



Third Person Controller - Basic Locomotion

(v2.5.6 - 01/02/2021)

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

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

With this template, you can set up a 3D Model in just a few seconds, without the need of knowing advanced scripting or wasting time dragging and dropping game objects to the inspector, instead you can just focus on making your game.

--- Invector Team ---

FIRST RUN	3
CREATING A CHARACTER CONTROLLER	5
HOW IT WORKS?	7
HOW TO APPLY DAMAGE TO THE PLAYER	8
CREATING A NEW CAMERA STATE	11
XBOX CONTROLLER SUPPORT	15
INPUT MANAGER	15
HEAD TRACK	16
FOOSTEP AUDIO SYSTEM	18
CREATING A RAGDOLL	21
HOW TO ADD NEW ANIMATIONS/ACTIONS?	23
RAYCAST CHECKERS	24
TOPDOWN / 2.5D / POINT & CLICK CONTROLLER / MOBILE	25
MOBILE CONTROLS	27
CAMERA CULLING FADE	30
GENERIC ACTION (HOW TO INTERACT WITH OBJECTS)	33
ANIMATOR TAG	37
ANIMATOR EVENT MESSAGE/RECEIVER	38
MESSAGE SENDER/RECEIVER	38
BODY SNAPPING ATTACHMENTS	40
SNAP TO BODY	43

FIRST RUN

IMPORTANT

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

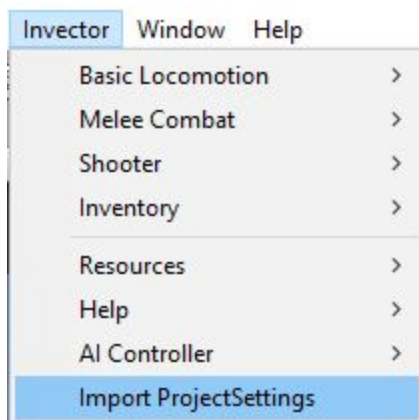


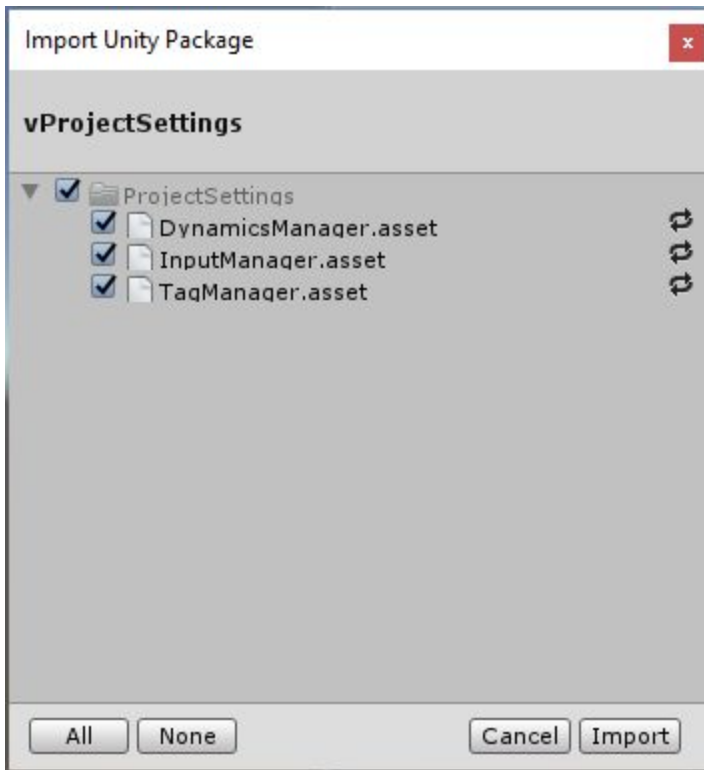
- *Importing on an existent project*

There are basically 3 files that are **extremely necessary for the correct functioning of this template**.

- ***InputManager.asset*** - We have a custom input mapped to support the Xbox360 controller, without it you will receive errors about missing input.
- ***TagManager.asset*** - Includes all the necessary Tags and Layers for the project to work correctly.
- ***DynamicsManager.asset*** - this will update the Collision Matrix of our Layers, for example we need the layer "Triggers" to not collide with the layer "Player".

After importing the template you can manually import those files by going to the tab **Invector > Import ProjectSettings**.



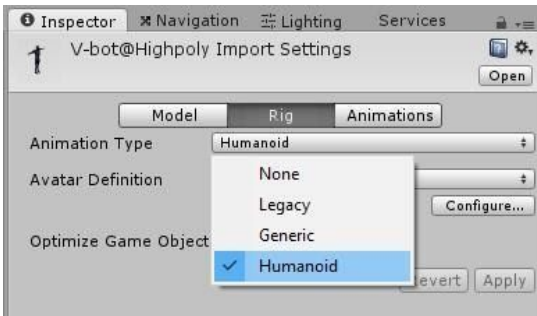


Now that you have imported the necessary files, you can explore the several demo scenes and figure out what kind of Third Person Game you want to create.

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

CREATING A CHARACTER CONTROLLER

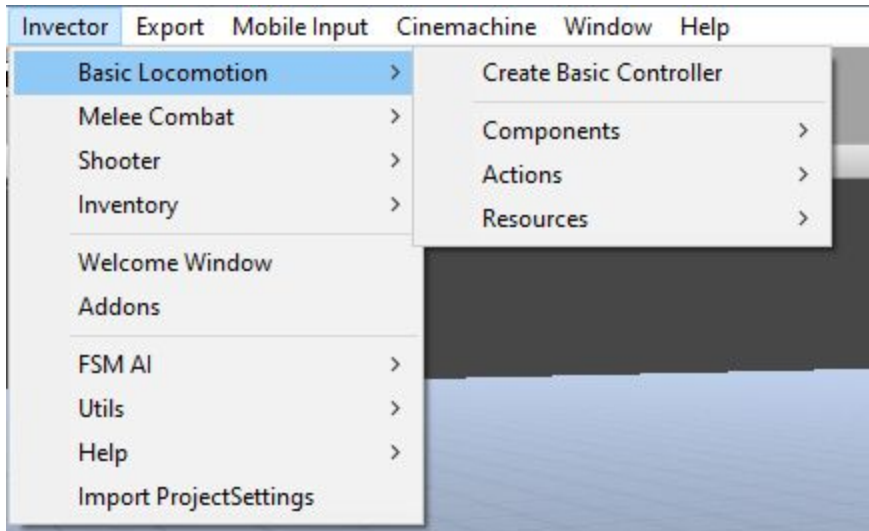
Make sure that your fbx character is set up as Humanoid



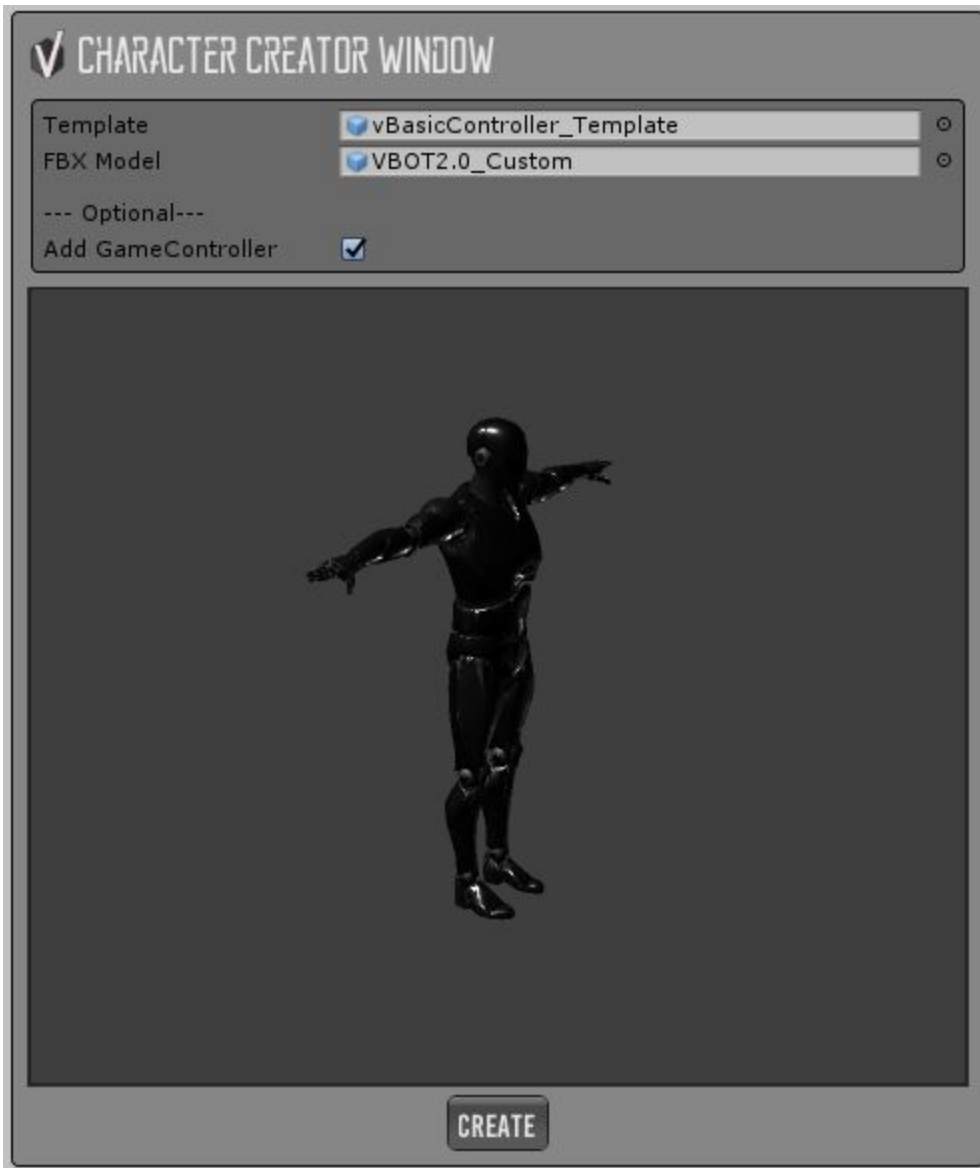
To setup a new basic character, go to the tab *Investor* > *Basic Locomotion* > *Create Basic Controller*

To setup a new basic character, go to the tab *Investor* > *Melee Combat* > *Create Melee Controller*

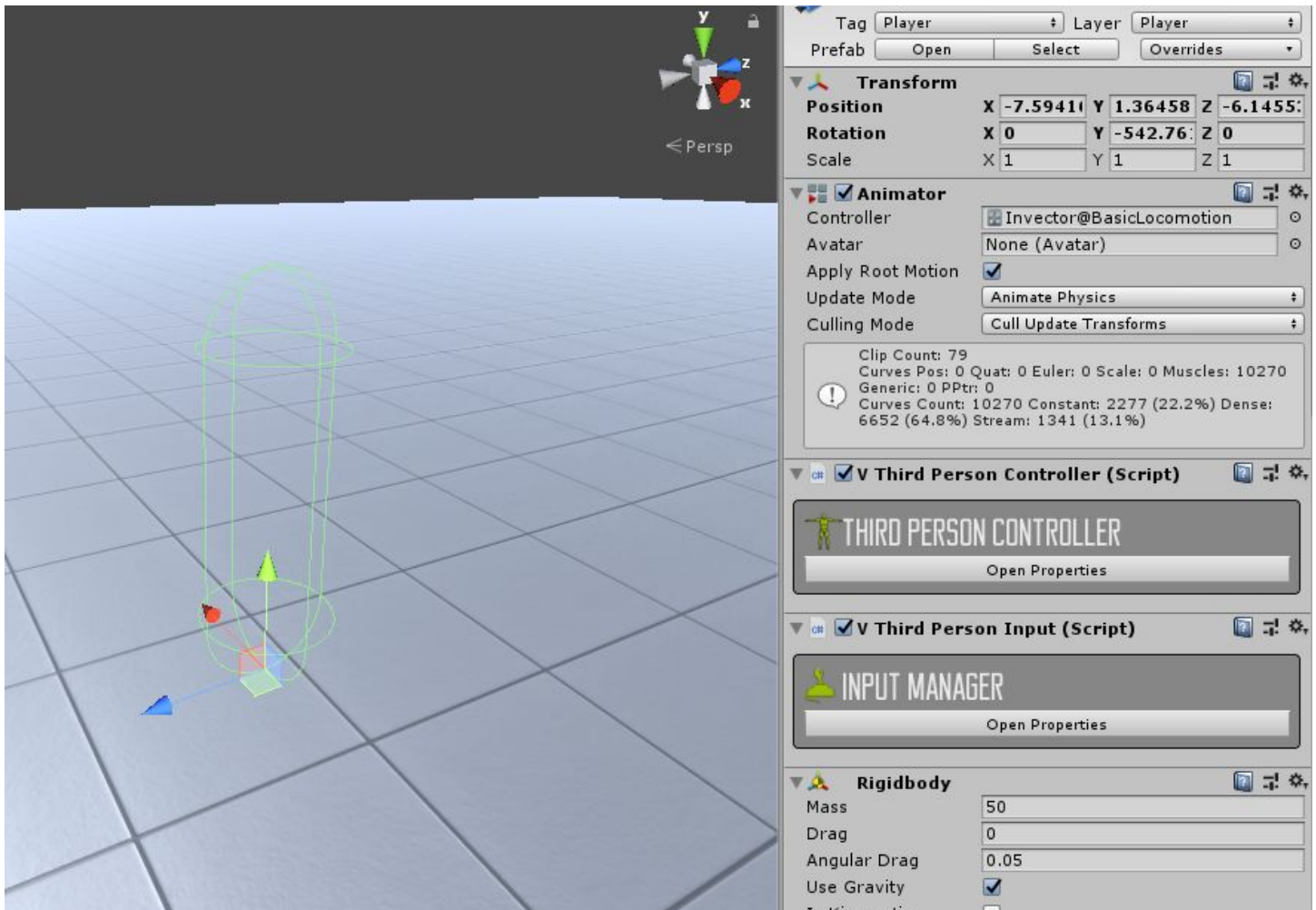
To setup a new basic character, go to the tab *Investor* > *Shooter* > *Create Shooter Controller*



Make sure your Character is **Fully Rigged** and set up the FBX as a **Humanoid**, then assign the FBX to the field "FBX Model" then Click on the button "Create" to finish the character.



