Corebound

Development Post-Mortem

Unity 101 Final Project | Unity 6000.0.23f1

Problem: floor labels competing with modal UI

Corebound is a 3D physics game in which a sphere collects eight energy cores and uses a directional boost to cross platform gaps. Labels on the floor identify sectors, checkpoints and the reactor. The game also has screen-space pause and victory panels.

During visual verification of the Windows build, floor lettering remained visible in the same area as the modal text. For example, the reactor label competed with the "STATION RESTORED" heading. The pause panel had a similar problem with checkpoint and sector labels. A recent respawn message could also remain visible when the game entered the victory state. The buttons worked, but the interface communicated several different states at once.

Investigation and an incomplete hypothesis

The initial automated run passed the gameplay checks, including pause, victory and scene transitions. This narrowed the issue to presentation: a panel could be active and functional while still being difficult to read. The rendered screenshots were therefore checked separately from the state assertions.

The floor labels were created in ProjectBuilder.WorldLabel using legacy TextMesh and its font material, while the HUD was created on a screen-space Canvas. The first working hypothesis was that changing the world text implementation would be sufficient to resolve the overlap. The labels were converted to text on World Space Canvases, keeping their position and orientation on the deck.

The next screenshots showed that this change alone was not enough. Floor lettering was still visible through the modal card background and continued to compete with its heading. This ruled out the assumption that changing the rendering approach would fully solve the readability problem. The world signs did not need to remain visible while the player was reading a pause or victory panel, so their visibility needed to follow the interface state.

Solution

GameHUD now caches the World Space Canvases in Awake. During its updates, it enables the floor-label canvases only when the game is neither paused nor won. Resuming play restores the labels; opening either modal panel hides them. The scene geometry can remain as a backdrop without additional floor text competing with the interface.

The same state rule is applied to temporary gameplay messages. GameHUD displays an empty message while paused or won, and GameManager.TryFinish clears its stored message when victory begins. This prevents a previous "Signal lost" message from appearing above the victory panel. The persistent Quit button remains outside the modal panels and is placed last in the HUD hierarchy so it stays accessible.

Result and lessons

Visibility follows the game state

```
label.enabled = !g.Paused && !g.Won;
message.text = g.Paused || g.Won ? "" : g.Message;
```

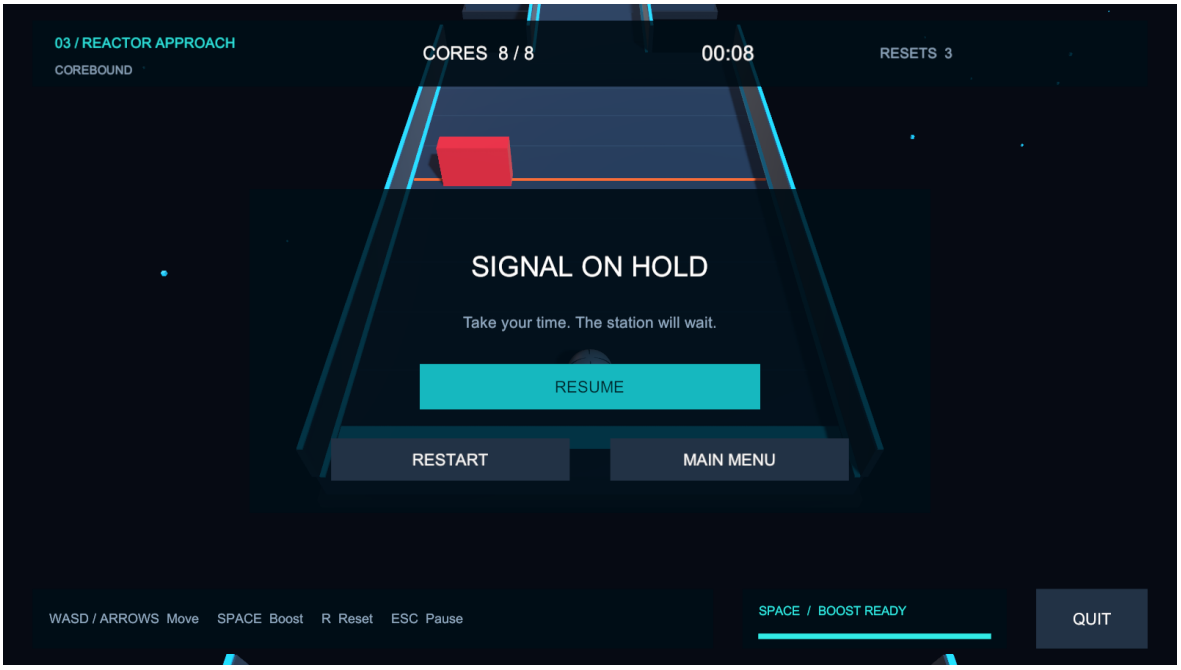


Figure 1. Final pause state from the running Windows player. Floor labels and temporary messages are hidden; Resume, Restart, Main Menu and Quit remain available.

Verification and result

The scenes were regenerated after changing the floor-label construction, and the Windows player was rebuilt after the final HUD changes. The opt-in verification runner exercised menu loading, physical boost flights, collision and collection triggers, checkpoints, pause/resume, victory, restart and return to the menu. All 28 checks passed in the final recorded run. The screenshots of the running player's pause and victory states were also inspected: the floor lettering and obsolete respawn message no longer competed with the modal text.

The automated checks are useful evidence of functionality, but they do not represent a complete human keyboard playthrough. Some checks reposition the sphere to isolate trigger behavior. Movement feel, difficulty and perceived audio loudness still require a manual review before submission.

What this taught me

A correct gameplay state does not guarantee a clear interface. World decoration, temporary messages and modal panels need a shared visibility policy. A rendering change can improve one part of a problem without resolving the player's actual experience. For future projects, visual checks should accompany automated state checks, especially when pause, victory or other overlays change what information is relevant.

Project evidence

Implementation: Assets/Editor/ProjectBuilder.cs, Assets/Scripts/GameHUD.cs, Assets/Scripts/GameManager.cs. Verification: Verification/results.json, Verification/TEST_REPORT.md and the final pause/victory PNGs.