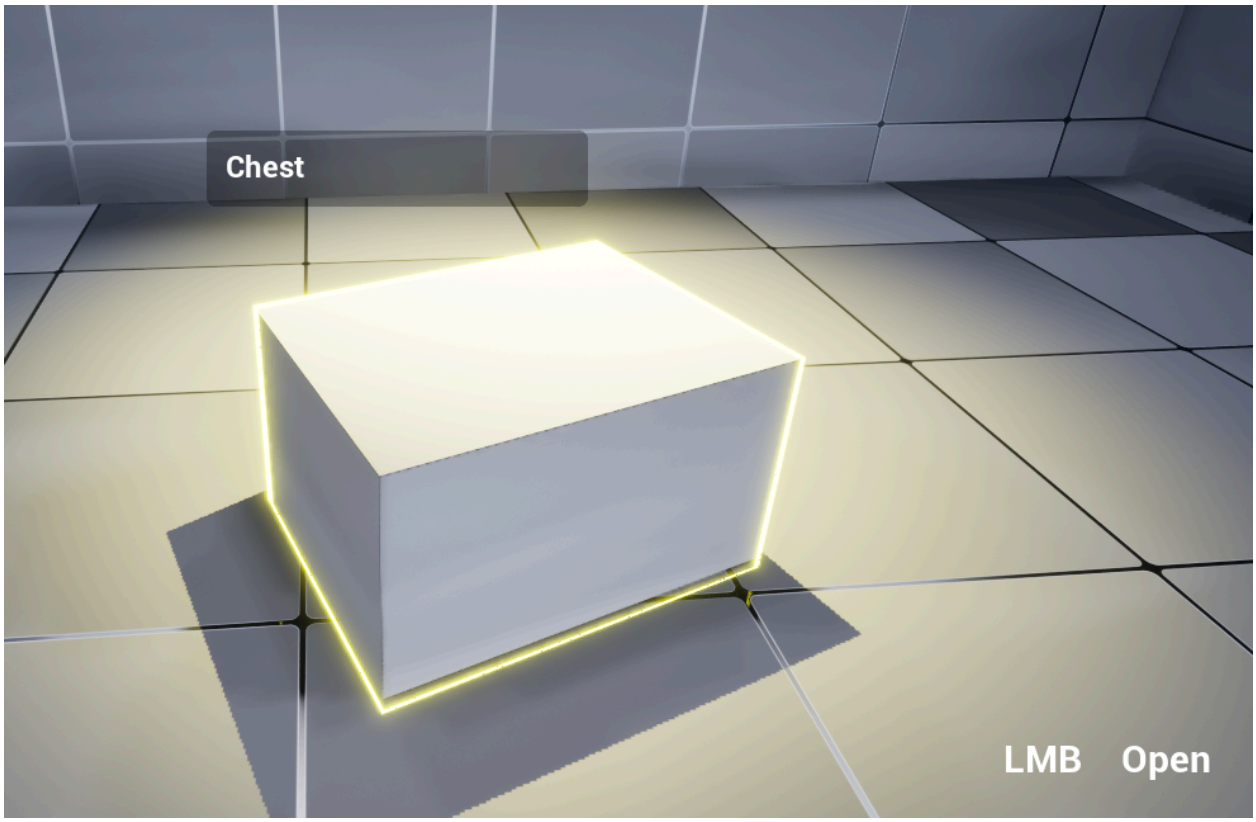
Gameplay Interaction Plugin

This plugin provides a flexible solution to handle gameplay interaction, based on the Gameplay Ability System (or GAS). It allows you to easily give multiple interactions through components and set them up quickly with Gameplay Abilities, making them reusable. It also provides a base for Interaction UI and Outlines.

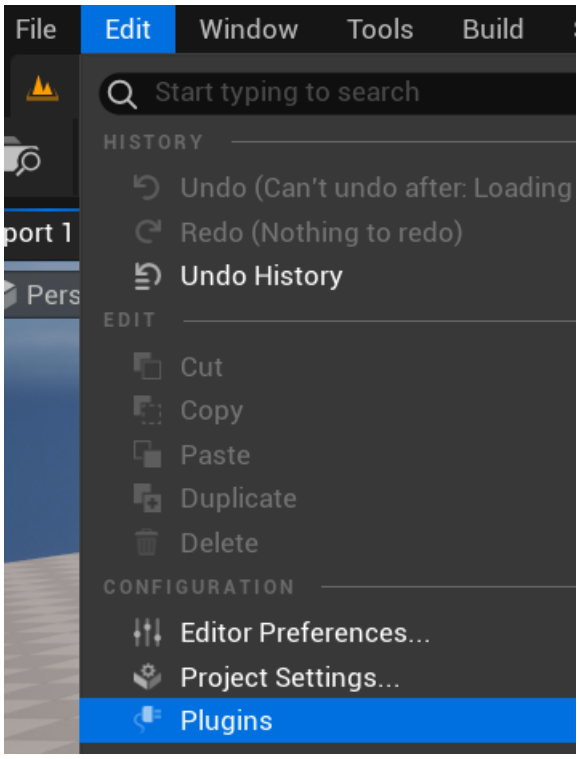
Disclaimer: This plugin is using the Gameplay Ability System and assumes that you have a basic understanding about Gameplay Tags and Gameplay Abilities.

Help: louisgirard.games@gmail.com



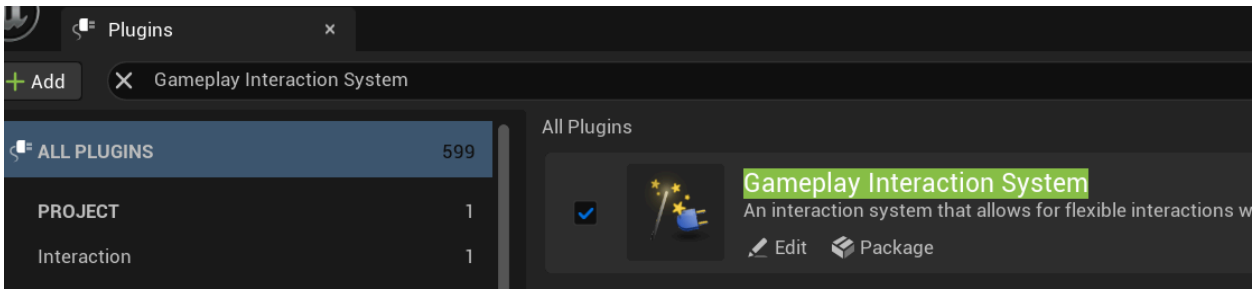
Activating the Plugin

Go to plugins



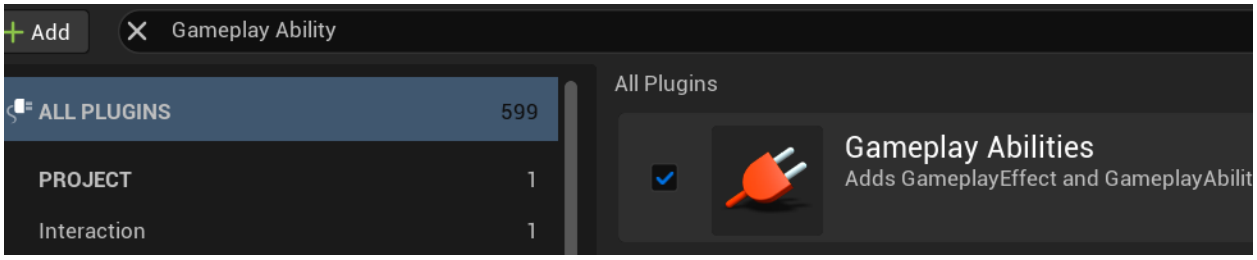
Enable the Gameplay Interaction System plugin

Search for Gameplay Interaction System and tick the box to enable it.



