Unreal Workshop

-04-
Blueprint Communication

By Linards Bitte

Session Expectation:

- Understand Line-Tracing nodes.
- Understand Blueprint Interfaces, How and why they are used & How to apply them.
- Understand Event Dispatchers, How and why they are used & How to apply them.

Areas we will cover:

Today we will look at:

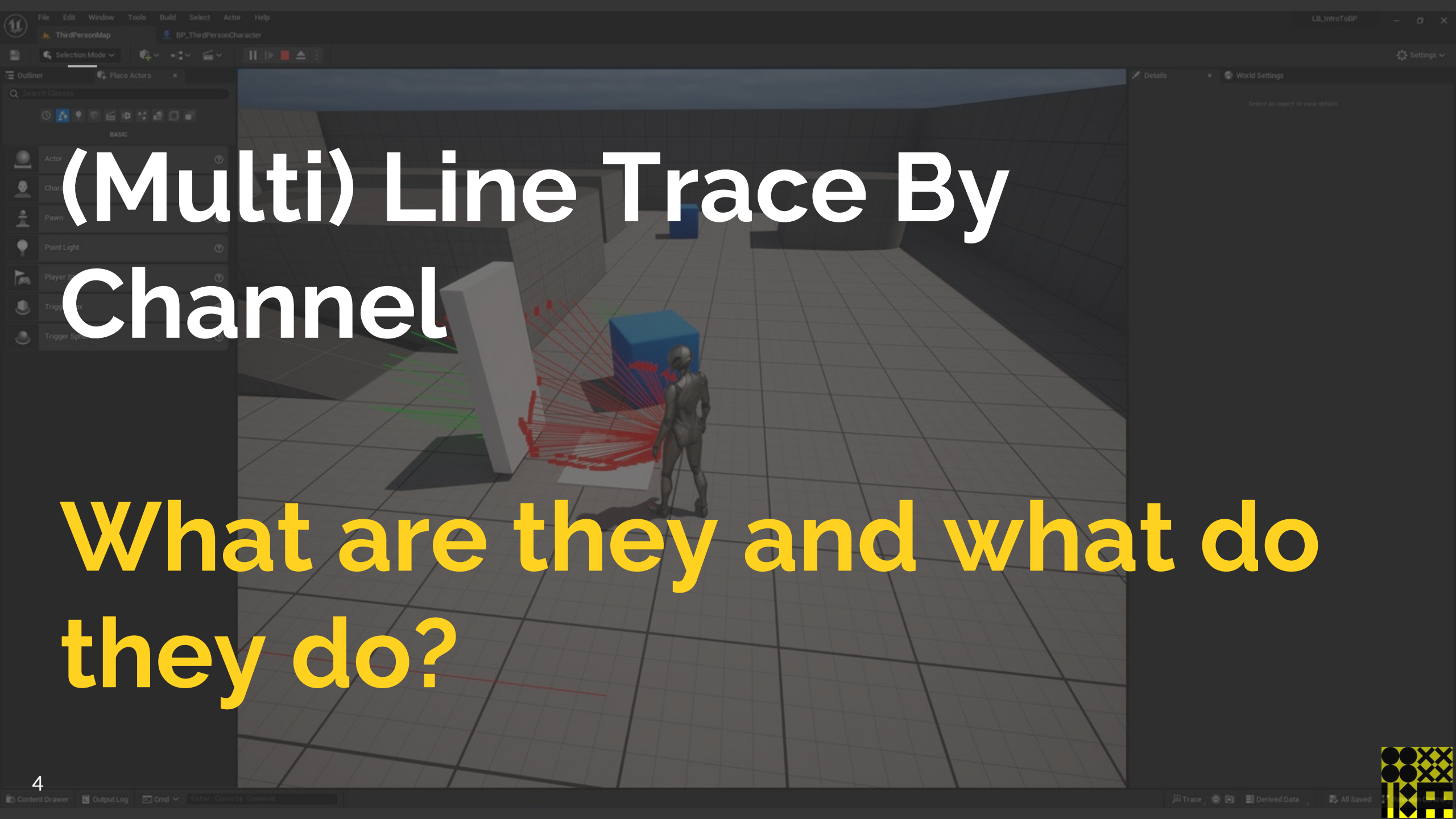
- Line Trace By Channel
- Multi-Line Trace By Channel

- Blueprint Interfaces
- Creating an Interface
- Attaching an Interface

- Event Dispatchers
- Creating an Event Dispatch
- Binding to an Event

- Reference Viewer
- Sizemaps

(Multi) Line Trace By Channel

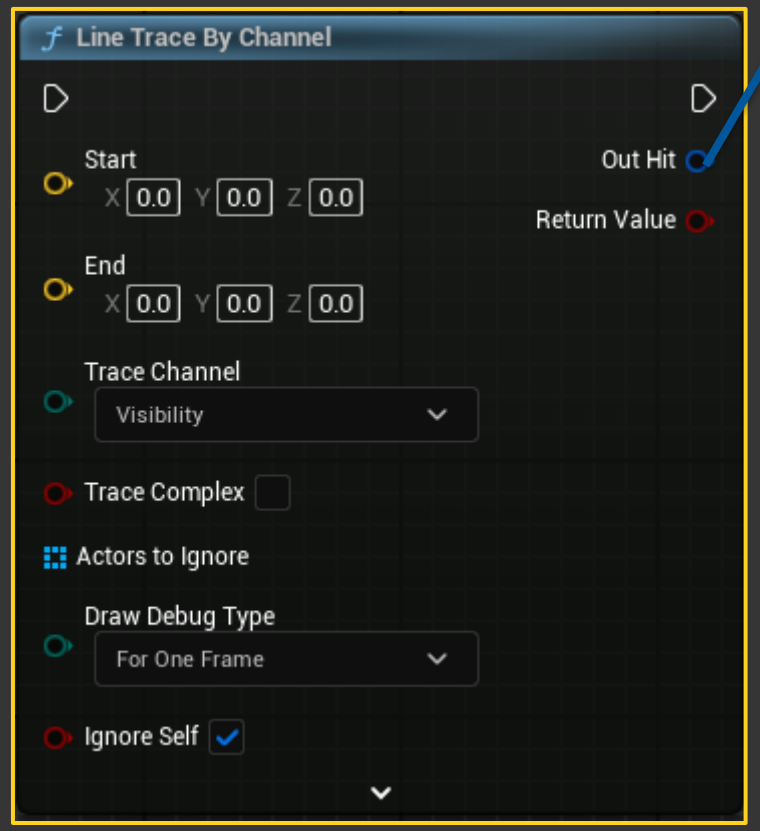


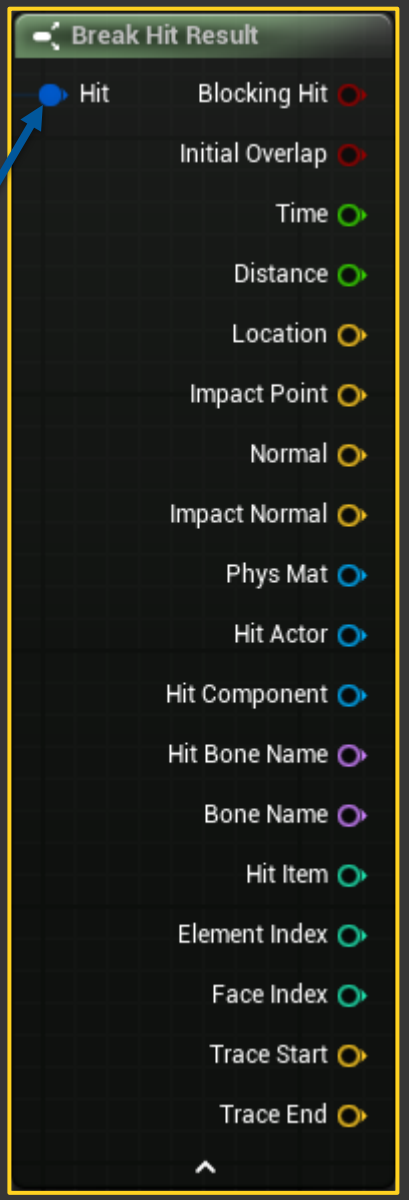
What are they and what do they do?

Line Trace – Explaining the node

- Allows us to 'trace' for data with a line.
- Using a 'World' start and end location, we can see if we 'Hit' anything.
  - The 'Hit' data will provide us with all the data we could use.

- [Traces with Raycasts in Unreal Engine | Unreal Engine 5.3 Documentation]