The `vBasicController_Template`, `vMeleeCombat_Template` and `vShooter_Template` depending on what controller you're creating, is already assigned in the Template field and is a Controller Prefab that contains everything ready to go, you just need to assign the model to update the Animator Avatar.



The **Character Creator** window takes care of all the hard work for you.

Depending on what Controller you created (Basic, Melee or Shooter) you will find different components inside, such as the Inventory or AimCanvas (Shooter requirement)

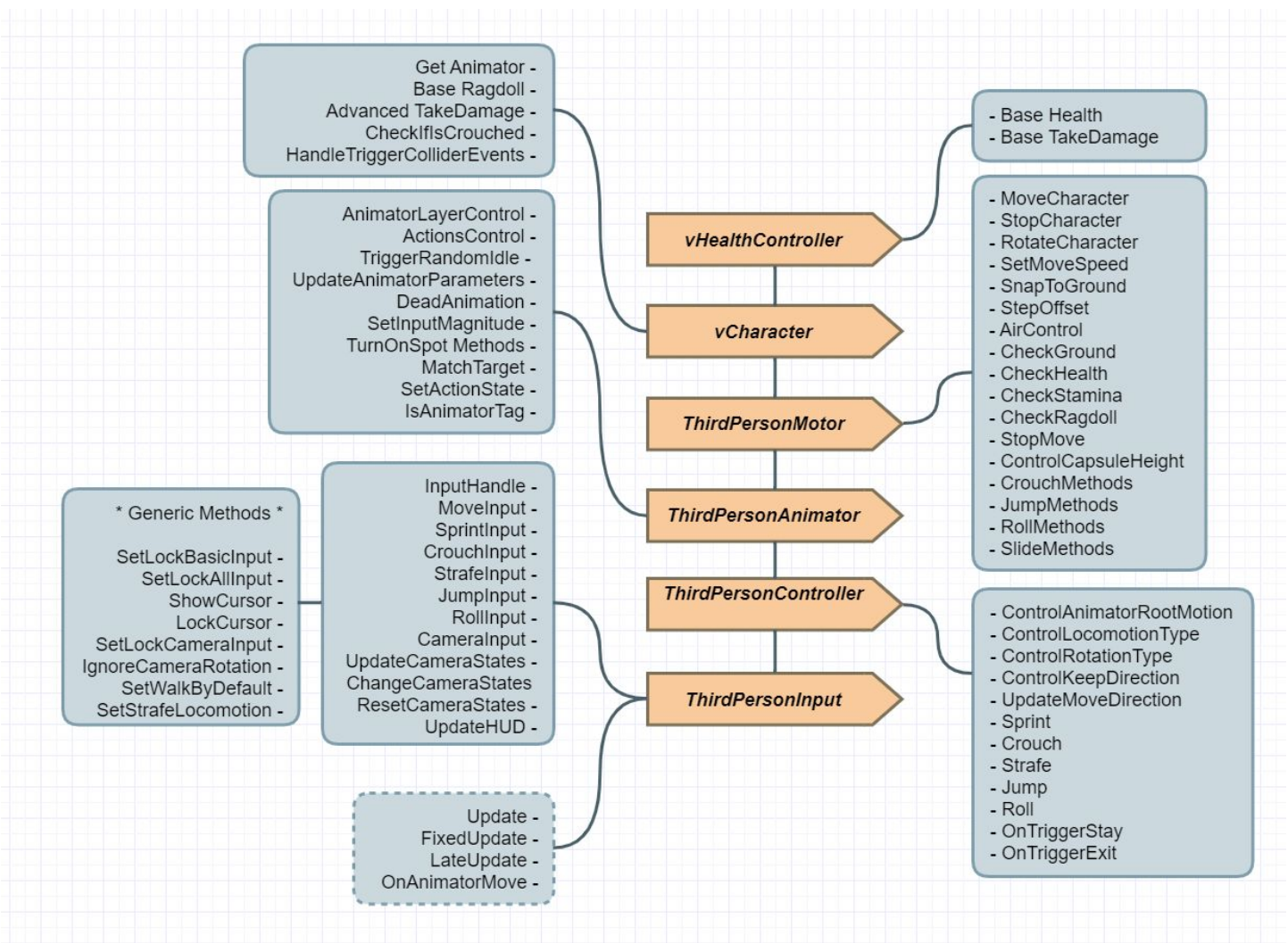


Hit Play and enjoy ☺

HOW IT WORKS?

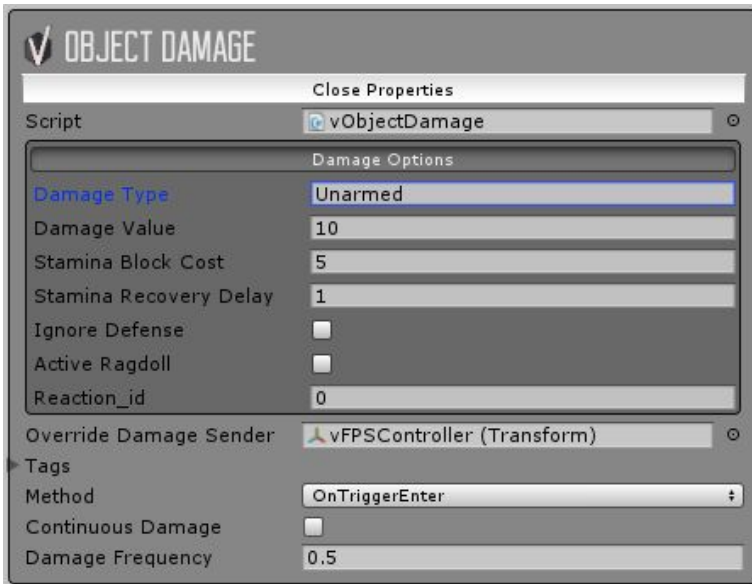
The Controller works with **six main scripts**:

- 1- **vHealthController** takes care of Health/Stamina and has the method TakeDamage to apply damage.
- 2- **vCharacter** - it prepares the vHealthController to be a vCharacter using our animator parameters, ragdoll and action system.
- 3- **vThirdPersonMotor** handles all the information of rigidbody, colliders, verifications of ground distance, stepoffset, slope limit, etc...
- 4- **vThirdPersonAnimator** is responsible to control the behavior of animations
- 5- **vThirdPersonController** manages methods like sprint, crouch, roll, jump, etc...
- 6- **vThirdPersonInput** receives all the input and calls every method of the other scripts on Updates.



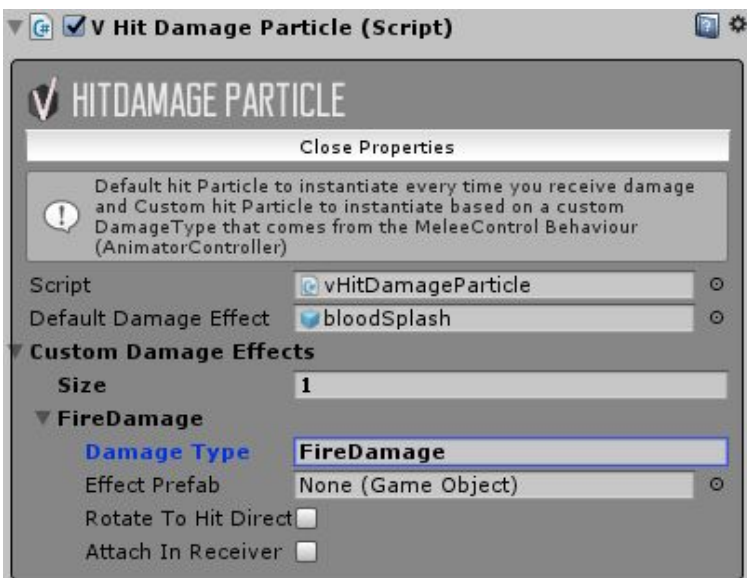
HOW TO APPLY DAMAGE TO THE PLAYER

We have a few examples on how to apply damage to the Controller, basically you need the **vObjectDamage** script which is attached to the Pendulum and Spikes, an Object Damage can send damage to any object that uses a **vHealthController**.



- **DamageType**: Used together with the HitParticleDamage component, you can trigger different particles for different types of damage.

For example if you have an area with fire, you can add a vObjectDamage there and add a DamageType of "FireDamage", then add a Custom Damage Effect with the same DamageType to your HitDamageParticle on your Character and add the particle effect to burn your character.

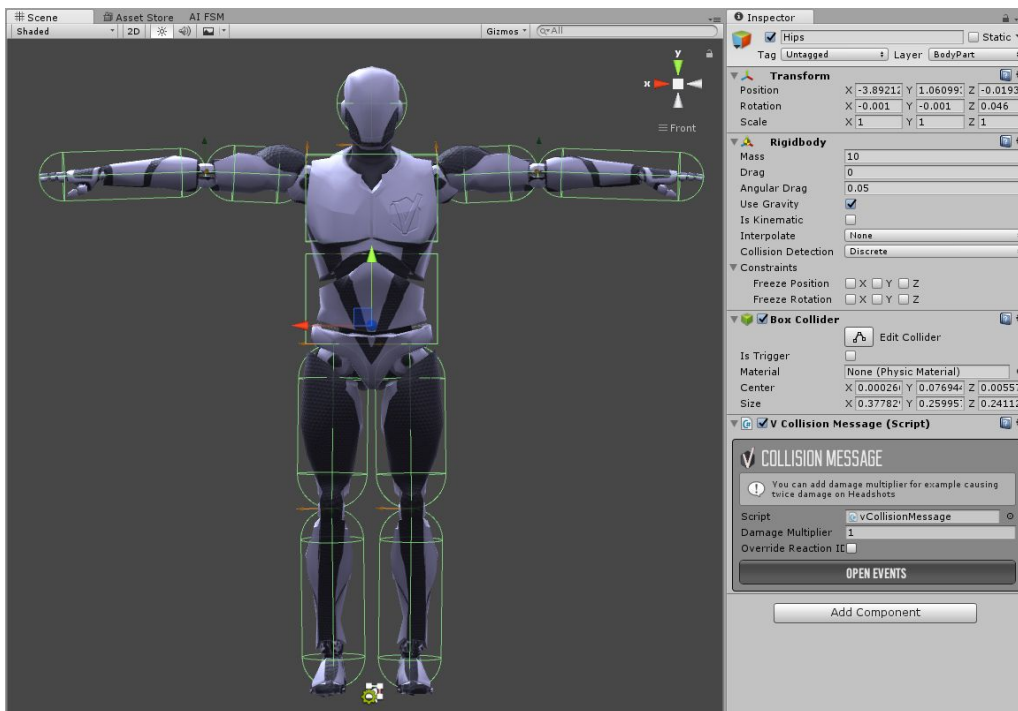


*This component is attached to the Controller or any object that contains the HealthController

- **DamageValue:** How much damage it will be applied to the vHealthController of the target.
- **Stamina Block Cost:** You can ignore that option, it's only for the ThirdPersonController.
- **Stamina Recovery Delay:** You can ignore that option, it's only for the ThirdPersonController.
- **Ignore Defense:** If you're using a MeleeCombat Controller, it will ignore the defense and apply damage anyways.
- **Active Ragdoll:** It will activate the ragdoll on your character, if it has one.
- **Reaction ID:** You can trigger specific hit reaction animation, you can use -1 if you don't want to trigger any animation.
- **Override Damage Sender:** Assign the root object otherwise the AI will target the object that has this component instead. For Example: If you apply the vObjectDamage to be a Hitbox of a LeftHand of a character, the vHealthController or AI will have the LeftHand as the target instead of the GameObject parent.
- **Tags:** What tags you will apply damage to
- **Method:** OnTriggerEnter or OnCollisionEnter
- **Continuous Damage:** Useful for fire damage for example
- **Damage Frequency:** Frequency to apply the damage, if Continuous Damage is enabled.

Using the Ragdoll Colliders as BodyParts to inflict precise damage.

SHOOTER > If you want to cause damage for each body member using the ragdoll colliders, **UNCHECK** the "Disable Colliders" and you can add damage multiplier on each member.



* You must use a different Layer in the Ragdoll Colliders like "BodyPart" and another for the main capsule collider like "Enemy", this way the Detection will detect the Enemy object as a target, but the ShooterManager will actually apply damage to the "BodyPart".



Bodyparts use the Damage Receiver to receive damage.

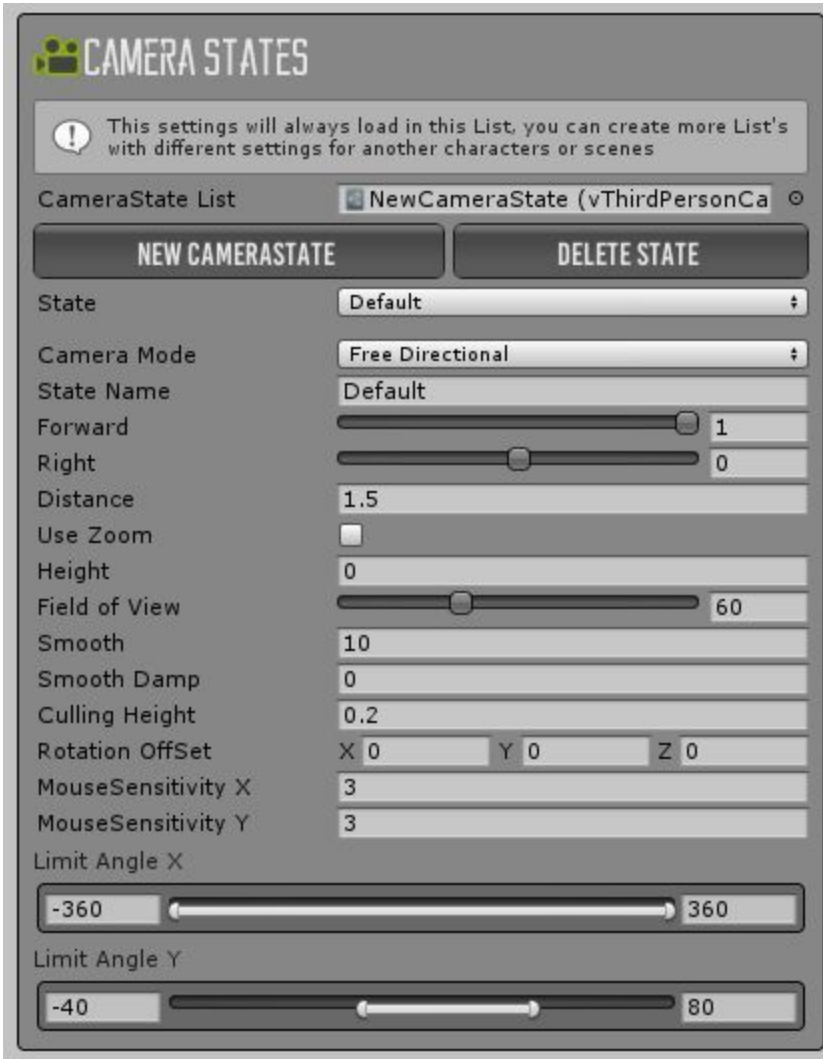
A DamageReceiver is attached to each ragdoll collider, this will allow to Player or AI to apply damage to each bodypart instead of the CapsuleCollider.

