Development Journal - Infernal Cleaner

Week 1

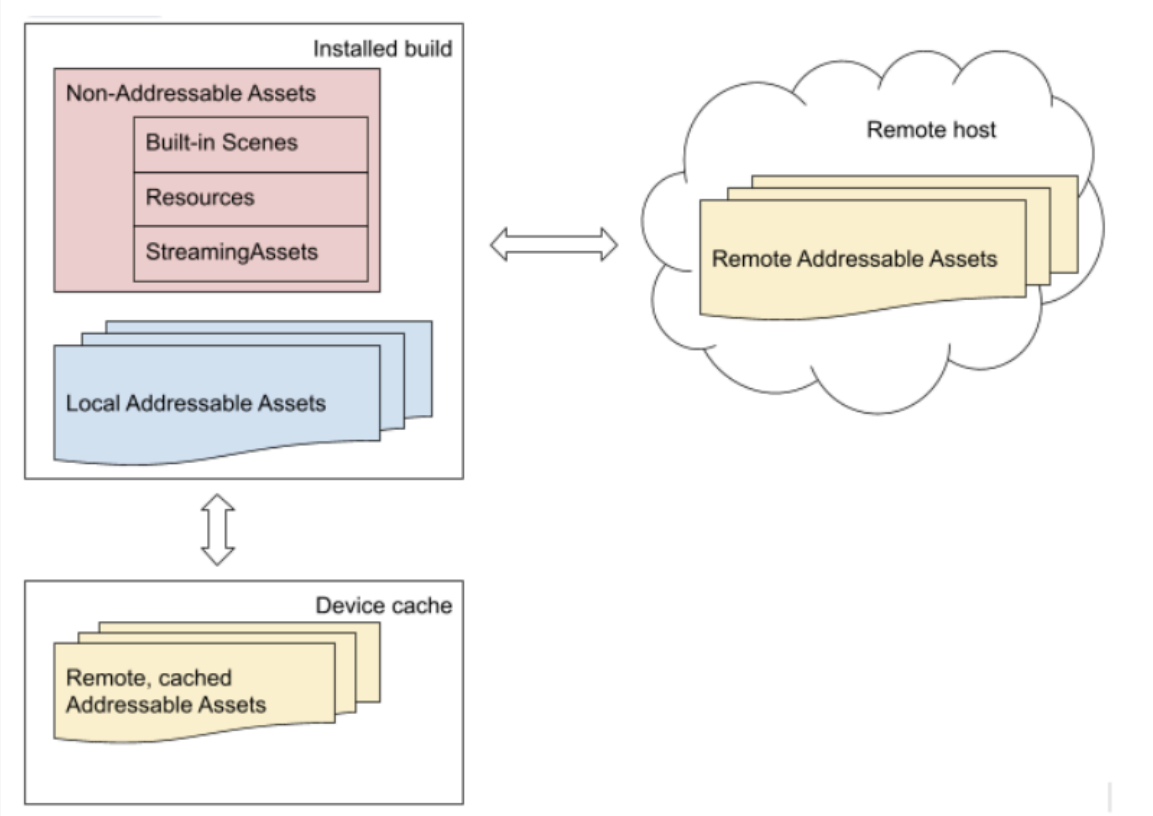
We encountered an issue with Git LFS corrupting Unity scene files during merges. To resolve this, we created a new repository and capped our scene sizes to under 100MB. This prevented excessive file sizes from causing conflicts and improved project stability.

I've identified the cause of the build time issue -- we were creating too many shader variants of the master_paintable_shader which causes a significant increase in shader compilation time. I have deleted many of these, and we will need to set up the materials again.

