Flex Marks 3

Added Button Indicators

Why: Players were greatly confused as to what the controls were, and as our games get more complex we are having to reach out to other less expected inputs. Indicators will allow Players to know what buttons do what without having to rely on us developers.

Customization

For reusability purposes, I made sure the following was available for customization:

- The Key text (the text within the circular icon)
- The Descriptor text (the text beside the circular icon)
- The Color of the circular icon

Two Button Indicators which have been set to have a different colored background, key, and descriptor

Throughout the process, I realized that not all inputs can be properly communicated through text, especially since special Xbox inputs are not available as a font. My solution was to give developers the option as a boolean to turn off the Key text and Circular icon, and allow them to fill a Texture variable with the icon. It also allows the developer to chose a custom shadow texture for it.

A Button Indicator with the background hidden and using an image instead of text.

Versatile Buttons

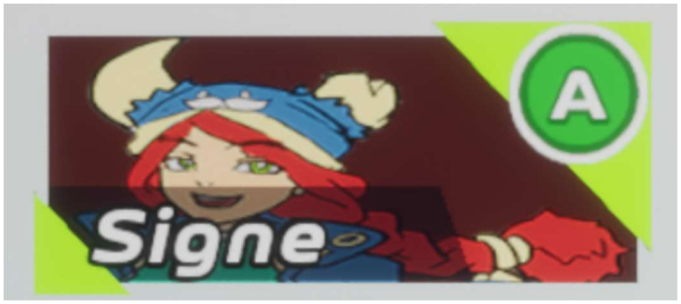
Players seemed to also get confused as to what buttons to hit, and even in some instances what button was focused. To fix this, I added indicators to Versatile buttons that are equally customizable to those used outside of the Versatile button.

These indicators are only visible when the button is actively being hovered by a user.

On top of all basic customization features listed in the section before (except for the indicator descriptor), you can customize the alignment of the indicator as well as an offset to push the alignment.

Main Menu Versatile Button with the Indicator aligned Center-Right with an offset of (X = 100, Y = 50)

Level Select Versatile Button with the Indicator aligned Bottom-Center with an offset of (X = 0, Y = -170)



Character Select Versatile Button with the Indicator aligned Top-Right with an offset of (X = 25, Y = 25)

