2309516 - Environmental Storytelling

Figure 1: Final render of the factory floor environment.

Exploring (Research & Inquiry)

Concept & Inspiration

- **Mood Boards**:
    - As a group we put together a Figma board which has mood boards for the factory environment, UI elements such as widgets and fonts within the original game and a storyboard of how we wanted the initial cutscene to play out.
    - Link to Figma board

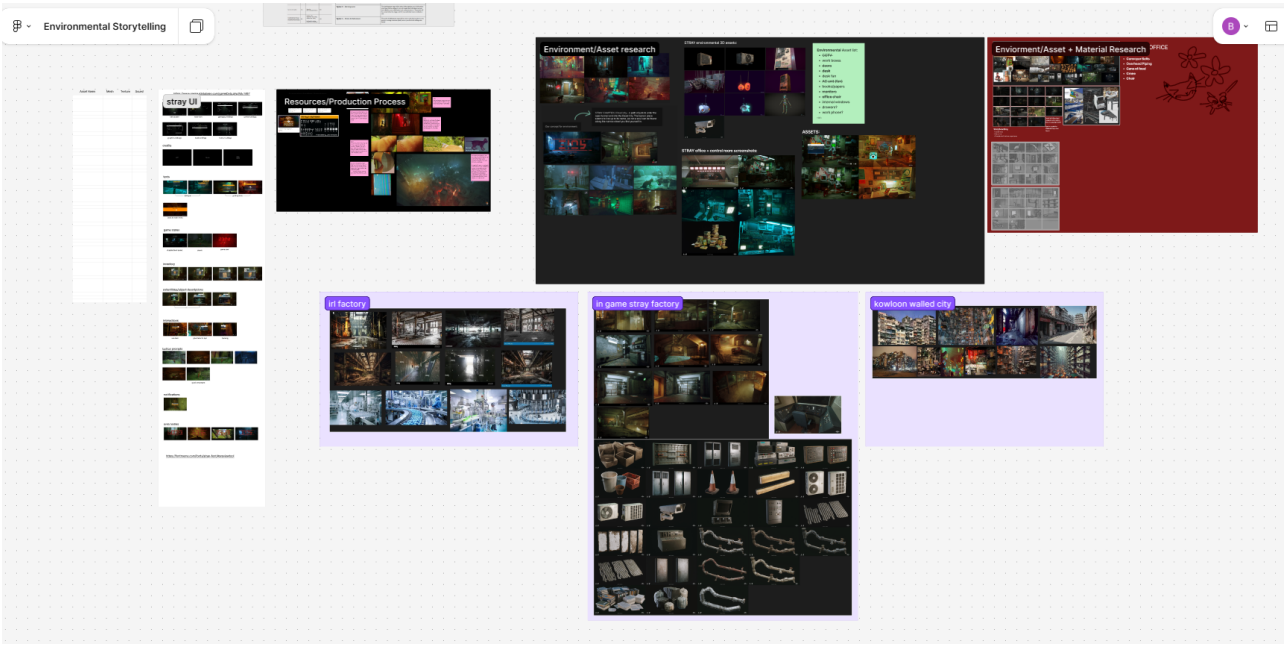

Figure 2: The team's Mood Board and Storyboard on Figma.

- **Genre Analysis**:
    - At the start of the first week we each took time to look at how environments tell stories within existing games to try and emulate a similar feeling. Games that came to mind were *Control* (Control - Remedy Entertaiment, s.d.) and *What remains of Edith Finch* (Edith Finch, s.d.).

(Control - Remedy Entertaiment, s.d.)

Figure 3: Floating bodies in Control with its uncanny atmosphere.

(Edith Finch, s.d.)

Figure 4: A detailed bedroom environment in What Remains of Edith Finch.

- In *Control* the environment is used to build a sense of corporate dread and surrealism through scattered memos and office layouts that feel both mundane and impossible. You might find a simple sticky note that reveals a terrifying supernatural containment failure which makes the cold concrete hallways feel alive and dangerous. On the other hand Edith Finch uses every inch of the family house to tell a more intimate and personal history. Each bedroom is a time capsule tailored to a specific character where the posters on the walls and the clutter on the floor explain who they were without ever needing a line of dialogue. While *Control* uses the world to explain its complex lore and mechanics Edith Finch uses it to create an emotional connection to a family that is already gone.

