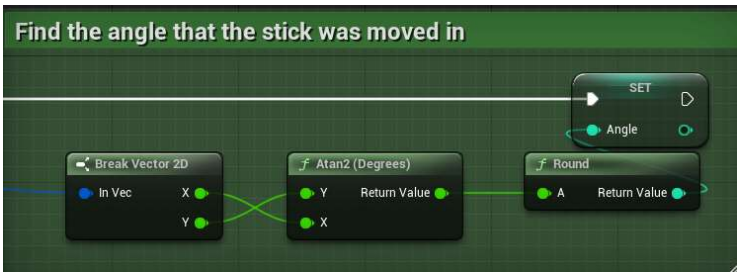
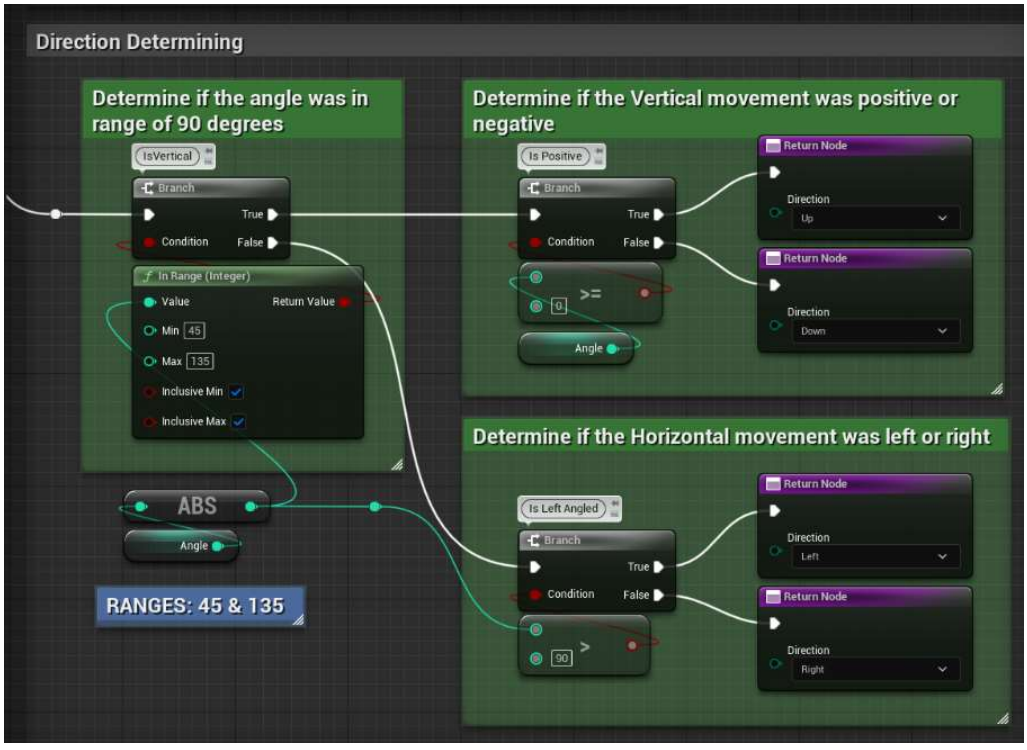
Flex Marks - 1