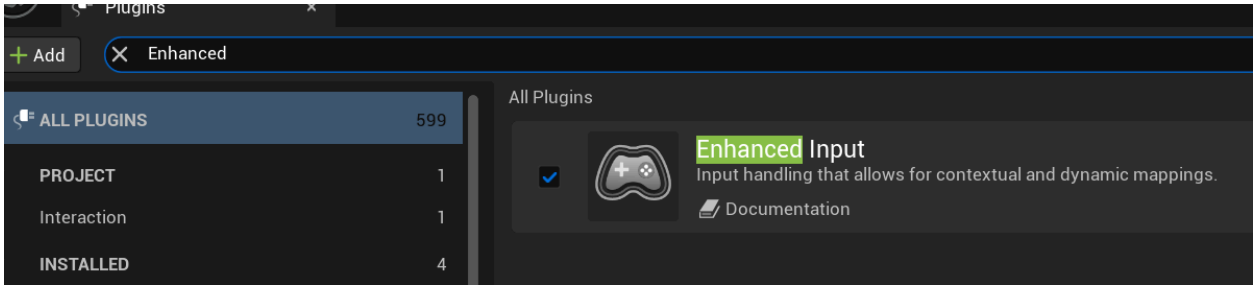
Enable the Gameplay Abilities plugin

As this plugin relies on the Gameplay Abilities plugin you will want to enable it. Search for Gameplay Abilities and tick the box to enable it.



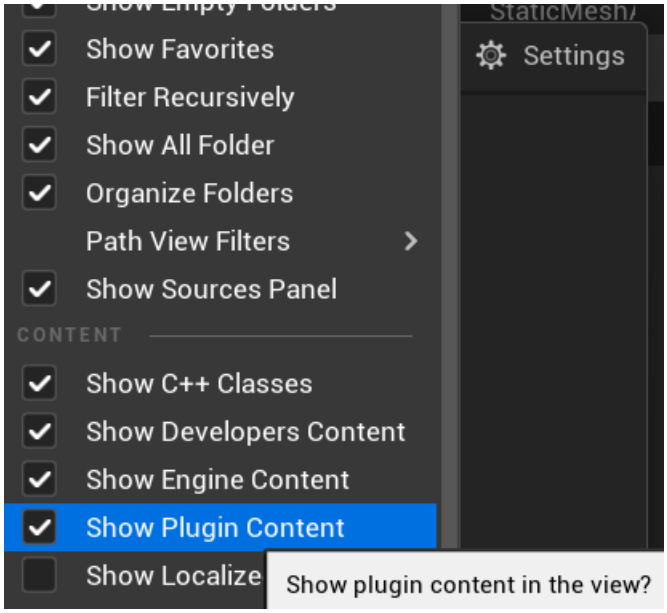
Enabled the Enhanced Input plugin

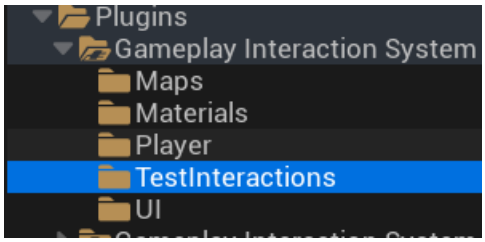
As this plugin uses Enhanced Input you will want to enable it. Search for Enhanced Input and tick the box to enable it.



Sample Content

Once the plugin is enabled you can enable the Plugin Content by going in the Content Browser and tick “Show Plugin Content”.





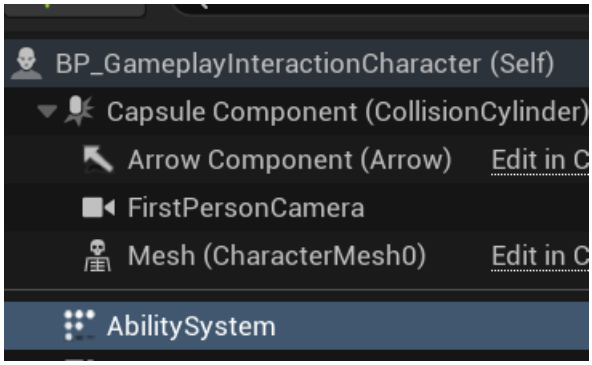
You can launch the **L_GameplayInteractionSystem map** and start exploring the interactions, or follow the documentation on how to set up your character and an interaction.

Character Setup

We will use the BP_GameplayInteractionCharacter as an example for you to set up yours. It uses Camera Inputs and Movement Inputs that can be found in the Unreal First Person Template so we won't look into that.

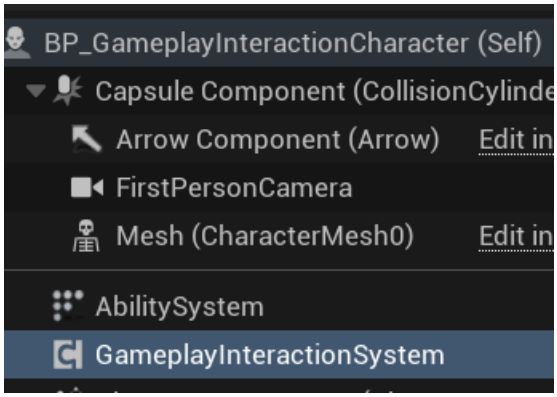
Ability System Component

First you will need to add the AbilitySystem component so that your character can be given the interaction abilities.



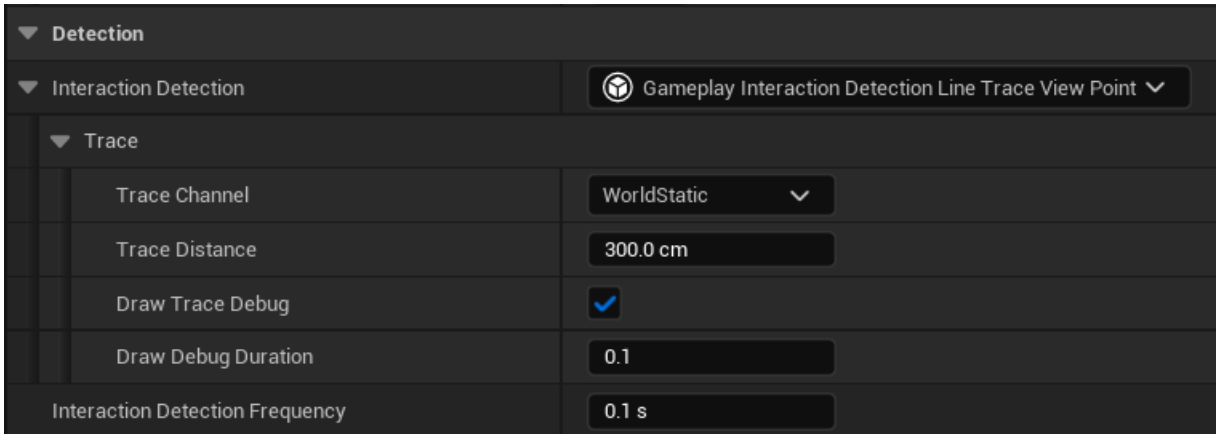
Gameplay Interaction System Component

Let's add the Gameplay Interaction System Component and look into its options.



Interaction Detection

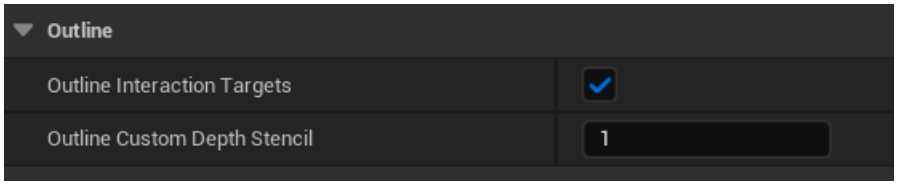
By default, the detection of interaction is done with a Line Trace from the View Point of the Character (usually the camera). You can configure the Trace Channel to detect Interactions and the Trace Distance.



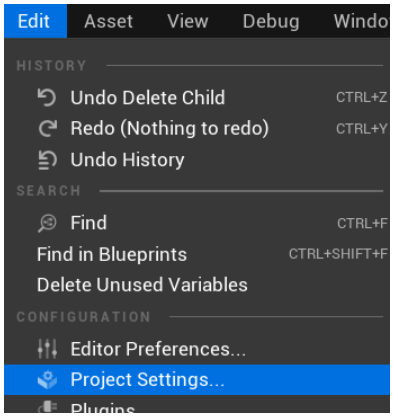
If you want to detect Interactions in a different way you can also create custom detections.