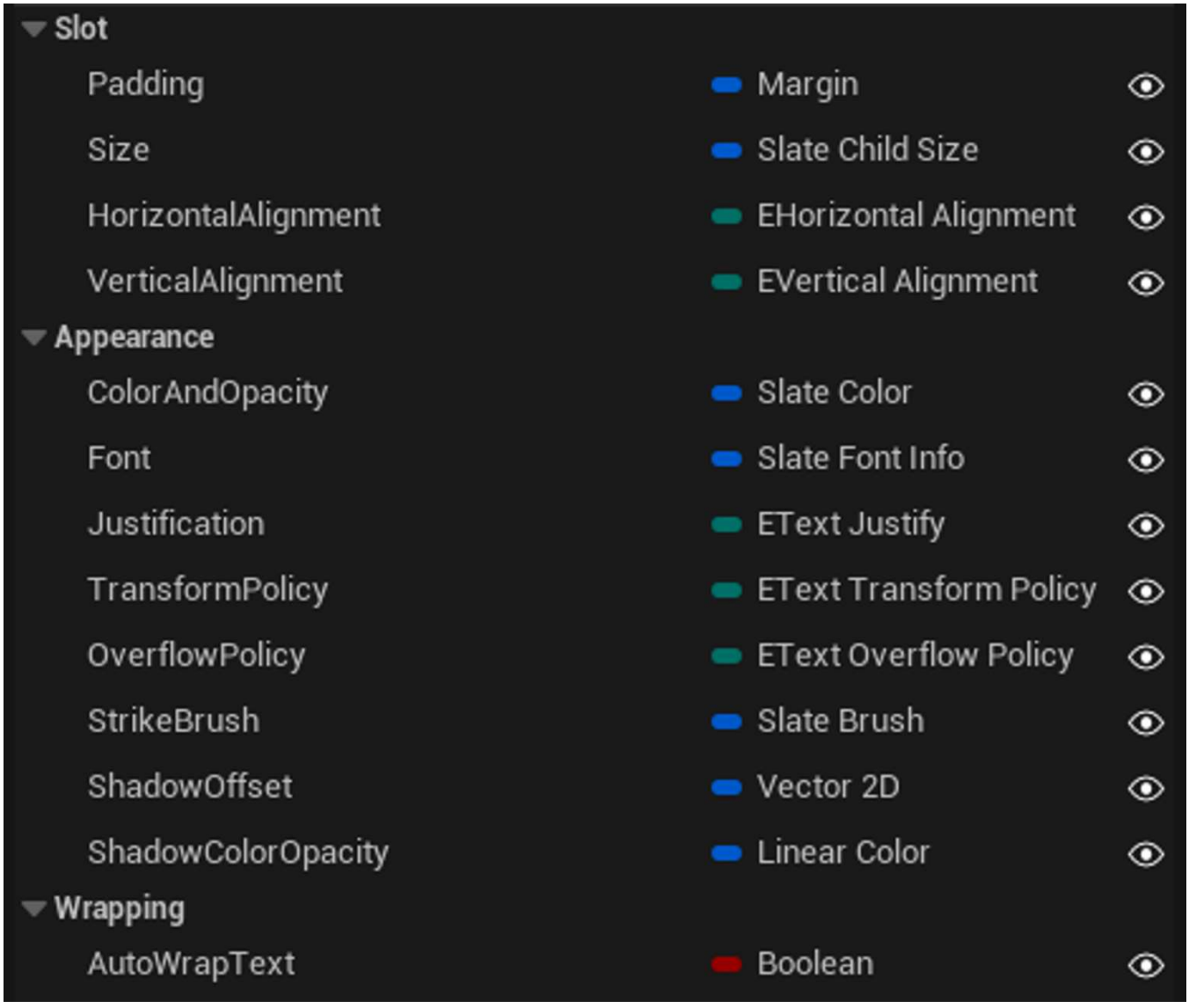
Expanded the appearance customization of VersatileButtons

Why: It was requested to me that we be able to better customize versatile buttons, as it was brought to my attention that most customization has to be done by embedding the Versatile button within an overlay and then customizing the button there.