- **Mechanic Research**:
  - When coming up with the puzzle ideas and storytelling elements I had to look into different ways a character such as a robot could interact with the environment without using common methods such as waypoints. Games like *No Mans Sky* (Analysis Visor - No Man's Sky Wiki, s.d.), *Cyberpunk 2077* (Scan results - Cyberpunk 2077, s.d.) and *Death Stranding* (DEATH STRANDING Tips: Connecting the Eastern Region | Kojima Productions, s.d.) all use a scanning feature to reveal hidden details about the environment and objects.

(Analysis Visor - No Man's Sky Wiki, s.d.)

Figure 5: The Analysis Visor scanning interface in No Man's Sky.

- In *No Man's Sky*, the scanner is a multi-tool utility that pulses a local area to highlight nearby resources, mission objectives, and points of interest through a visual HUD.

(Scan results - Cyberpunk 2077, s.d.)

Figure 6: Scanning interactive objects in Cyberpunk 2077.

- In *Cyberpunk 2077*, the scanner allows you to identify interactive objects, hackable devices, and lootable containers within your field of vision. It also provides a tactical overlay of NPCs, revealing their vulnerabilities, resistances, and any active bounties or "Wanted" statuses.

(DEATH STRANDING Tips: Connecting the Eastern Region | Kojima Productions, s.d.)

Figure 7: The Odradek Scanner revealing terrain traversal in Death Stranding.

- In *Death Stranding*, the Odradek Scanner is a pulse from the players location that analyzes the landscape to identify traversable paths, marking hazards like deep water or steep slopes with color-coded icons (blue, yellow, and red).
- I preferred the feel of the scanner in *Death Stranding* because it provided a better sense of spacial awareness to the player especially as we decided on the game having a darker and more somber lighting scheme. I wanted the scan to also be able to highlight specific objects in the environment more closely to how *Cyberpunk 2077* does it with the deep red outlines. Finally I wanted the scan to also flow over objects as it expands outwards from the players location.

Making (Production & Iteration)

The core mechanics of the environment were built to support the narrative of discovery and interaction with the abandoned facility. The technical implementation focused on a modular "Scanner" system and a series of interconnected puzzle elements.

1. The Scanner Mechanic

The primary tool for navagating the environment is the Scanner. The scanner blueprint manages both the visual effect and the gameplay interaction.

- **Logic**: The scanning process is driven by a custom event `DoScan`, which plays the `ExpandScan` Timeline. This timeline controls the growth of the scan radius over time.