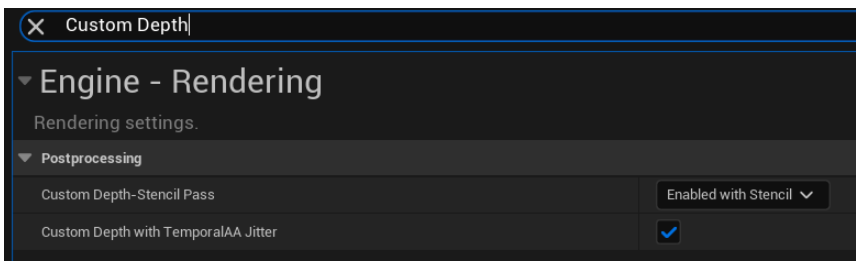
Outlines

If you wish to have an outline on the Interactions you can enable it with **Outline Interaction Target.** You can leave the **Custom Depth Stencil** value as is, it is exposed only for you to modify it if you already use it elsewhere in your project.

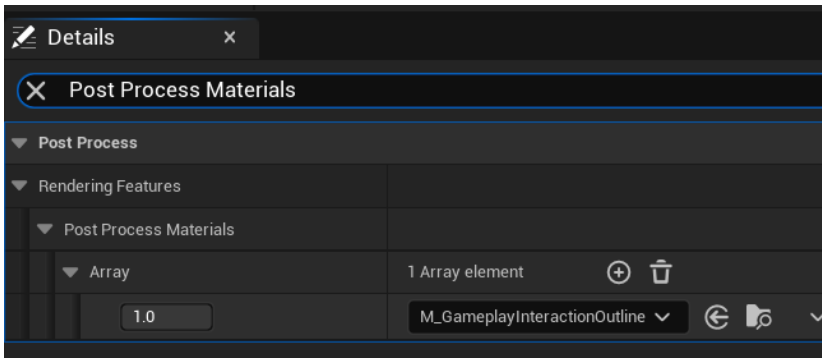


You will also need to go to the project settings, look for Custom Depth-Stencil Pass and set it to “Enabled with Stencil”.

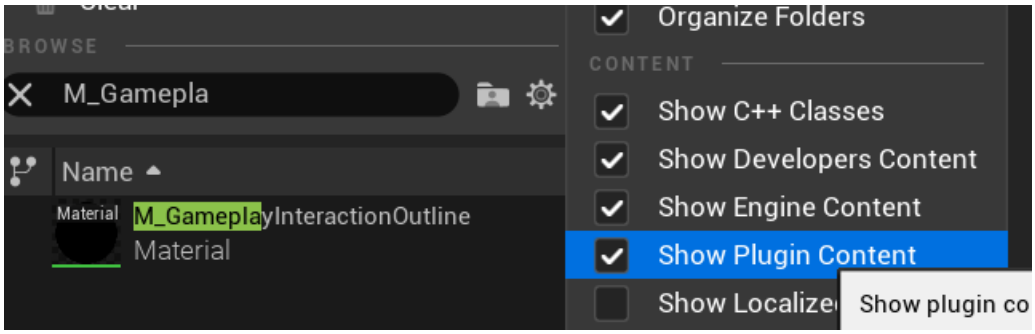




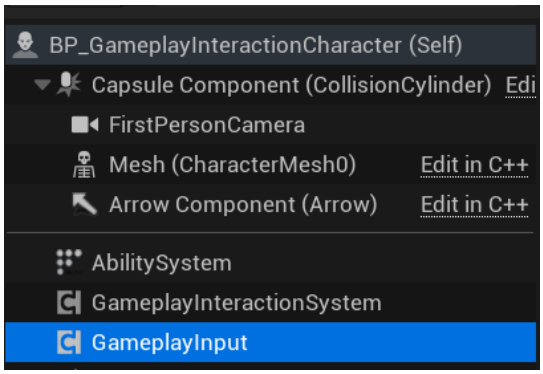
Lastly you will need to go to the camera of your character, look for “Post Process Materials” and add the M_GameplayInteractionOutlines as an entry.



If you don’t see it at first don’t forget to enable the plugin content.

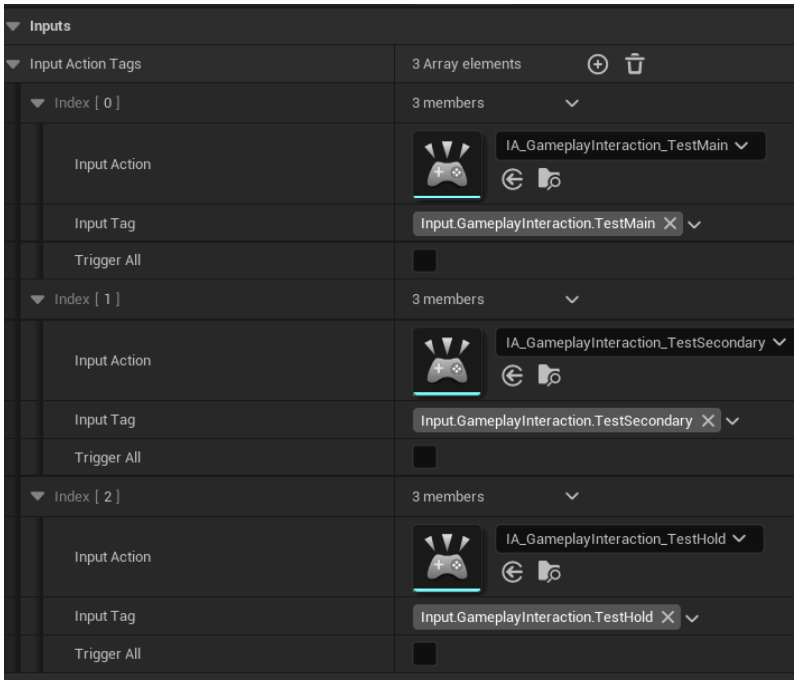


Gameplay Input Component

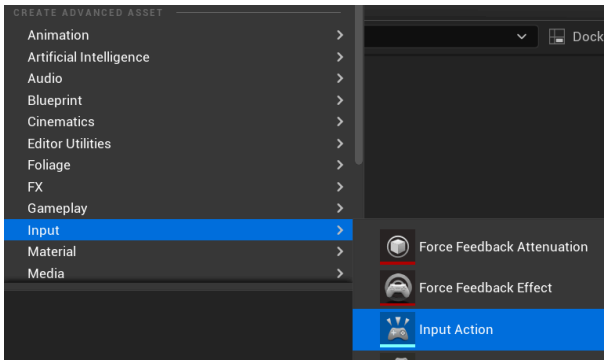


To trigger the interactions by inputs we will be using the Gameplay Input Component that comes with this plugin. It’s a component that allows to bind Input Action with a Gameplay Tag to trigger Gameplay Abilities that have the input tag. *It’s a flexible component that you could use for other systems than the Interactions.*

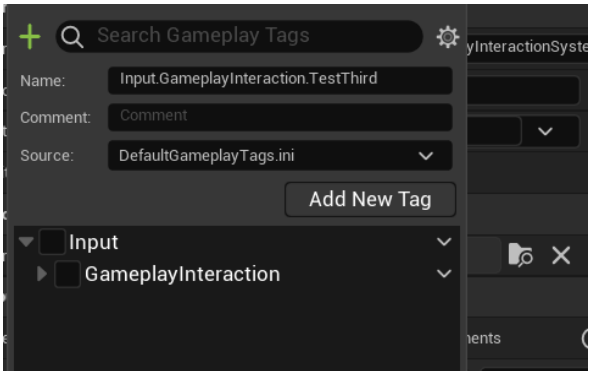
Let’s look at the list of Inputs available.



The first property is the **Input Action** that will be used to trigger an interaction. If you wish to create new Input Actions for your project you can by doing the following.



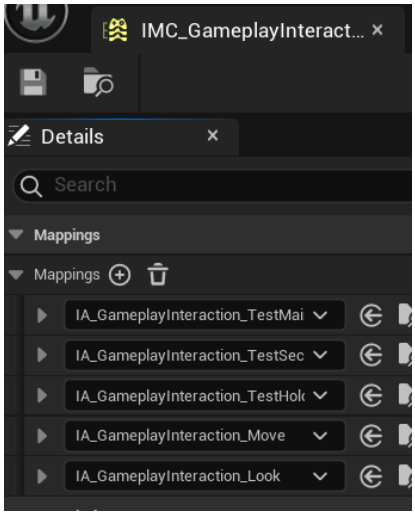
Then the **Input Tag** is a Gameplay Tag that will be used to determine which Abilities to trigger on Input. You can create a new entry and configure its Gameplay Tag by clicking on “None”, then on the green “+”. Then you can name it and click “Add New Tag”.