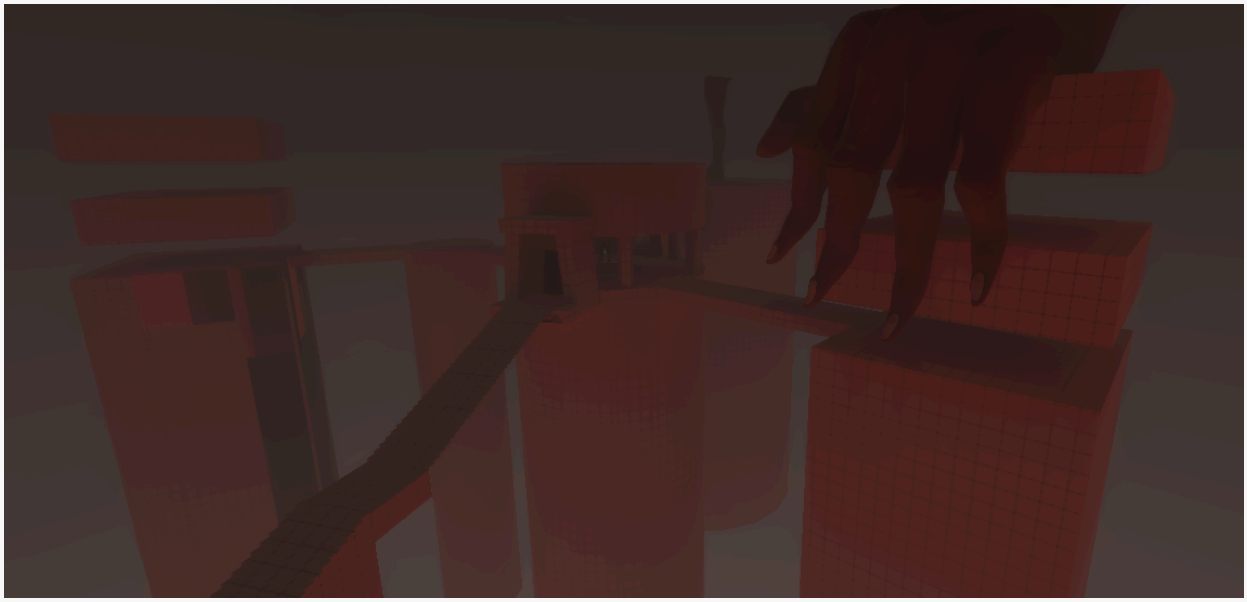
- Scene change triggers
- Scene partition using "neighbours" to determine which scene addressable to load and unload when triggered
- Save disabled save/load for now, since too many things changed. It should not take too long to re-introduce the feature and I was happy with how it worked previously (in terms of save file size, load / save speed).
- Wrote a scriptable object that 'records' all of the scene addressable used and store into an array that can be referenced in runtime.