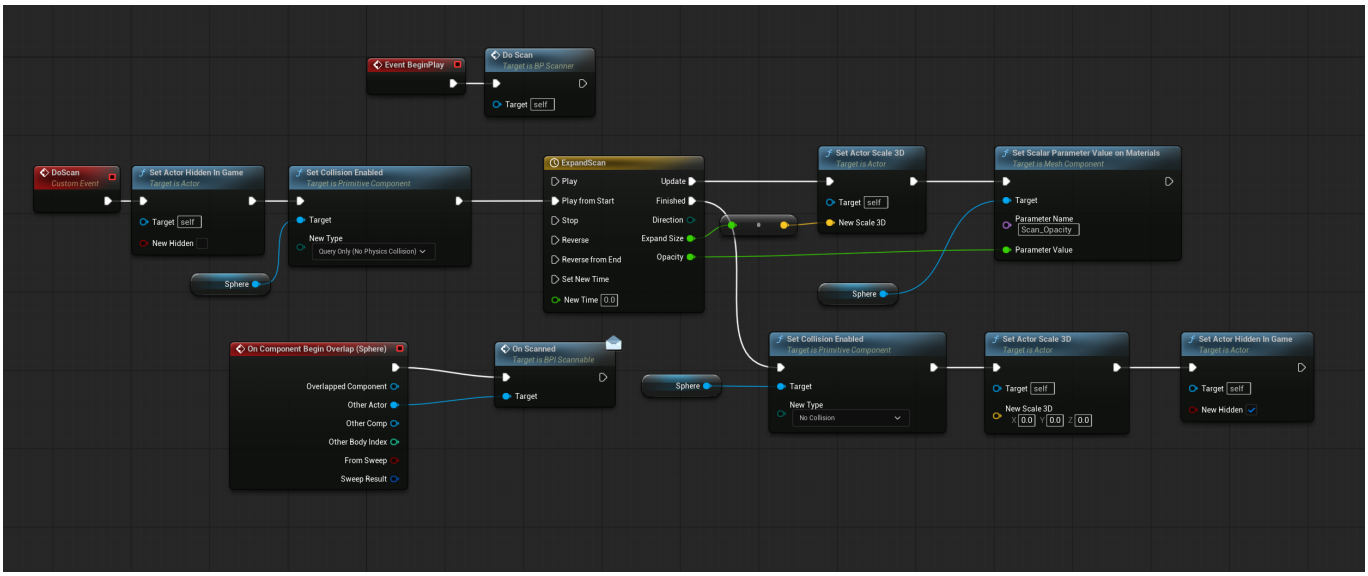

Figure 8: The BP_Scanner Event Graph controlling the scan logic.

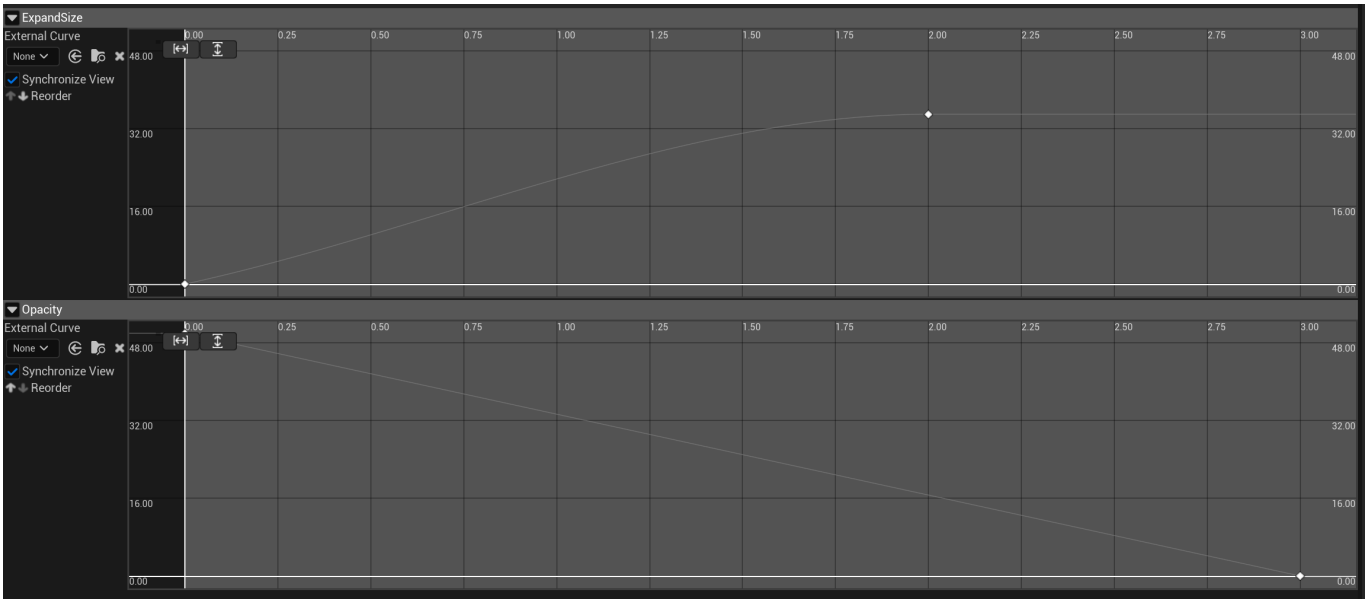

Figure 9: The Timeline curve controlling the scan radius expansion.

