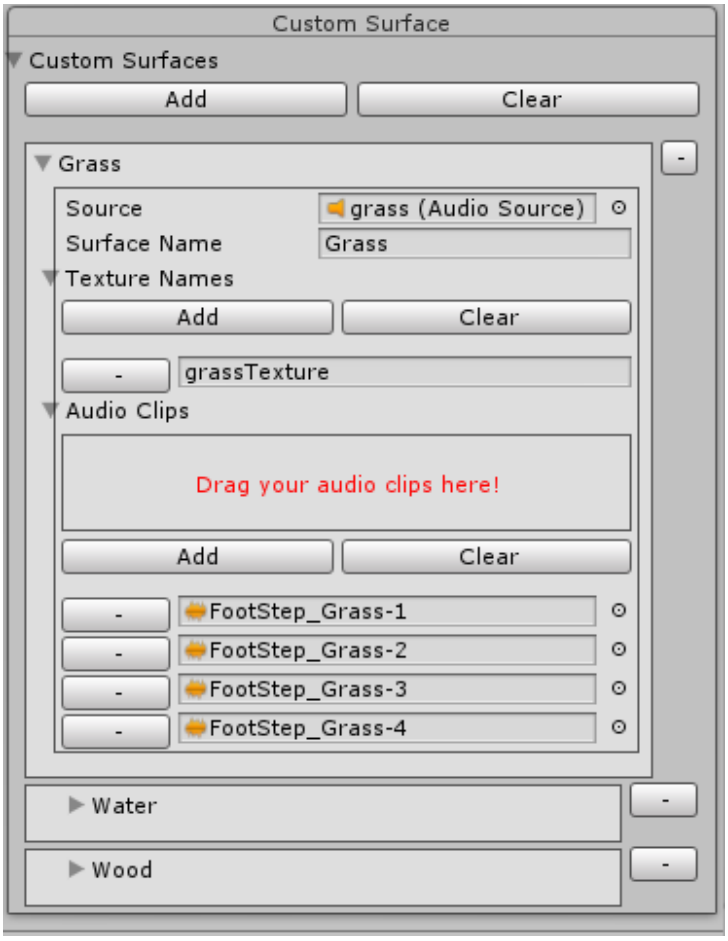
2- Drag and drop the AudioClips into the area "**Drag your audio clips here!"** or just add one by one. Click on the "**–**" button to remove the audio clip.

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.



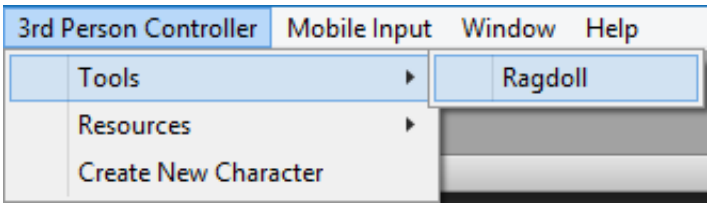
Now you can create **Custom Surfaces**, to play other audioclips based on the **texture** that the sphere collider will hit.

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.