- **Damage Multiplier:** multiply the damage value
- **Override a Reaction ID:** Check this option to override the hit reaction animation to trigger a specific animation.

For example, if you want to cause 2x damage and trigger a specific reaction animation when shooting in the Head, simply change the values in this component.

CREATING A NEW CAMERA STATE

With the Third Person Camera you can create new CameraStates to manage different values, states like “Default”, “Aiming”, “Crouch”, to set up new camera position, distance, height, etc.



You can call the method `ChangeCameraState()` from the `tplInput` via Triggers or Custom Scripts.

Example:

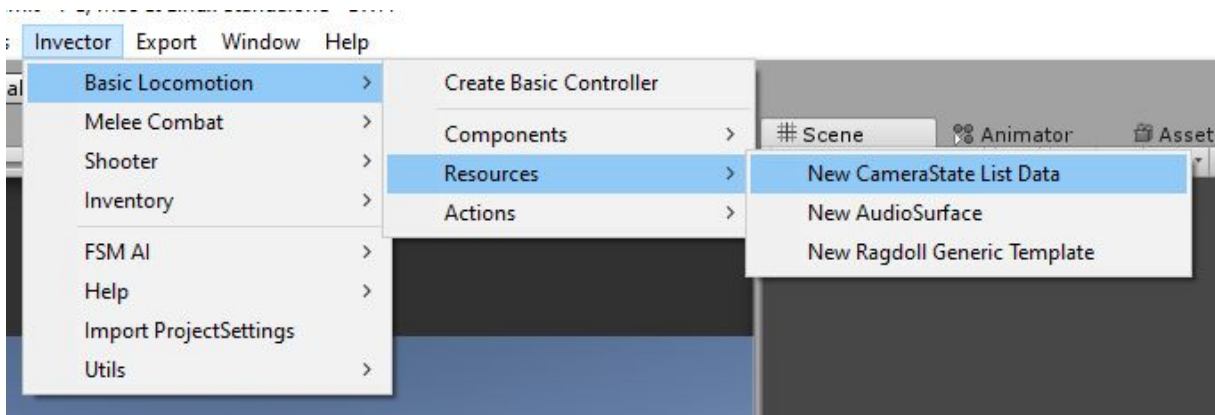
```
tplInput.ChangeCameraState(string cameraState, bool useLerp = true)
```

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

You can also call a method to reset to the defaultCameraState

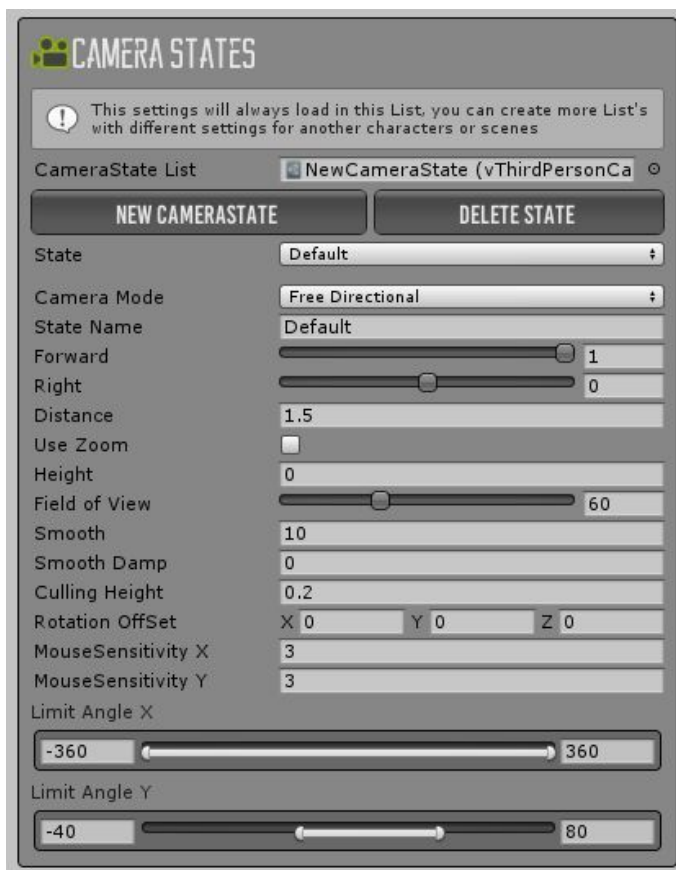
```
tplInput.ResetCameraState();
```

You can create a new **CameraState List Data** and assign it in the CameraState List field on TP Camera Inspector.



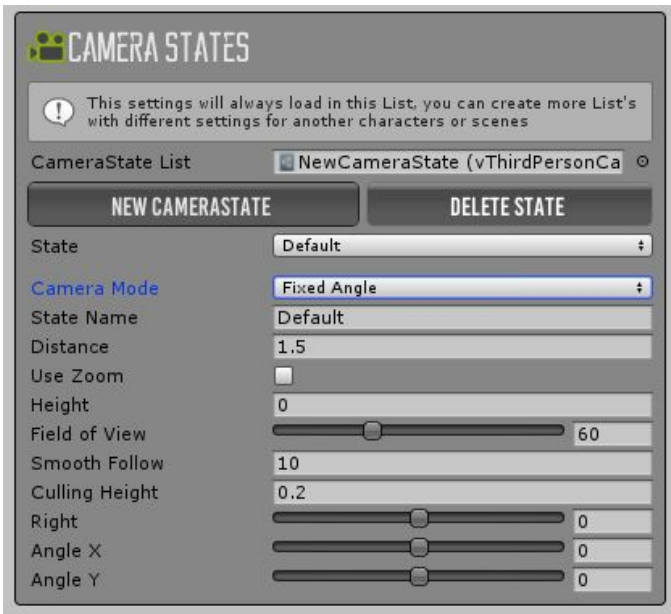
CameraMode - Free Directional

This CameraMode offers 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...



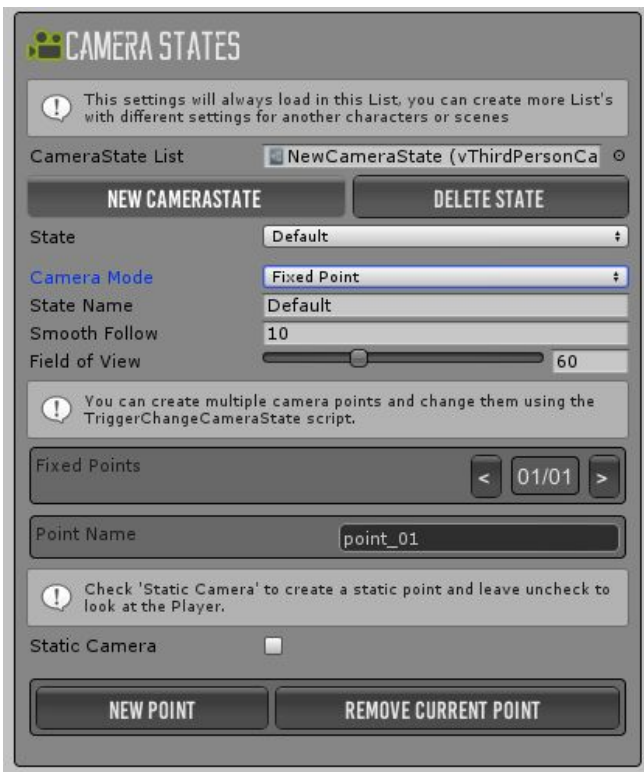
CameraMode - Fixed Angle

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.

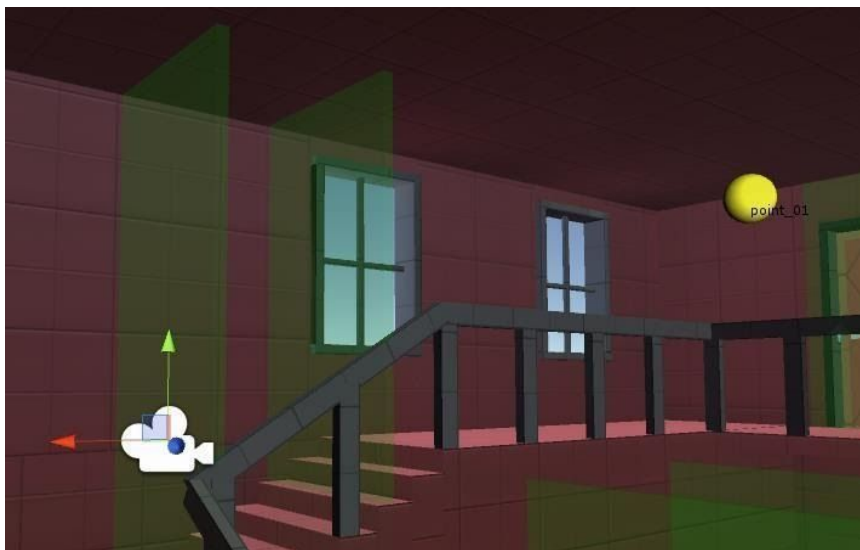


CameraMode - Fixed Point

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 **XboxOne controller**.

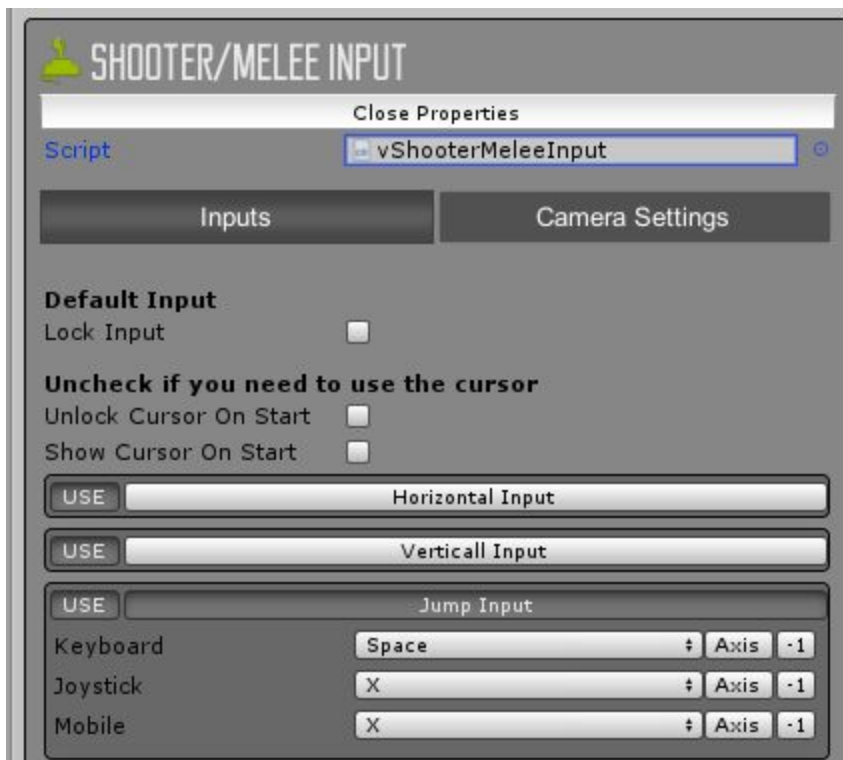
Make sure you imported our InputManager by going to **Investor > Import ProjectSettings**.

For other controllers like PS4, Logitech or Generic you need to remap the InputManager based on your controller inputs.

INPUT MANAGER

The InputManager manages input for the character actions and movement.

