Flex Marks - 2

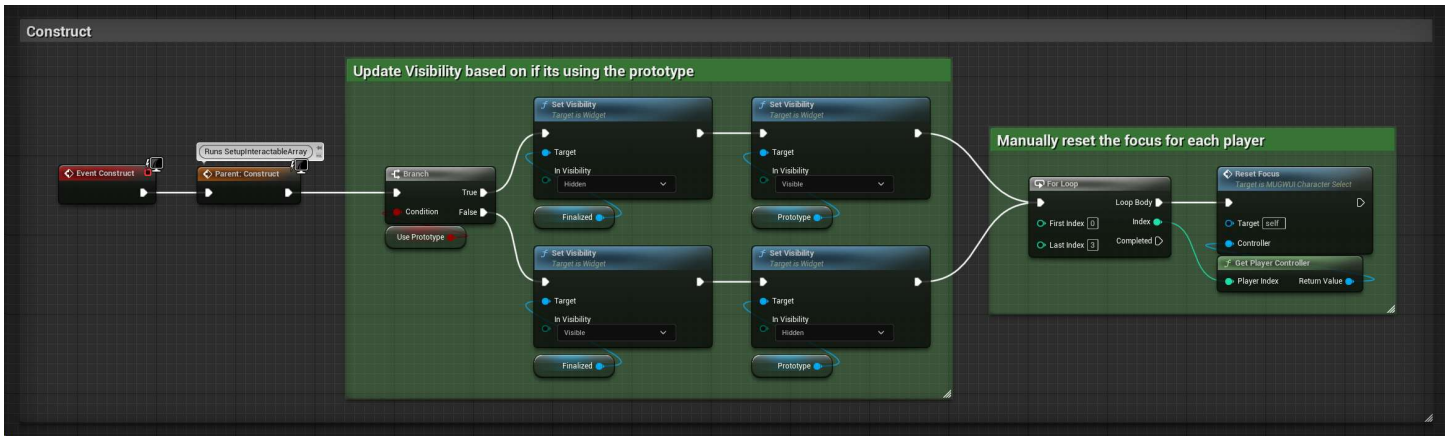
Summary

These past two weeks I mainly focused on the improvement of the UI system. I mainly focused on bringing the CharacterSelect into a 3D widget, fixing the mouse breaking the focus, and adding D-pad and Arrow keys to the input.

3D Space

When we moved the CharacterSelect widget into a 3D space, it came to our attention that the input was no longer reaching the widget. This was because we needed to turn on the ReceiveHardwareInput flag of the Widget component, and fix the issue that came with it; the events CharacterSelect::ResetFocus and CharacterSelect::Construct were being called out of order, which meant that the Interactables array and those like it were empty when the user was initially meant to be sent to the button at the top of their character selection strip.