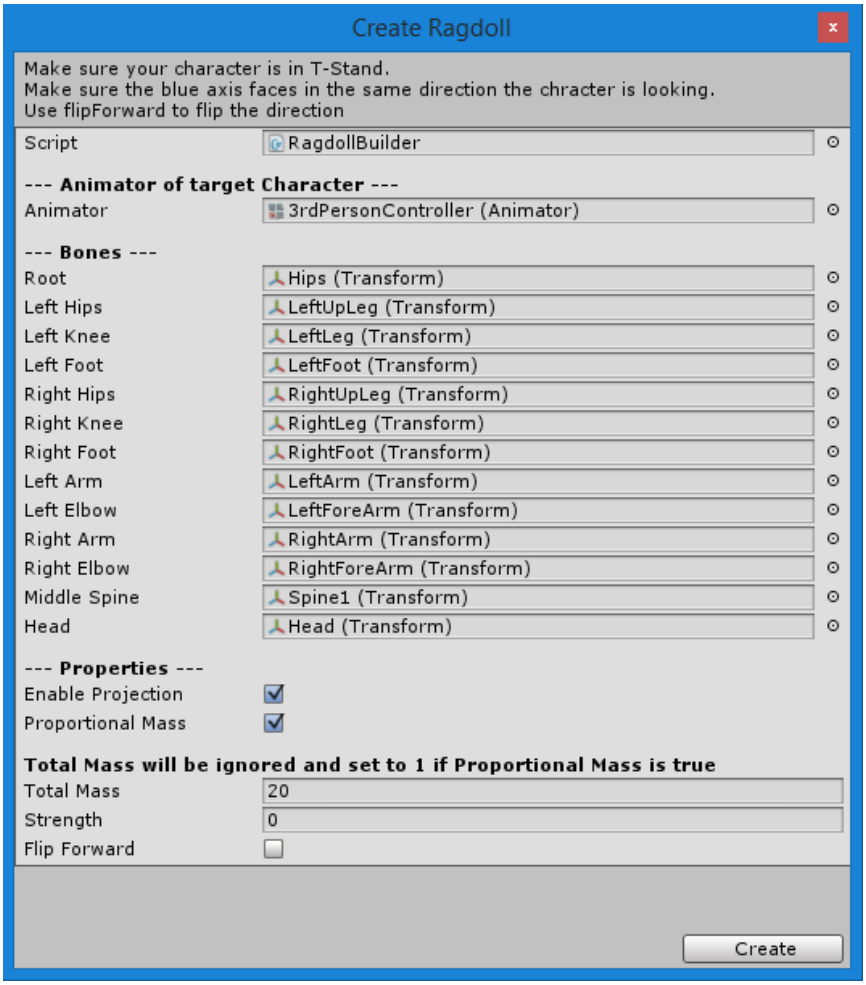


Creating a Ragdoll

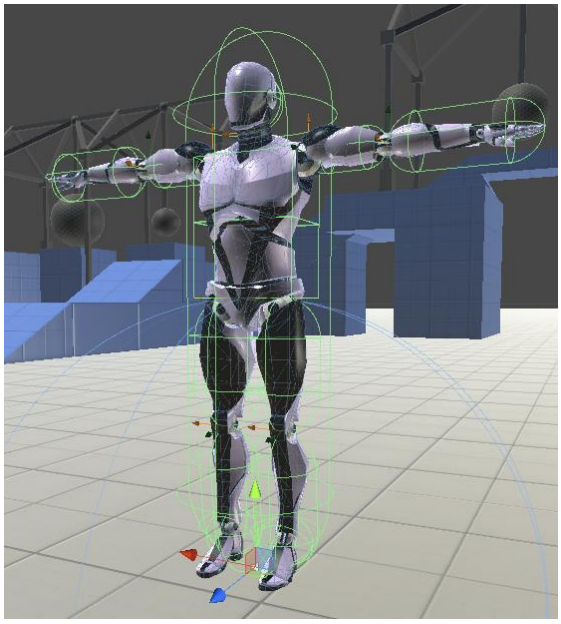
Creating a Ragdoll is just easy as creating your Character, just go to the tab “3rd Person Controller” > “Tools” > “Ragdoll”.



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.