- **Visuals**: The timeline updates the actor's scale (`SetActorScale3D`) to simulate the expanding wave. Simultaneously, it drives a Scalar Parameter `Scan_Opacity` on the material `M_Scan`. This material uses a Depth Fade and Emissive Color (`Scan_Glow`) to create the holographic, "x-ray" aesthetic that reveals hidden geometry.

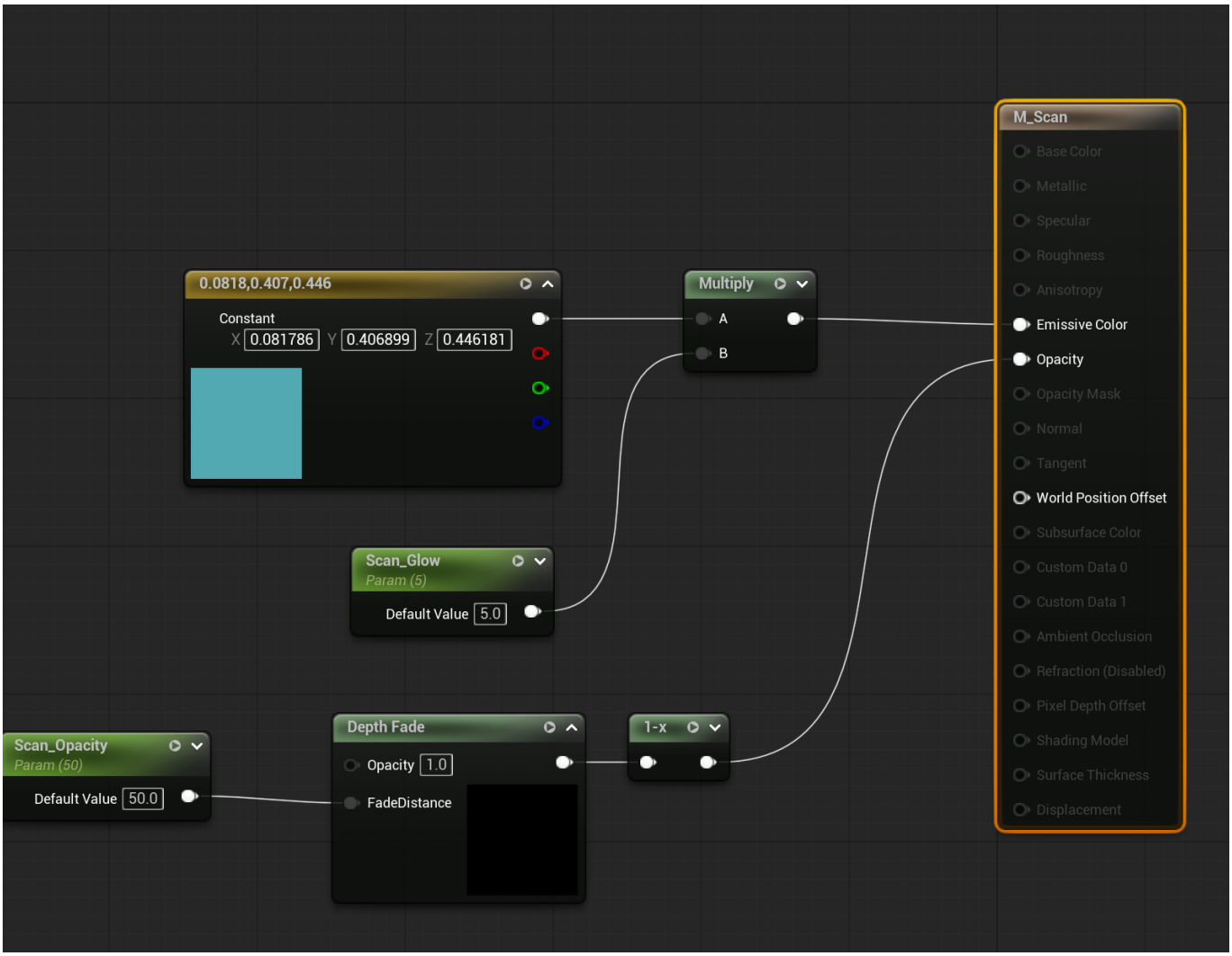

Figure 10: The M_Scan material graph creating the holographic effect.

