Report - Individual

In this sprint, the assigned task was creating and finishing the prototypes of core menus in the front-end level, including the main menu, level selection menu, and character selection menu. To do this, a versatile button that different menus could use for their own unique use cases was created. Additionally, another task assigned was to finish the Snowboarder's move-set, which included implementing her distinctive snowboard projectile attack. This added extra complexity, as for realism other moves needed to be aware if the snowboard was currently out as a projectile.

User Interface Updates

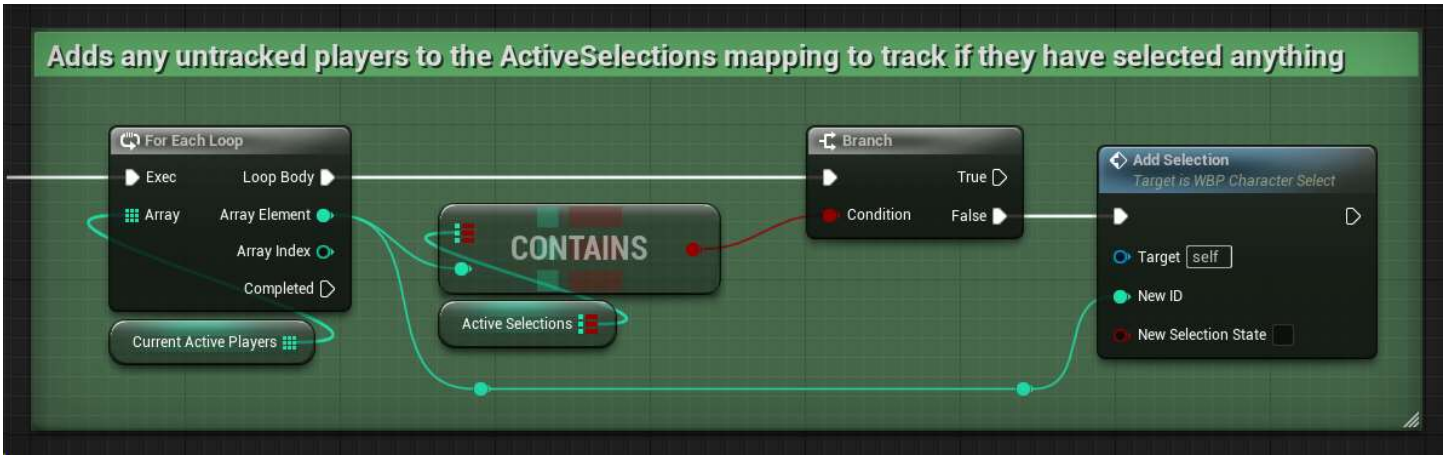
Player Activation

To address the issue of controllers without a character being selected causing the match to break, a solution was devised to initially flag each controller with an "is active" boolean.

When an inactive player interacts with their gamepad, the game will then switch this flag to active, allowing them to interact with the UI and select a character.

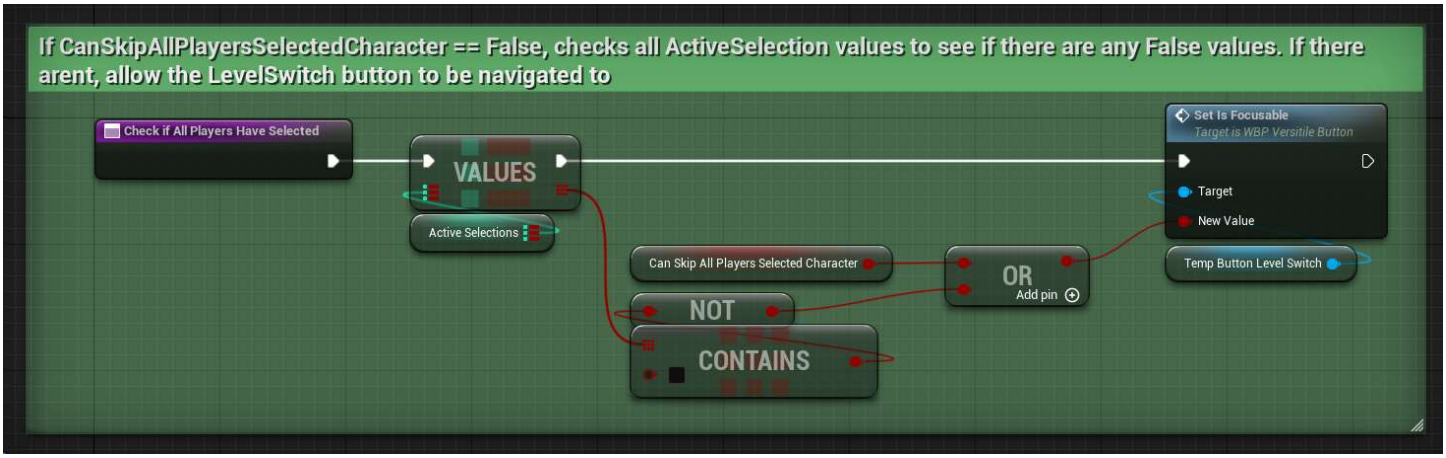
One player activated vs four players activated. The left image shows player one hovering over the button. The right image shows player one has selected Kite as their character, and players one through three are not focused on this button.