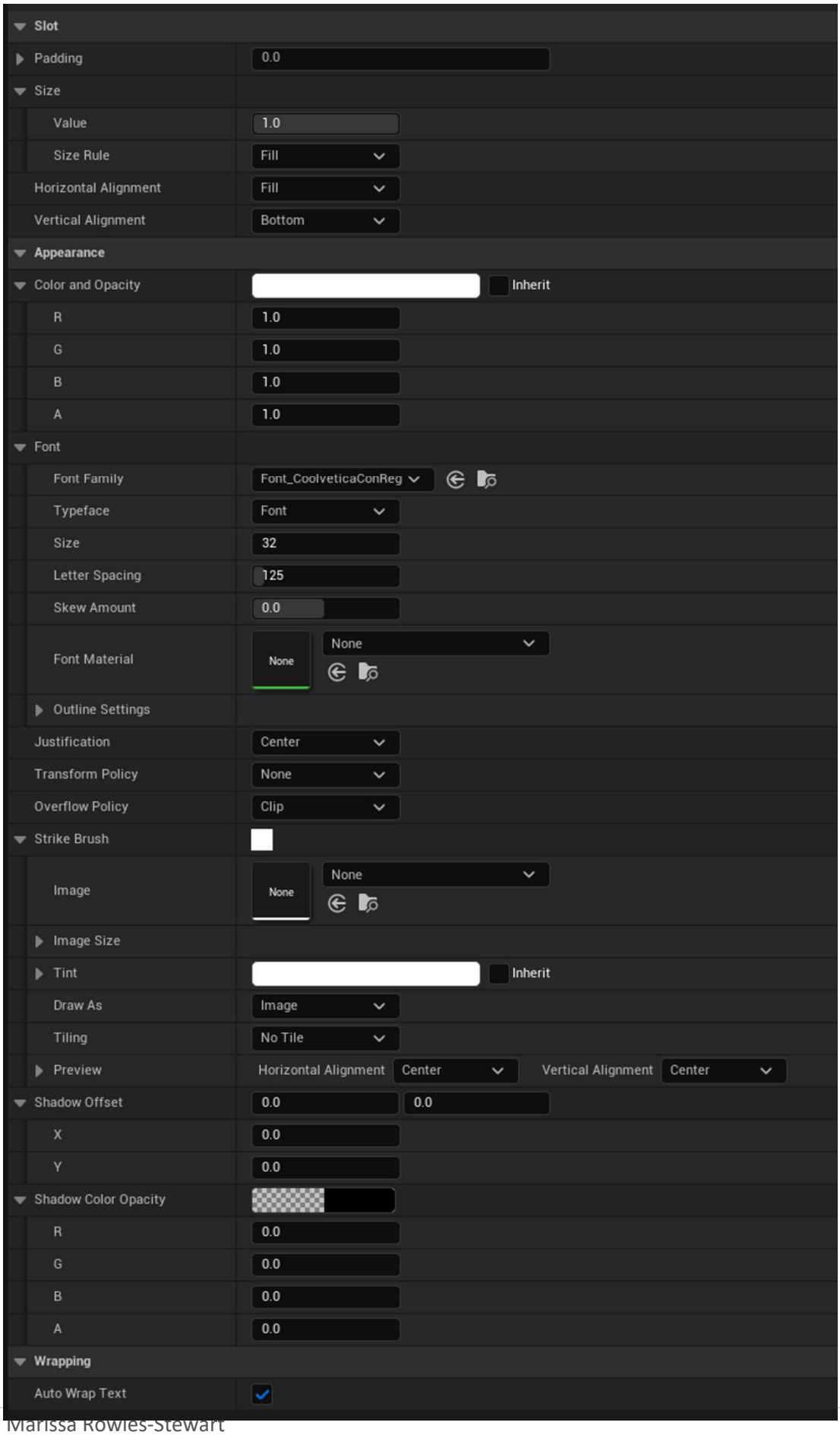
Specifically, I added the ability to customize the texts fonts and create overlays between the buttons background and text.

Font Style Presets via Data Asset

Initially I had Font Styles as a set of Variables within the Versatile Button. During testing, I found that integrating these fonts between other buttons was rather tedious, especially because copying the settings of each variable under the category often caused the engine to crash. I decided to get around this by making a data asset named `DA_InteractableFontStyleBase`, which contains every core variable used within the `TextBlock` Widget element, including variables to change the text's alignment.