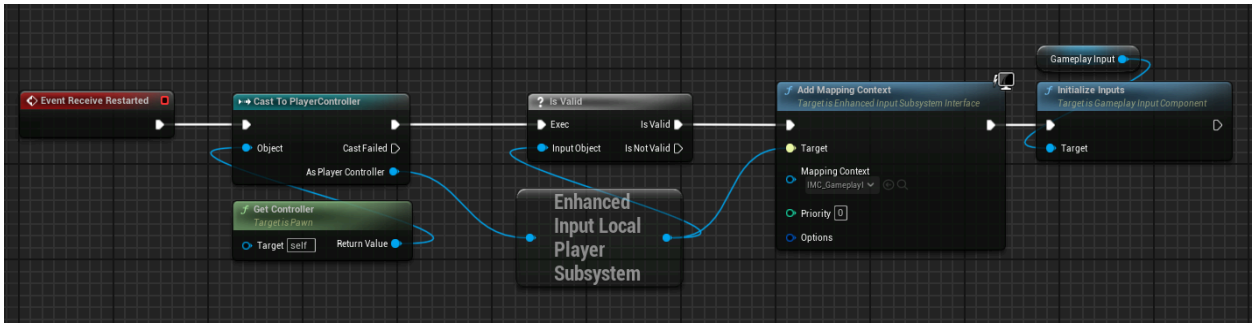
Trigger All gives you the control if you want to trigger all the abilities with the Input Tag on Input, or only

the first activatable one.

Once all your inputs are added you should create an Input Mapping Context (or use an existing one) and add them in there. This will allow you to map your interaction inputs to real keyboard or mouse inputs.



Finally in your Character, on Receive Restarted you can add your Mapping Context and then call **Initialize Inputs** on the **Gameplay Input Component**.

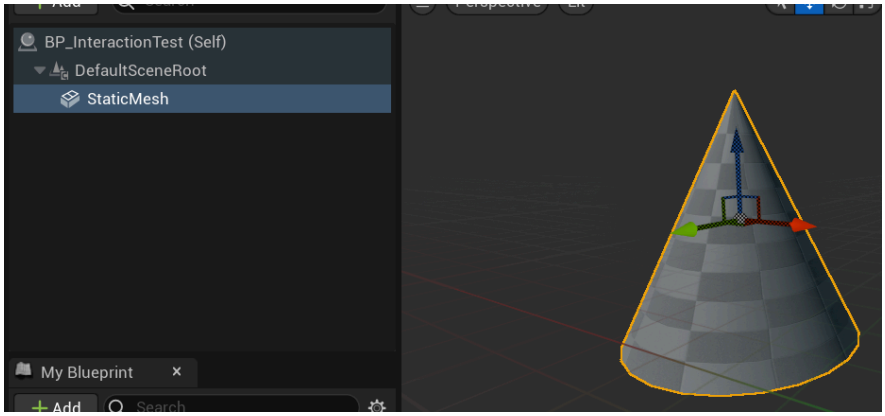


Interaction Objects

Let's create a basic interaction that will display a debug text, for you to see how to configure a simple one, then feel free to explore the interactions in the sample content.

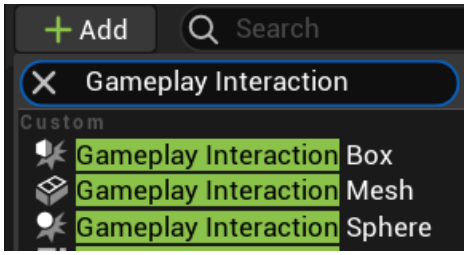
Note - If you are using the Open System: You might see the examples are present in the OpenSystem folder. This is because the examples were reflecting a simplistic version of the OpenSystem which should not be used if you are using the OpenSystem.

First let's create a simple actor with a Static Mesh Component.

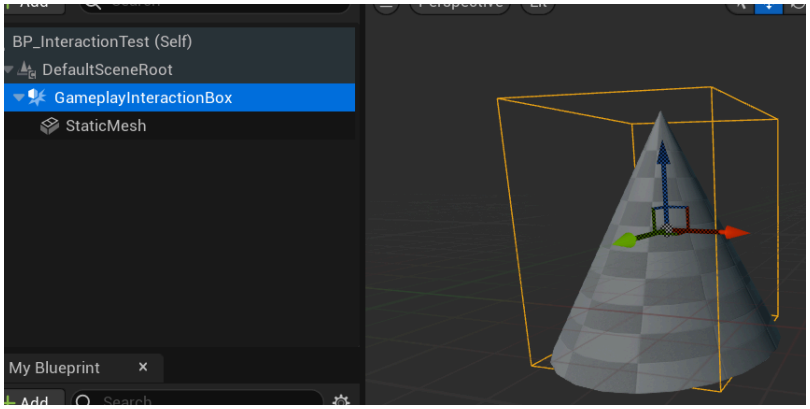


Gameplay Interaction Components

You will be able to add any of the three Gameplay Interaction Components available.



They all offer the same functionality, but differ from the Shape. You can either choose a Box, a Sphere or even a Mesh itself. They will be used to detect an interaction from the Character's Line Trace. Let's choose the Box and parent it to our mesh.

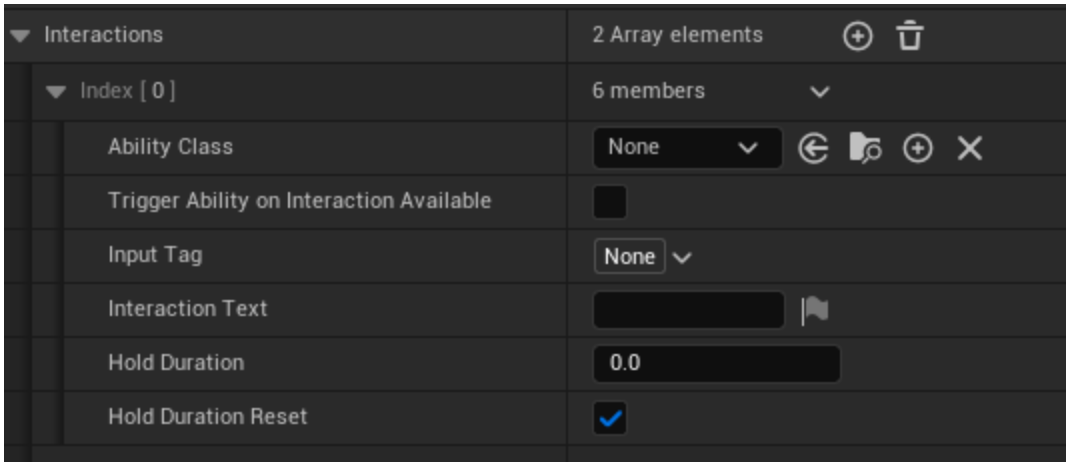


Collisions

If you are using the Line Trace for Detection you may want to update the collisions on your component to include the Trace Channel.

Interactions

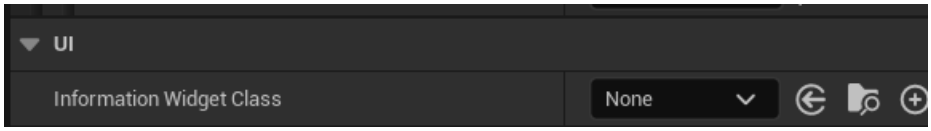
In those components you will be able to define the different interactions.



First you can set up the **Input Tag**, that should be matching one of the Tags you set up previously in the Gameplay Input Component of your character.
To define the behavior of the Interaction itself, this will be done with an **Ability Class**, that we will see in the Gameplay Abilities section.
You can tick **Trigger Ability on Interaction Available** to activate the ability as soon as the interaction is possible. This allows for custom behaviour while you have the interaction but have not pressed the input yet.
You can also set a **Text** representing the Interaction (like Open Door), this will be used later on for the UI.
Finally if the interaction is a **Hold Interaction** you can set its duration and if the duration should reset when releasing the hold or not. We'll see in the Hold Interaction section how to set up a Hold Interaction ability.

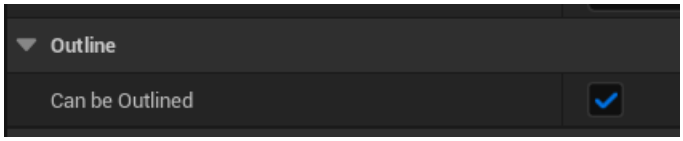
Information Widget

If you want you can define a widget to display extra information regarding the Interactions of this component. We will see later on how this can be displayed in the UI.