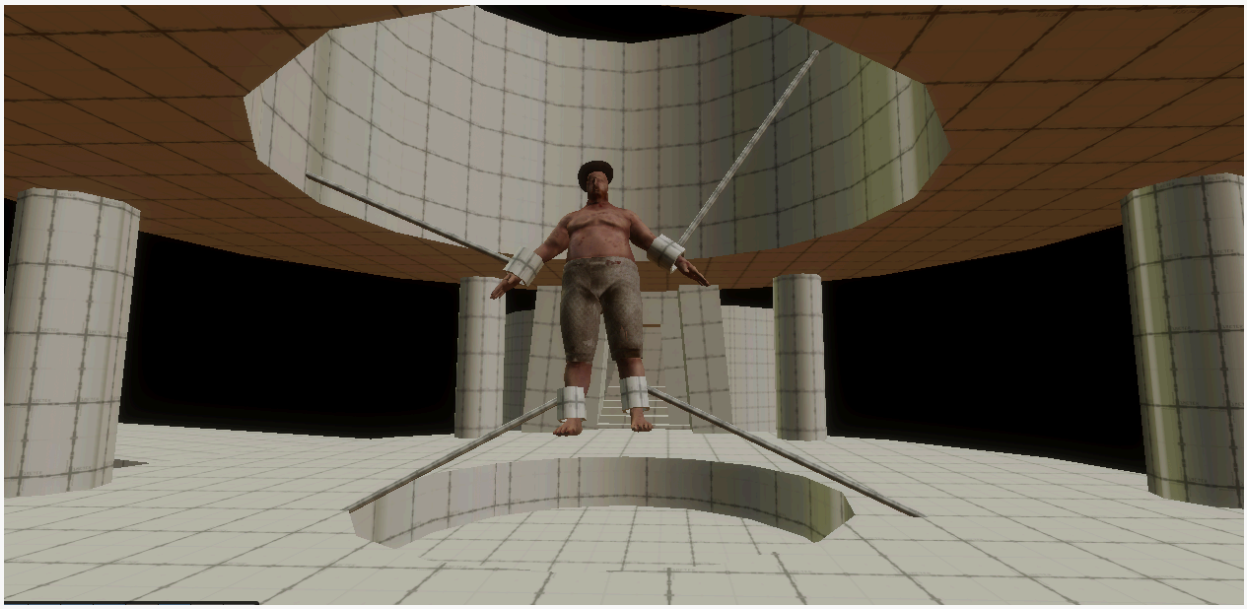
Week 2-3

We optimized level loading by dividing scenes into smaller chunks and loading them asynchronously using Unity's Addressables system.



The Hell-themed level was greyboxed and served as a test environment to demonstrate this approach. This method