3 - Data attributes

Variables can be called and set. They can be made instance editable by clicking the 👁

Variable/ data types:
Boolean= true or false
Integer= whole number
Float
Name/ string/ text.
String can be searched and manipulated (can use substrings)
Text supports formatting and can be used as character dialogue on the screen
Name is used for identifying objects
Vector 3, could be location or rgb colour
Rotator- 3 values that represent rotation in 3d space
Transform= location, rotation, and scale

There are also way more, like colour

4 - Using the event graph

Event graph and construction script. Event graph runs when its simulated. Construction scrips runs when ur building the level, (design time).

Begin play executes when it plays
Event tick executes every time there is a frame.

Print string- prints text on the screen. Can plug variable into it. It automatically coverts to a string. U can change the duration.

Can use subtract node by plugging hp in, and sting it after subtracting 1.

5 - The construction script

public variables 👁
Can look at what objects u have in the bp, when selected u can see what properties u can change on the details panel.
How to do it:
Drag and drop the component onto the construction script to get a node that references the component.
Drag from the node and then u can search for the property u want to change. If ur changing light colour u can choose set light colour
Where new colour is u can right click and choose promote to variable. Then u can make it instance editable, so duplicating the blueprint u can choose

diff colours.

6 - Using arrays

Good for storing a list of keys or other stuff. Can be searched, can get random values from list.

Create an array:
In event graph, create variable, type string, convert it to array by changing the button from single to array.
Add array elements by pressing the + button

When getting array drag from the variable node and choose get a copy. It has a number which refers to the index of the array.
Can connect this to print string node to see that its being retrieved when simulated.

Instead of get a copy u can use other nodes. They are listed below:

Another thing u can do is instead of the get node use an add node. U can also use add unique to make sure that value isn't already in the list

Can make array variable public too see whats going on/ being added when playing

The other node is insert. This is like add but u can choose the index number it will be, moving other values.

Also there is remove index node. It will remove any item thats in the chosen index.

Can u send remove item to remove a specific item. This has a boolean pin which shows if the item was removed or not.

Can also use the clear node to remove all items from the array.

Can use the contains item node. It returns true or false if it contains the item or not.

Find item node. Include item name, it returns the index of the item if its found, but if it isn't found it returns -1.

Can also use length node to display the number of items contained in the array.

Last index node will display the last index. So if u have 3 items last index is 2 because 0 is included in index

Is empty/ is not empty returns boolean that tells if its empty or not.

Is valid index can say if the index u put into it is valid (exists) or not

Misc nodes:

Shuffle randomised the order of elements of the array.

Random chooses a random index and return it.

7 - Code execution flow:

Sequence node executes code one after the other. Can add more pins.

Flip flop. Alternates between a and b each time the node is called. Can double click to add a node redirect before the flip flop node, and connect path A output to it. Dont connect both or you'll make an infinite loop and break it. This node is good for light switches

Do once node makes sure code is only run once. Ex. Plugging it into event tick means it doesn't happen every tick, just the first tick.

Do N means u can choose how many times the code is executed. It has a reset pin as well

Branch node. Executes based on of condition is true or false

For each loop node. Executes for every element in the array. Also has a complete pin

For loop node same as for each loop but u can choose which indexes its between

While loop node executes as long as the condition is true. If condition never turns to false then it might crash unreal.

The switch statement. If u have an array connected to a random node u can connect that to a switch on string node.
U can add elements to it and can match the index number to each case output.

8 - User defined data types:

Enumerator (enum). User defined list of options.

To make it right click content browser, go to blueprints, enumeration.
Double click it to open it. To add items click add enumerator
In the blueprint u can make variables. When choosing the variable type u can type in the name of the enumerator. U can then make it a public variable. Now in the viewport u can pick from a dropdown which item of the enumerator.
If u drag this variable into the construction script and get it, u can drag out the pin and use a switch node.
Example in the video is making one enumerator output make the light a random colour, and other output lets u pick the colour.
Also u can turn off run construction script on drag off in the class settings if u want.

Structure (struct)
can contain different types of variables inside of it.