Outline

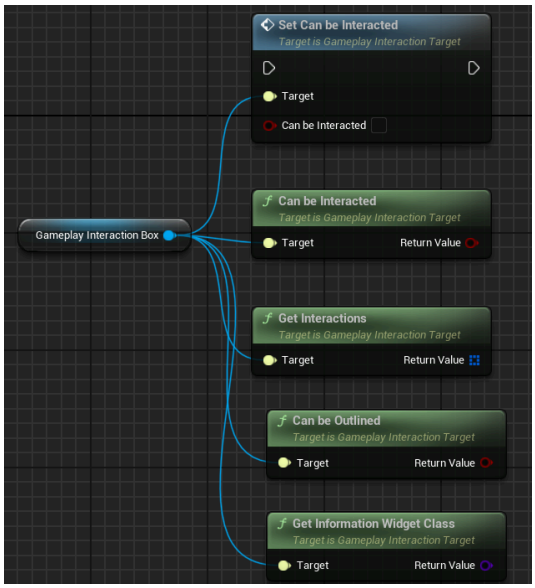
If you are using outlines, you decide if this component's Interactions should be outlined.



Take into consideration that if you are using a Box or Sphere component, all the Mesh that you want to be outlined should be children of the Gameplay Interaction Box/Sphere Component.

Interaction Target Interface

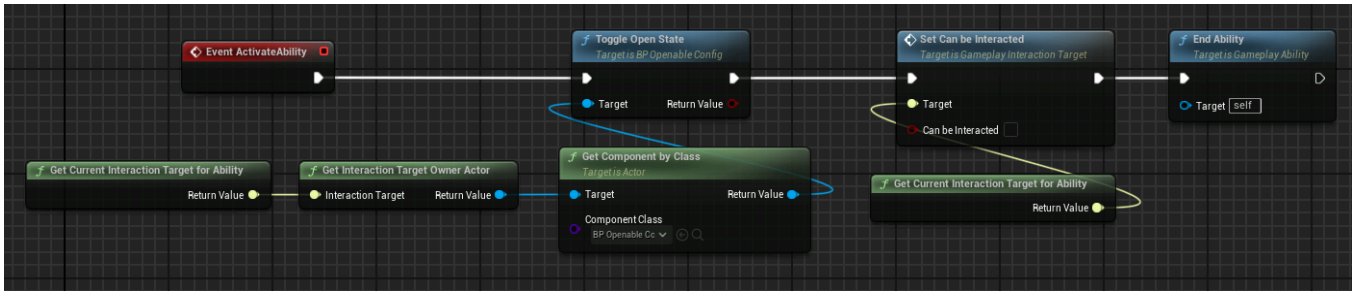
These components are implementing an interface called **Interaction Target** allowing them to reuse the same behavior and functionalities. You can easily change if the component **Can Be Interacted** or not, get its current Interactions or its Information Widget Class.



Gameplay Abilities

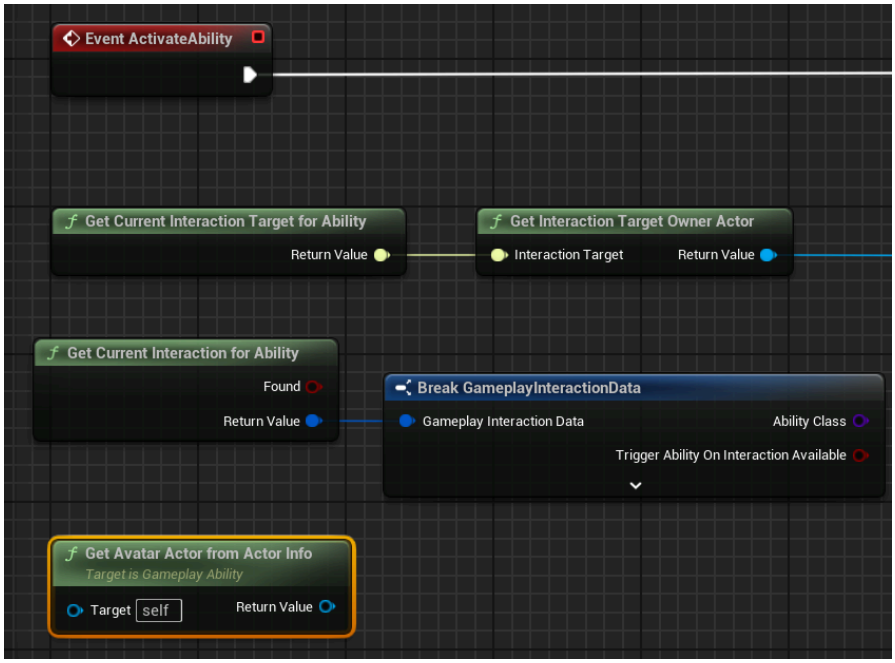
This part assumes that you have a basic knowledge of how Gameplay Abilities work.
Each interaction's behavior will be set up with a Gameplay Ability, allowing them to be very reusable (like an Ability "Open" that could be used for a Door or a Chest).

Activate



Let's take a look at the ability to Open. We take the Interaction Target (usually Interaction Component), using **GetCurrentInteractionTargetForAbility**, then get its owner with **GetInteractionTargetOwnerActor**. Finally we change its open state and set can be interacted to false.
The most important part here is to not forget the **End Ability** for the Interaction to be considered over.

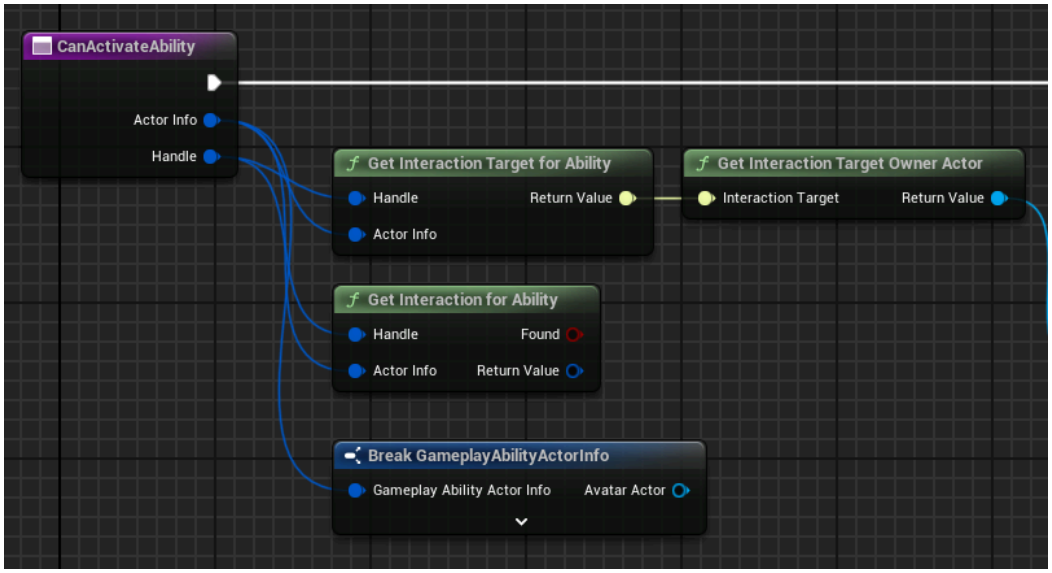
Useful information you may want to use in an interaction ability:



With **GetAvatarActorFromActorInfo** you can get the one that triggered the Interaction, most of the time the player.
GetCurrentInteractionTargetForAbility gives you the Interaction Target owning the interaction, most of the time the Interaction Box/Sphere/Mesh Component.
GetCurrentInteractionForAbility gives you the actual Interaction Data associated with the interaction.

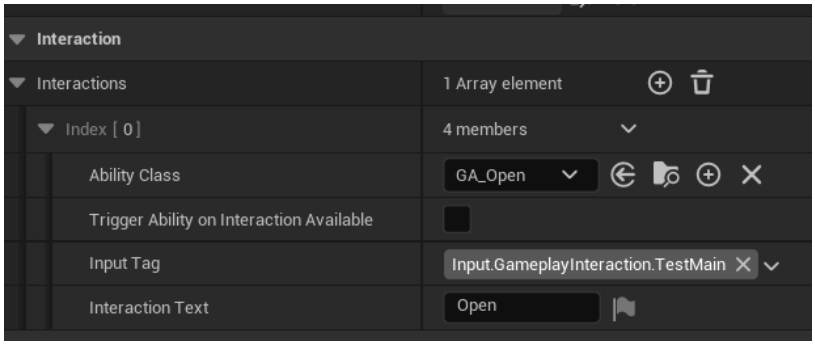
Can Activate

If you wish to give an interaction but have it activatable only on certain conditions you can modify the CanActivateAbility. For example you could want to have the Open Interaction only activatable if the thing to open is not locked.
You will find similar options to the Activate, but you should be using these functions.