reduced memory usage and initial loading time.





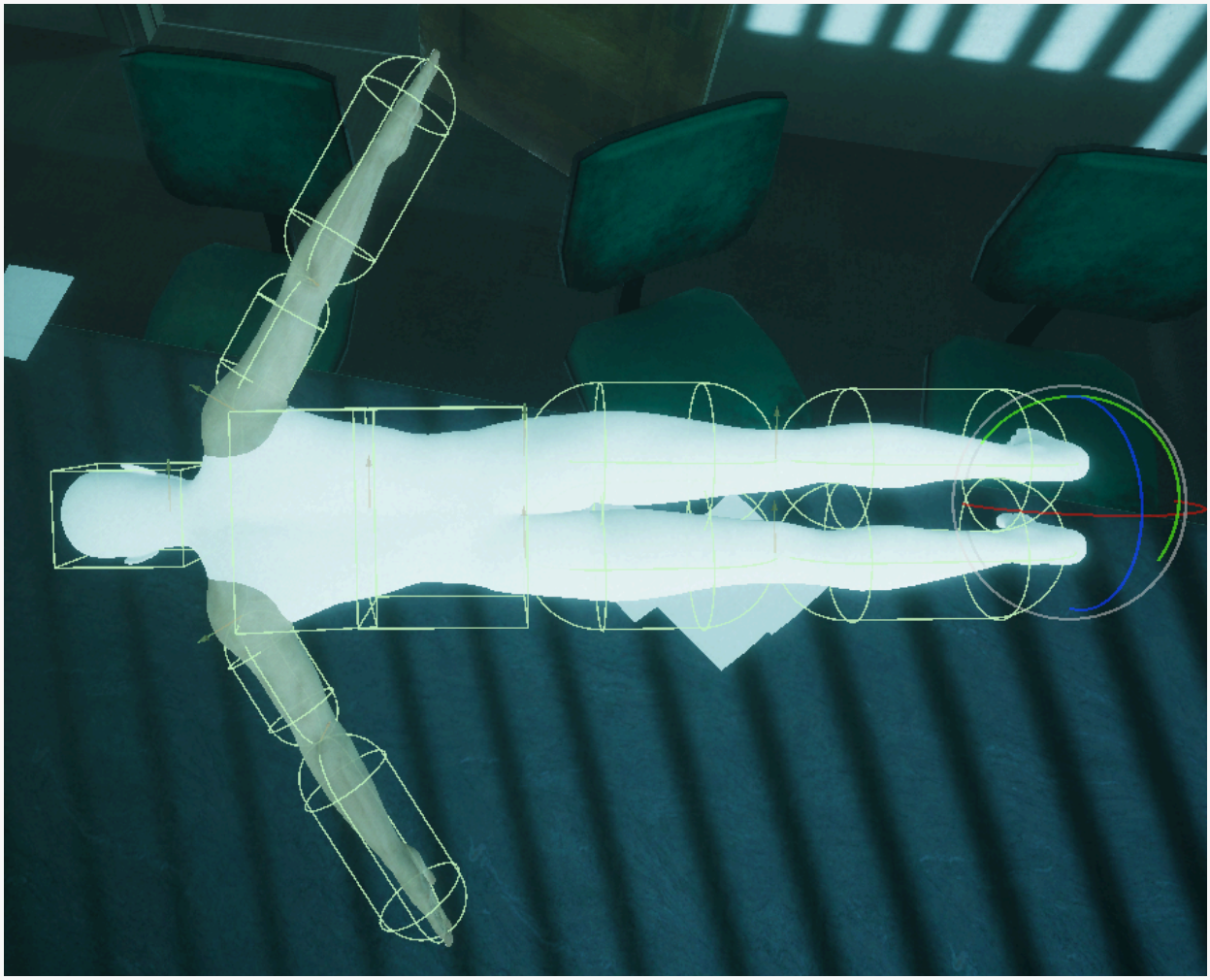
Implemented toothbrush mechanics today, it was relatively simple, I just used existing blaster code and made it sway from one side to another while ray-casting. Exposed its own variables as a separate tool.