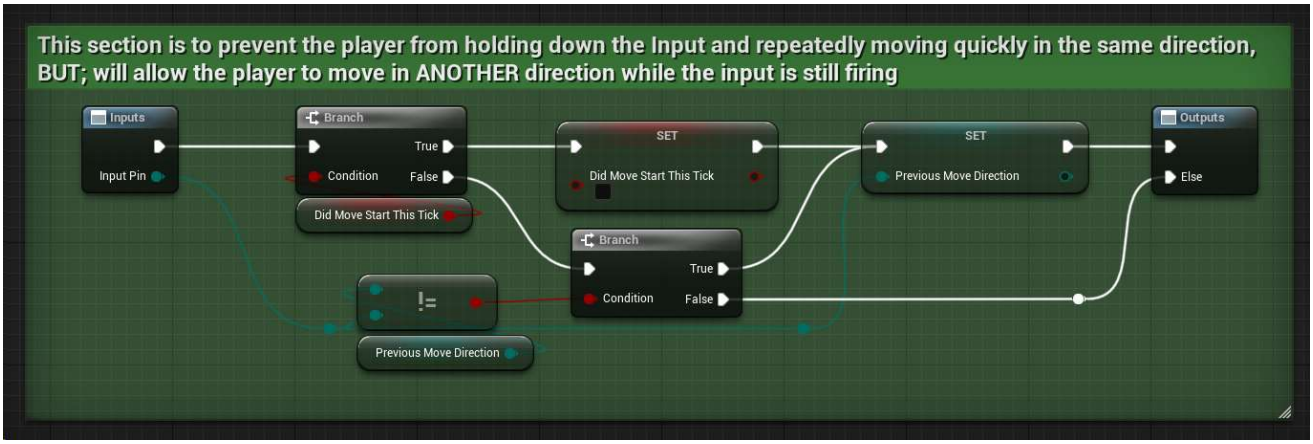
With the way that Epic Games set up Unreal Engine's widget system, it is very hard to make a gamepad (a controller) interact with UI in a consistent, satisfying manner, let alone four of them at once. I spent a considerable amount of time in this portion of this project working on the UI system, allowing gamepads to interact with it and for multiple players to use the UI at once. I named this the MUGWUI system, standing for *Multi-User Gamepad Widget User-Interface*.

MUGWUI is made up of two core components, the base, and the user interactable base.
The base serves as the container and directory for any objects within and tells any interactable base objects inside how to react to a user's inputs.
The interactable base serves to react to the inputs given. Each interactable base has a structure of surrounding objects, which contain what interactable base object will be focused on if the player was to move in that direction.

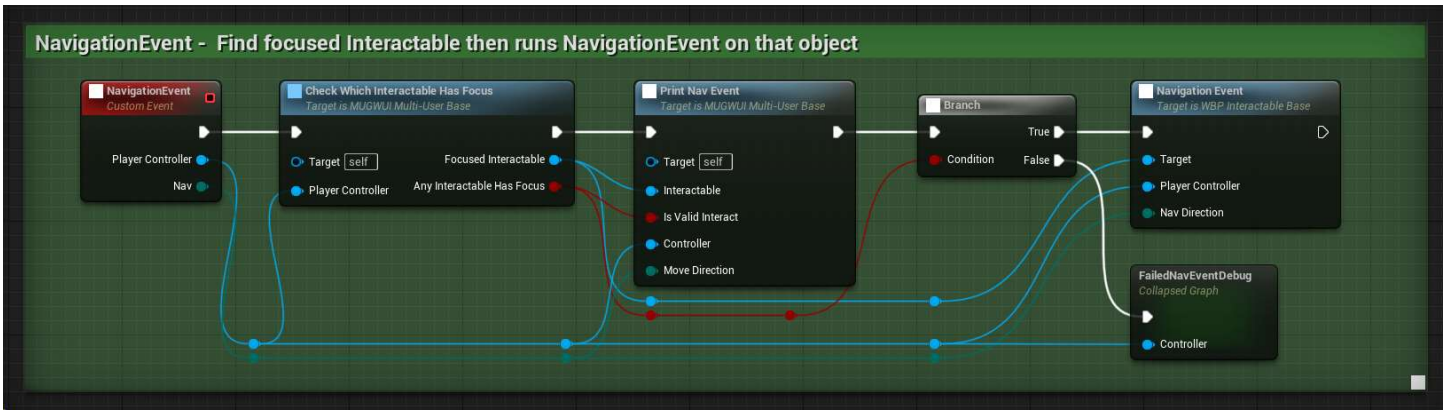
I took advantage of Unreal's new Enhanced Input System to feed input actions into the MUGWUI system. While with a keyboard or d-pad, movement is quite simple, with a joystick however, it is different as you must consider all the entire 360 degrees it can move in.
To determine this, after the user gives a movement input; I use a rounded Atan2 node to calculate the angle. With this angle, it is then determined what direction the joystick is moved in.
It also must be considered if the player has already navigated once, as the Enhanced Input System sends an event every tick the button is triggered.