ActivateCurrentPlayersOnButtons: Compares ActiveSelections to see if there are any more or less CurrentActivePlayers.



From the previous image: if ActiveSelections length was less than CurrentActivePlayers, adds any untracked players to the ActiveSelections mapping to track if the player has selected anything.

Inactive players are all but ignored when it comes to passing the block that allows transitioning into the match level, ensuring that the game will progress smoothly without being blocked by players that do not exist.



CheckIfAllPlayersHaveSelected: If CanSkipAllPlayersSelectedCharacter is false, checks to see if all ActiveSelection values are false.

Creating a Versatile button

In the earlier sprint, it was noted by a fellow UI programmer that buttons created for the menus lacked flexibility in their implementation. When needing to convey data to its parent widget, each subclass of the interaction base had to be tailored to that specific task, which included re-implementing what was usually similar graphics.

This approach would have led to numerous unnecessary one-use-only subclasses, leading to an increase in file size in its builds. To resolve this issue, a new subclass was introduced from the interaction base. This subclass shares core functionality that would be present in all interaction base subclasses, including visuals and selection capabilities, which show the number and identities of activated players (see the previous section for explanation on activation). The appearance of this new class adapts in response to user actions– hovering over the key, pressing the button through input, and, if enabled, selecting the button.

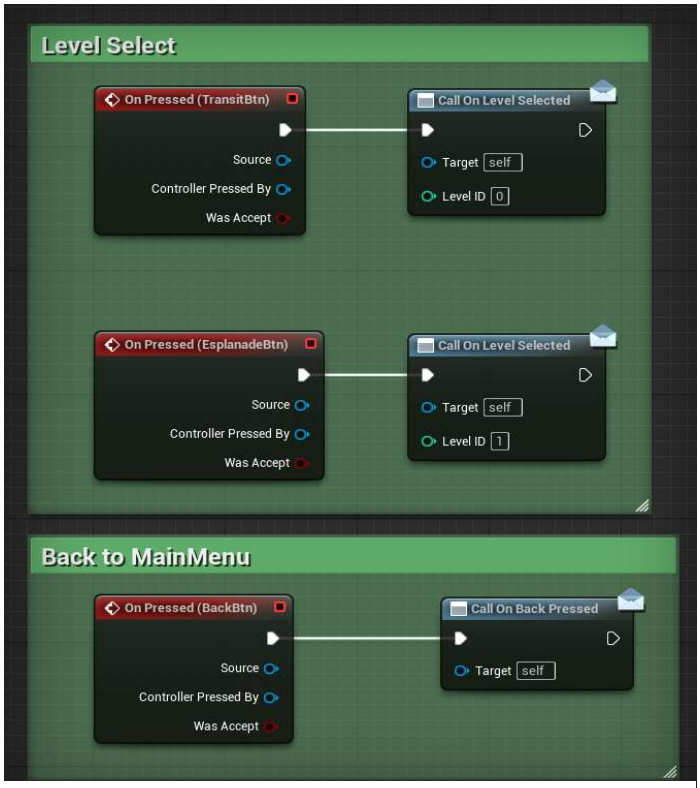
In the parent (multi-user base) widget, one of its jobs is now to figure out how it will respond to a button press event. You can see how this new system is implemented reading the following section.

Menu Updates

In this sprint, one of the primary goals was to develop the main menu and level selection widgets, while also bringing the prototype character selection widget to its full functionality.

After completing the versatile buttons development, the required menus were built with high efficiency. The buttons in the main menu and level selection all share similar functionality that transitions the player between different widgets. In addition, in the level selection screen, the buttons also communicate the level selected to the game mode.

In character selection, the versatile button was used once again. Each button is mapped to a specific character class, instead of being internally imbedded as a variable in a one-use interactable base subclass. When a player presses one of the buttons, the menu will communicate the selected character to the game mode, which will later notify the game instance. This will then allow the transition to the match to assign the chosen character to its corresponding player when the match starts.



Versatile Button events being used to navigate to the next widget, with Transit and Esplanade button press events also communicating which level was chosen.

Versatile Buttons being mapped to their relevant character when the widget has been constructed, in the Character Selection menu.