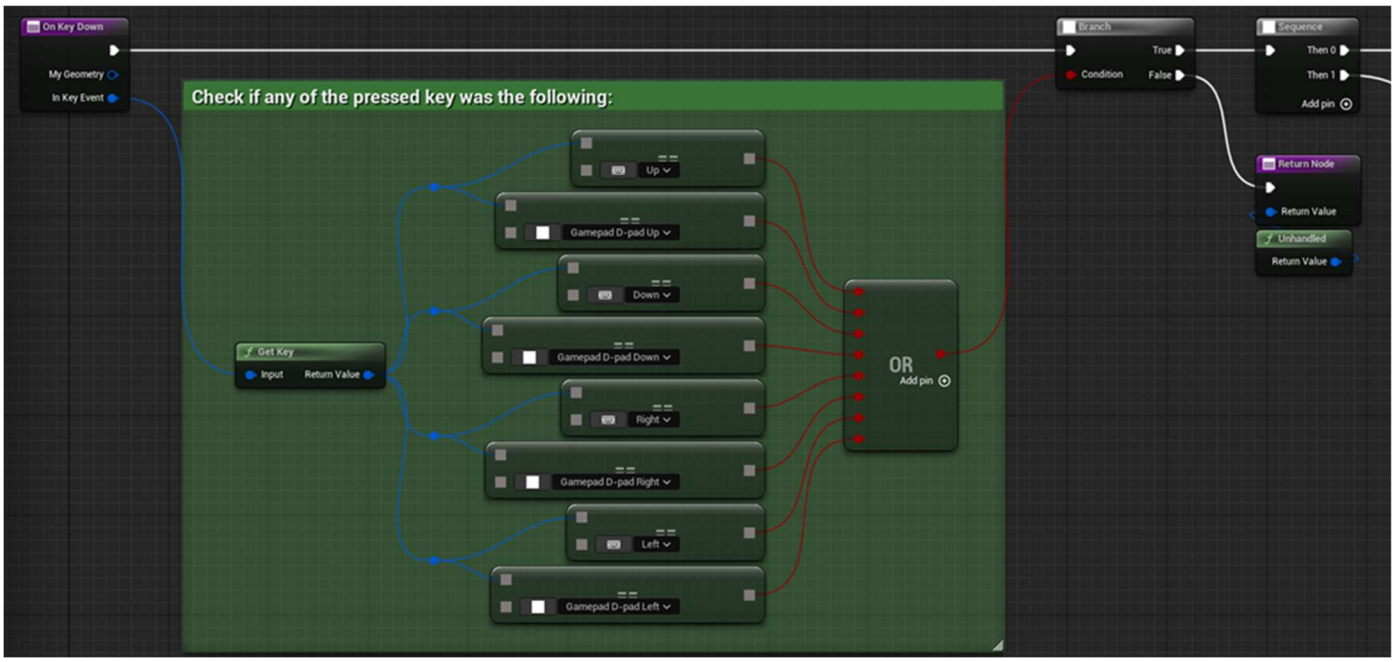
I fixed this by adding an extra call to the CharacterSelect::ResetFocus event at the end of CharacterSelect::Construct, that ensures the arrays necessary for the system – to correctly find a button for the player to focus on – is full.



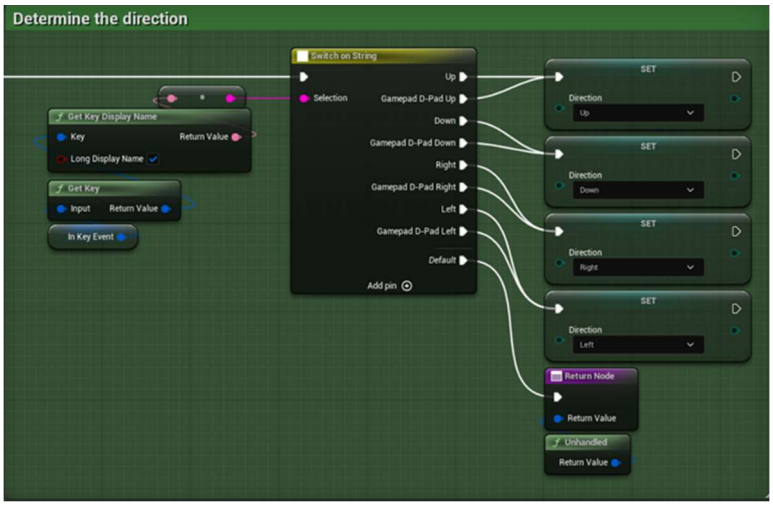
D-Pad and Arrow Keys

When trying to add these inputs to the UI input mapping, we discovered that UI takes input priority over the Enhanced Input.