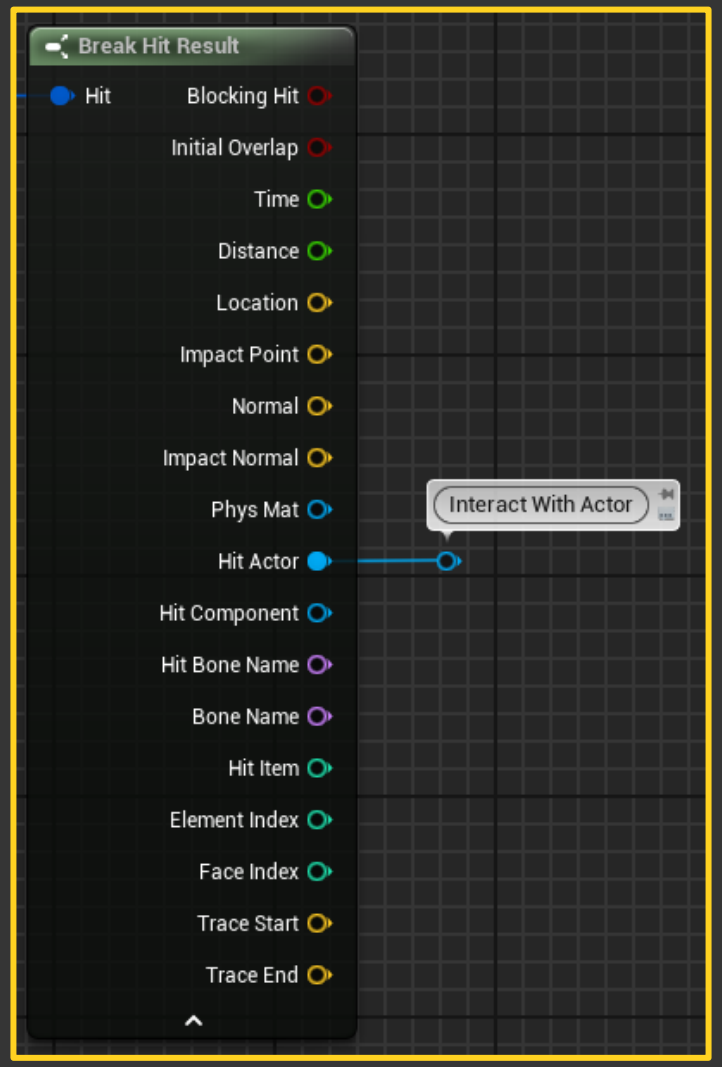




Line Trace – Hit Actor

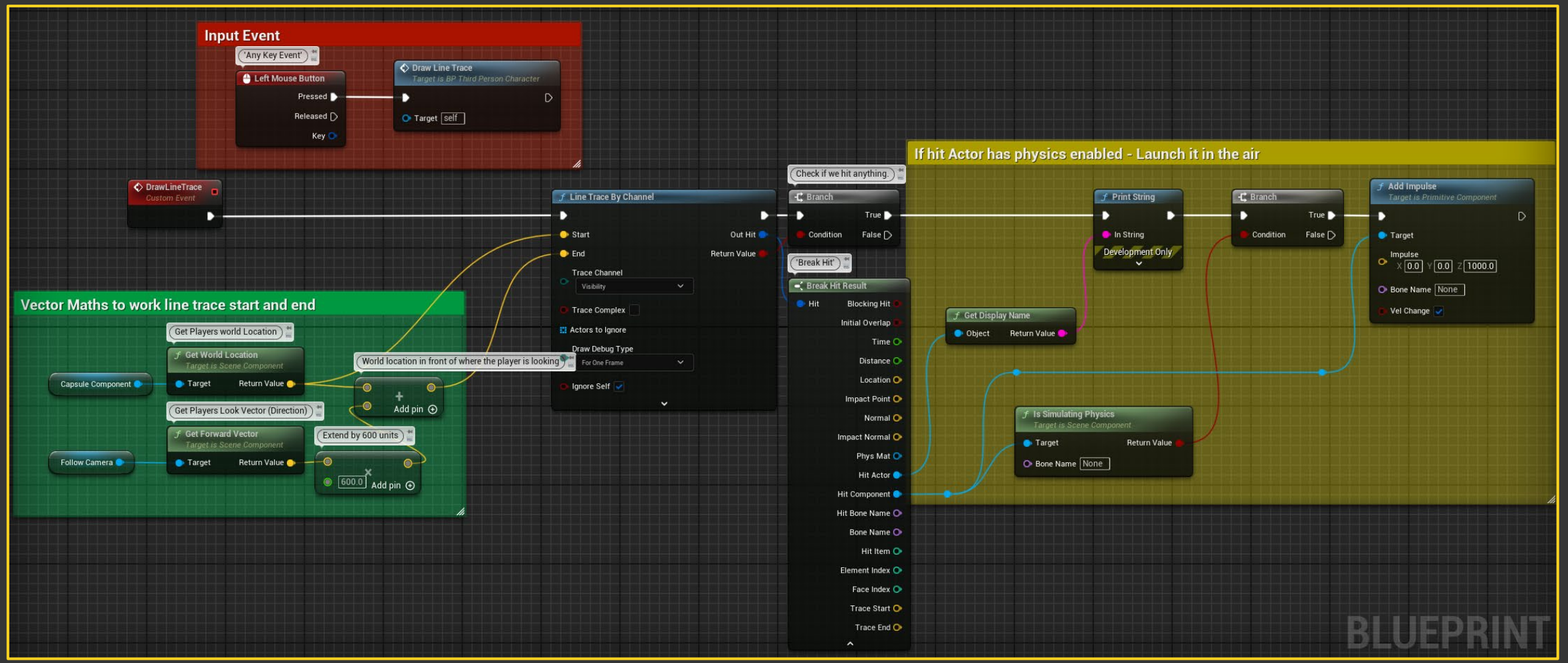
- We can use the ‘Hit Actor’ actor reference to see what our line trace has hit.
- **Example:**
  - **Hit Door:** We can try and open the door.
  - **Hit Button:** Try and interact with the button.

- You can use **‘Cast To’** to see what we interact with. Or we can use a ‘Blueprint interface’ to ‘Interact’ with said actor.



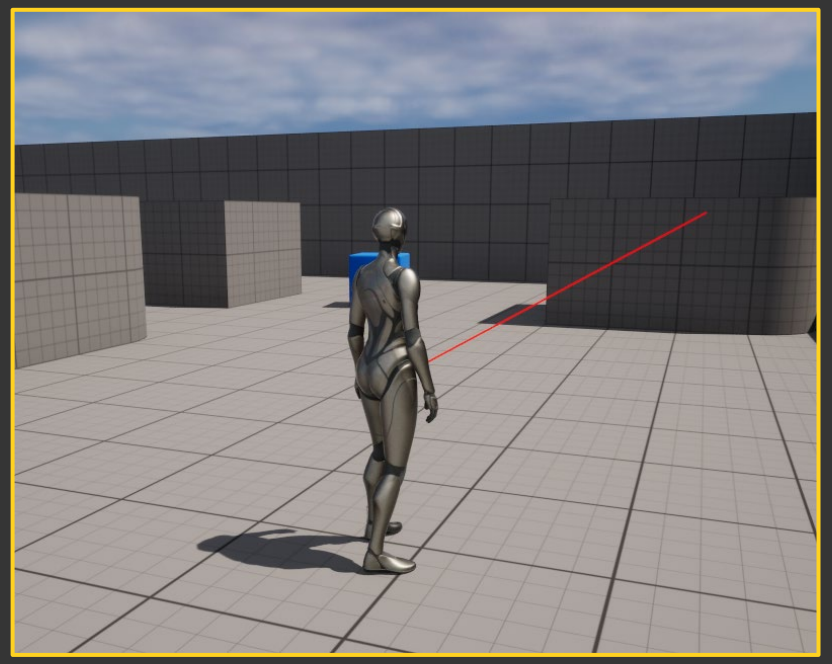
Line Trace – Setting up a line trace.

Blueprint inside the 'Third Person Character'.