Use **ActorInfo** to get the **AvatarActor.**
Use **GetInteractionTargetForAbility** instead of **GetCurrentInteractionTargetForAbility.**
Use **GetInteractionForAbility** instead of **GetCurrentInteractionForAbility.**
This is needed because CanActivateAbility will be called on a non instanced Ability, so we don't have access to the other functions.

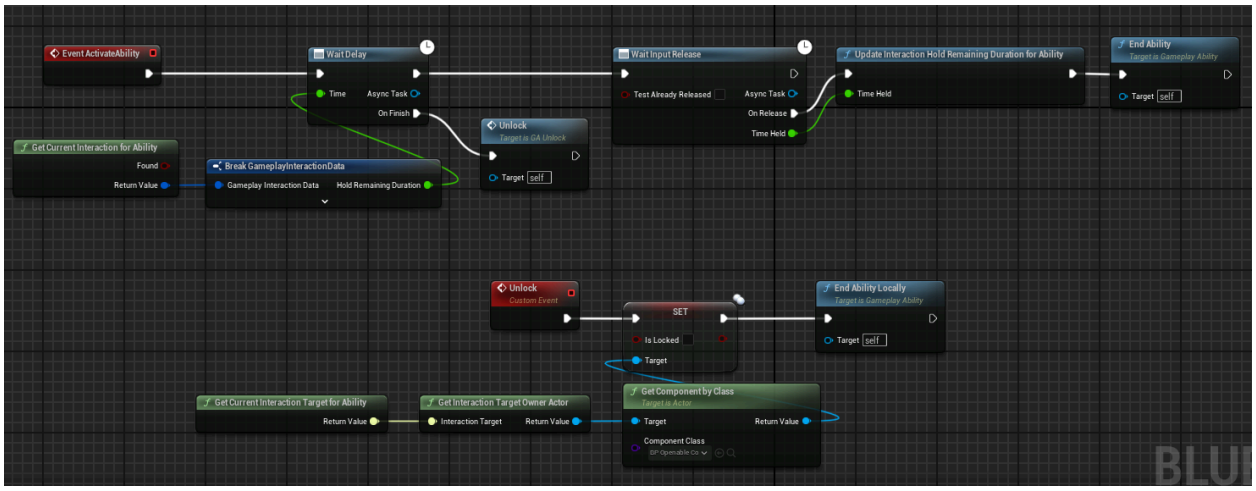
Then you can set the gameplay ability in any Gameplay Interaction Component for you to use.



If you want you can add multiple interactions with the same Input Tag, and it will always detect and trigger the first activatable one.
You can also add multiple interactions with the same Ability Class, and it will always detect and trigger the first activatable one.

Hold Interaction

If you want to have an interaction that requires holding the input, you can set up the ability for it easily. Let's take a look at the Unlock ability that does that.



We first use the **Wait Delay** ability task to wait for the Hold Remaining Duration. On finish we do the interaction logic, in our case Unlock.
Then we use the **Wait Input Release** ability task to wait for a release of the input, to end the interaction.
We need to call **Update Interaction Hold Remaining Duration For Ability** to update the remaining duration of this hold interaction.
The most important part to end the interaction normally is to use **End Ability Locally.** That way the client

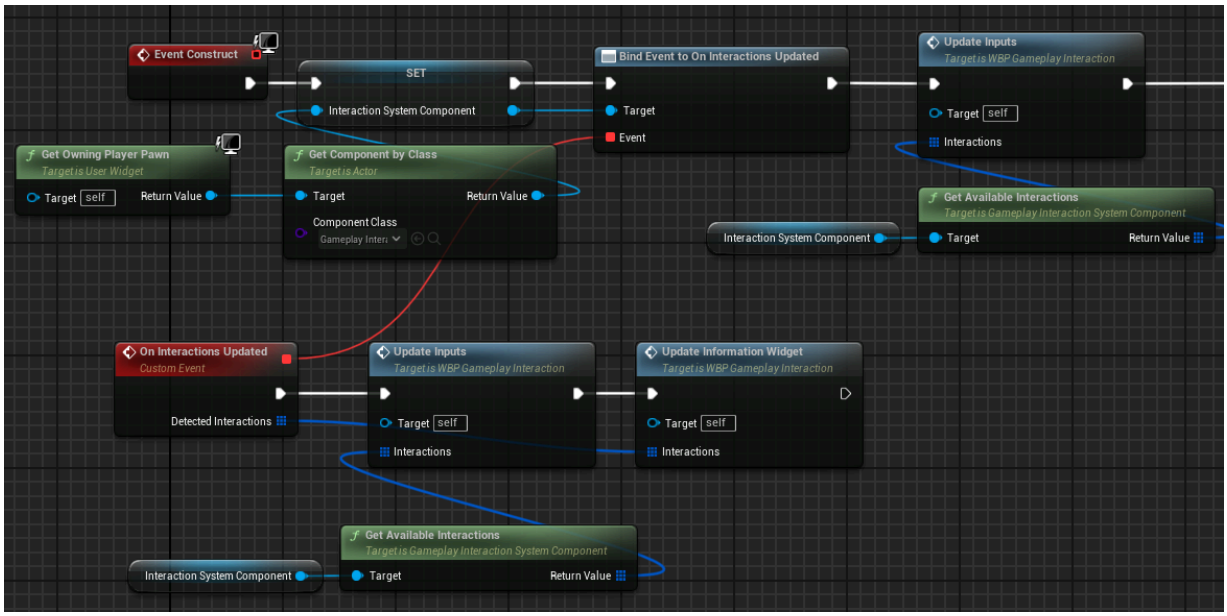
can finish the interaction and let the server finish the Hold Duration and end the ability normally on its side as well.

UI

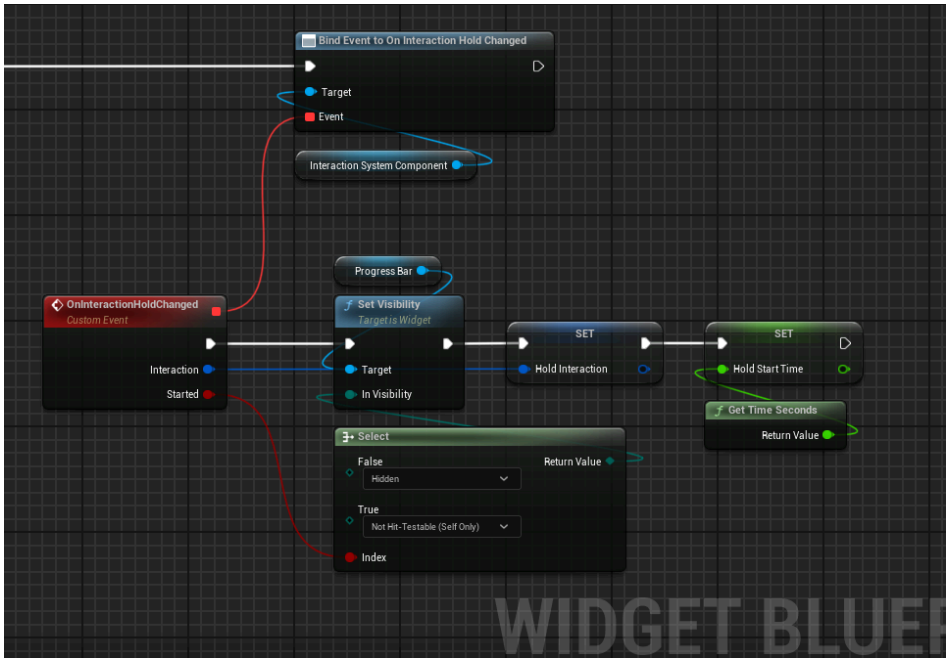
This part will show you a basic setup that you can use to display Inputs for your Interaction and extra Information.