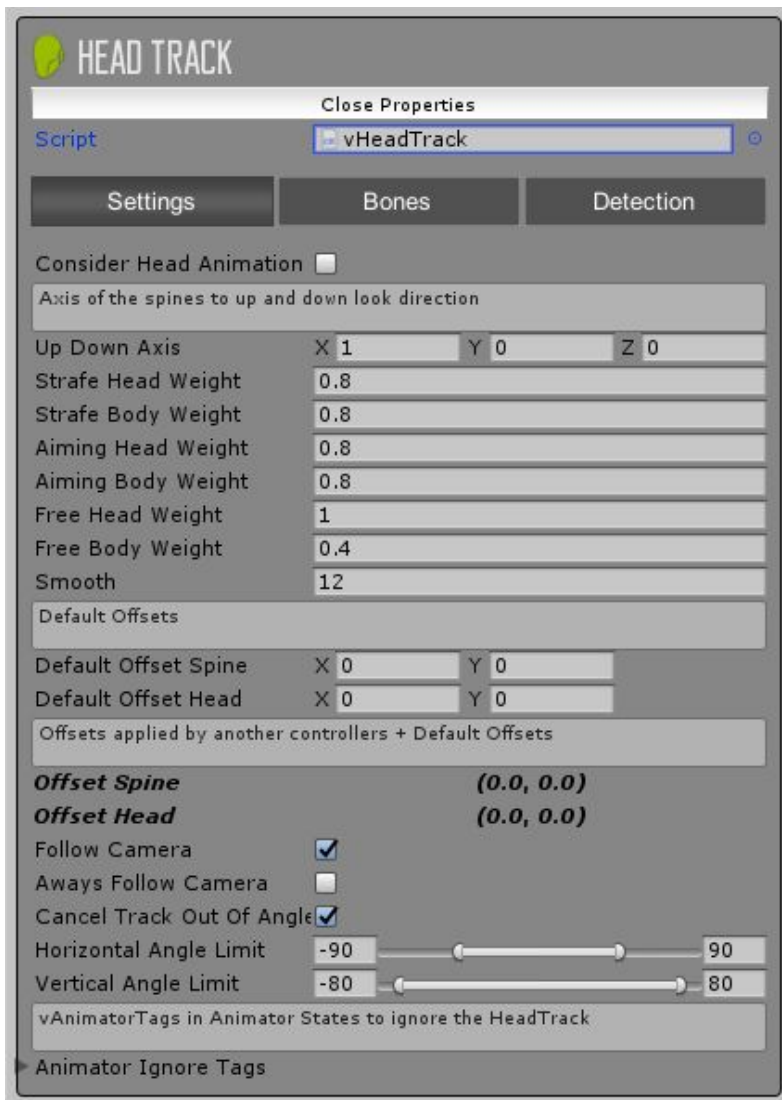
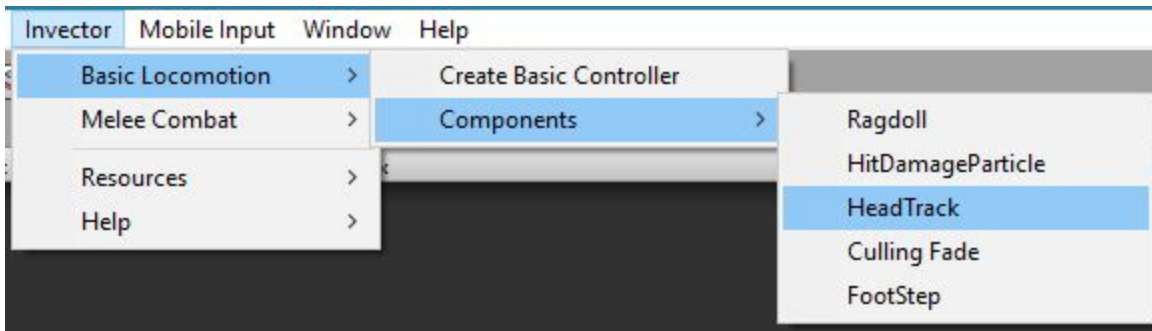
If you created a **Basic Locomotion** it will use the **vThirdPersonInput**, if it's a **Melee Combat** character it will use the **vMeleeCombatInput**, and the **Shooter** uses the **vShooterMeleeInput**.



You can remap the input and disable it if you don't need to use it for your game.

HEAD TRACK

*Shooter - automatically add the headtrack in order to aim up/down



UpDownAxis: Depending on your character's rig, the orientation could be different so if you're looking up/down and the character is responding left/right instead, try changing the value 1 of the X, Y, Z until the character looks normal.

Weights: You can set different weights depending on the **Locomotion Type** (Free Movement or Strafe)

Angle X and Y: You can limit the max angle to look up/down and left/right

Smooth: the speed to look at something

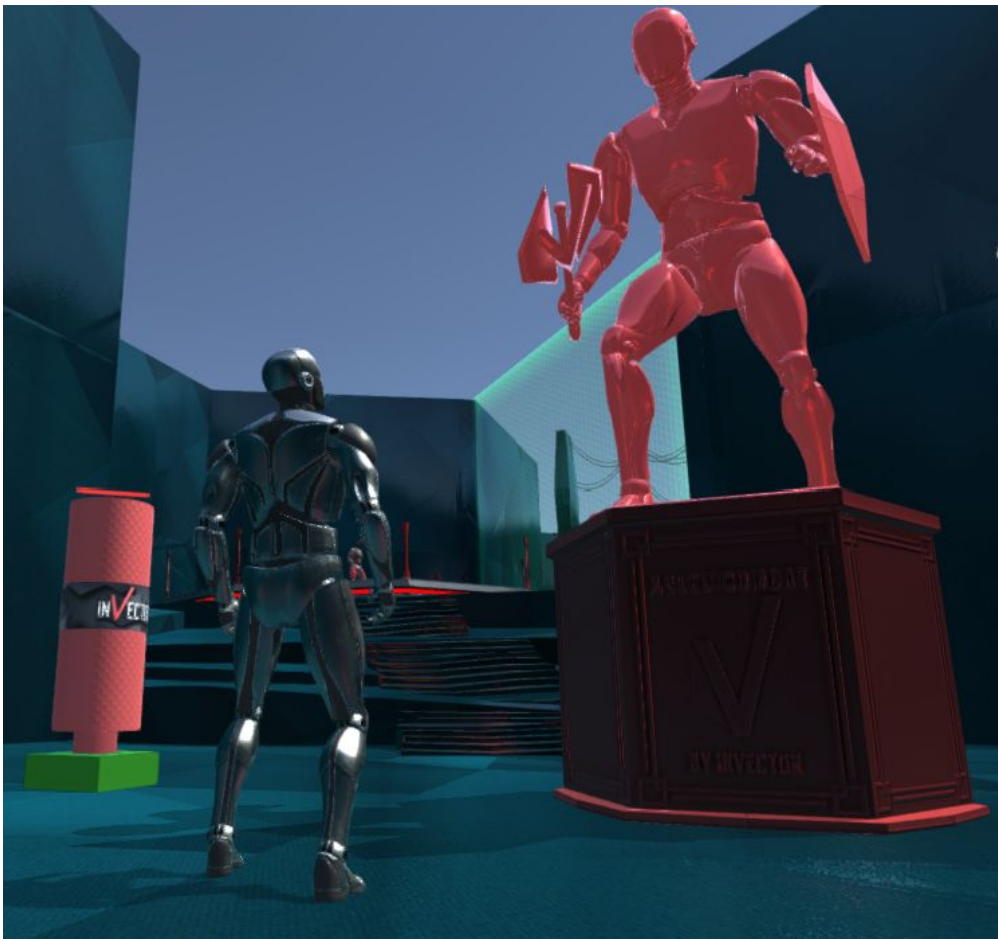
AnimatorTags: Which animator tags will ignore the Headtrack, for example you have an animation that the character opens a door and look at the knob, it's best to not use the headtrack while this animation is playing, making the animation play as it was designed without the headtrack interfering.

Offset Spine//Head: more offset options in case you want to fine tune your headtrack.

Follow Camera: tracks the camera forward, disable when using a topdown view

Always FollowCamera: force the controller to always follow the camera regardless of AnimatorTags

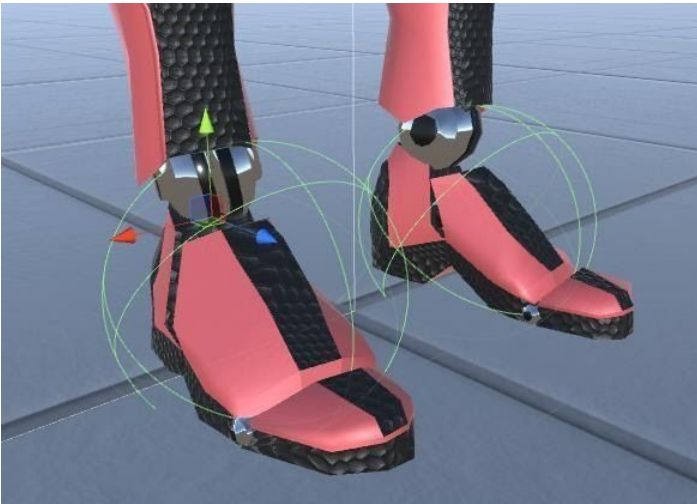
To make the character look at an object, you need to add the component **vLookTarget** into the object, you can take a look at several examples in the DemoScenes.



FOOSTEP AUDIO SYSTEM

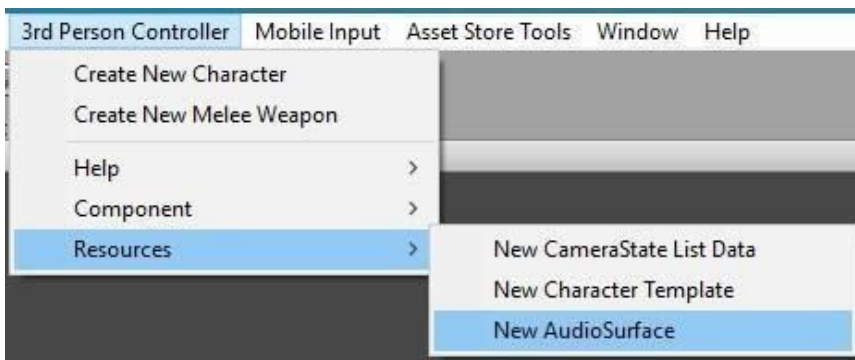
Video tutorial: <https://www.youtube.com/watch?v=gxesgNH0UBM>

When you create a new Character the FootStep component will be already attached, if you want to add a component into another Character go to the *3rdPersonController Menu > Component > FootStep*. The component will automatically create a **sphere collider** on the foot of your character, but you need to make sure that the Radius and Position of the sphere is **touching** the ground.



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

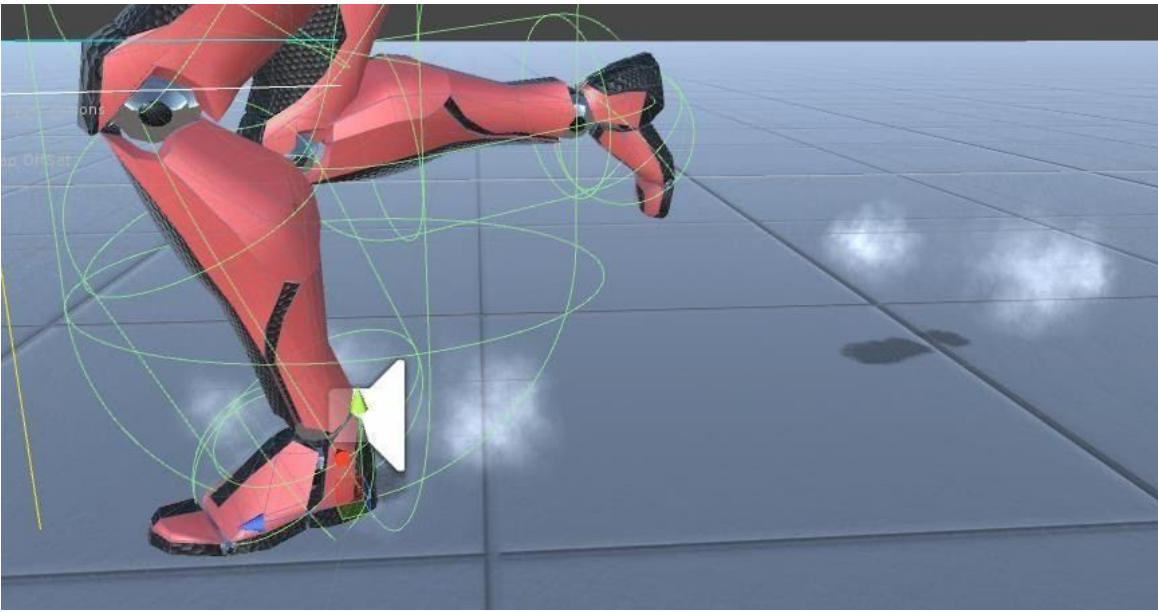
To create a new AudioSurface go to the 3rd Person Controller menu > Resources > New AudioSurface.



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

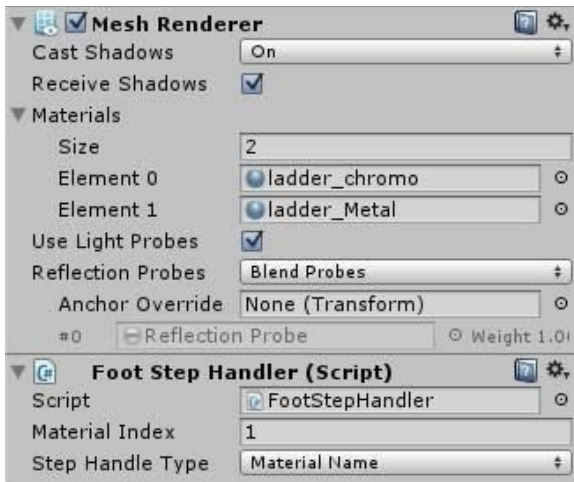


You can assign an AudioMixer for better control surfaces, and you can instantiate a **Particle** as well, see the example on the DefaultSurface call 'smoke' that also uses a **StepMark** sprite call SimpleStepMark.