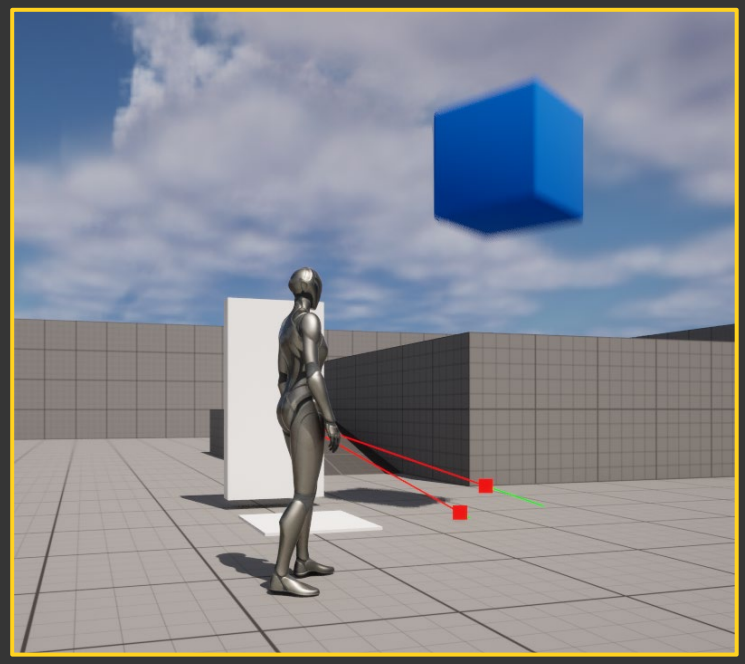


Line Trace – Previous Blueprint

- When I press ‘Left Click’ I trace a line in front of my character

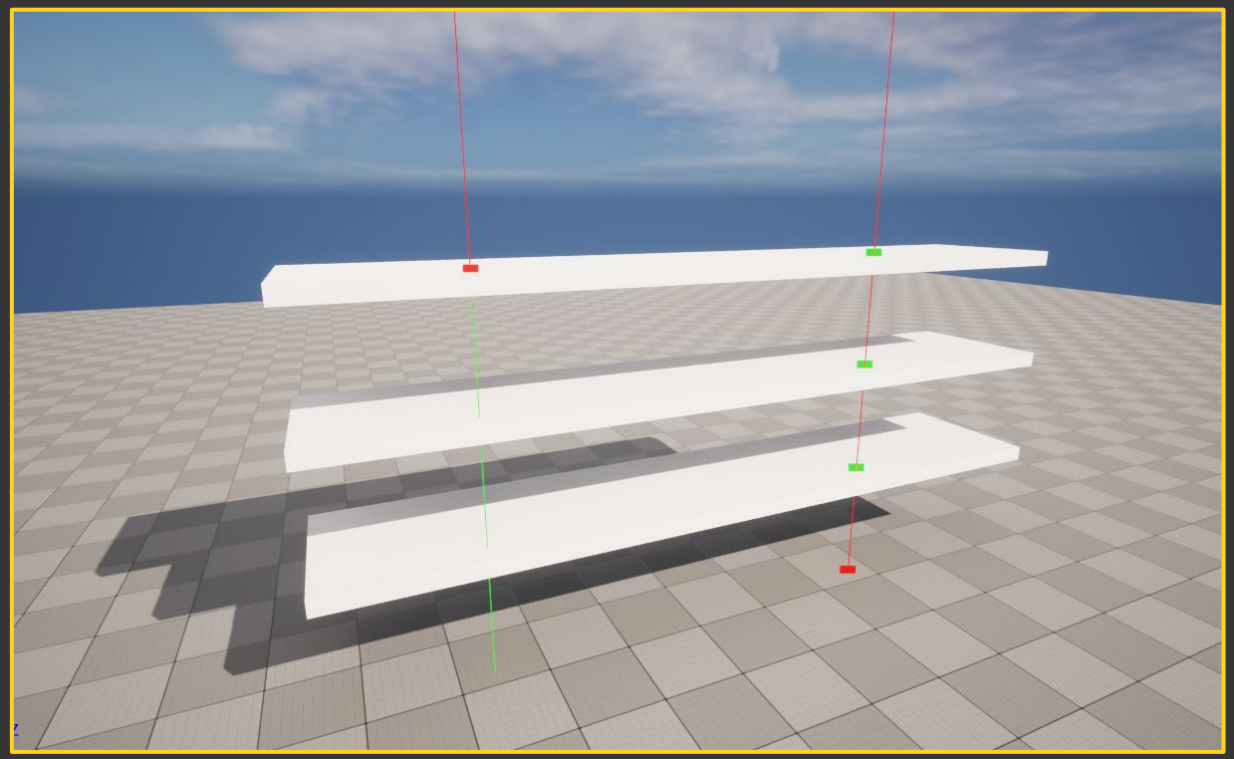


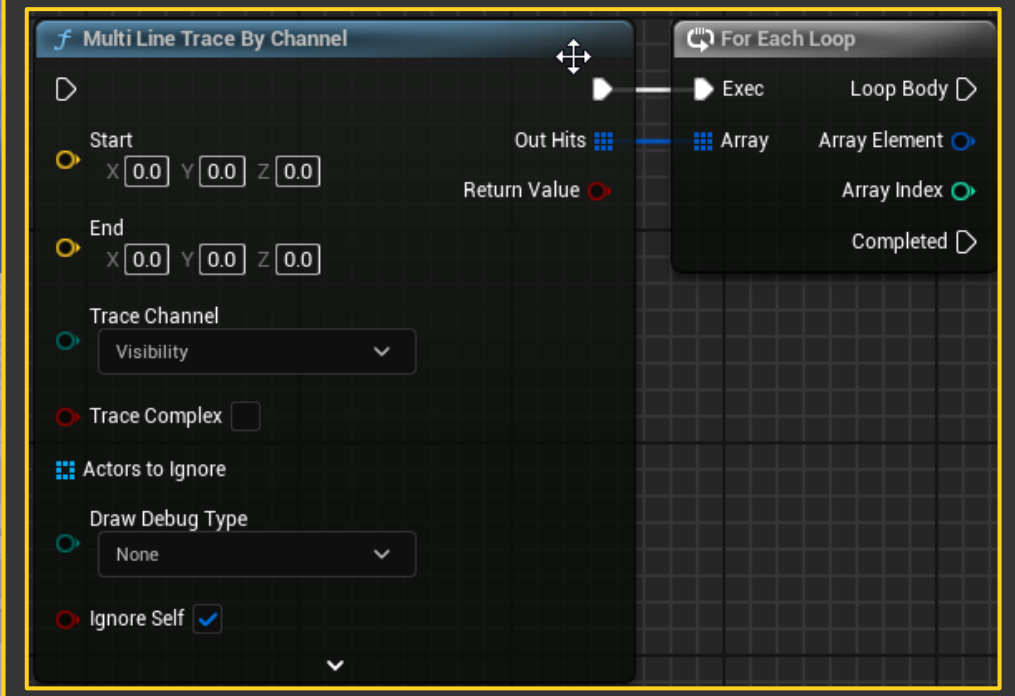
- When I ‘Hit’ an object, the object is launched in the air.



Line Trace – Single vs Multi

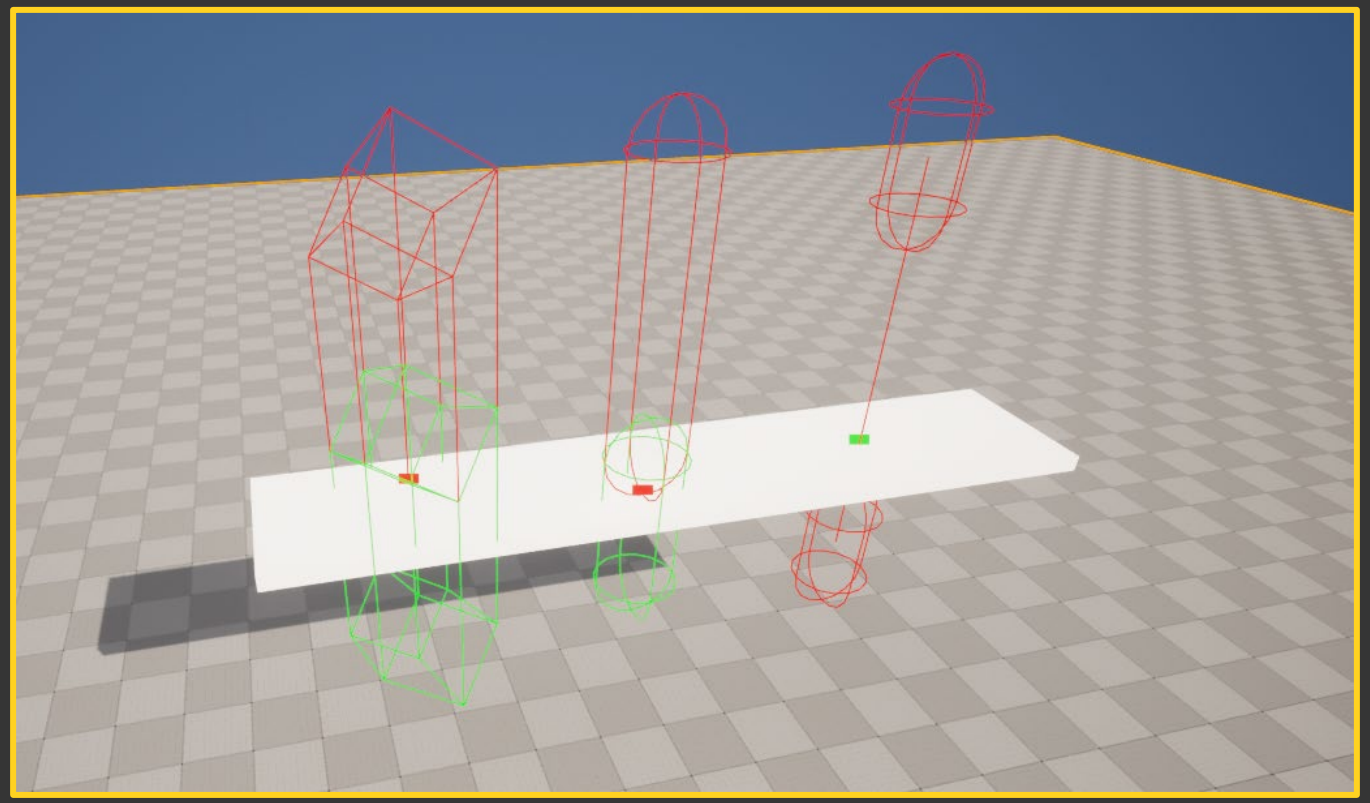
- Multi-line trace provides an ‘Array of Hits’





Shape Trace – Section name

- There are also Box, Capsule & Sphere traces you can use



[Traces with Raycasts in Unreal Engine | Unreal Engine 5.3 Documentation]


Tracing is great for collecting data to use later.
Now unto Actor Communication!

Actor Communication – Intro

- There are 3 main ways of getting actors to communicate in Unreal Engine.

- **Direct Casts** – Getting a direct actor reference from an 'object' we've seen*.
- **Blueprint Interfaces** – Sending a message to an actor and hoping they are listening.
- **Event Dispatchers** – Send out an event, All actors that are listening will receive it.

- **Currently we've covered Direct Casts & Actor References.**

*Seen in the terms of our actor has a reference to it through a trace, a collision, or other manners.

Blueprint Interfaces

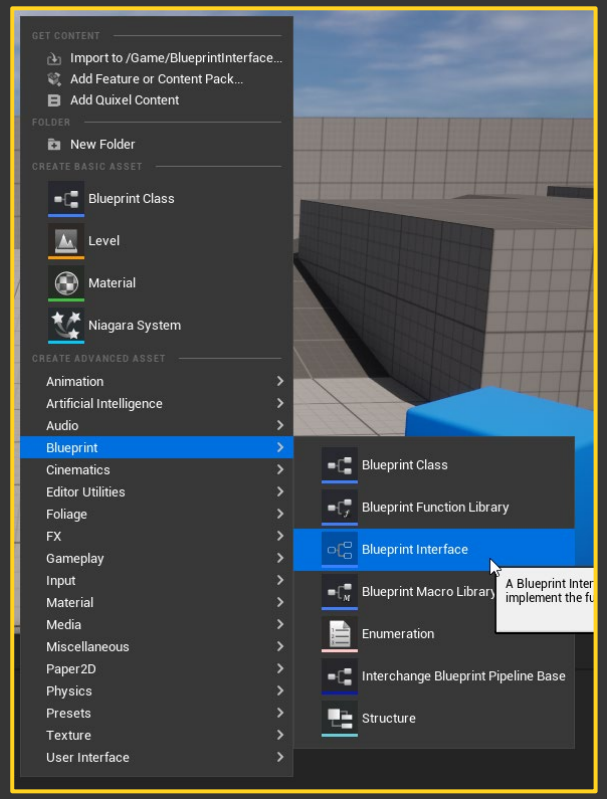
Sending a message in the hopes someone hears.