It can also contain enumerators.
To make it: right click content browser, blueprints, structure.
Double click to open it.
Click add variable to add variables.
Choose the default values in the default values tab.
Now u can make a new variable and set the variable type to the name of the structure that was created.
Can make this variable public. Can drag this variable into the graph and get it. When dragging from the pin u can use break struct. To break it into the separate variable pins.
You can have structure inside structure.
Just create another one, add new variables, make one of the variable types the name of the old struct.
Again u need to make a new variable with the struct as the variable type.
When dragging from the struct variable, instead of breaking it, u can also set members in it. In options of this node u can tick as pin on the variables u want to change.

9 - Functions and events:

Functions can be called. Can make local variables INSIDE the functions.
In event graph go to functions (above variable) and

click the + and name the function. Using fn as preser can help with searching for them.
Now u can drag and drop the function into the event graph
In details panel u can add inputs and outputs.
Inside function node u can add local variables under variables panel.

Custom events:
Right click and search for add custom event node. Name the event. Prefix it with ev to make searching easier. Attach the code to be ran to the event node. Then u can call the event by dragging from a node and search for it.

Differences: can only add input events to event node
Events can be called in editor, if u make it public. (Unless it has inputs)
Functions cant have delays as it executes all code in the frame its called.

10 - Time based code:

Delay node lets us delay executing portions of code. Can change duration with a float variable. If u want the delay to be random then drag from the float pin

and search for random float in range.
Retriggerable delay node will reset the wait duration every time its called.
Repeating code can be done by adding a delay node to the end and calling the event again after to make a loop. But u can also do this with the..

Timer node, which can loop. Right click and search for set timer by event node. So now u just connect the event, enter the time (for delay) and check the looping box. U can expand for more options.
U can also search for the set timer by function name node. Instead of connecting the event, u type the name of the function or custom event. Same result.
With the timer u can right click the return value output pin and promote to variable.
This variable can be used as a remote, to press pause and play or stop the timer.
Drag the variable into the graph and drag and search for pause timer by handle. Then u can frag from the variable again and search for unpause timer by handle. Finally u can add the clear and invalidate timer by handle node
Now to pause and play from editor add 1 custom event for each node and choose call in editor for all 3 of them.

11 - Smooth movement

Interpolation- estimate values between start and end values.

The platform:
Add platform shape, add a variable called end location. Make the variable type vector, make it instance editable, and choose show 3d widget checkbox
Now there's a physical widget that represents the end location.
Now in the event graph use the vinterp to (v stands for vector) node.
Drag in the platform cube to the graph and set its location with the node set relative location . Select it and connect it to event tick. And make the new location the result of the interpretation node.
Make the target location of the interp node the end location variable.
Current location can be set by again dragging in a reference to the cube, and using the get relative location node
Delta time, right click and search for world delta seconds and plug that into delta time in the interp node.
For interpretation speed add 1

In this case the "current value" is changing as the cube moves, so it needs to be connected to the event tick so its constantly being called.

There is also the vinterp to constant node. Its the same but there's no easing.

Linear interpolation:
Search for the lerp (vector) node
This is similar to the interp node but it uses an alpha, a value between 0 and 1. When its 0 the return value is A, when its 1 the return value is B
Can connect same end location vector to end location, start location needs to be static so make a new vector variable called start location, set it on event begin play. Get cube relative location and use it to set the start location on begin play.
Now plug the new leap node into the set relative location node instead of the interp.
To make the alpha value increase smoothly promote it to a variable. Make the variable increase every tick by a small value by setting it.

The platform continues indefinitely because the alpha can go above 1.

To limit the alpha to 1 use the clamp (float) node.

The timeline node is good for making it go back and forth. So search the node and click add timeline. And u can name it.
U can double click the timeline node and u can add a track. For changing the alpha add a float track.
Can right click the graph and add key. Make it time 0 value 0. Add another and set it to time 1 value 1.
Aligh the view with the arrow buttons. Can adjust the length of the timeline to 1 for 1 second. Now there is a new pin that represents the graph on the output of the node
Now u can connect the timeline to the aloha of the lerp and to the set relative location node.
Attach a new custom event to play and call it on begin play.
To make it reverse u can create a custom event and connect it to reverse, and call the custom event when the timeline is finished.
To make it go indefinitely add a branch.
Drag from the direction pin of the timeline and search == and choose equal (enum) node. This returns weather the timeline is going forwards or

backwards. So connect it to the branch condition and call the desired event in each case.
To smooth the movement open the timeline and right click on the keys, and change it from linear to user for easing in and out.
U can change the play rate by dragging and dropping the tineline into the graph, choosing get, drag the pin out and search for the set play rate node.
Can promote new rate to a variable and make it instance editable. Now different platforms can go at different speeds.
Play rate works like a value of 1 is 100%, 2 is 200% ect.
U can add delays to the events being called after the branch to make it wait before moving back. Can add another one in front of the first event call as well.
And can make variables to edit both in design mode.