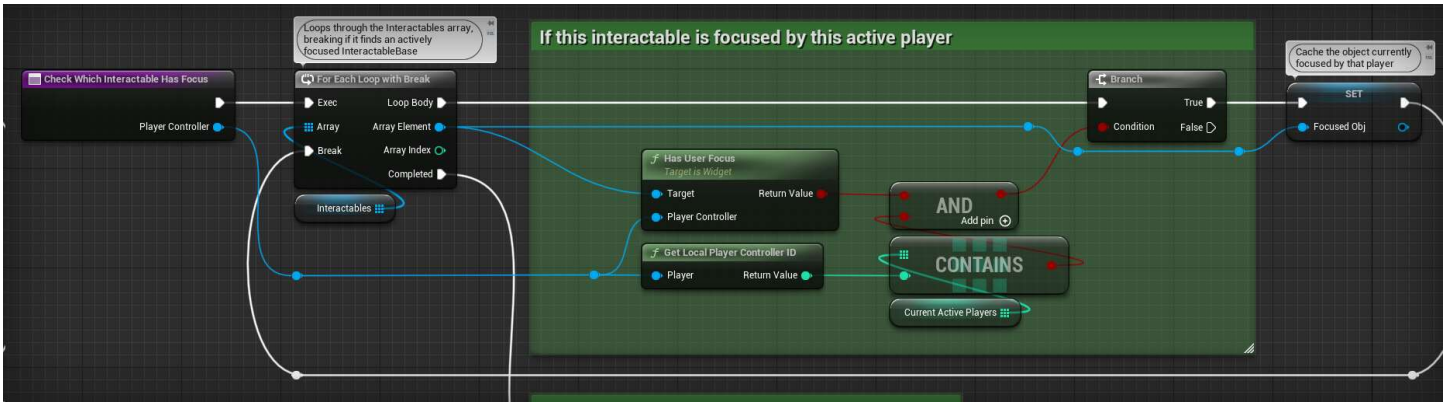




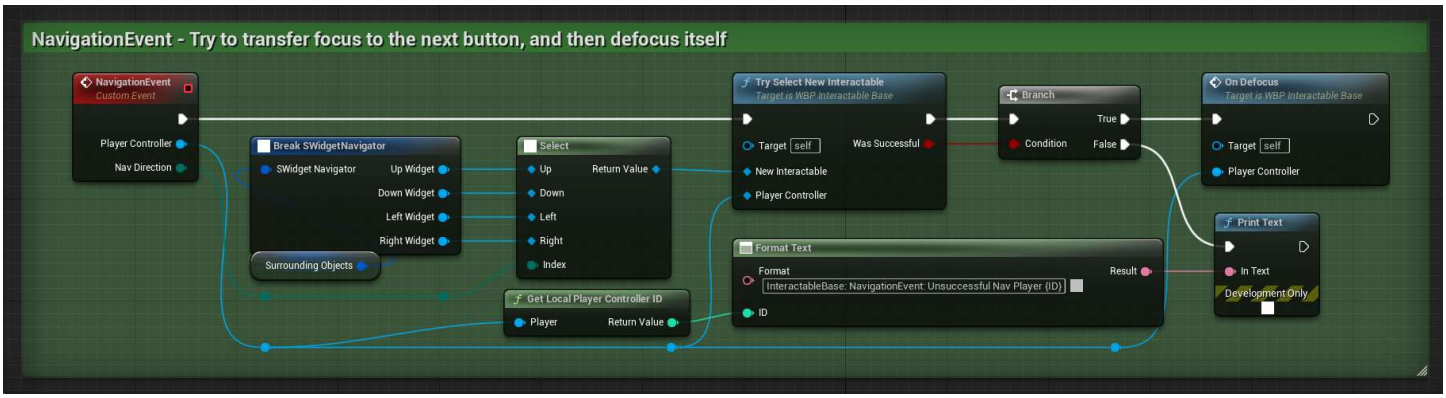


After calculating the direction, it then gets sent to the MUGWUI base, which will iterate through all of its interactable base objects to see which interactable base the player is actively focused on. Then, using the found object, it then calls the navigation event on that interactable base





Inside the interactable base, the navigation event uses the direction calculated in the player controller to determine what of the surrounding objects it should switch its focus to, if the interactable base corralating to the direction is not null and the interactable base is enabled.



With the swapping of widgets, the visuals need to update as well, but the interactable base is just that, an abstract base. This is why I made children inheriting from the class, versatile buttons. These give the interactable base a visual component.