- **Interaction**: To detect objects, the scanner uses a `Sphere` component. On `OnComponentBeginOverlap`, it checks if the overlapping actor implements the `BPI_Scannable` interface. If it does, the `OnScanned` function is called on that actor, allowing for distinct behaviors for different object types without hard references.

Figure 11: In-game view of the scanner highlighting the environment.

2. Interactive Objects

To bridge the gap between "looking" and "doing", I implemented a physics-based interaction system in `BP_ThirdPersonCharacter`. This allows the player to pick up specific objects such as the Levers and Tablets scattered around the environment.

- **Trace & Grab Mechanics**:
    - **Input**: The interaction is triggered by the `Left Mouse Button`
    - **Detection**: A `LineTraceByChannel` is fired from the camera's center forward vector. This ensures the player picks up exactly what they are looking at.
    - **Physics Handle**: If the trace hits a valid physics object, the `PhysicsHandle` component is activated. `GrabComponentAtLocationWithRotation` attaches the object to the character's `HoldLocation` scene component.
    - **Tick Update**: In the `ReceiveTick` event, the system continuously updates the Physics Handle's target location (`SetTargetLocationAndRotation`) to match the `HoldLocation`. This creates a smooth, laggy "towing" effect where the object physically follows the player's view rather than snapping rigidly.

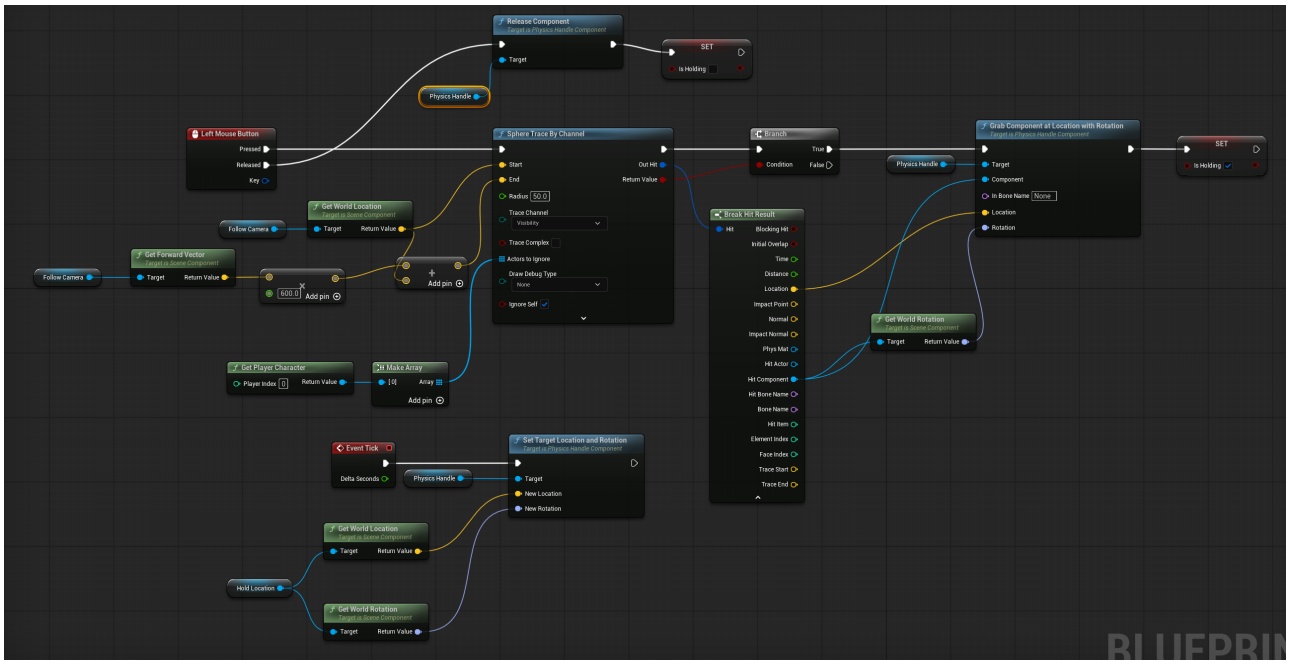

Figure 12: Interaction and physics logic in BP_ThirdPersonCharacter.