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

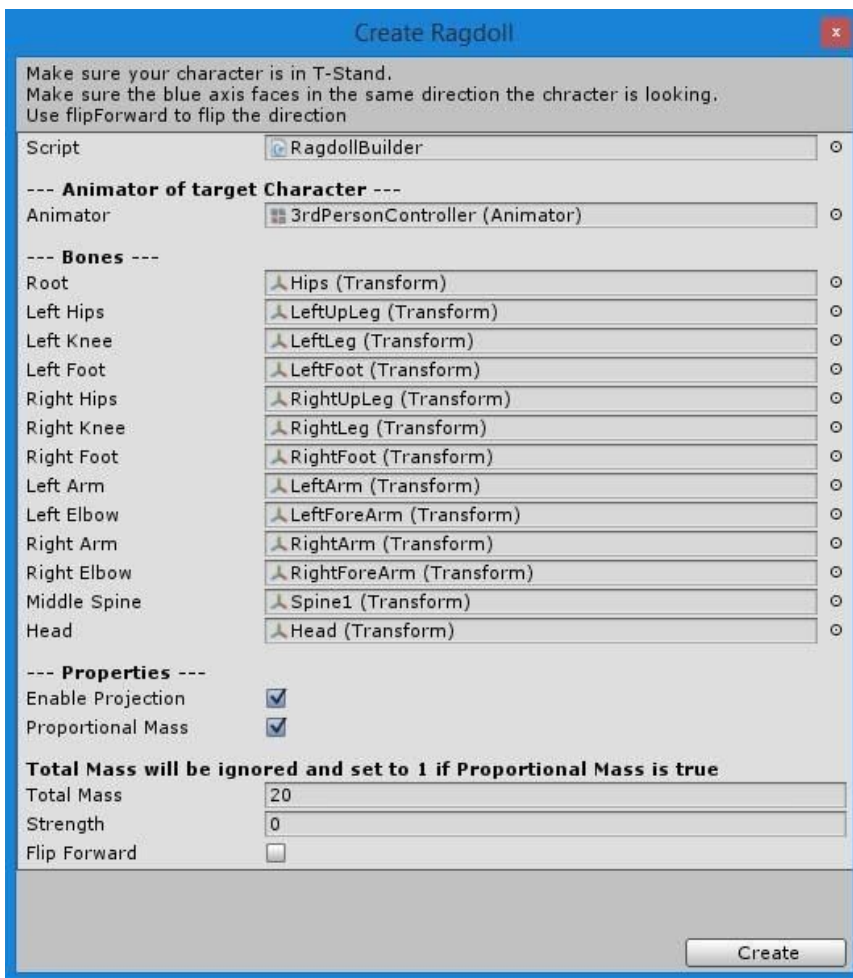
If your gameobject has multiple materials and you need to play a specific material, you can use the FootStepHandler script and set the correct Material Index of your object. (*See example on the Ladder prefab)



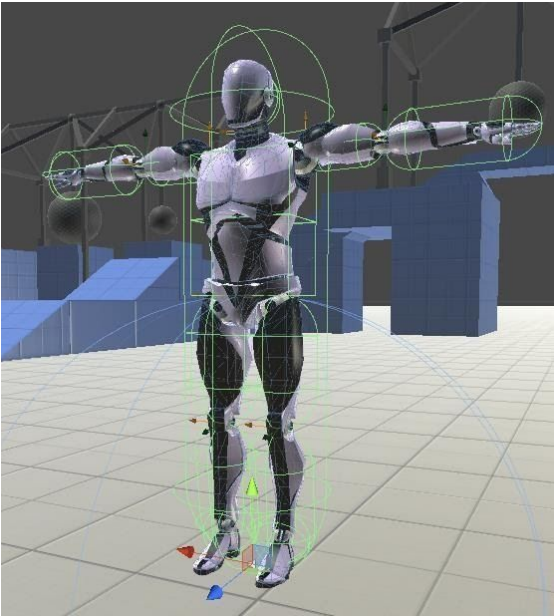
CREATING A RAGDOLL

Creating a Ragdoll is just as easy as creating your Character, just go to the tab *Inspector* > *Basic Locomotion* > *Components* > *Ragdoll*.

If you have your character selected on the Hierarchy, all the fields will **autofill**, if not, just click on your character and it will autofill for you, this template was designed to **save time**, so you don't have to waste your time dragging and drop every bone, instead just hit the "Create" button and it's ready to go.

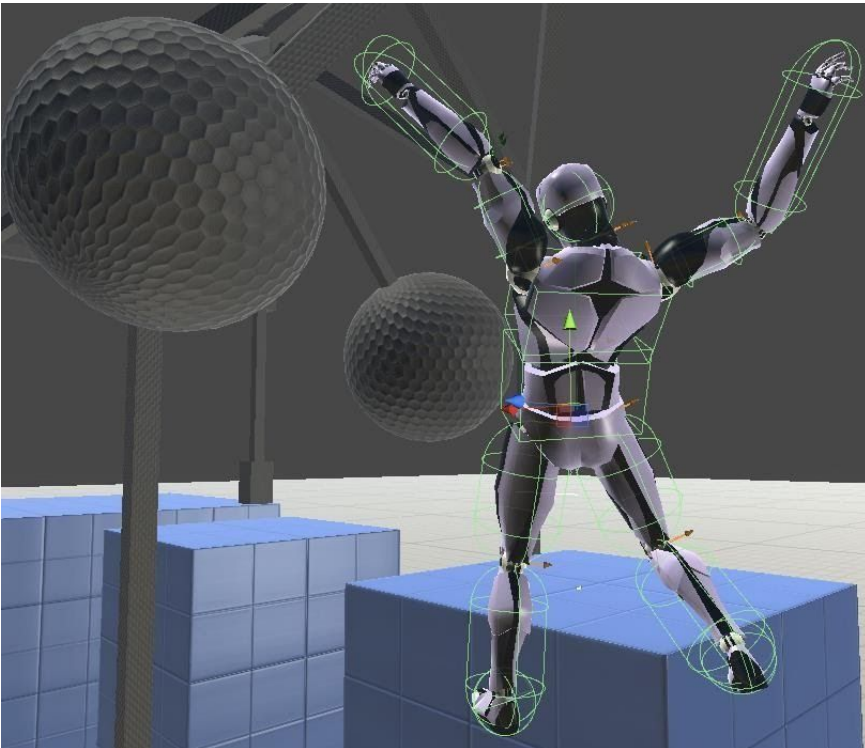


We strongly recommend keeping the **Enable Projection** and the **Proportional Mass** enabled, and do not forget to use **Scale Factor 1** on your **fbx Model**. This you provide better behavior of your ragdoll.

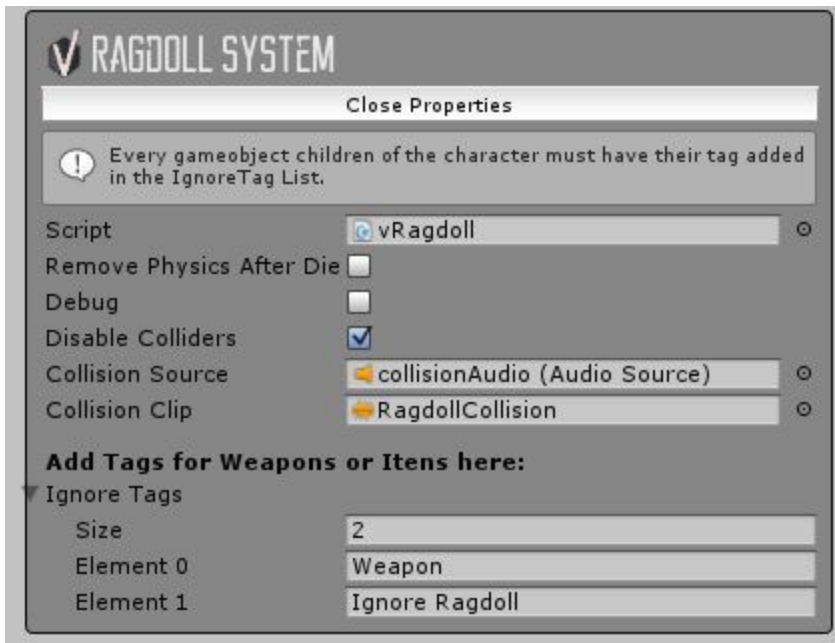


To enable the ragdoll, you can use the Script `ObjectDamage` or just call this line on the `OnCollisionEnter` method.

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



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



* **SHOOTER** > If you want to cause damage for each body member using the ragdoll colliders, UNCHECK the “Disable Colliders” and you can add damage multiplier on each member.

Ps* Don’t forget to add the Layer “BodyPart” for each collider.

HOW TO ADD NEW ANIMATIONS/ACTIONS?

We have 2 excellent video tutorial showing examples on how to add simple and complex animations

Simple > <https://www.youtube.com/watch?v=VVqkSIQ4x2M>

Complex > <https://www.youtube.com/watch?v=hLLWnsIQz-c>

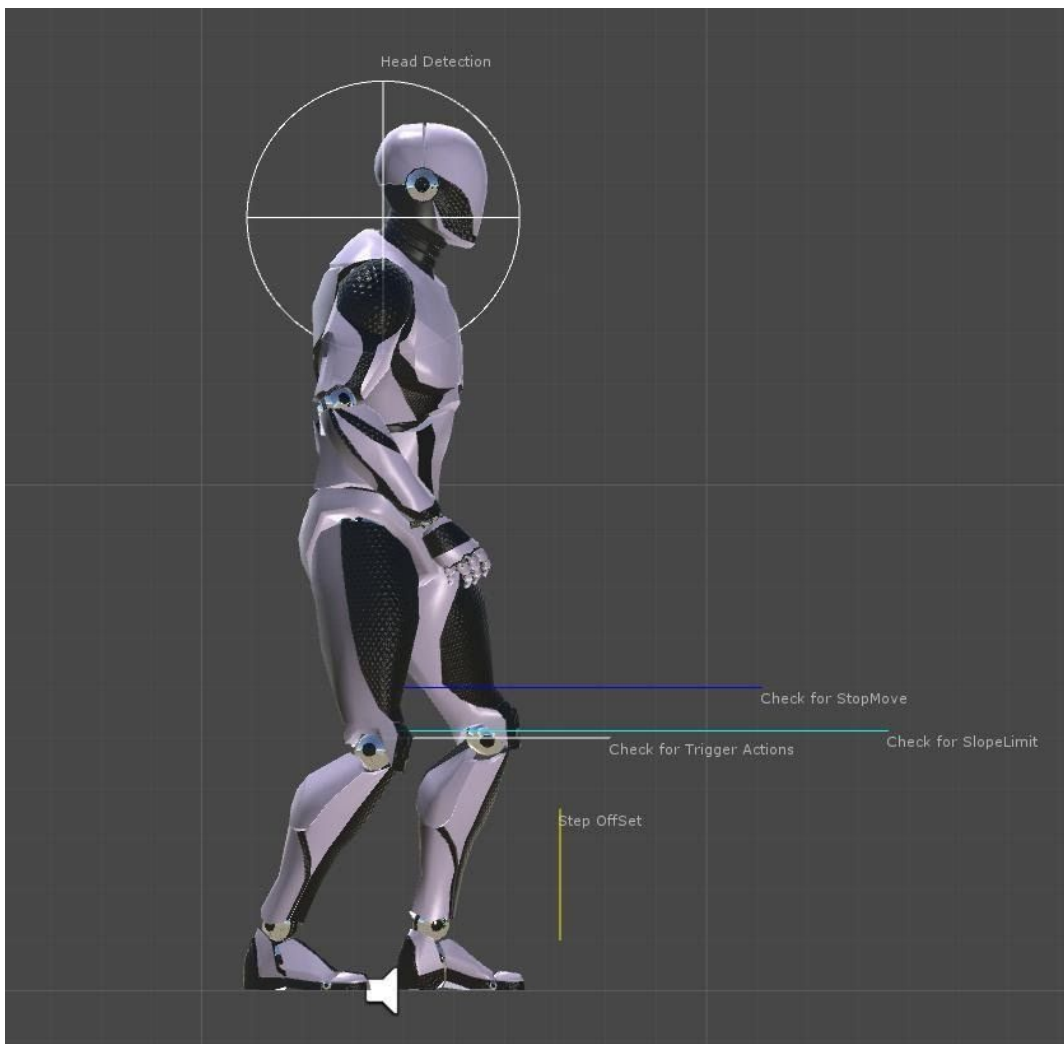
RAYCAST CHECKERS

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

StopMove is a Raycast that detect any object with the layer (Default, StopMove) to prevent the character to walk in place, you can use a StopMove in an invisible wall for example, and the camera will not clip, because the culling layer is set to "Default".

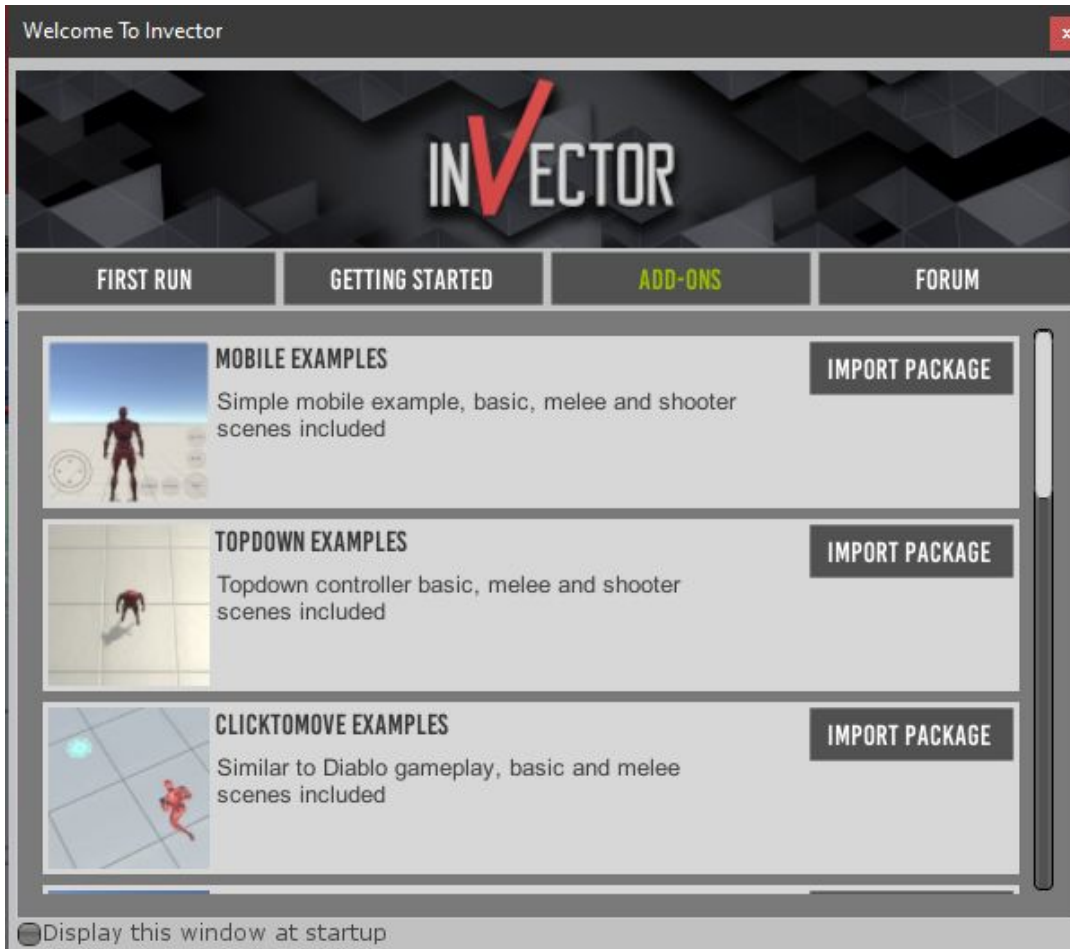
SlopeLimit will prevent the character from walking in absurd angle heights, float customizable on the Player Inspector.