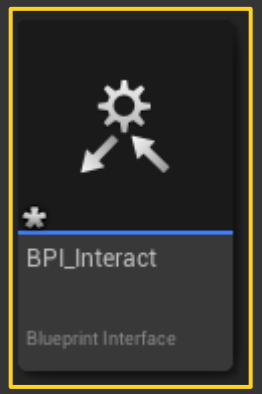
Blueprint Interfaces – Basics

- A collection of one or more functions that we can add to blueprints.
  - These functions work only as calls without ‘implementation’
- They allow blueprints to share data and ‘**call events**’ without explicitly having direct class-type references.
- The reference part is important because it allows us to limit our ‘reference chains’.

Blueprint Interfaces – Creating an Interface

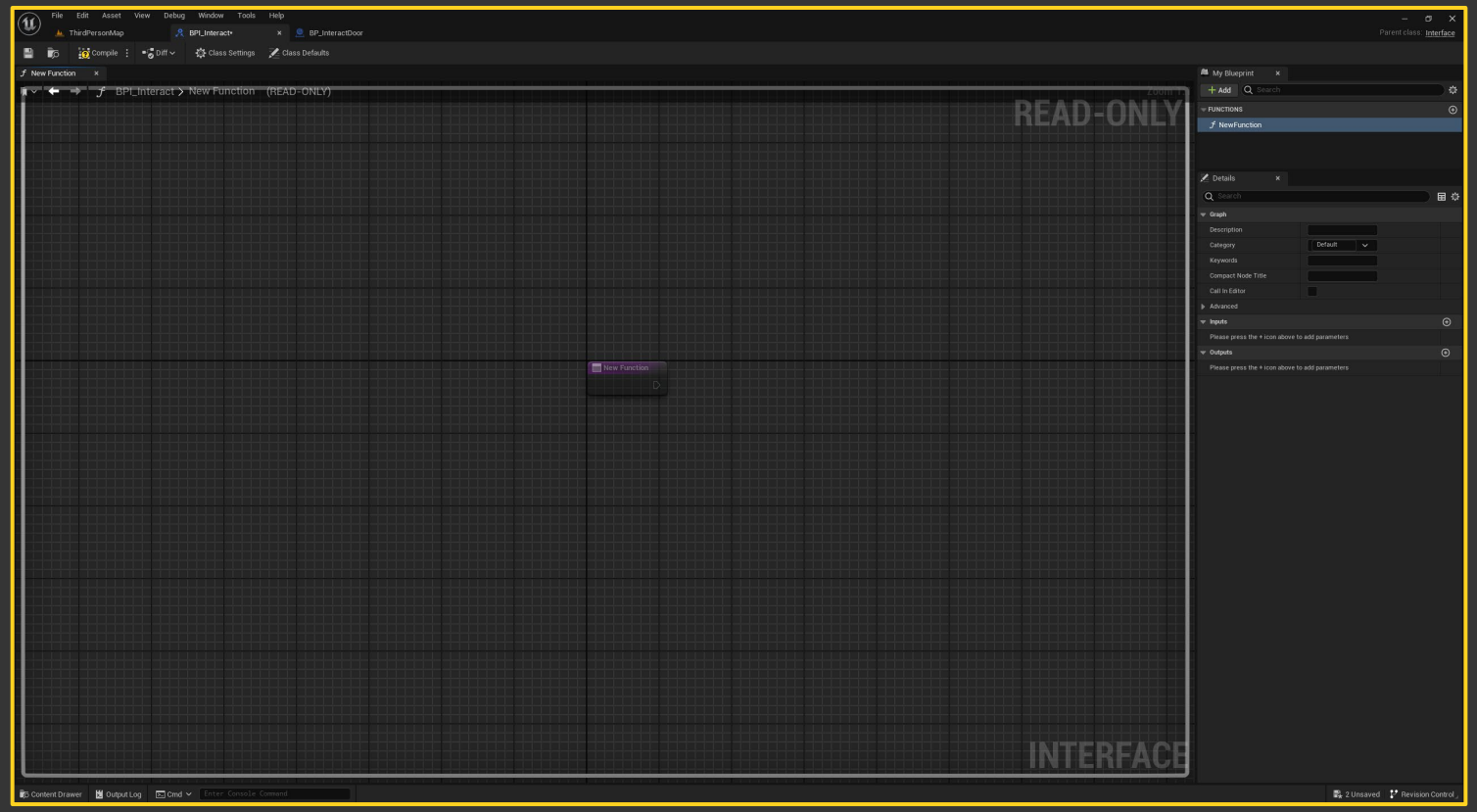
- To create a blueprint interface. **Right Click** the **content browser**, Select **Blueprint**, Then select **Blueprint Interface**.





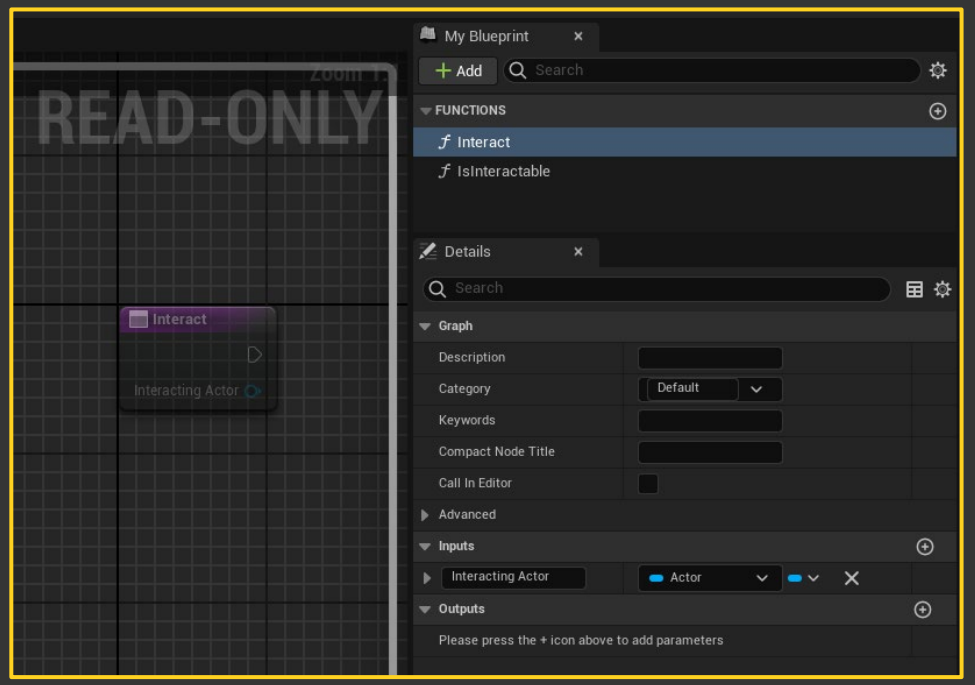
Blueprint Interfaces – Interface Functions

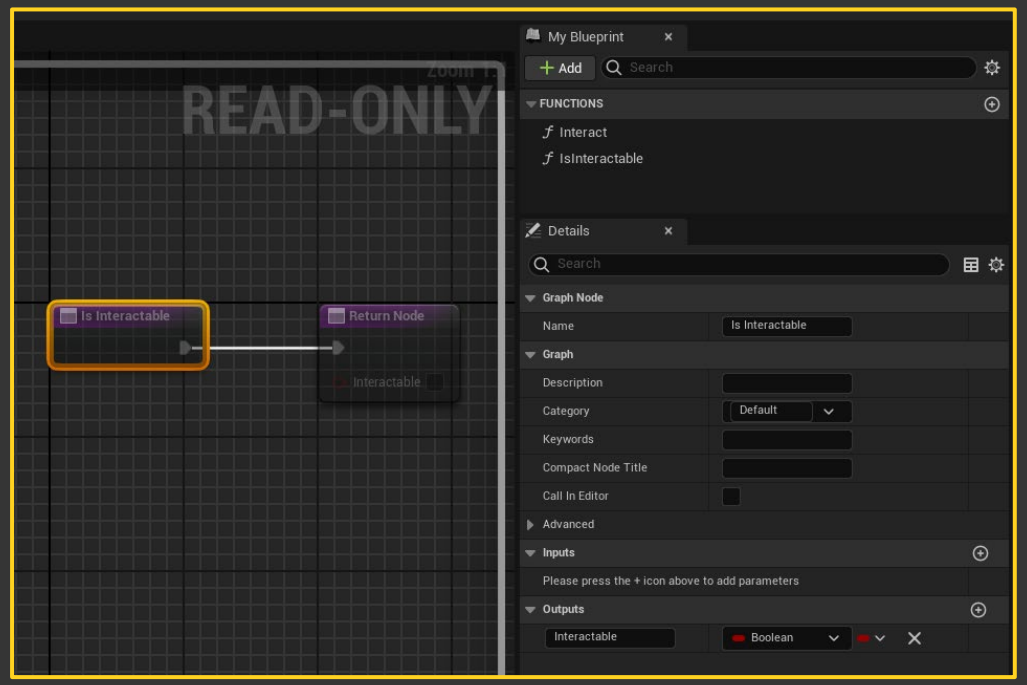
- Opening an Interface, you'll find that you can only create 'function calls' with parameters and not any specific functionality.
  - **You cannot add nodes***



Blueprint Interfaces – Interface Functions

- Add the following Functions:
  - Interact: with Actor Input called ‘Interacting Actor’
  - IsInteractable: With Bool Output called ‘Interactable’