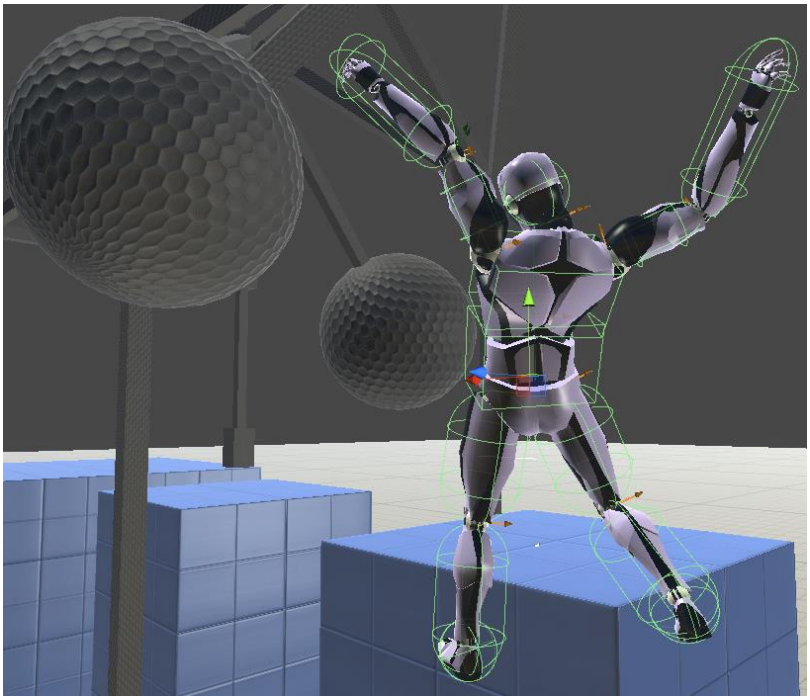


We strongly recommend keep the **Enable Projection** and the **Proportional Mass** enabled, and do not forget to import the **PhysicsSettings** as was said earlier. This you provide better behavior of your ragdoll.



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

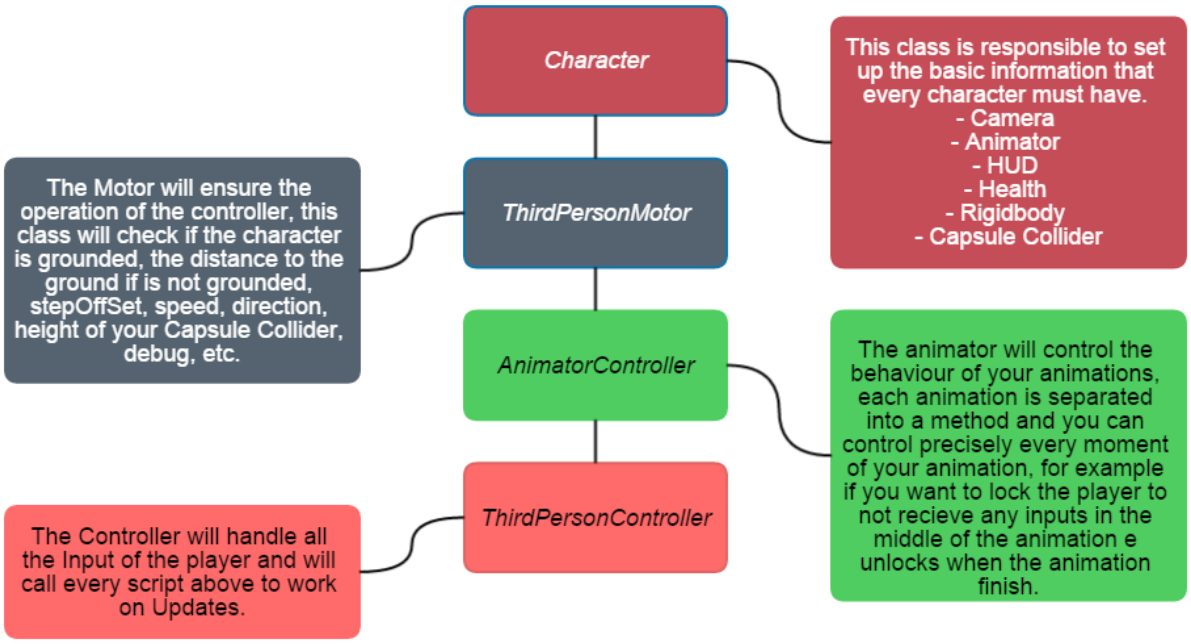
hit.transform.root.SendMessage ("ActivateRagdoll", SendMessageOptions.DontRequireReceiver);



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:

https://www.youtube.com/watch?v=W2IX7peS37s

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.