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

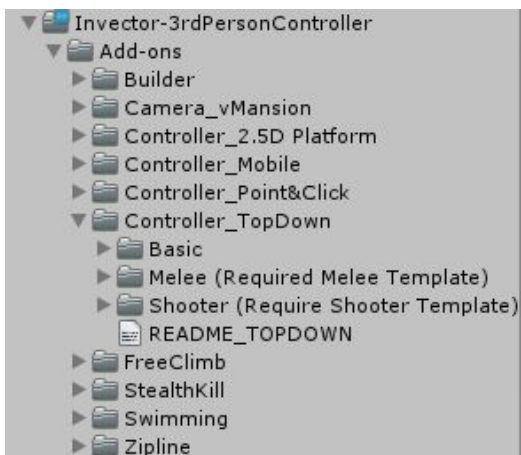


TOPDOWN / 2.5D / POINT & CLICK CONTROLLER / MOBILE

We have different types of controllers that are included in the project as a bonus, you can import those packages by going to *Invector* > *Add-ons*



If you don't own the Melee or Shooter templates, you don't need to import their folders.



To turn your Third Person Controller into a TopDown or Isometric controller just go into your ThirdPersonCamera and change the CameraState to **TopDown@CameraState**, **Isometric@CameraState** or **2.5@CameraState** depending on what controller you want.



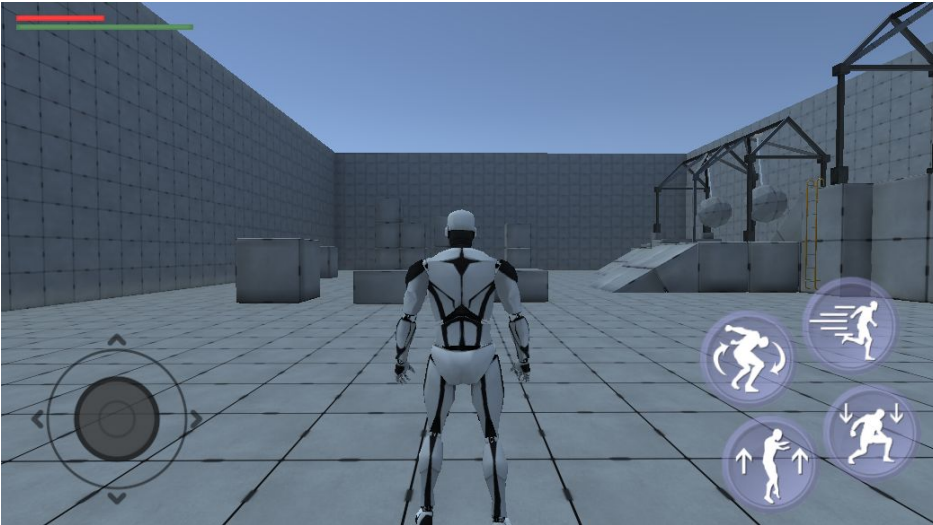
Go to the add-on folder you want and replace your current **vThirdPersonController** component for the **vTopDownController** or **2_5Dcontroller** in the Player Inspector.

To use the **Point&Click** you can still use the **vThirdPersonController**, but you will need to replace the Input to **vPointAndClickInput** or **vMeleePointClickInput** (if it's a melee character), for more information check the **Invictor_Point&Click_Melee**.

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

MOBILE CONTROLS

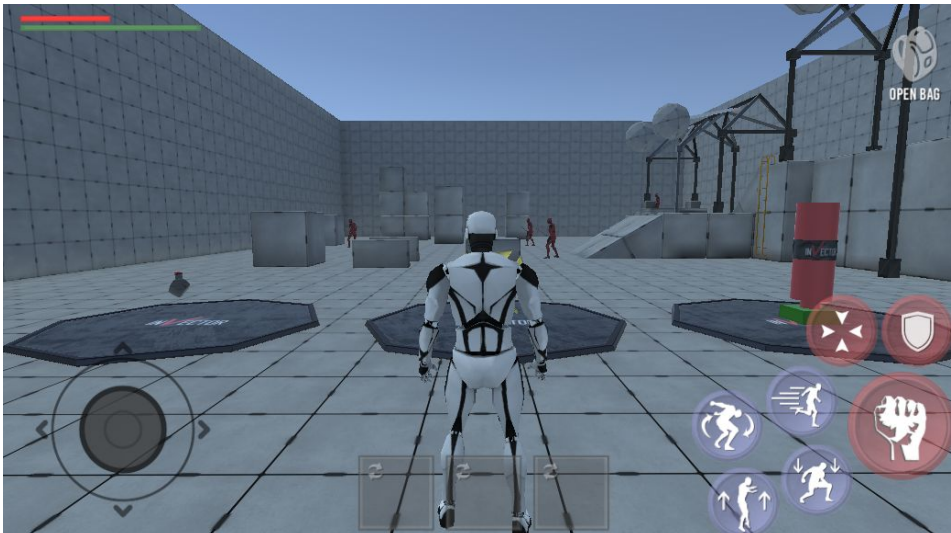
You can import the Mobile demo scenes by going to the menu *Investor > Addons*.



After importing the mobile package, you can create a regular `ThirdPersonController` using the Character Creator Window and drag and drop a Mobile UI prefab inside your character.

You can find the MobileUI prefab examples inside each folder Basic, Melee or Shooter

ps You must own the `MeleeCombat` or `Shooter Templates` in order to use their prefabs, otherwise you will receive warnings about missing content.*

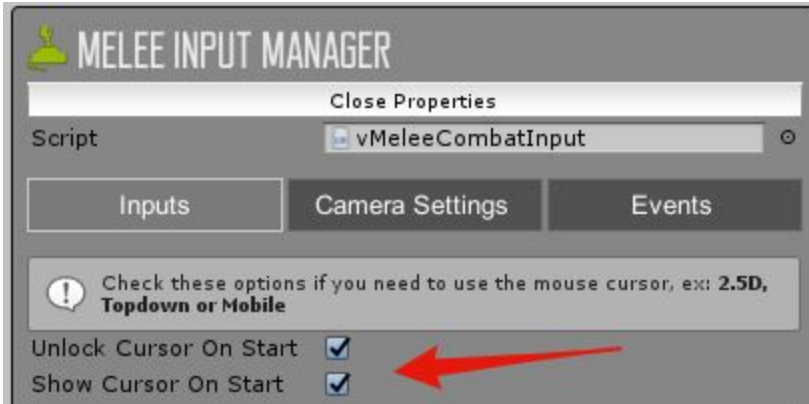


Make sure to check the options “Unlock Cursor and Show Cursor” when using the Mobile UI.

vThirdPersonInput > *Basic Controller*

vMeleeCombatInput > *Melee Controller*

vShooterMeleeInput > *Shooter or Shooter w/ Melee Controller*

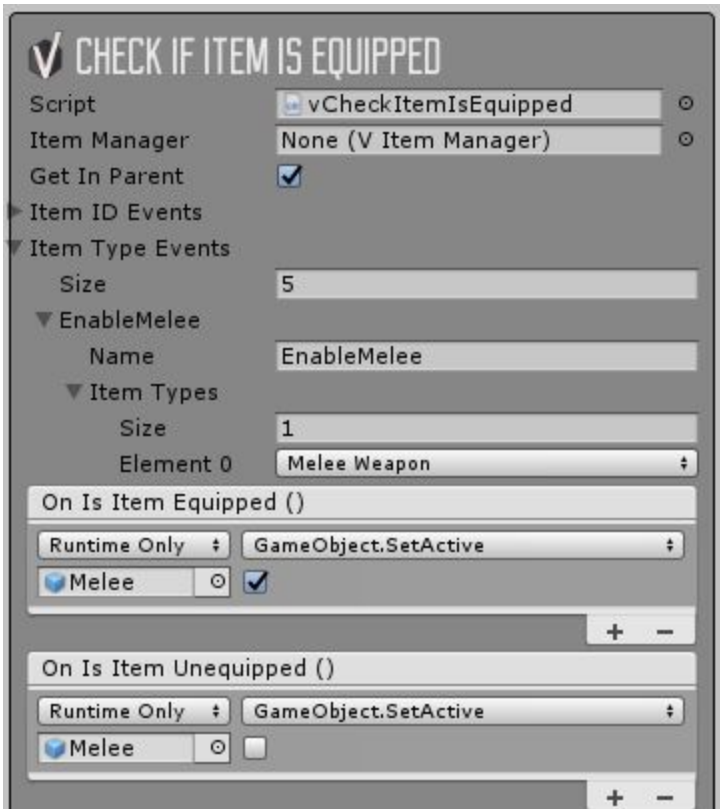


After setting up the controller and aligning the handlers, search for the Mobile UI Controls prefabs:

MobileUI_ShooterMelee_Inventory - if you want to use Inventory

MobileUI_ShooterMelee_NoInventory - if you don't want to use Inventory, check the demo scenes to see how it works and choose one style for your game.

You can use the component **vCheckItemsEquipped** if you're using the Inventory to turn on/off mobile buttons:



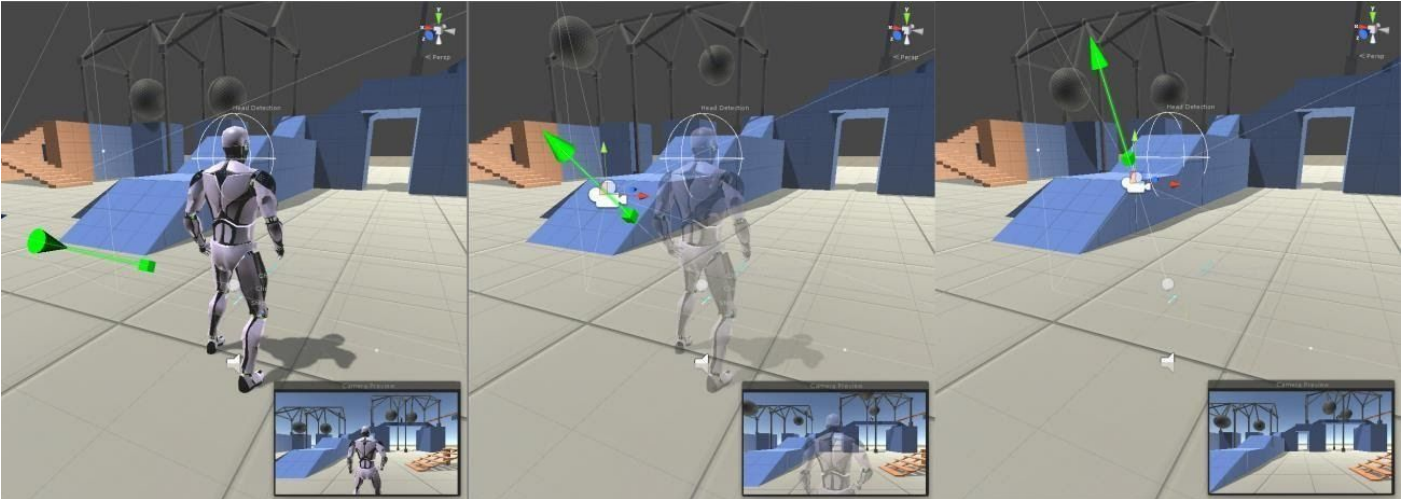
If you're not using the Inventory, you can use the `vCollectShooterMeleeControl` Events to manage the buttons, it identifies if you're using a shooter weapon or a melee weapon only.



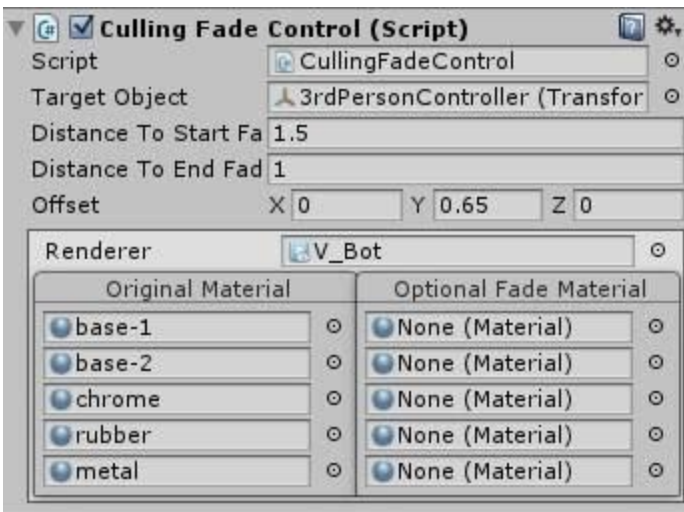
CAMERA CULLING FADE

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

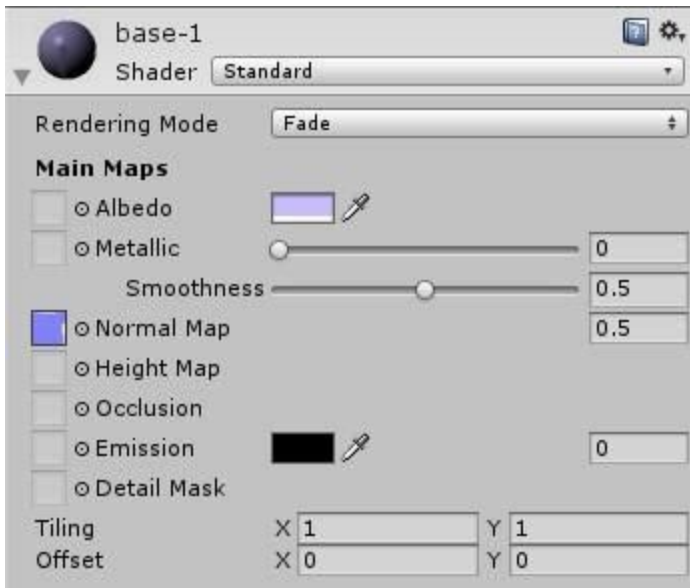
Example:



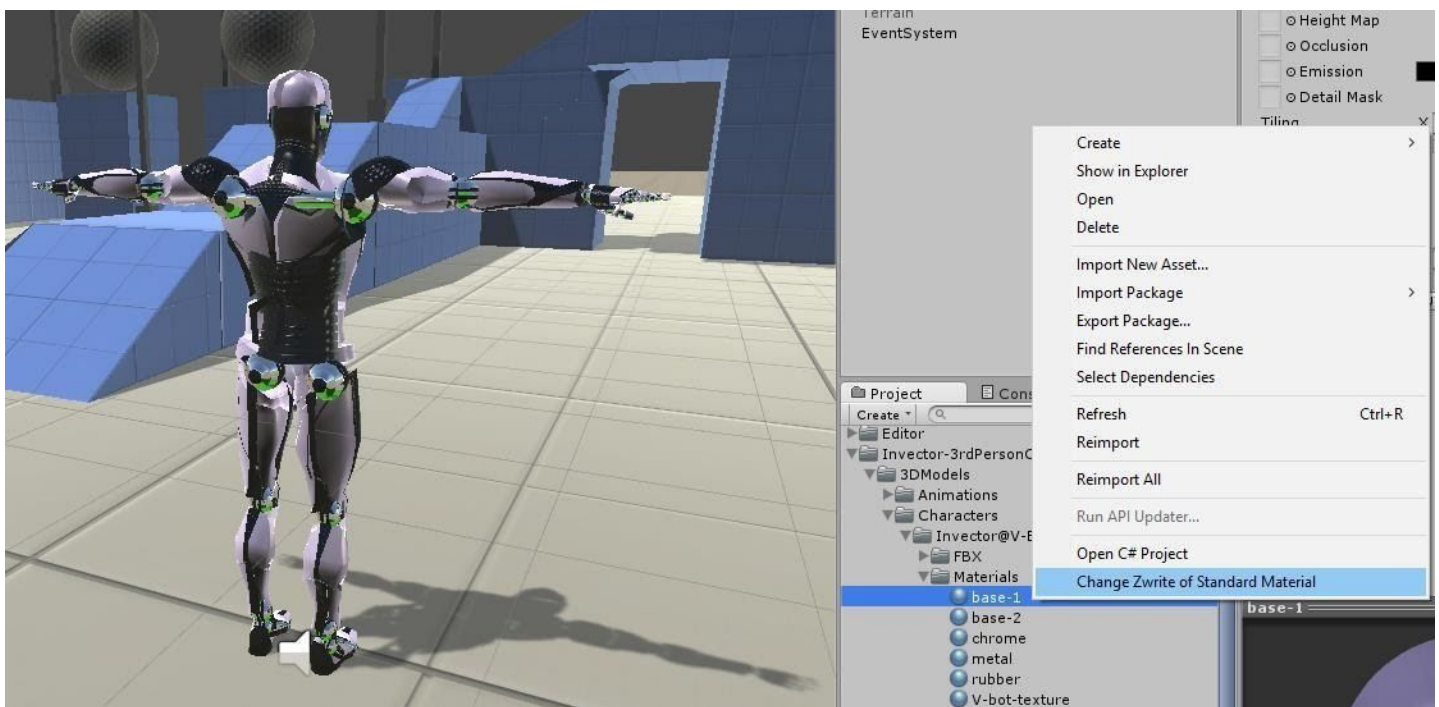
Our Culling Fade will set up automatically for the default Standard Shader of Unity's, but you also can use custom shaders, just make an additional copy with the fade material and assign in the "Optional Fade Material" field.



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”.

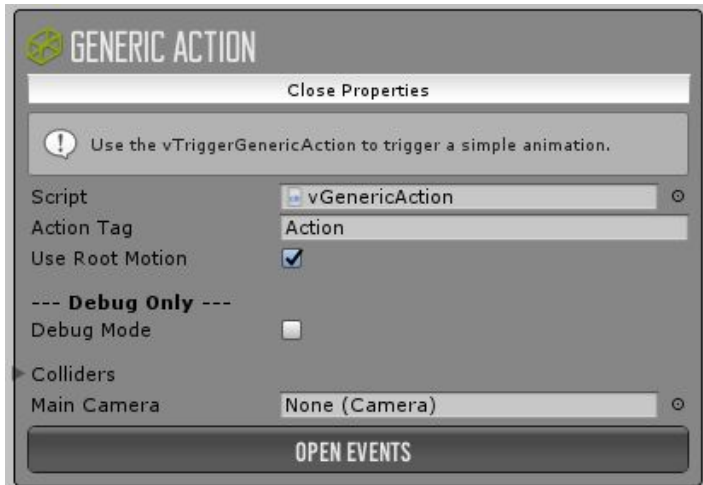


UPDATE V1.1B - now the script will be attached into the Controller just like the Ragdoll and the Footstep, It's a modular feature.

GENERIC ACTION (HOW TO INTERACT WITH OBJECTS)

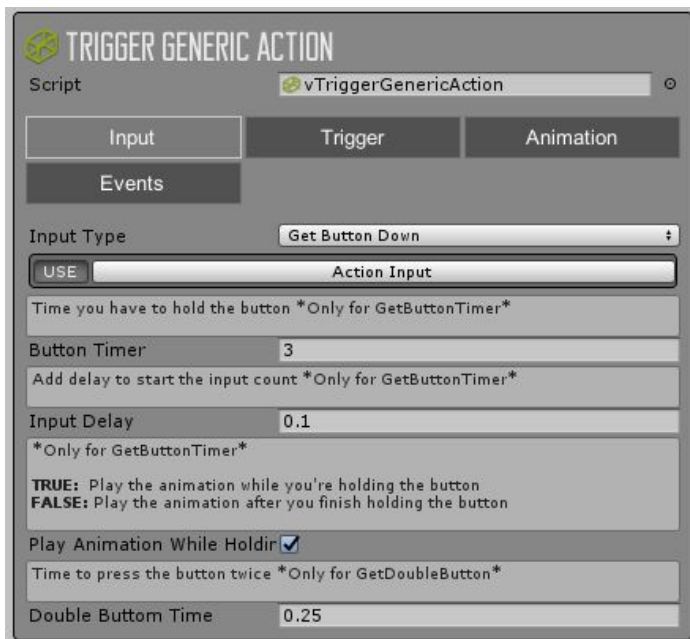
Video tutorial is also available: <https://youtu.be/hLLWnsIQz-c>

The GenericAction is a system designed to trigger simple interaction animations such as opening a door, jump over an obstacle, push a lever, etc...

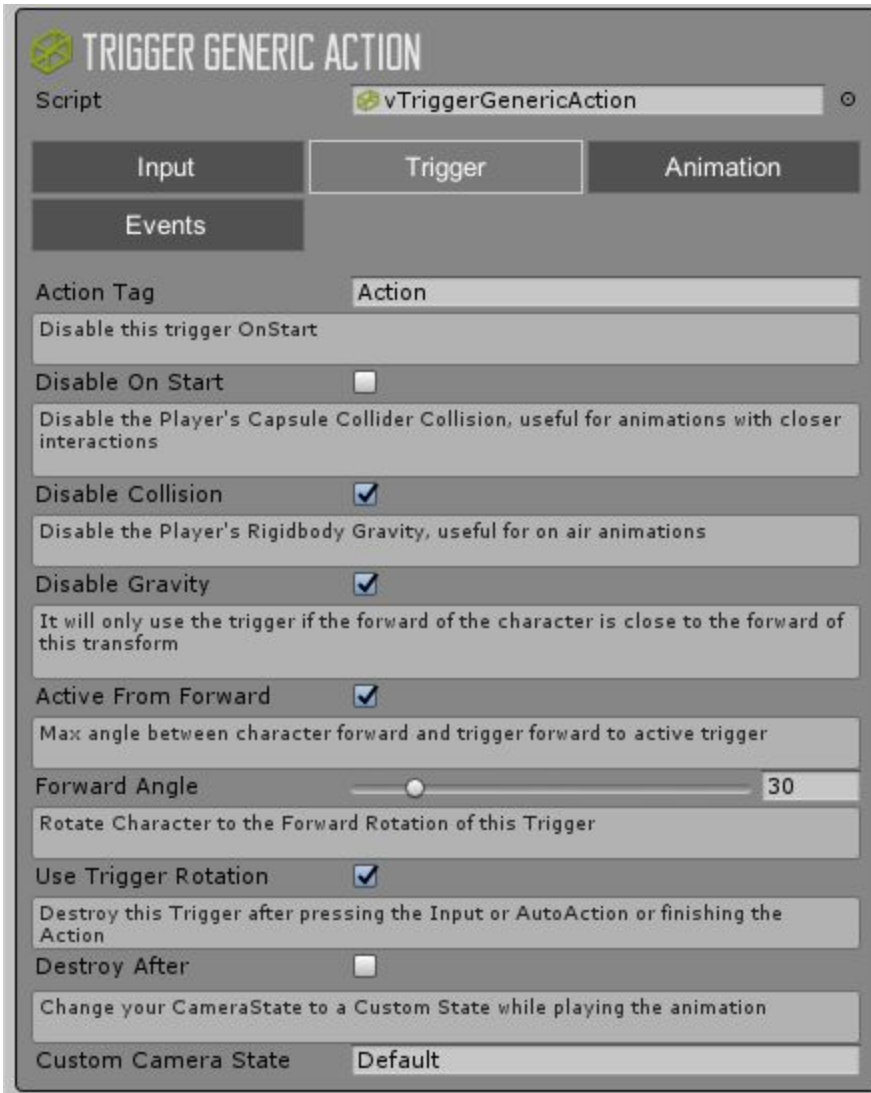


(This component is already added to a Player when creating a new controller)

You need to add a **vTriggerGenericAction** component on a collider **IsTrigger** to perform the action. In the **Input** tab you can assign an **ActionInput** to trigger the action and select between the **InputType** **GetButtonDown**, **GetDoubleButton**, **GetButtonTimer** or perform an **AutoAction** as soon as the player enters the trigger.



There are several options in the **Trigger** tab such as disabling the player gravity and collision, only performing the action if the player is facing the trigger forward, destroying the trigger after the interaction, etc... every option has a description of what it does:



In the **Animation** tab, you must assign the name of the animation clip you want to play it (*not required, you can perform an action without playing an animation as well*)

If you need to align the player into a specific position you can use the [MatchTarget](#) option or our Curve system to place the character into a custom transform while playing the animation.

TRIGGER GENERIC ACTION


Script

Input **Trigger** Animation

Events

Trigger a Animation - Use the exactly same name of the AnimationState you want to trigger, don't forget to add a vAnimatorTag to your State

Play Animation

Check the Exit Time of your animation (if it doesn't loop) and insert here.
 For example if your Exit Time is 0.8 and the Transition Duration is 0.2 you need to insert 0.5 or lower as the final value.
Always check with the Debug of the GenericAction if your animation is finishing correctly, otherwise the controller won't reset to the default

End Exit Time Animation

Use a ActionState value to apply special conditions for your AnimatorController transitions

Animator Action State

Reset the ActionState parameter to 0 after playing the animation

Reset Animator Action Sta

Use Animator Match Target

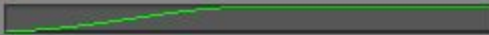
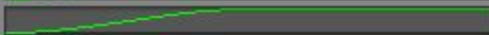
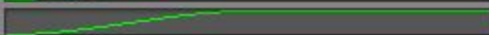
Use a empty transform as reference for the MatchTarget


Match Target

Select the bone you want to use as reference to the Match Target

Avatar Target

Curve Match target system

Use Local X
Use Local Z
Match Position XZ Curve 
Match Position Y Curve 
Match Rotation Curve 

 **These properties are related to the animator's matchtarget system.**
To use our new system curve based, *uncheck the UseAnimatorMatchTarget*
This properties will be removed in next update

Animator Match target system

Check what positions XYZ you want the matchTarget to work

Match Pos X Y Z

Rotate Weight for your character to use the matchTarget rotation

Match Rot

Time of the animation to start the MatchTarget goes from 0 to 1

Start Match Target

Time of the animation to end the MatchTarget goes from 0 to 1

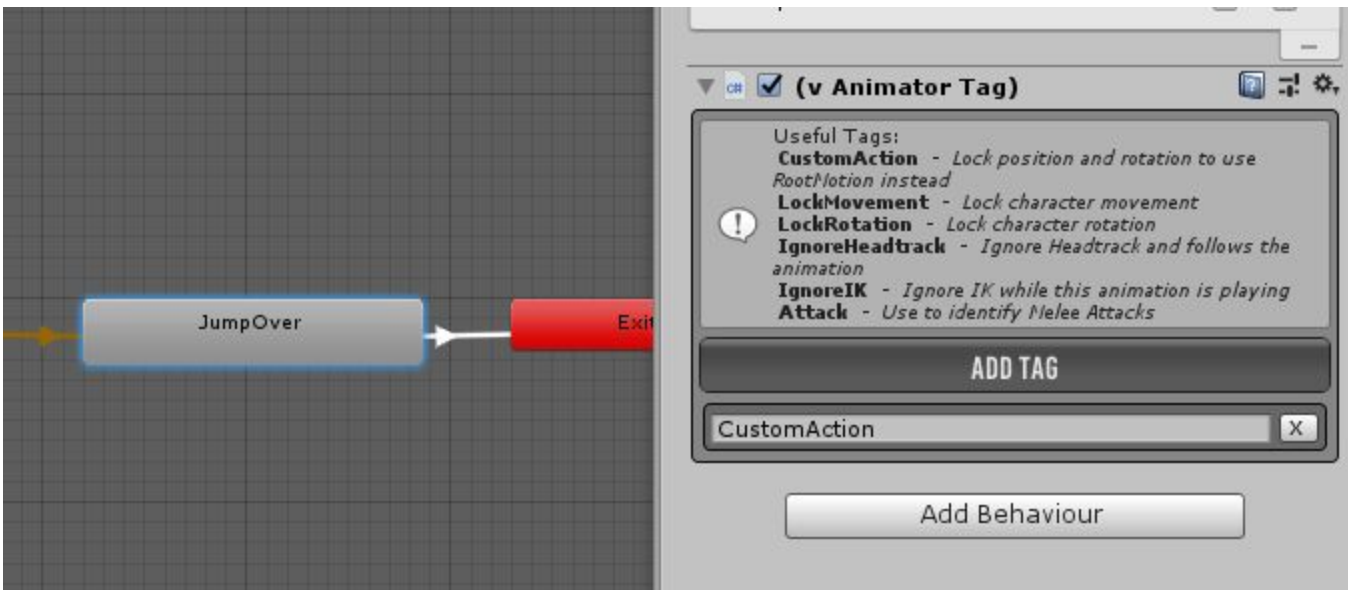
End Match Target

And in the Events tab you can trigger things like Playing a sound or particle effect once you PressTheActionInput, StarAnimation, EndAnimation, FinishTimerInput, etc...

