How to Begin Designing UI for Virtual Reality Games

Tutorials

- Matt Aspland target game: https://www.youtube.com/watch?v=3OCisqQMaVc
- VR Playground
  - Interactable UI Creation: https://www.youtube.com/watch?v=H5nVjSwM_Uk
  - Pop-Up Wrist Menu: https://www.youtube.com/watch?v=lgsR1g-5HKE

A Standard Beginning

- I created a start menu, and these are the steps I took. They are no different from standard UI creation in Unreal Engine.
- To start, create a Widget blueprint in your blueprints folder in the Content Browser. This can be done by right-clicking inside the Content Browser, navigating to the User Interface submenu, and selecting Widget Blueprint.
- Open it. Make sure you're viewing the Designer panel instead of the Graph panel. This is located, by default, to the right of the viewport, above the Details Panel.
- In the Palette section, search for Canvas Panel. Add it to the widget.
- From here, you can place Buttons by searching for them in the Palette menu, and adding them as children of the Canvas Panel.
  - In the Details panel for the buttons, I recommend changing the colors to better assist with readability. You can change their colors when clicked on, hovered over, or their normal appearance.
  - You can also add events to the Blueprint Graph Editor from here as well. We will do this later.
- To add a Text Box to the Buttons, simply search for "Text" in the Palette panel. Drag the corresponding Text element onto the button, creating a Text element as a child of the Button element.
  - You can change the displayed text in the Text details panel, as well as how it is aligned to the button.
- I created two buttons. One labelled "Start Game," and the other labelled "Quit Game."
- Select your first button. For me, it was the "Start Game" button. In the Details panel, find and add the "On Clicked" Event. By default, it is the bottom of the panel.
  - Now, your viewport will switch to the Graph Editor. If it did not automatically switch, this is located to the right of the Designer Editor.
  - Here, you can edit the functionality of the button to enable it to do things within the game.
  - Since it was a simple "Start" button for me, I just created an "Open Level" node, and selected the level I wanted to open when the player clicked the button.
    - If you type this into the node manually, make sure that your level name matches the name of your level exactly.

        - For my “Quit Game” button, I created a “Quit Game” node. I left the quit preferences the same.
- From here, you can proceed to either “Enabling a Floating Menu for VR,” or “Attaching a UI Menu to the Player’s Wrist.”

Enabling a Floating Menu for VR

- This tutorial will guide you through the steps of placing a permanent UI interactable menu in the VR level.
    - It will not be able to be moved.
    - The player can interact with the menu by using either hand. This will be detailed in a different section.
- With your UI menu created, it’s time to place it in the world so the player can interact with it.
- Create a new Actor Blueprint in the Content Browser. Right click inside the Content Browser, navigate to Blueprint Class and enter the option selection, select Actor. Open the asset.
    - I named this asset “BP_WidgetSpawnUIElement”
- In the Components menu, add a “Widget” component. Select it, and navigate to its Details panel.
- We need to adjust the collision of the Widget component.
    - Set it to Custom.
    - Set the Object Type to World Dynamic.
    - Ensure 3DWidget is blocked in the Collision Responses table.
- Once collision is set up correctly, navigate to the “User Interface” section. Ensure that the Widget Class linked to the Actor object is the Widget Blueprint that was previously created.
- Now, place it into the world. From here, you can continue with “Interacting With a VR Menu.”

Attaching a UI Menu to the Player’s Wrist

- This tutorial will walk you through the process of having a permanent UI menu attached to the player’s wrist (developer’s choice).
    - It will only be visible if the player angles their UI wrist towards their headset’s front-facing camera.
    - The player will also be able to use their non-UI hand to interact with the menu using debug lines. This will be detailed in a different section.
- With your UI menu already created, open you VR character controller. By default, this is the VRPawn.
- In the components menu, find your MotionController(Direction) component. If this component is not available, add it to your pawn.
    - This would be, by default, labelled as MotionControllerLeft or MotionControllerRight.
    - For my project, I used my left motion controller. MotionControllerLeft. All instructions going forward in this section will assume the left hand is used, but it will work the same on the right hand if that was your choice.
- Select your hand component of choice. You’ll need to add some child components, listed below.
    - By default, HandLeft (B Mannequins XR) should be added. If not, add it in.
    - Add an Arrow (Arrow Component).
    - Add a Widget (Widget Component).

        - Do not add the Widget Interaction Component to the hand you attach your UI menu to.
- From here, you'll need to angle the Arrow component to be perpendicular to the hand component. Facing upwards. This may need to be fine-tuned to have it work properly.
- In the Widget component, find the User Interface tab in the details panel. In the Widget class, add your created widget blueprint.
- Now, go to the event graph. Create a function to detect your hand widget. This function will detect the direction of your Arrow component in relation to your headset camera.
- Drag this function onto the main event graph and attach it to the Event Tick. From here, you'll need to do two things:
    - Event Tick Functionality: Copy and paste this functionality and attach it to your Event Tick. Make sure the name of the function matches, and add variables as needed.
    - DetectHand Function: Copy and paste this functionality into your DetectHand function. You'll be using references to the Arrow and Camera components, not creating separate variables entirely.

Interacting With a VR Menu

- The final steps can be accomplished in the VR Pawn blueprint.
- In the Components panel, find the two "WidgetInteraction(Left/Right)" components.
    - If they are not there by default, create each of these components as children of their respective "MotionController(Left/Right)Aim" component.
- Go to the Details panel.
- Under the Interaction submenu, ensure that each WidgetInteraction() component has a different Pointer Index (right hand = 1, left hand = 0), and set the Trace Channel to WorldDynamic.
    - Adjusting the Interaction Distance can assist with interacting with the menu if it is farther away from the player.
- By default, this will enable the trigger buttons on your headset controllers to act as a standard mouse left-click.

Helpful Tips

- To enable debugging, go to the details panel of the WidgetInteraction() components in the VRPawn blueprint. In the Debugging submenu, enable Show Debug. This will project a line trace from the index of your controller to the end of the Trace Channel (settings found in the Interaction submenu).
    - You can also adjust the colors of each hand, to better see which controller has which functionalities enabled.
    - You can adjust the thickness of the debug traces if they are too difficult to see with the headset.

Troubleshooting

- If your menu widget is not appearing in-game, try these steps.

- Change the material of the widget in the Actor that spawns the widget from the default one-sided material to the opaque material.
    - “Widget3DPassThrough_Opaque” is the material label.
    - This worked for me.
- Ensure the Canvas Panel is set to “Enabled” and “Visible” in the Behaviors submenu.
- Increase the Draw Size of the user interface. This is done in the actor blueprint details menu, under the User Interface submenu.