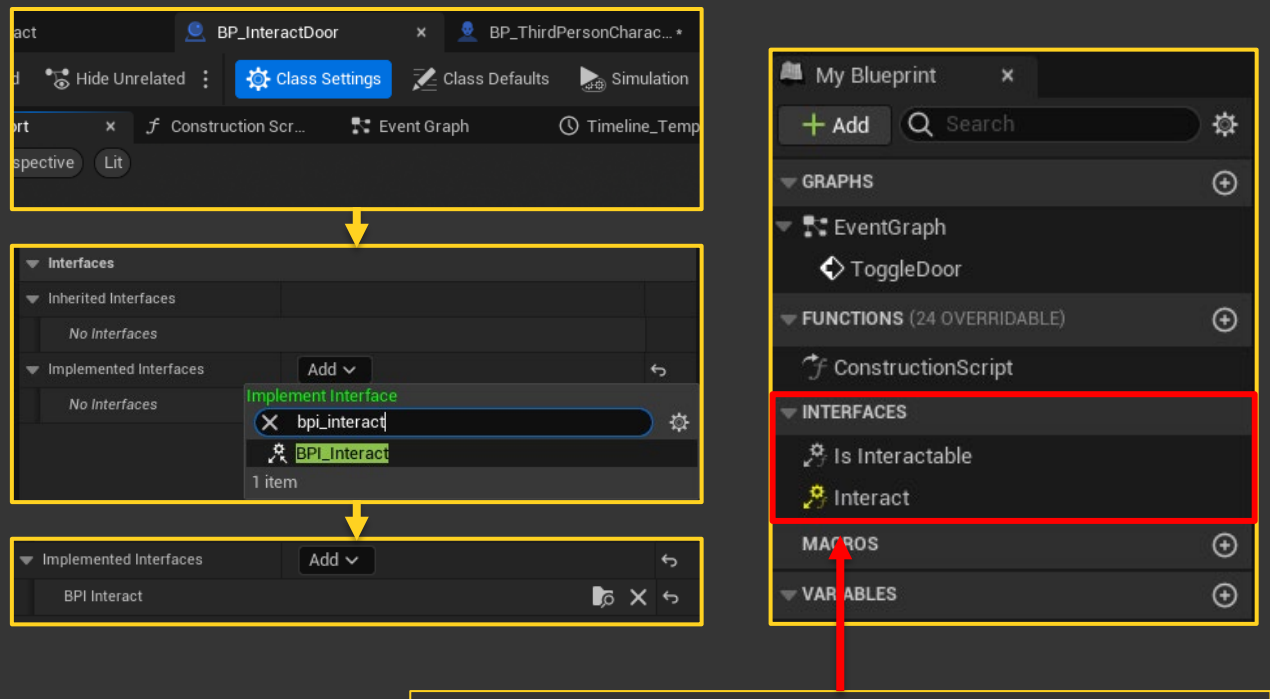




Blueprint Interfaces – Adding an Interface

- We need to '**implement**' the interface in the actor that will use the functions in the interface.
  - **Example**:
    - The door needs to be 'interactable' so the interacting interface will be implemented in the door.
    - **Not the actor 'interacting'**

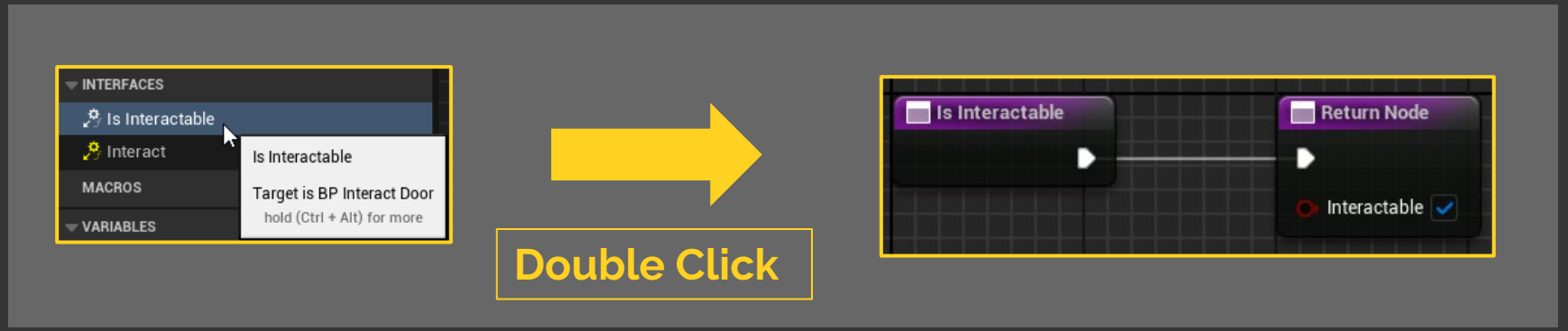
- **To add the interface**, go to Class Settings. Then in the details panel at the bottom. 'Implement' The interface you created.



You should now find the functions you created in 'Interfaces' section of 'MyBlueprints' panel.

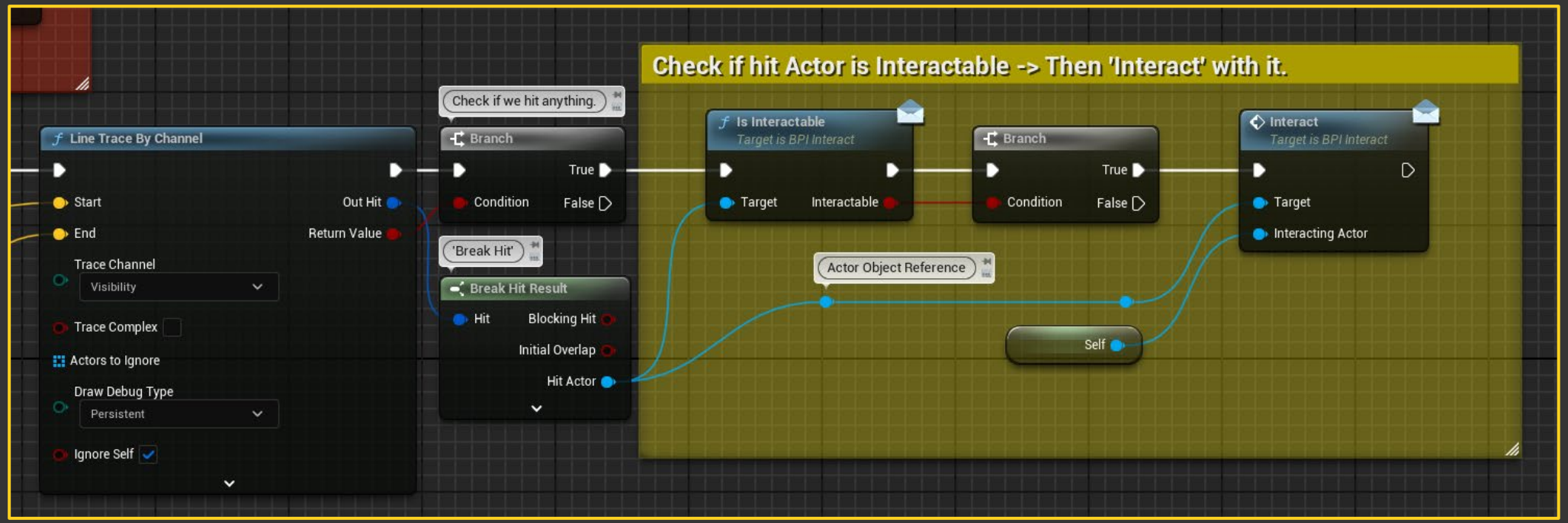
Blueprint Interfaces – Using the Events/Functions

- We can now use and 'script' with the Events/Functions from the interface.
  - **Interface Function with no return will be an 'Event'.**
  - **Interface Function with a return will be a 'Function'.**



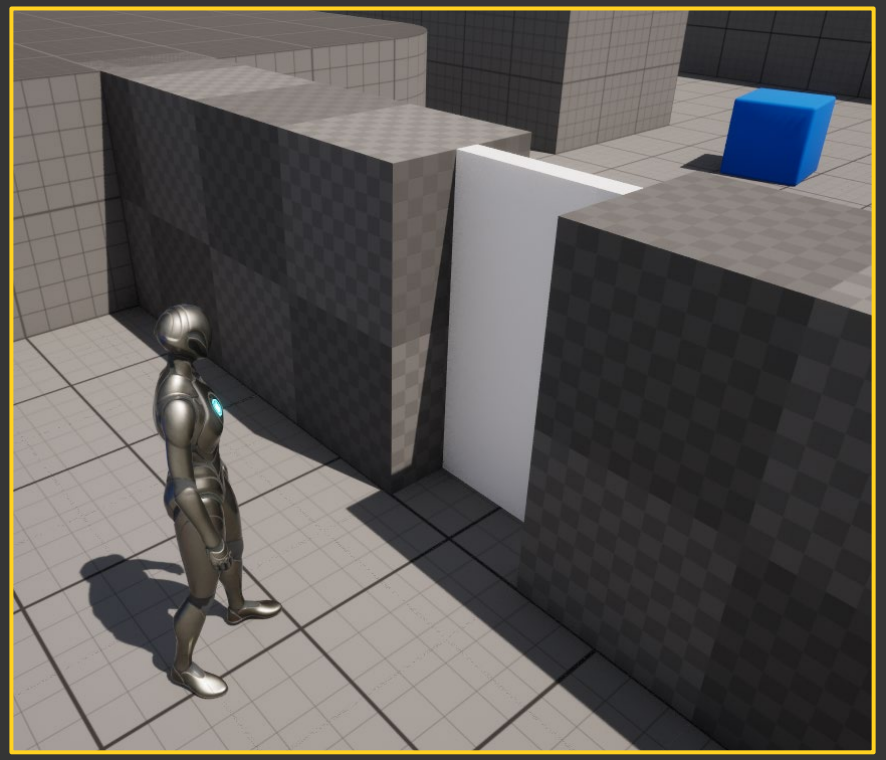
Blueprint Interfaces – Calling an Interface