Main Widget

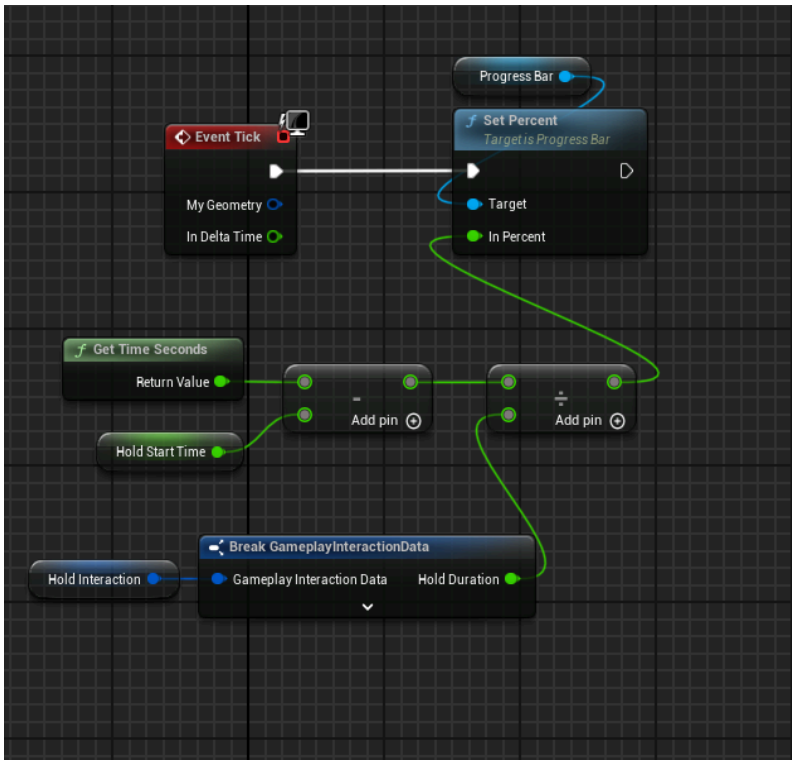
WBP_GameplayInteraction will be the main widget that will display Interaction Inputs and the Information Widget.



First we get the Gameplay Interaction Component, listen to the event **On Interactions Updated**, and initialize the Inputs with **Get Available Interactions**, to display the inputs for the interaction that can be triggered. We also initialize the Information Widget with any **Detected Interactions** to display information about interactions that can be triggered or not.



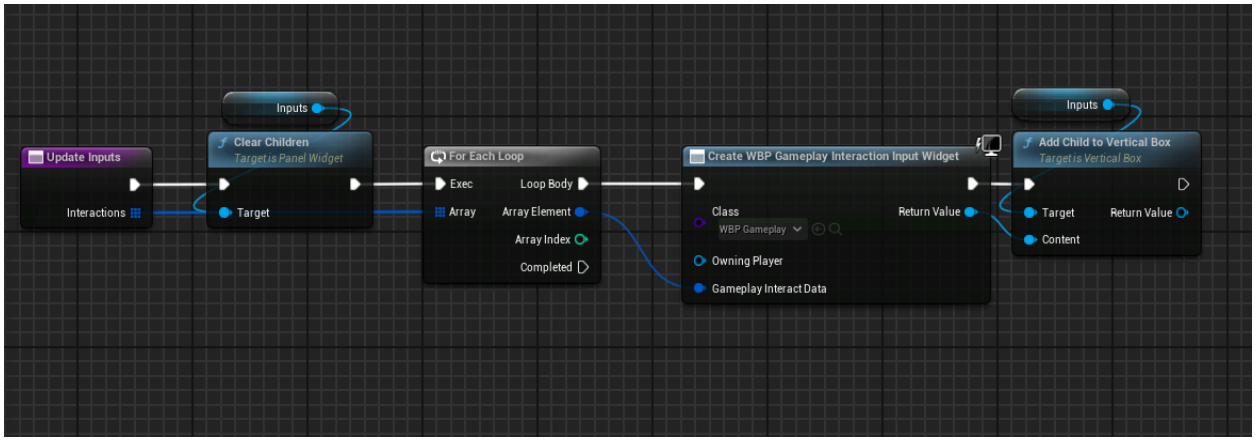
We listen to **On Interaction Hold Changed** to detect when an Hold Interaction started or stopped. If it's started we display the Progress Bar, and initialize the **Hold Interaction** and **Hold Start Time**.



On Tick we just need to update the Progress Bar to have its percentage match the hold duration.

Update Inputs

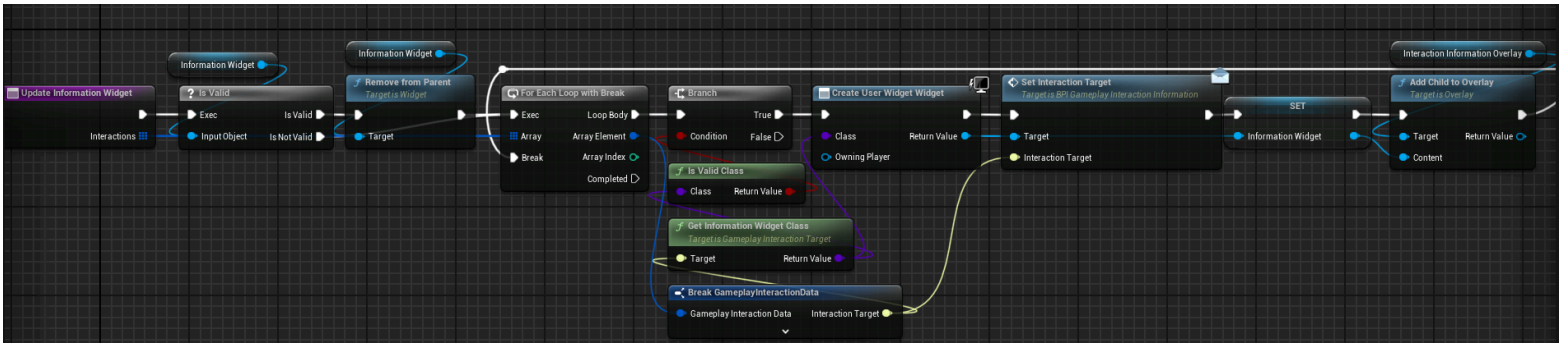
Here we just take every interaction and create a [WBP_GameplayInteractionInput] for each of them.



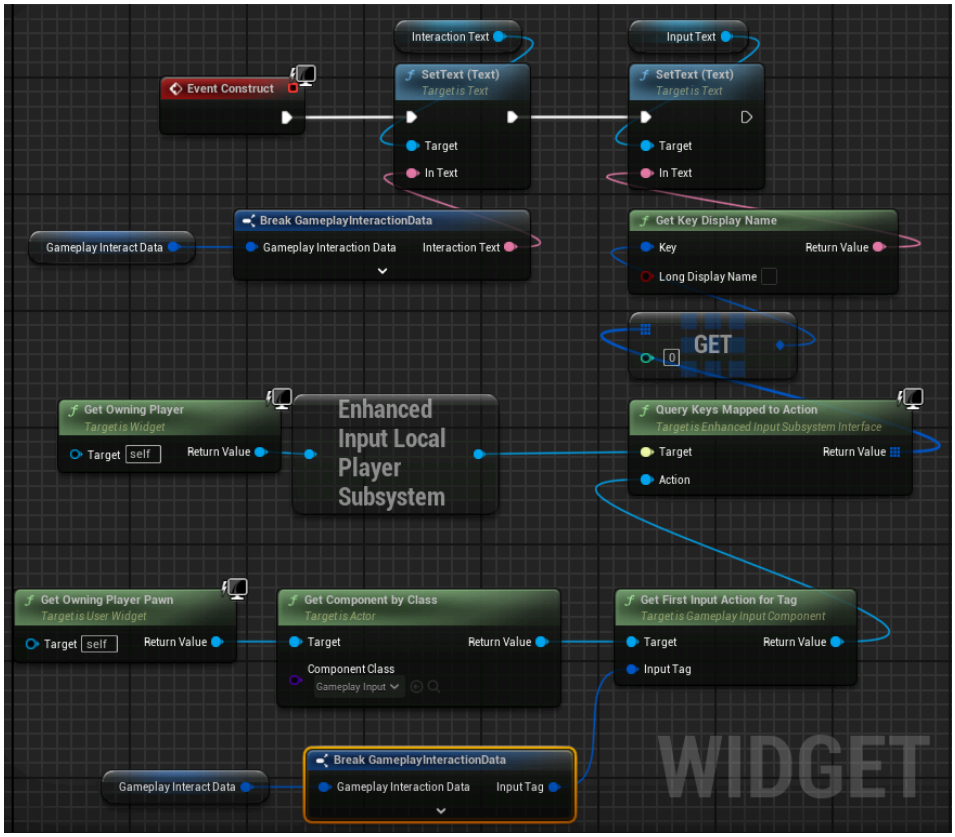
Update Information Widget

Here we look for the first interaction that has an Information Widget Class, create the associated widget and initialize it using a very basic Interface to pass it the Interaction Target. This widget will then be displayed in the middle of the screen to show the information of the current interaction.