12 - Collisions and overlaps

Different object types have different responses for other types. U can add custom object types. Can use hit and overlap events.
Example is with a floor/platform blueprint.
With the cube in the bp editor u can scroll to

collision settings. If u choose custom from the collision preset menu all options become avaliable to edit. Also there's object type. We can choose what types it blocks as well.
Anything u want to hit it needs to have physics enabled in the details panel.

How to make custom object type:
Go to project settings, under engine choose collision, under object channels choose new object channel and name it and set default response.

In the blueprint now u can see the new object type, named boulder in this case.

U can change the object type of objects to boulder.

Detecting blocks and overlaps:
U can right click the cube/ component and go to add event, u can choose add on component hit, on component begin overlap, abd on component end overlap to make unreal actually fire the event when it happens click the component (cube) and in the details make sure generate overlap events and simulation generates hit events are both enabled. Generate overlap events ALSO needs to be enabled on the object that is overlapping as well.

On the on component hit node there is a struct called hit which u can break, to see a LOT of info

Append node takes two strings and returns it as one string.

U can use a box collision to make use of the overlap events, but make sure it moves with the platform by parenting it to the cube by clicking and dragging it into the cube.

13 - Direct actor communication

Direct communication- holds a reference to the target actor. If u reference a parent actor (like bp base platform with child bp actors platform A and B) then u are referencing the children too.

To assign a parent bp to a bp, go to class settings and under parent class choose what u want.

So in the parent class u can make custom events, and in the child class u can call them by going to functions, override tab, and choose the event u just made.

Alternatively u can use a boolean variable, make it in the parent class. To get it in the child class choose the gear icon above the variables menu and choose show inherited variables. Now u can use it in the child class.

Referencing

So making a new bp now (static box collision the player will stand in the make platforms move), to reference in the base platform bp u need to add a variable, and when choosing the variable type search for the name of the bp u are referencing, and choose object reference. Make the variable public and back in game mode u can choose which child bp actor u want to choose to reference
Make sure that with the new activation bp that is referencing the base platform has the nodes to activate it. Get the platform base reference variable abd drag out from it and search for the event (in examples case ev_activate) and connect that to on component begin overlap.

If u want to activate both child actors at the same time u can use an array.
Duplicate the reference variable- named platform list for this example (right click and choose

duplicate) and just change it from single to array.
Now drag and drop it to event graph and use a for each loop, and drag the array element pin to the custom event.
To make sure it only happens once, use the do once node and put it before the for each loop node.
Make the variable public and reference in the platforms to the list using the eyedropper tool
There is a way to add the instances automatically though.
To event begin play node attach the get all actors of class node. In the actor class part of the node specify bp platform base, it will return an array. U can then set the platform list variable we just made by connecting it to the node. This is apparently an expensive node so use sparingly

Using actor tags to specify which ones to move:
Select actor bp instance in the level.
Search for "tag" in details panel
U can use + to add a tag. For example we are adding room 1 and room 2 as tags to different platforms
Now back in activation bp, drag from the branch array elepment pin and search for has tag, and add the actor has tag node.
Now just type the tag ur searching for and use a branch node to set the outcome for if boolean is

true or false.
Also u can right click tab pin and promote to variable and make it public to change it in the level designer

Casting:

In the example an npc could activate the platform. To make sure only player can do it, drag from the other actor pin of the on component begin overlap node and search cast, then put the name of the player character, in this case BP_ThirdPersonCharacter
So if it can successfully convert the overlapping object (other actor) to a bp_thirdpersoncharacter class, which it will if the thing overlapping is the player, then it will output the first execution pin, if not it will output cast failed
Now only the player can activate the platform
What if we want it to only activate if the player has a key? There is an as BP third person character pin on the cast node which can reference data from the player actor.
So if u want to add a key then add a boolean to the player actor and check if its true in the activation actor using the cast node.
Do this by dragging from the as bp third person character pin and searching for the variable, in this