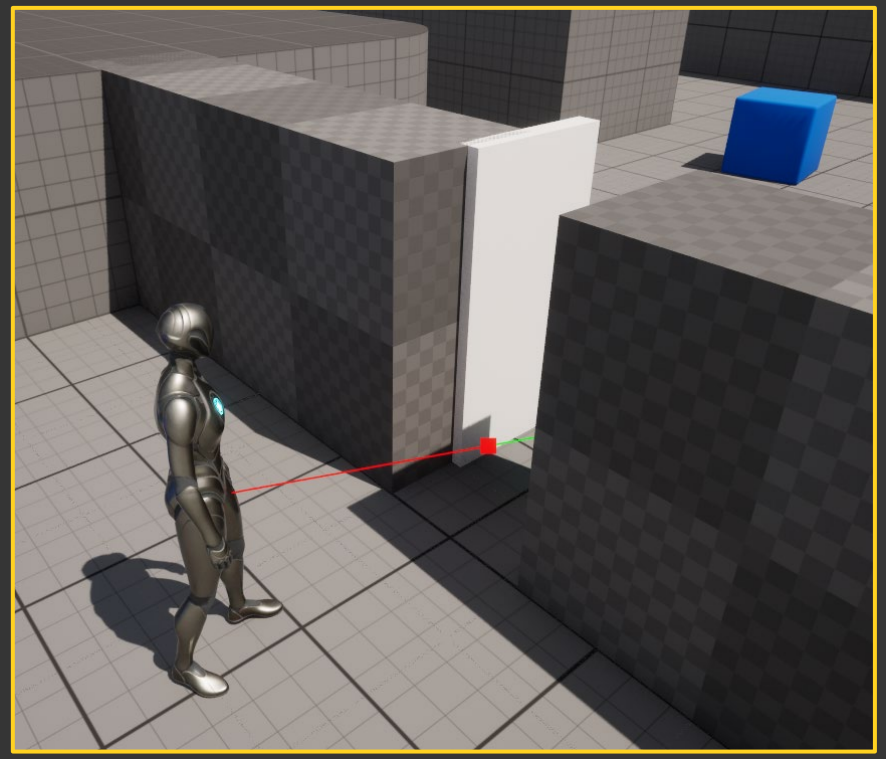
- So the main benefit of interfaces, is that we can call them 'Globally'.
  - We can 'attempt' to call the interface in any actor.
    - Notice how I call through an 'Actor Object Reference' using the line trace.
  - Only those with it implemented will react to it.



Blueprint Interfaces – Showcase

- Using the blueprint previously showcased ‘hitting’ the door with the line trace will result in the character ‘interacting’ with it – Opening the door.





Blueprint Interfaces – Usage

- Be sure to group your interface functions appropriately.
- **Don't have a single interface for:**
  - Interaction management
  - Damage Systems
  - Buffs & Status Effect Systems
  - Inventory System
- Only have functions in an interface that the actors using the interface will use.
- If you have to create separate interfaces -> **Good!** Keep things organised.

Event Dispatchers

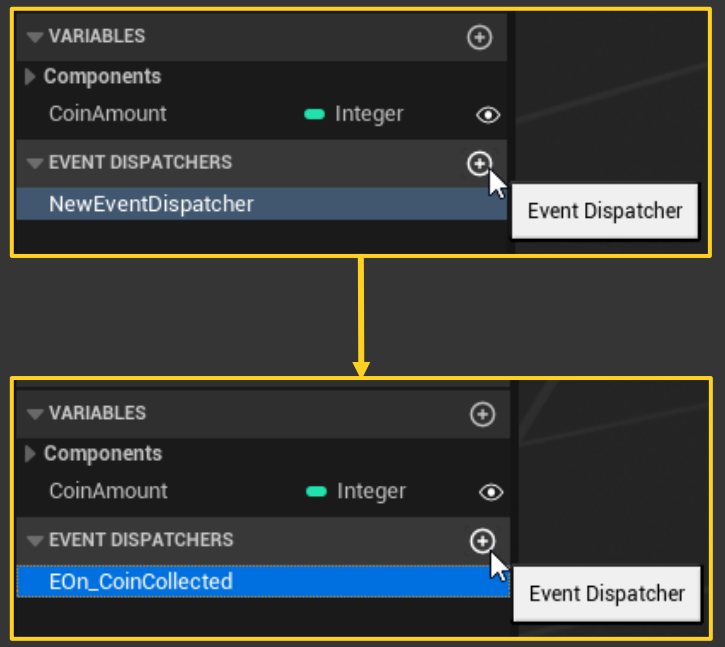
I'm doing something, now you do something!

Event Dispatchers – Basics (a.k.a Delegates)

- I like to use the following explanation when discussing 'Event Dispatcher'.
- An Actor with an Event Dispatcher is like a 'radio tower' that sends out a signal.
  - Only actors that are 'tuned in' to listen will be able to hear anything.
- **Why might this be useful – Instead of constantly checking for updates we should only update data based on when a condition is met.**
  - **Example 1:** UI Displaying Health, should we constantly get the Health amount to update the UI or only do so when the health value is modified?
  - **Example 2:** Should we constantly check where a player is in the level to update a checkpoint or only when they reach a specific trigger box?
- **Event Dispatchers need to be 'bound' to which means a reference to the actor is required.**

Event Dispatchers – Creating an Event Dispatcher

- To create an event dispatcher in an actor just create it in the 'event dispatcher' tab in the 'myBlueprint' panel.
- A good naming convention for Event dispatcher is 'EOn_*Dispatcher Name*'
- **This is done in a new coin actor that I've created.**

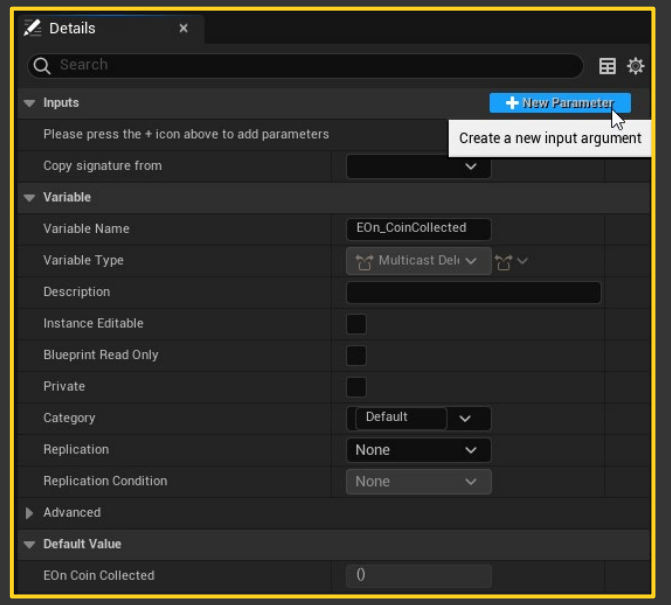


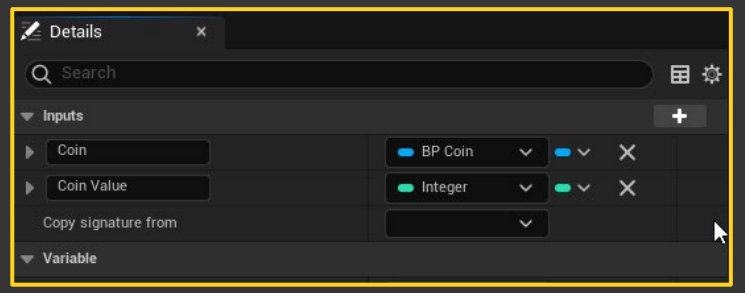
Event Dispatchers – Adding Parameters

- We can add Parameters to the Event Dispatcher in the details panel.

- I've added the following Parameters:
- Type '**BP_Coin**' named 'Coin'
- Type '**Integer**' named 'Coin Value'

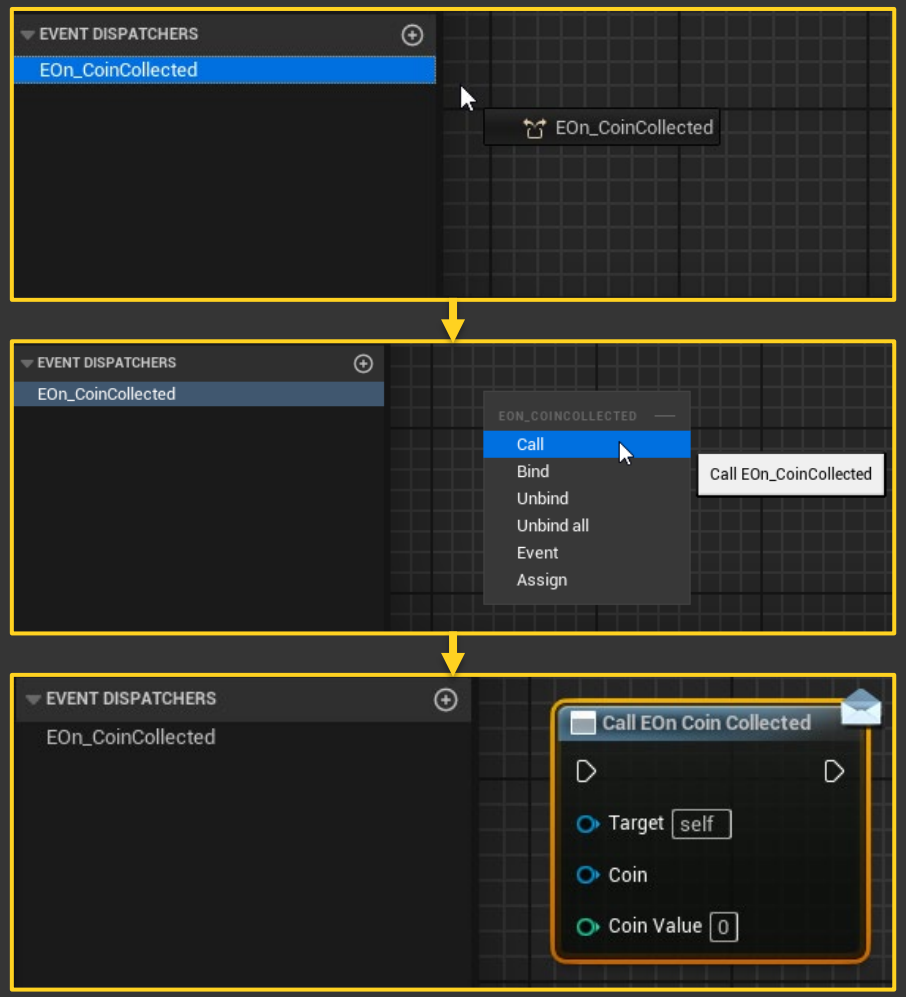
- **I will be using these parameters to send data through our Event Dispatcher.**