Gameplay Interaction Input

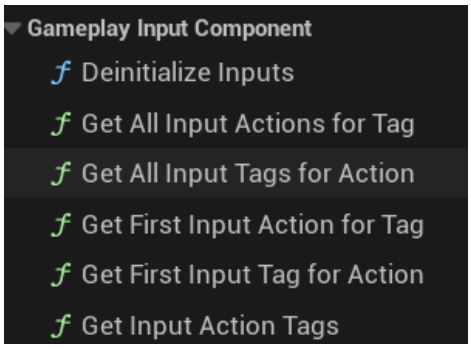


This widget contains two texts and will simply display the Input Text and Interaction Text for a given Interaction.



To get the Input Text we will look into the **Gameplay Input Component** and get the Input Action for the Interaction Input Tag. Then we query the Enhanced Input Subsystem to get the actual key binding to it.

There are many other options in the Gameplay Input Component to get Input Actions or Input Tags, feel free to use what works best for you.



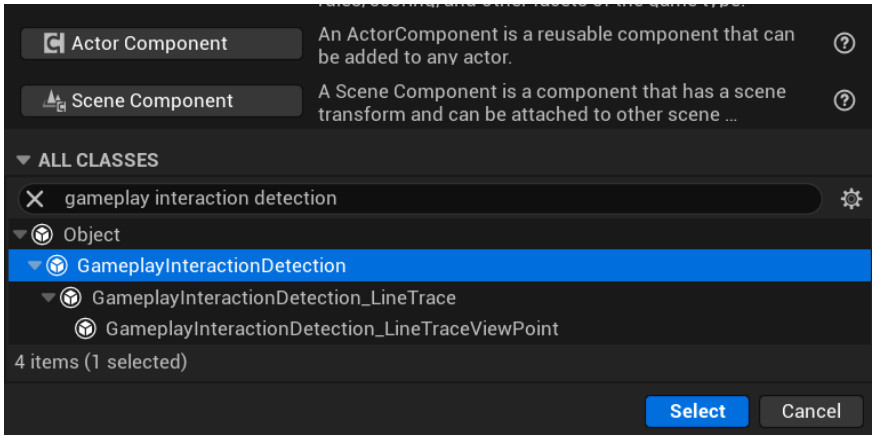
Gameplay Interaction Information

This widget could be a custom widget per Interaction, that you can define in the Gameplay Interaction Box/Mesh/Sphere Component in the **Information Widget Class**. It is mainly there to have the option to display extra information regarding the Interaction Target.

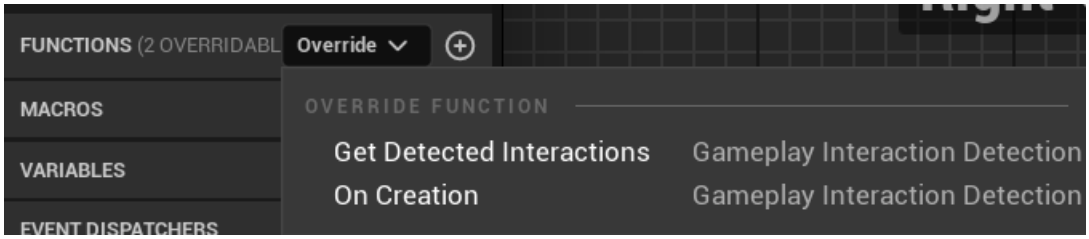
The default one will only display the name of the **Interaction Target**, but we could imagine getting specific components on the Interaction Target owner to display their information (like a locked Door that could get a Lock Component and based on its state display “Key needed to unlock the door”).

Custom Interaction Detections (Advanced)

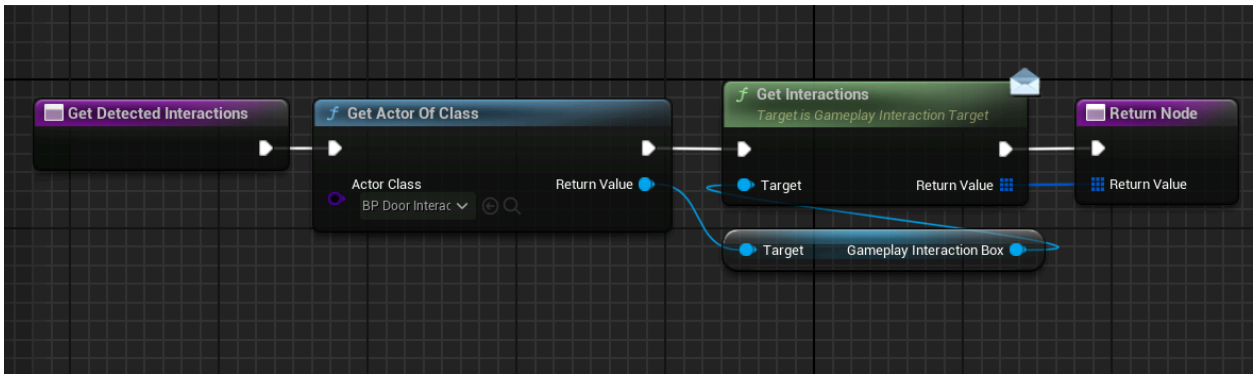
If you wish to detect interactions a different way than with a line trace you can create a new Interaction Detection class.



In this class you can derive from **On Creation** (acts as a begin play) and **Get Detected Interactions** to define the interactions that were detected.

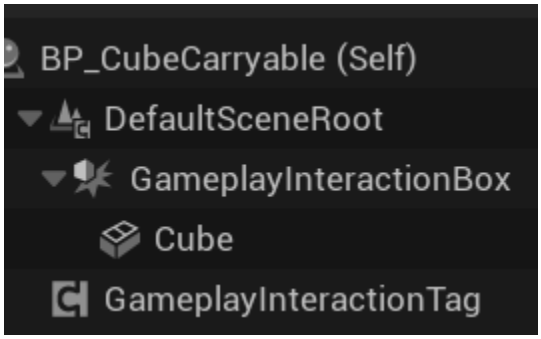


This example below is just to show how you can populate the return array, you could get the Interaction Targets on the actors you want and return their Interactions with **GetInteractions**.

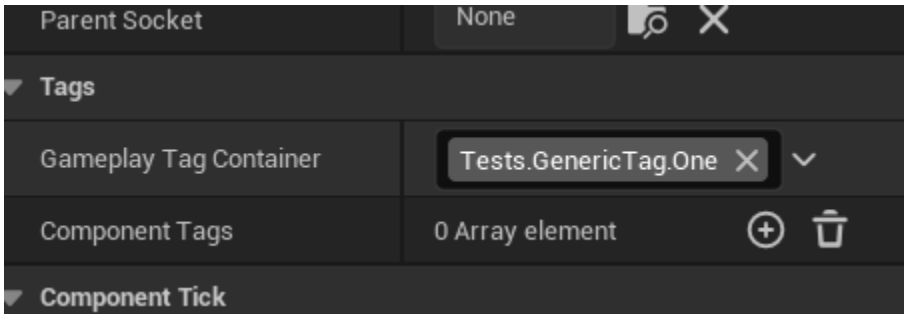


Gameplay Interaction Tag Component (bonus)

This component is an extra from the Gameplay Interaction System but can be very useful for any interactions so I included it in.



It allows an actor to be identified with Gameplay Tag. This is very similar to the default Component Tags that can be found on components but this new component uses Gameplay Tags that are more flexible.



FAQ

Why is the outline completely broken?

If you see the yellow outline doing a weird effect this is probably because you are missing the Custom Depth-Stencil Pass set “Enabled with Stencil” in the Project Settings.

With all that you should have all the information needed to start using the plugin and customize it the way you want. If you feel there is some information missing or you don’t understand some part, feel free to contact me at louisgirard.games@gmail.com.