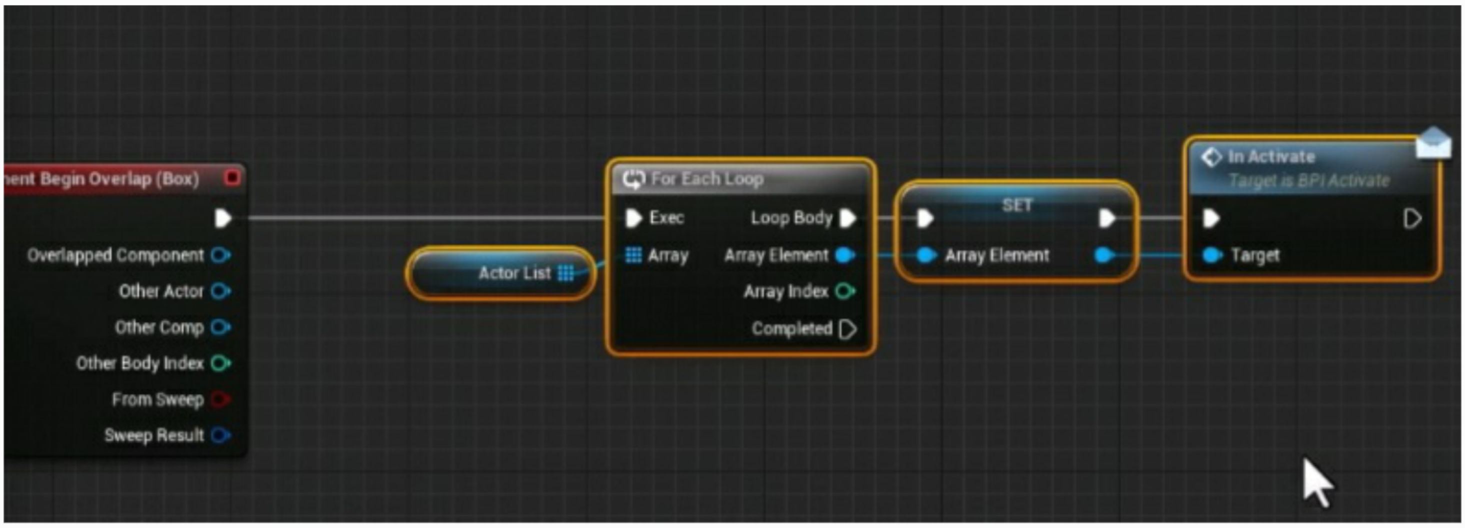
case "has key" also remember to create the variable in the character bp first
Choose get variable and use a branch node and connect true to rest of the code.

14 - Interfaces:

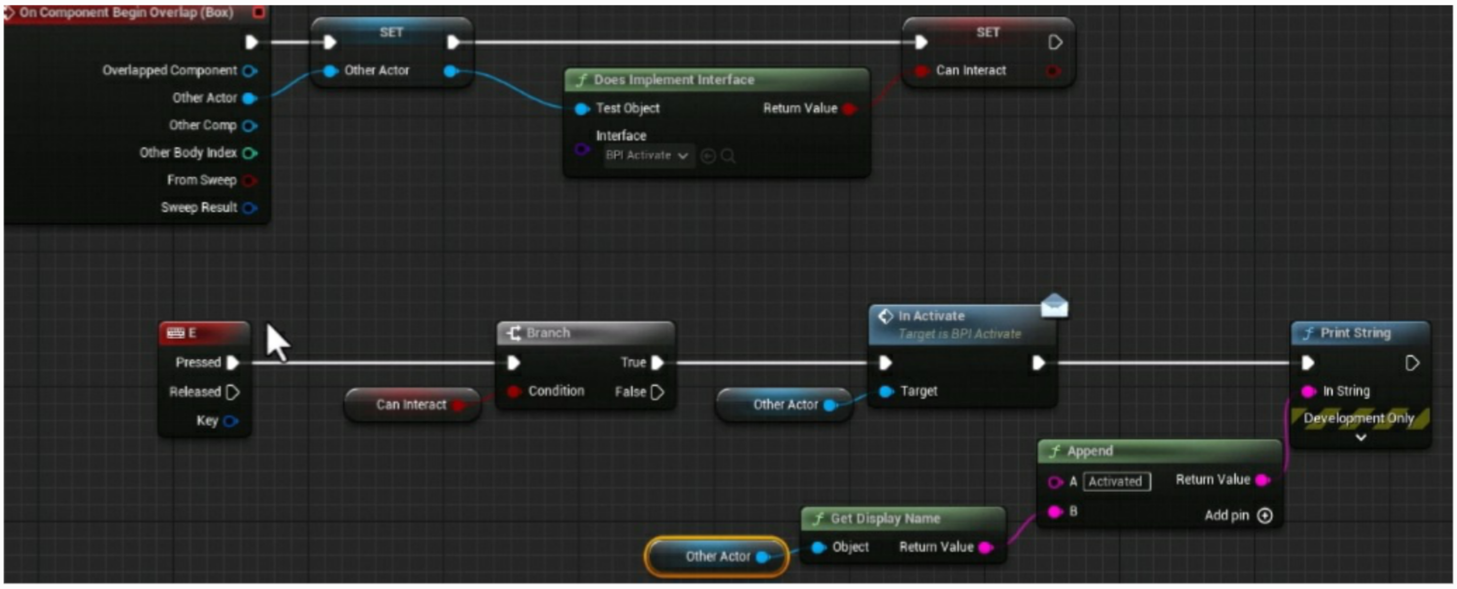
Bad way:
Make new activation bp, platform with box collision, create event for on component overlap of the box collision,
Make new variable, call it actor list, when choosing variable type search actor choose object reference, make it an array and make it public. Now whe u use the picker in level mode u can choose any blueprint. So now take the actor list variable, use for each loop node, promote array element to variable. Get array element variable. Attach that to cast platform base, Then drag from the pin and search for ev_ activate. What is this doing? I dont really know, but to add other stuff just duplicate the array elepment variable and cast to something else like bp_door. Also add the door to the variable with a picker tool, and use a sequence node to make it go through both codes. Also if its controlled by a boolean then just set the boolean at the end of the code

This is not efficient or good, so use interfaces instead. Interface contains the function name, and in individual bps like door and light, calling the function name may do different things, like opening door or switching on light.
Just a list of functions that each blueprint needs to implement

Create it:
Right click content browser, blueprint, blueptint interface.
Rename the function bt right clicking it.
Now implement the function in the blueprints u want to be activated.
In the bp go to class settings
Under interfaces there is add button. Search for the interface u just made
Now in event graph u can just right click and search for event, and type the name of the function and add it.
Then just add the code after. Like set a boolean or call an event.
To activate the interface, drag from the array element (of previous wring code, delete stuff after)
Abd search for in activate (which is a message for the interface, it should show under the name of the interface)