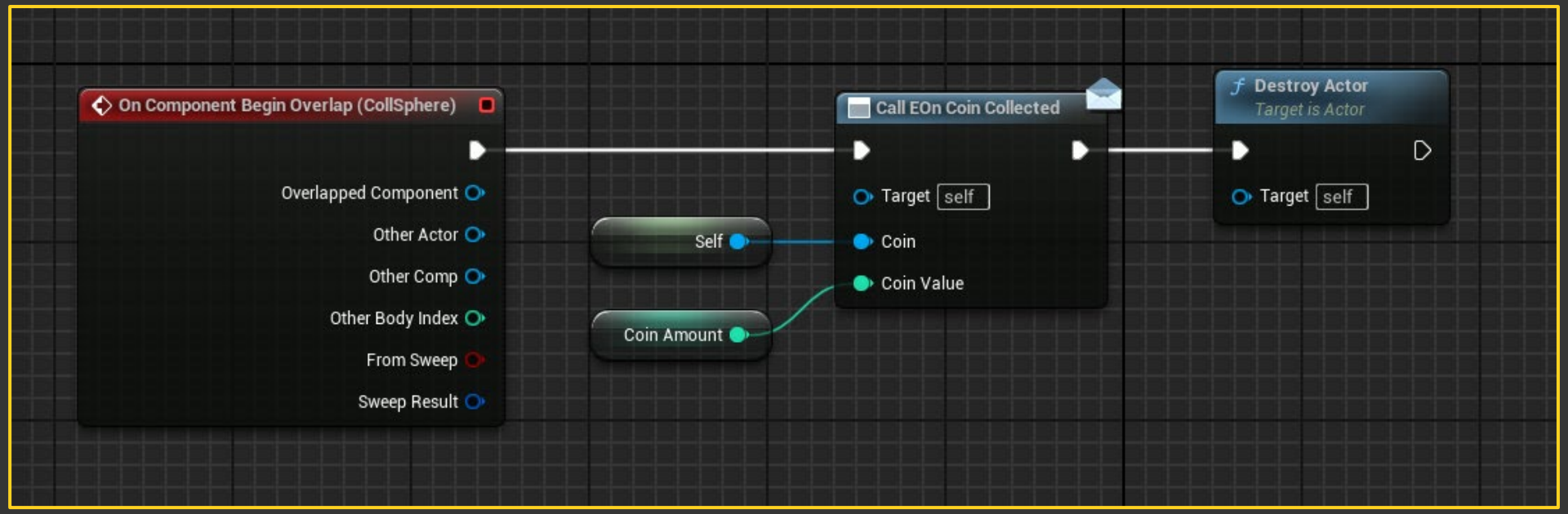
Event Dispatchers – ‘Sending a Call’

- To send a ‘call’ through our event dispatcher we need to ‘drag’ it into the event graph. Then select ‘Call’ which will create the node with our parameters already in.



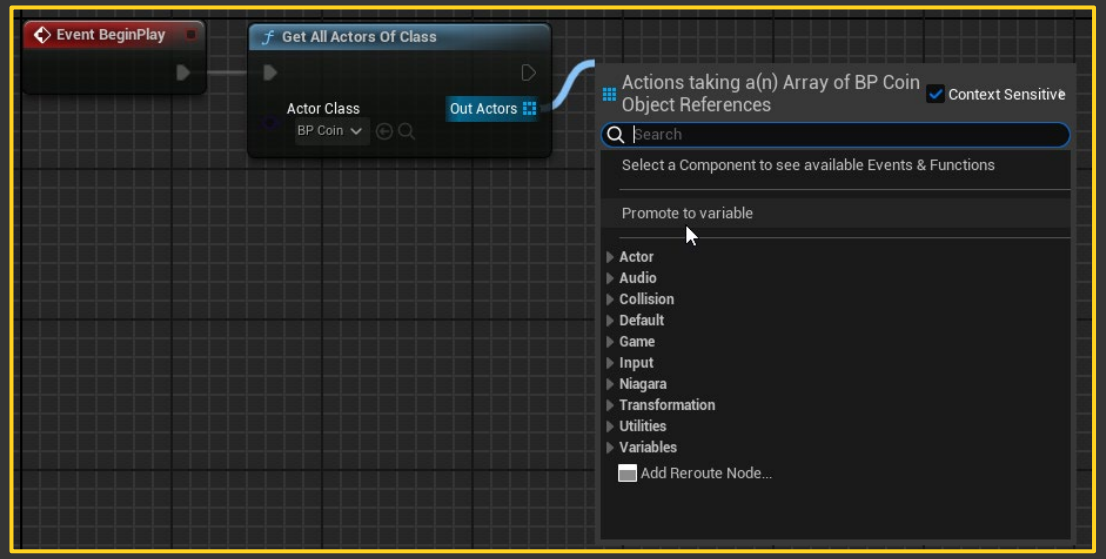
Event Dispatchers – ‘Sending a Call’

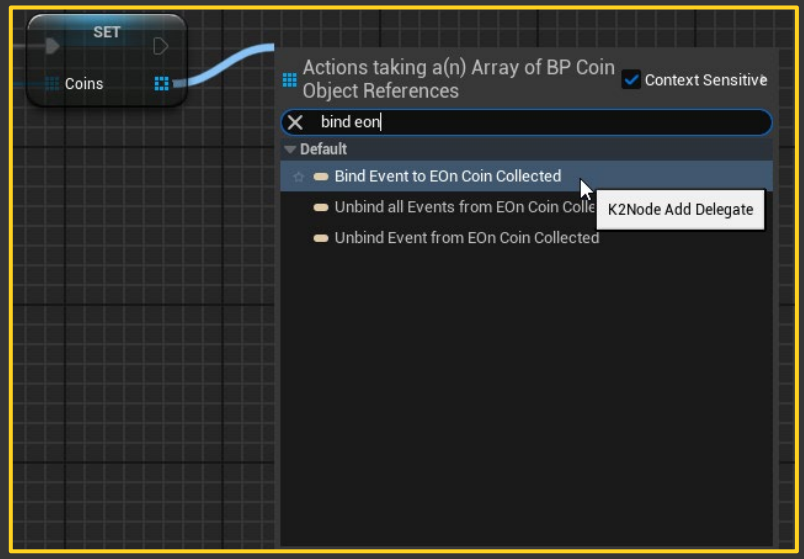
- **I’ve added the following blueprint to call our event dispatcher**
  - You’ll notice the ‘Target’ for the call is ‘Self’ because we are not sending the event dispatcher to anyone specific and are just making an announcement.



Event Dispatchers – Binding to an Event

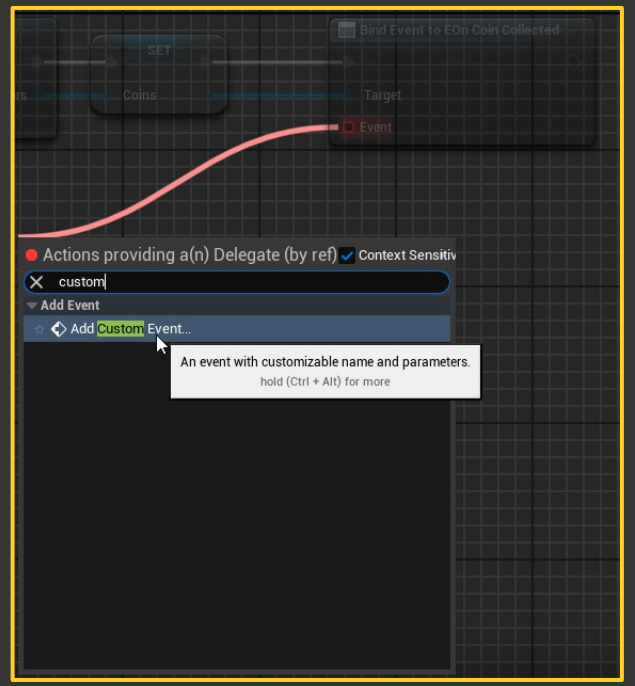
- I've also created a 'BP_CoinManager' actor that will listen to the coins and manage my score.
- My manager gets all the Coins in the level and 'Binds' to the 'EOn_CoinCollected' Event