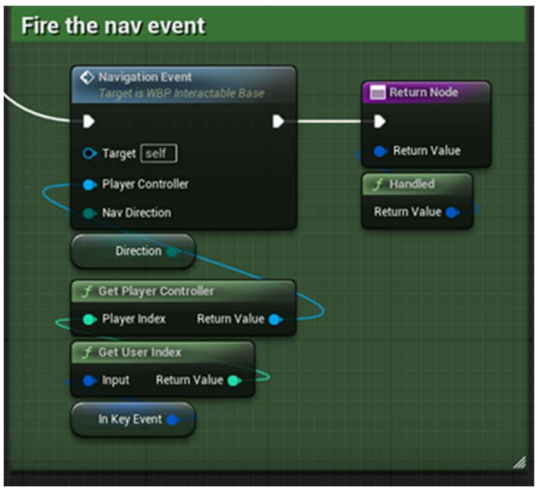
To fix this, we overrode the OnKeyDown event in InteractableBase and check if the key event it received was any of the inputs from the D-Pad or Arrow keys. If it is, it will continue into a sequence that will check which direction the input was (in literal terms of the key's name), and feed that direction into the InteractableBase::NavigationEvent like in PC_Menu. If the received key was not one of the D-Pad or Arrow keys, the event is returned as unhandled, which allows the input to continue to flow into the Enhanced Input System and trigger the Input Actions it has.
Not doing this final step blocks all input from the Enhanced Input System.



Determine if any of the D-Pad or arrow keys were pressed. If true, continue to a sequence of two. If false, let the input pass onto the Enhanced Input System.



Sequence 1: Determine the direction of the movement

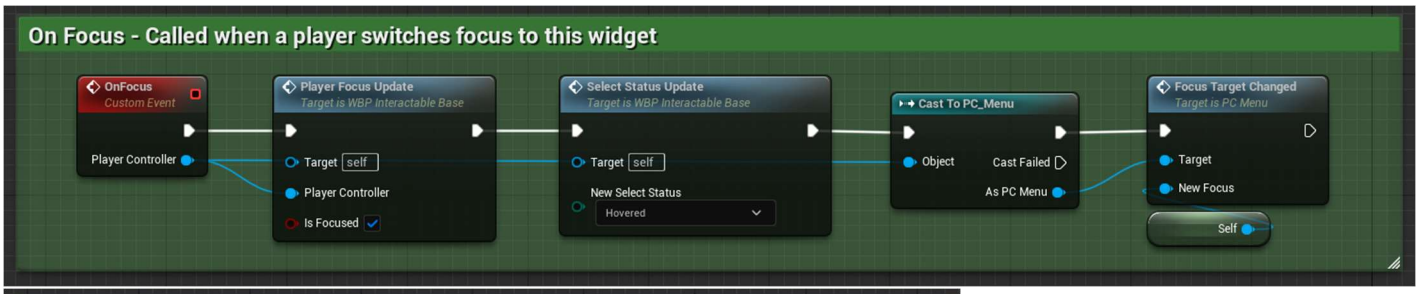


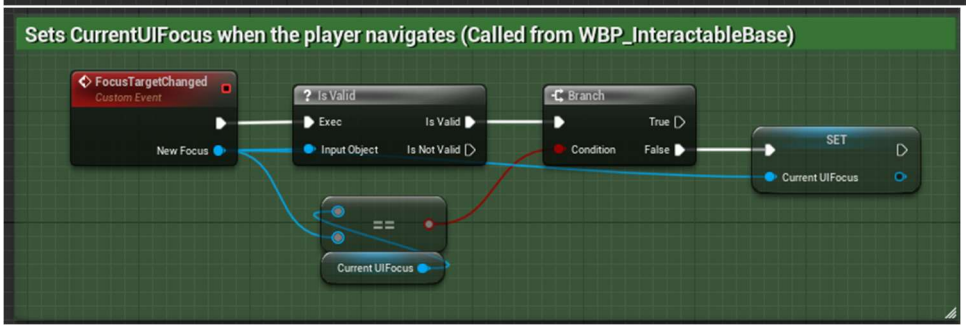
Sequence 2 : Output the direction of the movement into a NavigationEvent call

Focus Bug

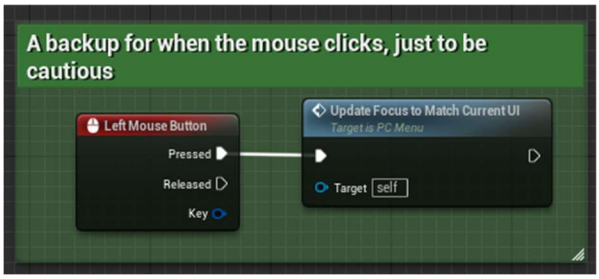
The last thing I fixed was the long existing bug that removed focus from InteractableBase buttons whenever the user clicked on the game screen.