- Implemented weapon range for both blaster and toothbrush

Modelled a simple brush using Boar’s fur texture for brush’s bristle, this is the material used in early toothbrush design.

- Oversight from async scene loading was the vista since 2 or more scenes can share the same vista/lighting/post processing volume, we should probably put these on a separated “vista_scene” with its own triggers and vista neighbours.

- Added ragdoll



- Implemented “Clump”: Cleanable mesh that deforms as they’re blasted.
- Ghost shader

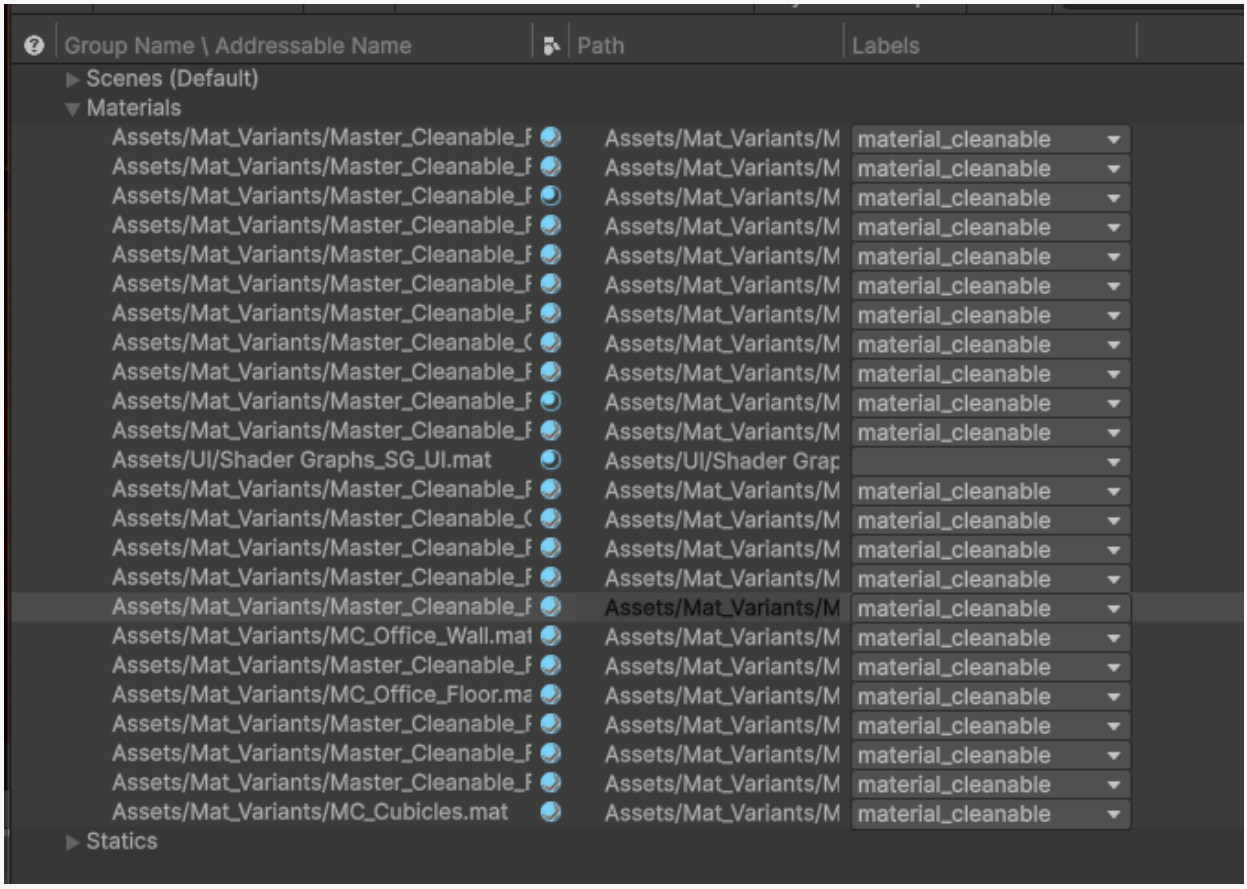
Week 4

Lighting consistency became a major challenge. Each scene initially required separate lighting settings, leading to visual inconsistencies. We solved this by implementing local volumes per scene with gradient sky and post-processing settings, creating a seamless transition between scenes.

To enhance gameplay variety, we introduced the Shotgun tool, a close-range weapon that balanced the existing cleaning mechanics. We also migrated to Addressables for asset management, significantly improving load times and reducing build size.

Week 5-6

- We expanded the level design by introducing the Hell-Corridor, a transition area between the dungeon and office. The toothbrush tool was added, providing a close-range cleaning method suitable for tougher grime types, encouraging strategic tool use.
- The Shotgun's functionality was refined, with bullet spread adjusted for a more consistent feel. We also enhanced the Blaster.cs script, exposing additional variables for future skill tree integration.




```
Addressables.LoadAssetsAsync<Material>( key: "material_cleanable", callback: null).Completed += handle =>
{
    allVariants = handle.Result.ToArray();
    UndoHighlightDirt();
};
```

- Fixed issue where 'Q' doesn't work in build -- we upgraded to addressable but it was still using /resources.

Week 7

Debugging tools were improved with the addition of an editor script for batch renaming objects, ensuring each GameObject has a unique identifier. This helped maintain scene organization and avoid save/load conflicts.

We reworked the HUD layout, centering the cleaning progress bar to increase visibility. We also implemented a currency system, where players earn money as they clean, providing immediate feedback.