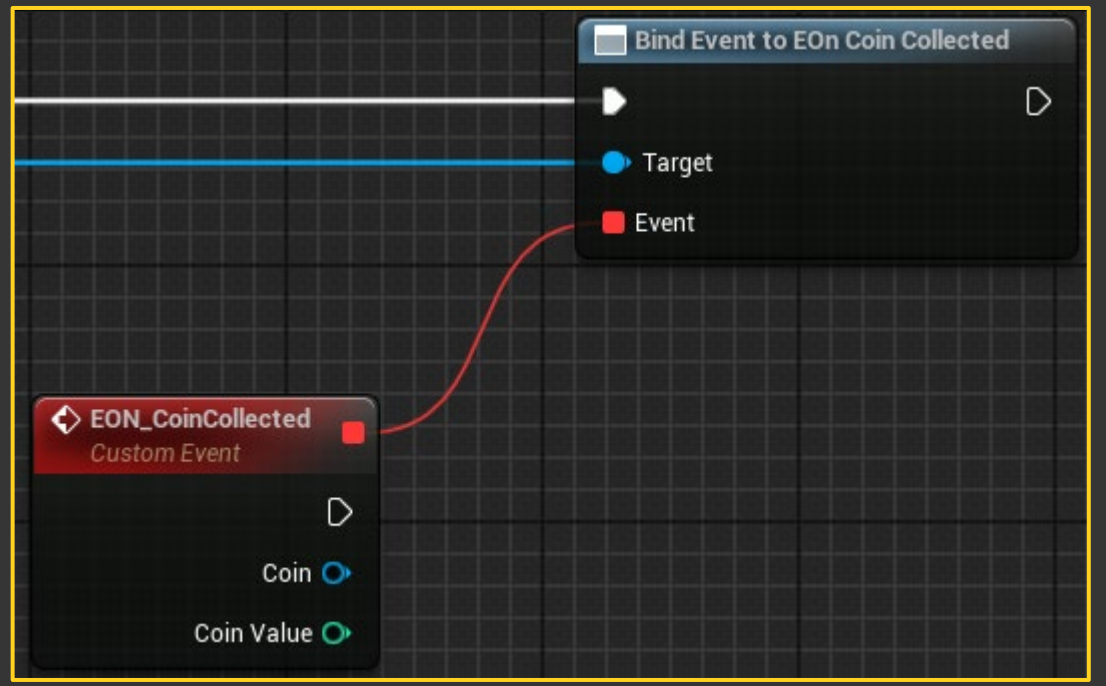




Event Dispatchers – Binding to an Event 2

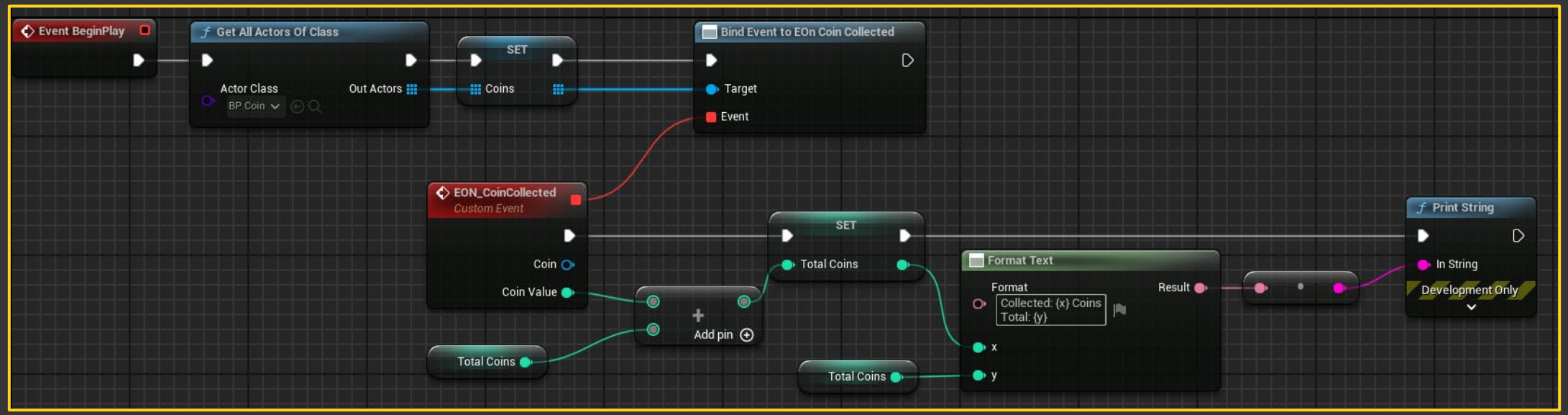
- The ‘Red’ Event pin allows us to specify what event we want our Manager to fire when the Coin collected dispatch is fired.
  - In this case I create a ‘Custom Event’ and Named it Eon_CoinCollected
    - Notice the parameters are added automatically.





Event Dispatchers – Binding to an Event 3

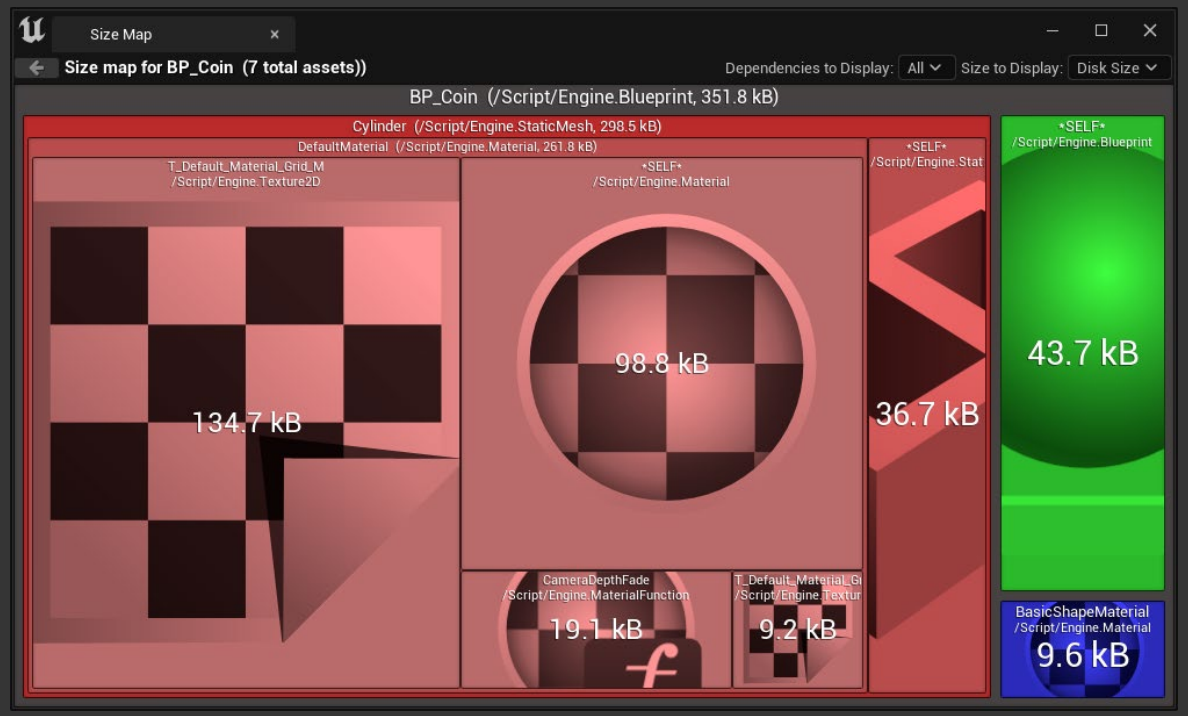
- Then I used the parameters to update a variable in our Coin Manager and print a debug message telling me the event has fired and parameters have been passed through.
  - **If your event is not firing. Double check your Manager actor is in the level.**



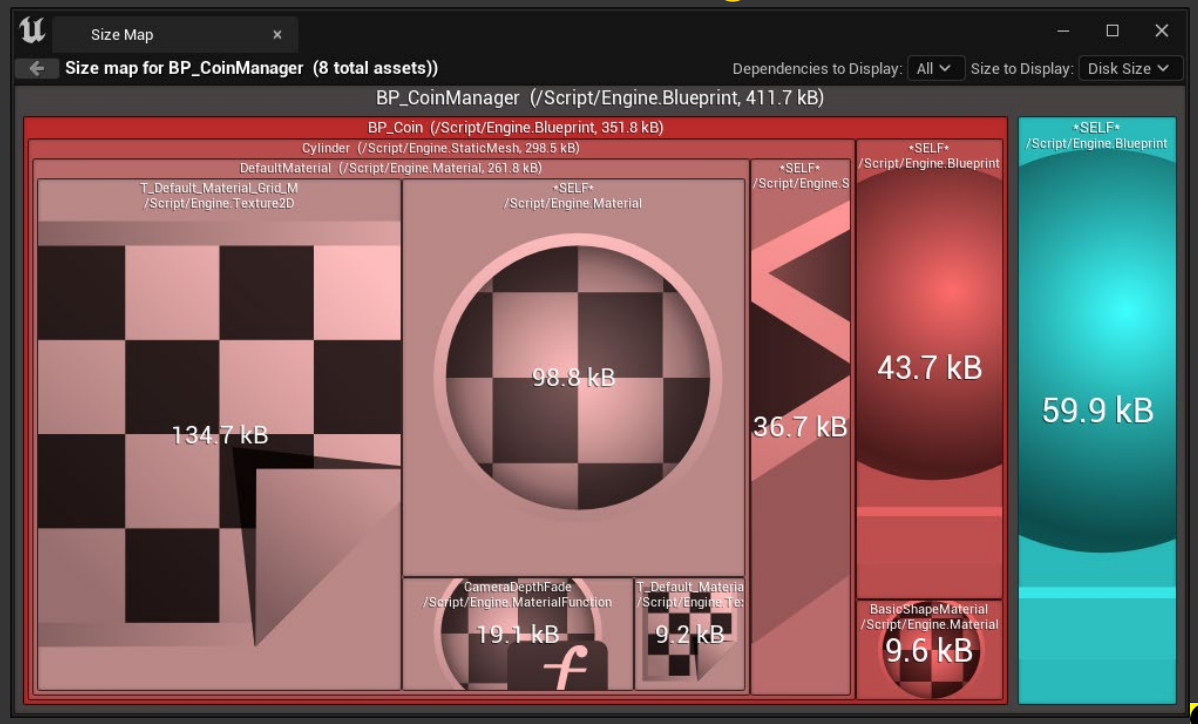
Event Dispatchers – References

- Notice how our Coin Manager has a reference to the coins. However, the coins don't need a reference to the manager.

BP_Coin



BP_CoinManager



—

Combine Event Dispatchers and Interfaces to create de-referenced communication.

Optimisations is great!

What's Next?– Next week's subject

For smaller projects optimising referencing is not necessary, but it's still important to practise the approach. Be sure to use both Interfaces and Event Dispatchers when setting up actor communication.

Consider how your projects could be optimised.

Next week we will focus on the following:

- **Functions**
- **Enumeration (Enums)**
- **Structures (Structs)**

If you have any Questions – Contact me!

You can find me at the following:

Location: Tech Support, SA1-03, St Andrews Building
(Hours may vary, Check VLE)

Teams: Linards Bitte

Email: L.Bitte@NorwichUni.ac.uk