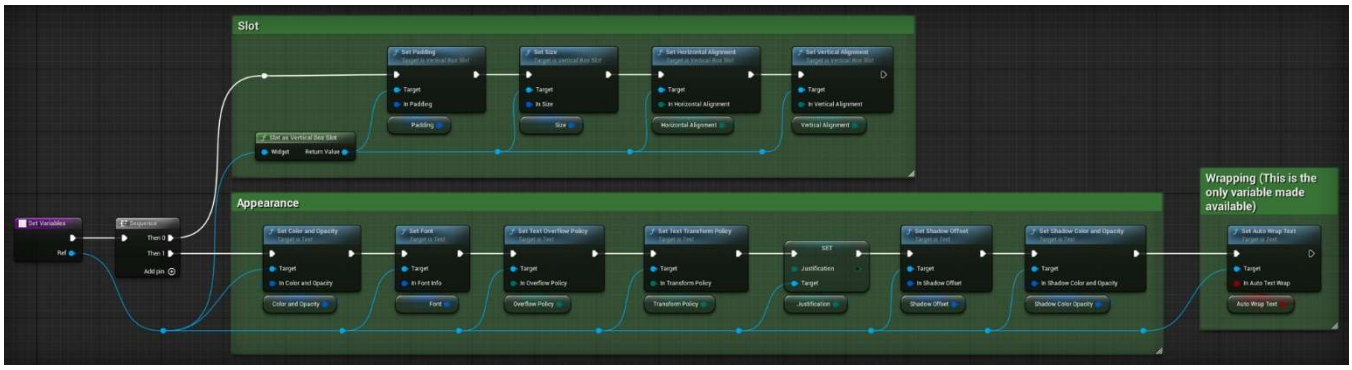


`DA_InteractableFontStyleBase` variables as shown within the class itself

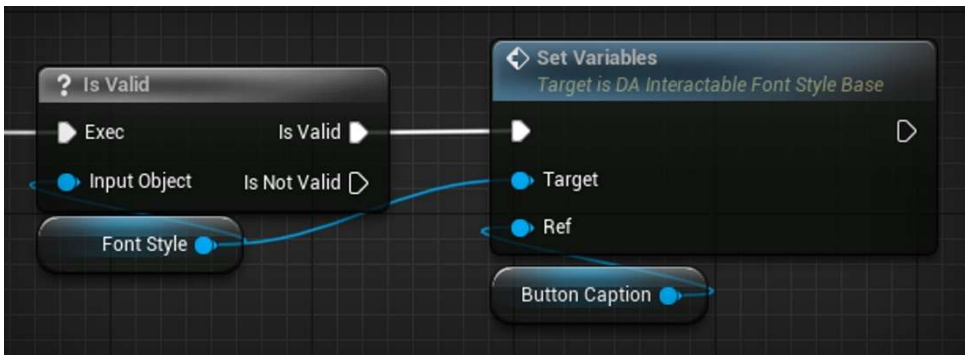

Slot
Padding
0.0
Size
Value
1.0
Size Rule
Fill
Horizontal Alignment
Fill
Vertical Alignment
Bottom
Appearance
Color and Opacity
Inherit
R
1.0
G
1.0
B
1.0
A
1.0
Font
Font Family
Font_CoolveticaConReg
Typeface
Font
Size
32
Letter Spacing
125
Skew Amount
0.0
Font Material
None
None
Outline Settings
Justification
Center
Transform Policy
None
Overflow Policy
Clip
Strike Brush
Image
None
None
Image Size
Tint
Inherit
Draw As
Image
Tiling
No Tile
Preview
Horizontal Alignment
Center
Vertical Alignment
Center
Shadow Offset
0.0
0.0
X
0.0
Y
0.0
Shadow Color Opacity
R
0.0
G
0.0
B
0.0
A
0.0
Wrapping
Auto Wrap Text

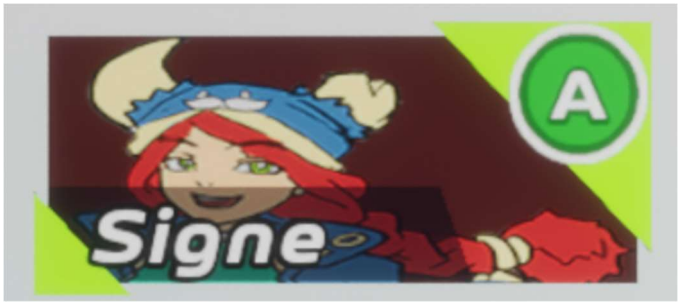
`DA_InteractableFontStyleBase` variables as shown from the `DA_DefaultInteractableFontStyle` DataAsset

I wanted the style to be easily accessible to apply to a text block, so that people wouldn't have to learn this whole new backdoor just to use it on their own content. This is where I discovered that Data Assets can have simple functions. I created a function that applies all of the variables above to a given text block.