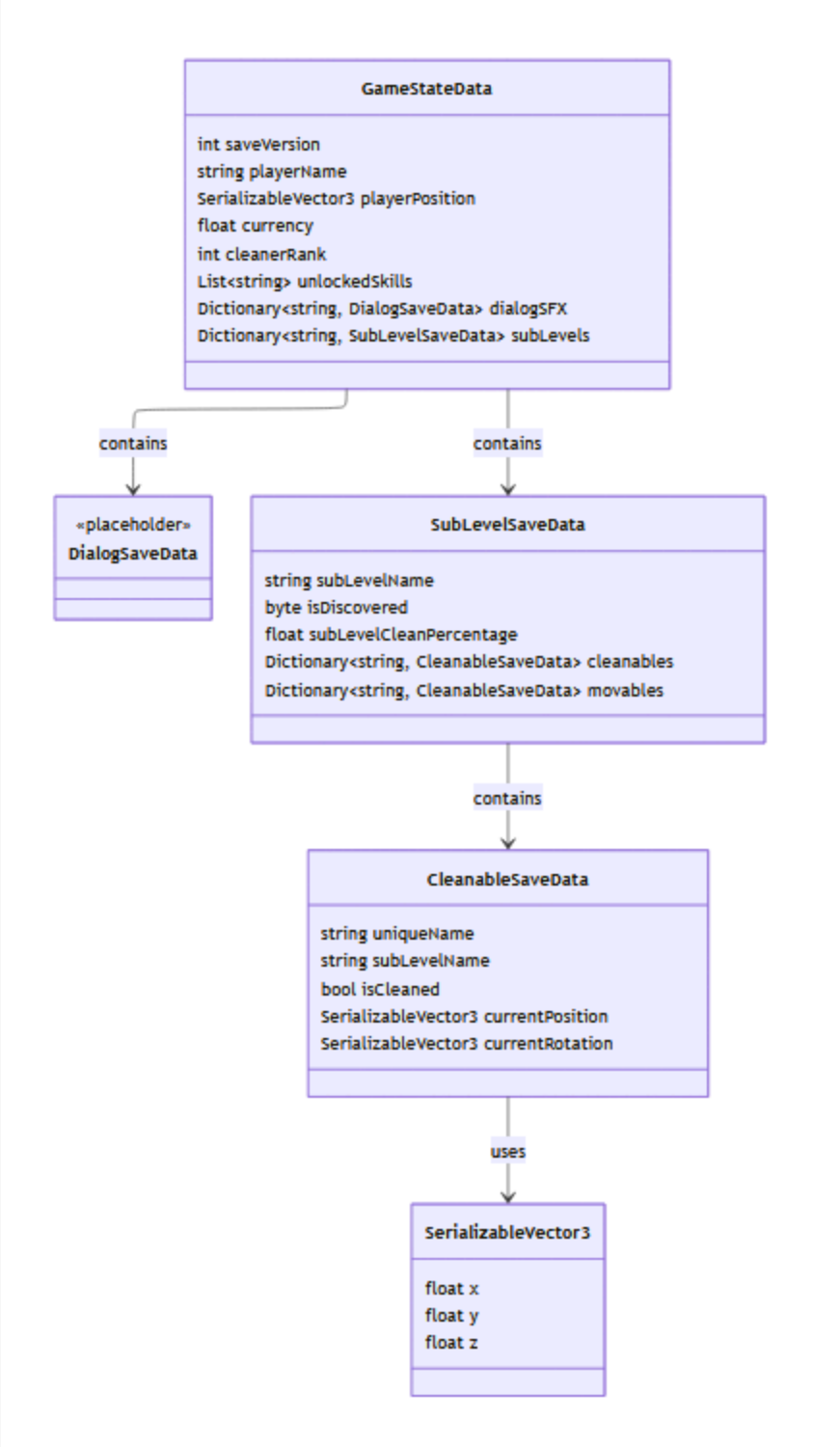
Week 8

Visual effects were further refined with the addition of splash VFX, created using Houdini's Vertex Animation Textures (VAT). These effects added a sense of physicality to the cleaning process.

We also modelled and implemented attachments for the power blaster, expanding player customization.

Week 9

I reworked the save/load system so it would work with additive scene loading / addressables. I added tracking for room cleanliness percentage, discovery state, cleaner rank, and room grade.



Physics interactions were enhanced, introducing ragdoll elements and improving force interactions on objects like chains, fans, and picture frames.

To streamline level creation, we created a Scene Validator Editor tool, which automatically detects and resolves naming conflicts and invalid cleanable objects.

Week 10

We refined the game’s structure by swapping level order, ensuring a logical progression and making the game fully completable from start to finish. We also added a screen fade at the end, smoothly looping the game back to the start for replayability. Additional polish was applied to power blaster attachments, and the elevator door transition was finalized.

Final testing and feedback gathering. Addressed minor bugs, polished visuals, and improved performance based on player feedback. These final adjustments ensured a smoother and more engaging player experience.