Post-Mortem: VR UI Interaction Not Working

Sara Naranjo

Link to Github: saranaranjo4430/LeafCatcherVR

Introduction

For this project, I developed a VR game called Leaf Catcher VR, where the player must catch falling leaves using baskets attached to their hands. The objective is to collect as many leaves as possible within a time limit. The game includes UI menus, allowing the player to start the game, pause, resume, and return to the main menu.

A major challenge I encountered during development was making the UI interactive using VR controllers. Initially, the UI buttons were not responding when trying to interact with them using the controllers. The issue was critical because it prevented the player from starting the game or navigating menus, making the entire UI system unusable.

Even after implementing a fix, clicking on buttons in VR is still a bit tricky. Sometimes, clicking doesn't register immediately, but adjusting the head position slightly or pointing at the button from a different angle helps. The buttons do change color when hovered over, which helps indicate interaction.

Problem: VR UI Buttons Not Responding

When testing the game in VR, I noticed that:

- The buttons were visible in the UI, but they didn't react when I pointed at them with the controller.
- The button highlight effect did not appear, meaning no interaction was being detected.
- Clicking the Trigger button on the controller did nothing.
- The issue occurred across all UI menus, including the Start Menu, Pause Menu, and Game Over Screen.

This issue made it impossible for players to start the game, pause, or quit, effectively breaking the gameplay flow.

Attempted Solutions

To solve this problem, I tested multiple approaches, trying to understand what was causing the buttons to be unresponsive.

1. Adding Box Colliders to the Buttons

Idea: I initially thought the problem was caused by the buttons not detecting raycasts properly, so I added Box Colliders to all UI buttons.

Result: The colliders were added successfully, but the buttons still did not respond to the controller ray.

2. Changing the Canvas Render Mode

Idea: The default Screen Space - Overlay setting for UI does not work in VR because it is meant for flat screens. I changed the Canvas Render Mode to World Space so that the UI could exist in 3D space.

Result: The UI became visible in VR, but button interactions still did not work.

3. Adding the Tracked Device Graphics Raycaster

Idea: I discovered that VR UI elements require a special Tracked Device Graphics Raycaster to allow interaction using XR controllers. I added this component to the Canvas.

Result: This allowed the UI to detect ray interactions, but buttons still didn't register clicks.

4. Replacing the Event System

Idea: The default Event System in Unity uses Standalone Input Module, which is meant for mouse and keyboard input. Since VR uses the new Input System, I replaced the Standalone Input Module with Input System UI Input Module in the EventSystem.

Result: The UI started detecting pointer interactions, but the buttons still didn't work.

5. Assigning the UI Layer in XR Ray Interactor

Idea: The XR Ray Interactor (the component responsible for VR ray interactions) needs to explicitly target UI elements. I checked the Interaction Layer Mask and noticed that UI was not included. After adding the UI layer, I made sure the XR Ray Interactors on both Left and Right Controllers were set to interact with UI.

Final Result: The UI finally became fully interactive, and I was able to highlight and click buttons using the VR controllers.

Final Solution

The problem was ultimately solved by implementing all the following fixes together:

- Changed the Canvas Render Mode to World Space.
- Added a Tracked Device Graphics Raycaster to the Canvas.
- Replaced the default Event System module with Input System UI Input Module.
- Assigned the UI Layer in the XR Ray Interactor's Interaction Layer Mask.
- Ensured both Left and Right Controllers had an XR Ray Interactor component.

After making these adjustments, I was able to interact with the UI using the VR controllers. Buttons highlighted correctly when hovered over, and clicking the Trigger button properly activated button functions like starting the game or pausing.

Conclusion

This issue taught me a lot about how UI works in VR and how it differs from traditional UI on a computer screen. Unlike normal UI interactions that use a mouse cursor, VR requires proper ray interactions, event handling, and input system adjustments.

This experience also reinforced the importance of step-by-step debugging. By systematically testing different solutions (instead of changing everything at once), I was able to narrow down the exact causes of the issue and apply the right fixes.

Even after implementing the correct fixes, clicking on the buttons in VR is still a little challenging. The button highlights when hovered over, but clicking sometimes requires moving the head slightly or adjusting the controller's angle. This is something that could be further optimized in the future.

In the future, I will approach UI interactions in VR with a clearer understanding of the required components, making it easier to implement interactive menus, buttons, and other UI elements in virtual reality.