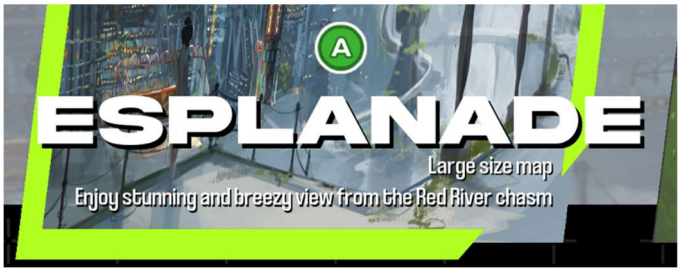


For the Font Style data asset to work, I added it as a variable to the Versatile button (Named `FontStyle`)





A Character Select Versatile button with its text aligned to the bottom-left



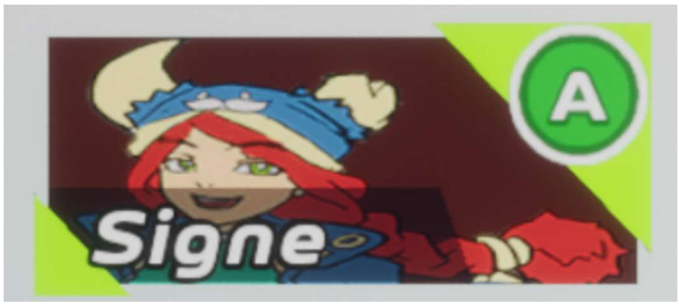
A Level Select Versatile Button with its text aligned to the center and slightly padded from the bottom.

NamedSlots

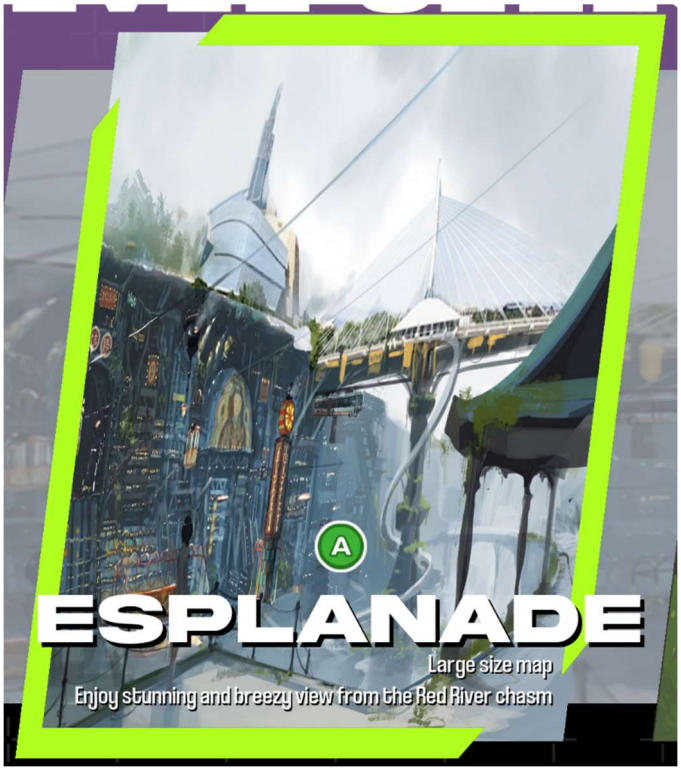
To create the ability to make custom overlays, I used Named Slots. As described by Unreal Engine:

> *This widget allows you to expose an external slot for your User Widget that can be populated with any other widgets and is useful for creating custom widget functionality.* - Unreal Engine

I added two named slots to the Versatile Button; Overlay and SelectionGraphic. Overlay is persistent on the Versatile Button, always visible unless the button itself isn't. Much like the new indicator, SelectionGraphic is only visible when the button has focus, hiding itself when losing focus.



Character Select Versatile Button
SelectionGraphic Slot: Two electric green triangles framing the box
Overlay Slot: Slanted black box that adds a background to the text.



Level Select Versatile Button
SelectionGraphic Slot: An electric green border around the button
Overlay Slot: The image present on the button and the subtext below the level name