The GenericAction is a very powerful tool for the template, and there are several generic actions included in the project, open the Basic Locomotion demo scene to see those examples in action.



When adding a new Animation to your Action StateMachine, make sure to use the vAnimatorTag “CustomAction” to it, so that the Controller knows you’re doing an Action.



ANIMATOR TAG

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

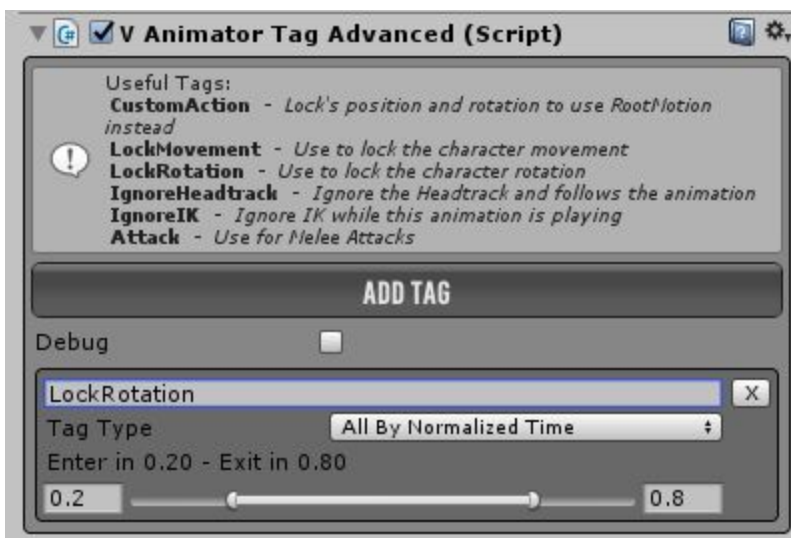
We have a few useful built in tags such as the image below in the infobox, but you can also create your own tag and verify in code, for example:

First access the `vThirdPersonController` via script so you can call the method

```
if(tpController.IsAnimatorTag("MyCustomTag")
{
    //do stuff
}
```



We also have a `AnimatorTagAdvanced` in case you want to check a tag but only during a certain period of the animation, every animation goes from 0 to 1 and you can filter the tag to only run your method during this time.



ANIMATOR EVENT MESSAGE/RECEIVER

Send a message from a AnimationState to any GameObject:

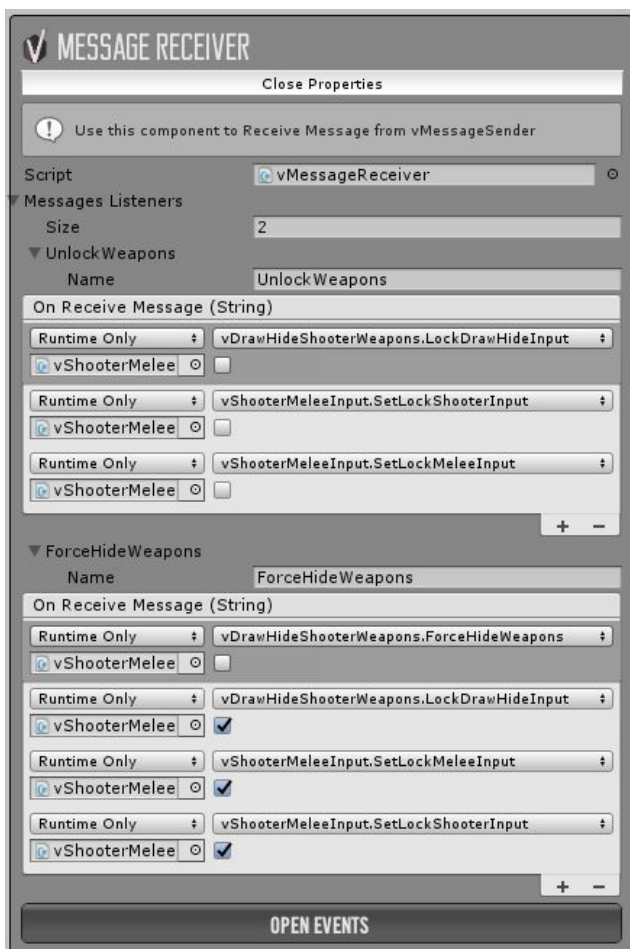
<https://www.youtube.com/watch?v=uZn53kKsl0I>

MESSAGE SENDER/RECEIVER

With this component you can create Custom Messages and call several public methods using Events and trigger them at any time by using a trigger for example.

In this example we have 2 Message Listeners:

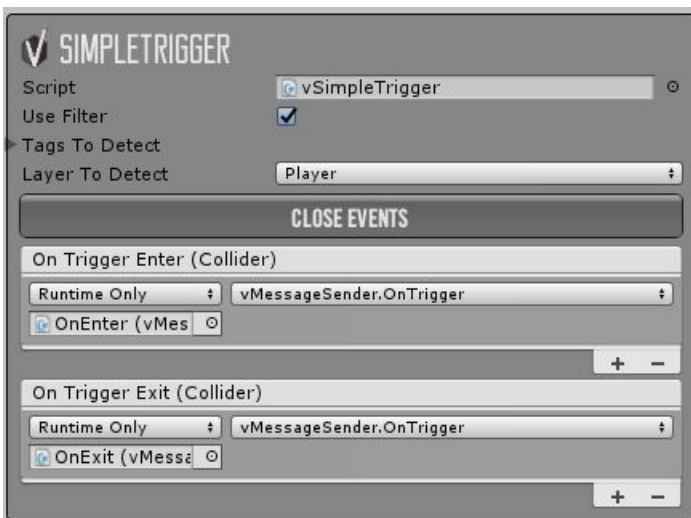
- ForceHideWeapons will lock the input of drawing weapons and call the methods to lock the Shooter and Melee input, it's useful for example when entering a npc area were you cannot attack anyone.
- UnlockWeapons were we call the method to unlock the hide weapons input and unlock the shooter and melee input.



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

Now to trigger those combined events of the MessageReceiver, we will use the vMessageSender to send the message and the vSimpleTrigger to detect collision with the Player.

We created 2 empty gameObject called OnEnter and OnExit, add the vMessageSender and create the Message ForceHideWeapons and UnlockWeapons, we also checked the option SendByTrigger since we're going to use the vSimpleTrigger to detect the player.



Now in the vSimpleTrigger we can use the option Use Filter to only detect the Tag and Layer "Player", create a box collider or mesh collider with the option IsTrigger checked and the layer "Triggers" that matches the size you want and call the method "vMessageSender.OnTrigger" for each Event OnTriggerEnter and OnTriggerExit.

BODY SNAPPING ATTACHMENTS

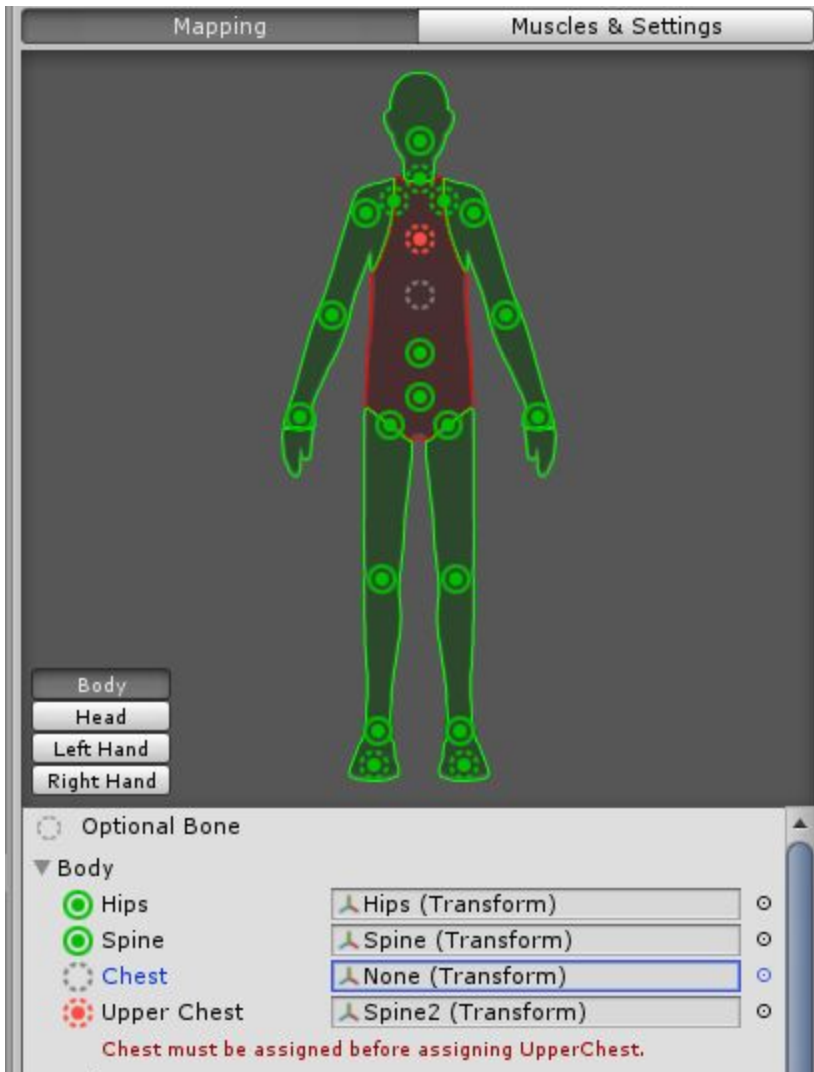
We created this feature to make it easier to transfer attachments from one controller to another.

This means that you can create a Prefab of a Character Attachments and quickly add to another character, without the need of adding attachments one by one on each bone.

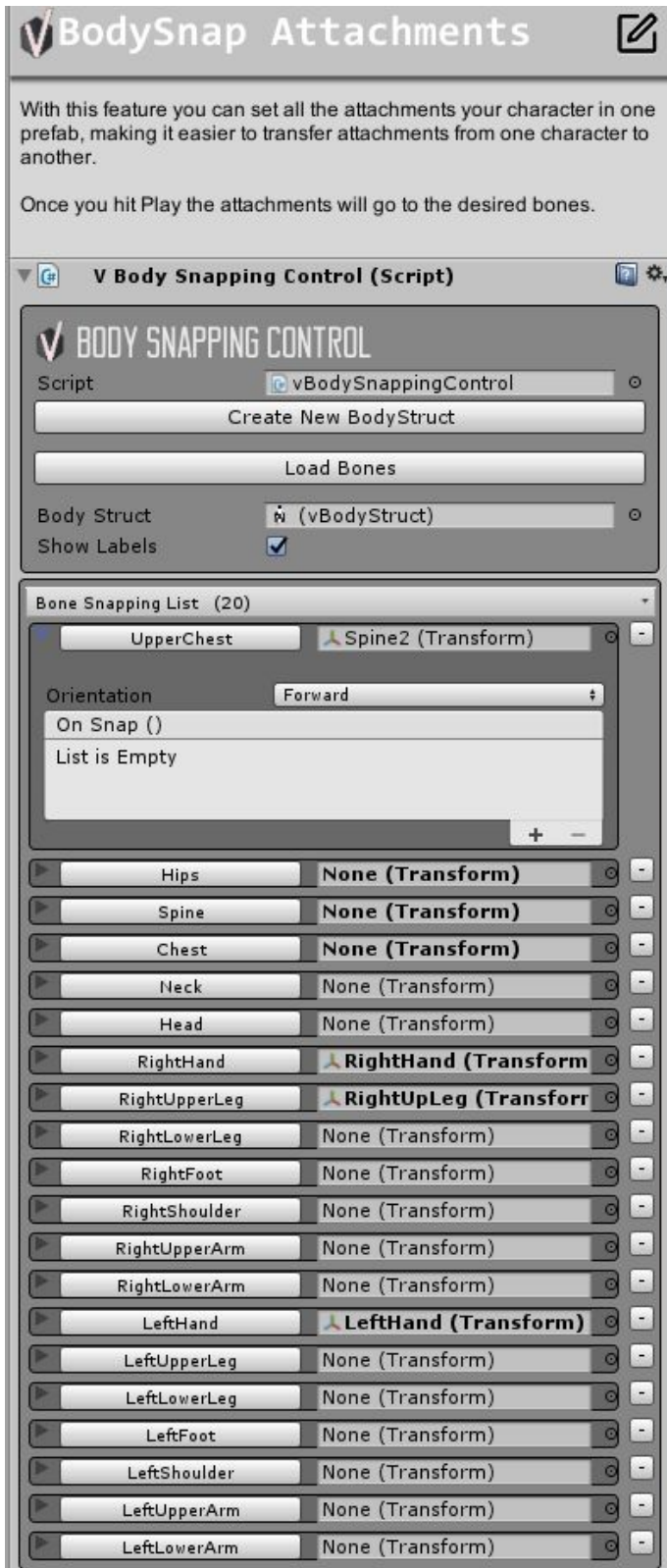
First, create an Empty GameObject inside your character, add the “**vBodySnappingControl**” and hit the “Create New BodyStruct”.



If your Avatar is already set up as Humanoid and all the bones are correctly mapped, it will all be automatically assigned for you, in some cases Unity doesn't recognize a Spine or Chest, so you need to fix by going to your Avatar and assigning the correct Bone, example:



Now going back to our Character BodySnap Control, you can add all your character attachments such as particles that activated on a specific bone, itemManager Handles, anything that you may use and assigned to a specific bone, once you hit Play that GameObject will be attached to the bone you assigned.



SNAP TO BODY

**Use this component with a vBodySnappingControl*

Attach this component to any object to make it snap to a specific bone when you hit Start.



For example, instead of manually drag and drop objects like Armour, Helmet, Capes, etc... to each bone of your character, you can create a Prefab called "Attachments" and add 'SnapToBody' on each object and assign where this object must be attached.

This makes the workflow much easier when replacing the character model for example.