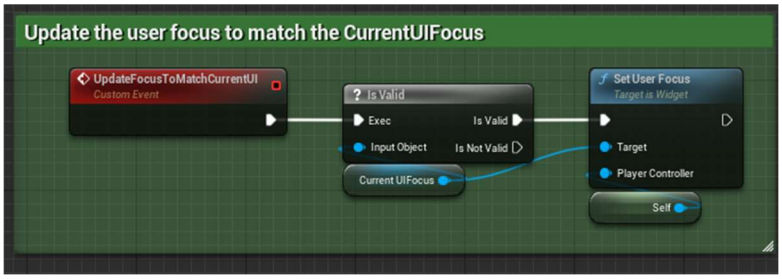
The solution came as adding an extra InteractableBase variable in PC_Menu named CurrentUIFocus, which whenever the player successfully transferred focus to another button, would call from InteractableBase::OnFocus to PC_Menu::FocusTargetChanged with the new Interactable to focus on.



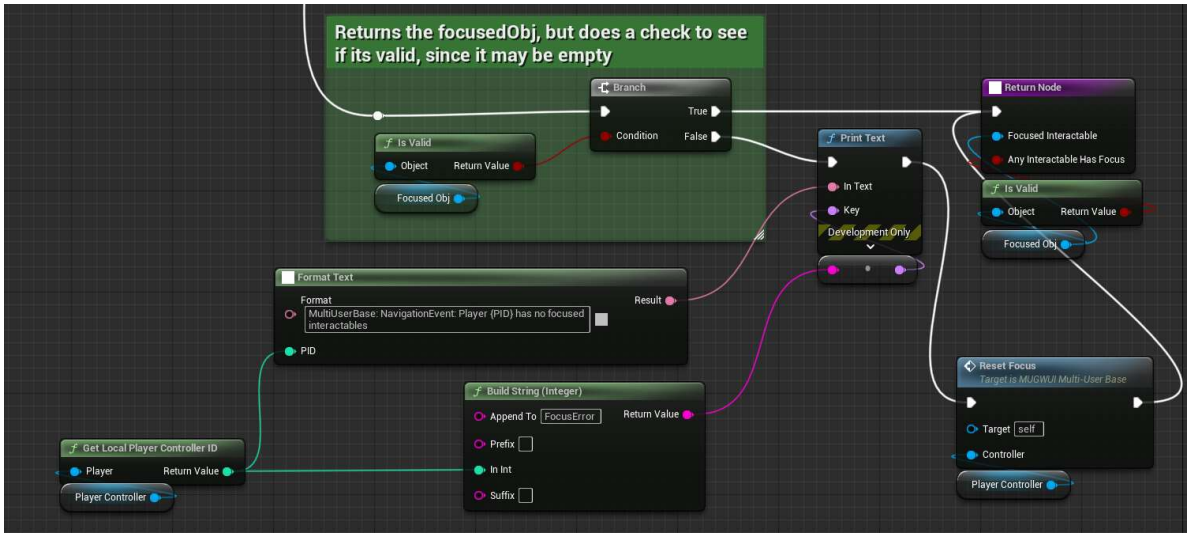


With the CurrentUIFocus updated, every 0.5 seconds OR when the player clicks their mouse, the UserFocus will be reset to whatever CurrentUIFocus is.





We also made changes that in MultiUserBase::CheckWhichInteractableHasFocus, if the player is found to have no InteractableBase focused within the MultiUserBase object instance, it will use MultiUserBase::ResetFocus to ensure it will be focused in the widget it should be.



After Iterating through all the items in the Interactables array, if no object inside the array is found to be focused by the user, it will alert the user and reset the focus.