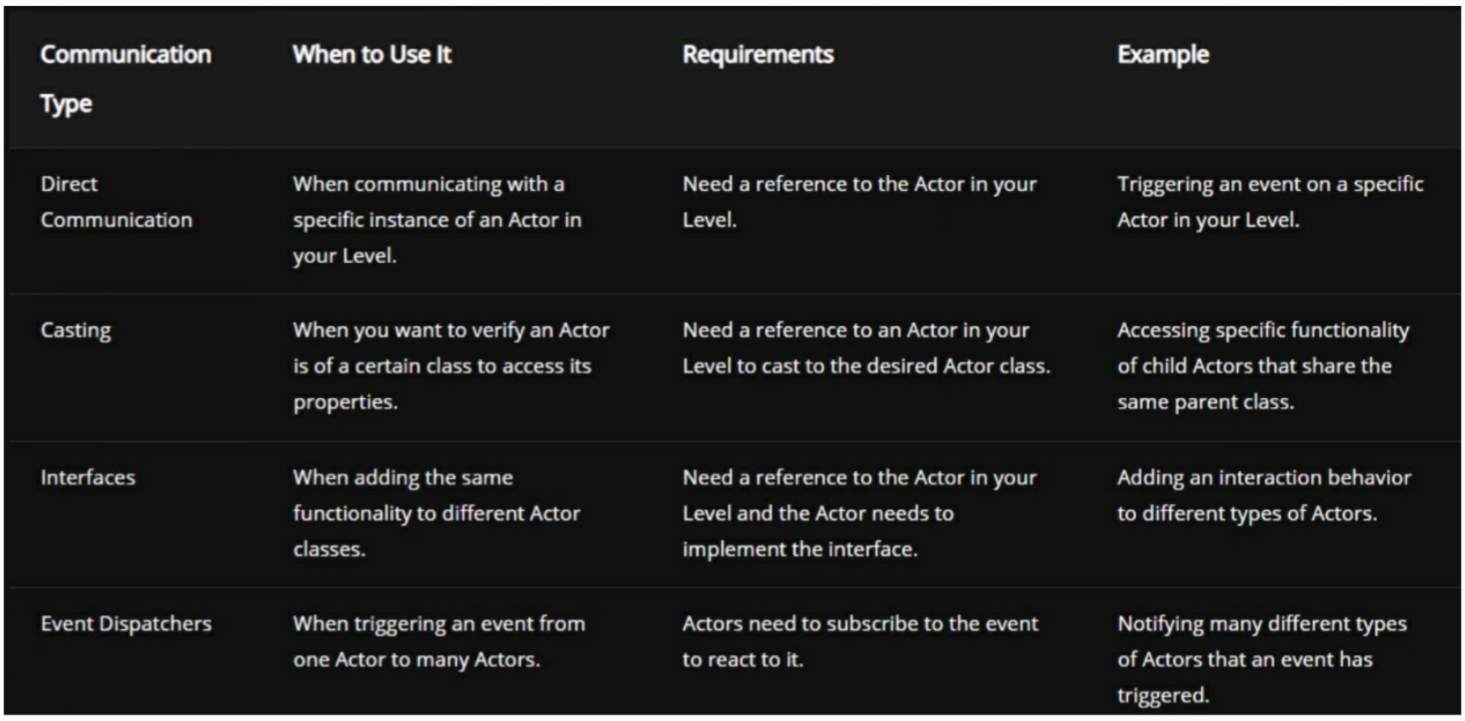
Can use this to let player press e to interact with stuff. Make a box collision in front of the player, enable able to overlap with everything, abd create on component overlap event. First part checks if the thing is in the interface and outputs a boolean, if it is then it will run the next part. Other actor is promoted to variable.



U can use interface functions to return values. By adding an output to the function

15 - Event dispatchers

To make it, make the bp that should activate the rest when triggered, under variable list there is an event dispatchers menu, choose add, name it, and drag it into the event graph.
Call triggers the event and any other bps that are bound to the event. So choose call
In the bp actor u want to bind, need to create a reference to the bp that has the event dispatcher.
So create a variable, in variable type search for the bp containing the event dispatcher and get object reference. Also this needs to be public so u can pick the object in the game making mode.
Get the variable in the graph, drag from it and search for the event dispatcher name and choose bind
Plug event begin play into white arrow and create a custom event for the event pin. So now when room cleared dispatcher is called the new event attached will also be called.



Communication Type	When to Use It	Requirements	Example
Direct Communication	When communicating with a specific instance of an Actor in your Level.	Need a reference to the Actor in your Level.	Triggering an event on a specific Actor in your Level.
Casting	When you want to verify an Actor is of a certain class to access its properties.	Need a reference to an Actor in your Level to cast to the desired Actor class.	Accessing specific functionality of child Actors that share the same parent class.
Interfaces	When adding the same functionality to different Actor classes.	Need a reference to the Actor in your Level and the Actor needs to implement the interface.	Adding an interaction behavior to different types of Actors.
Event Dispatchers	When triggering an event from one Actor to many Actors.	Actors need to subscribe to the event to react to it.	Notifying many different types of Actors that an event has triggered.

16 - Traces

Can add line trace by channel node as a result of pressing a key input like F.
U can also trace by object which has an object type input.
Multi line trace by object has out hits as an array, as it passes through objects and hots the one behind it as well.
To see whats happening set draw debug type to for duration.

17 - Custom actor components

Like a camera, it can be attached to any actor in the level within the bp.
When creating a bp, instead if actor or pawn, choose actor component
Parent of a component is called the owner. The owner is the bp with this component attached to it.
Search for get owner node. Drag from return value pin and search for bind damage, choose the bind event to on take any damage node. And connect the

event that will be called when the owner takes damage
To add the component to an instance in the level select the instance and in the details panel there is an add button at the top. Click it and search for the name u gave to the component
Instead of getting the component, you can search for it abd use an ? Is valid node to see if u should call it or not but using the get component by class node and choosing the comp from the dropdown menu
To add a comp to an actor DURING gameplay by in event graph searching for add (name of component) node