- **Interface Implementation**: `BP_Lever` implements the `BPI_Scannable` interface.
- **Feedback**: When the `OnScanned` event is triggered by the scanner, the lever executes its specific feedback logic. In this case, it calls `SetOverlayMaterial` on its Static Mesh, applying `M_HighlightItem`. This visual cue immediately informs the player that the object is significant and interactive. ![BP_Lever Event Graph showing OnScanned event and SetOverlayMaterial]

3. The Puzzle Loop (Generator & Crane)

Figure 13: Early prototype of the Crane puzzle area.

Figure 14: Final iteration of the Crane Puzzle environment.

- **The Generator (`BP_Generator`)**:
    - **Puzzle Logic**: The actor functions as a "counter" using a Trigger Volume. It casts overlapping actors to `BP_Lever` and increments an integer `ItemsOnPad`. Initially the generator started as a Pressure pad in which you put boxes onto but to better fit the theme of the environment I changed it to be levers that are replaces on a control panel which then allow the crane to function.

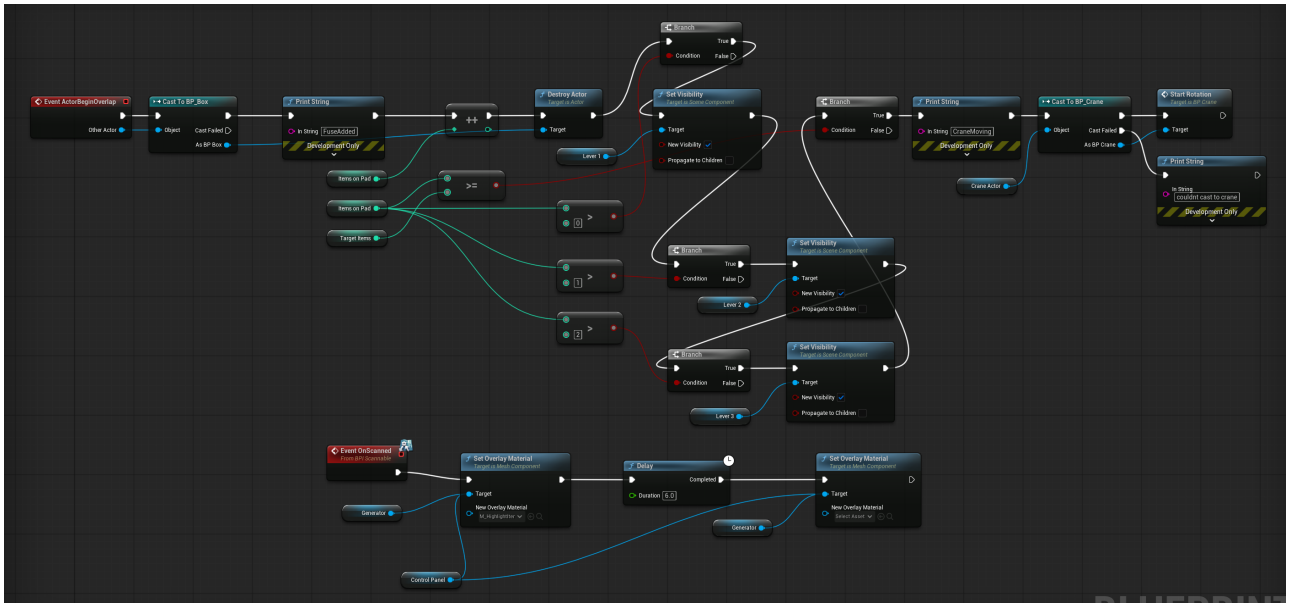

Figure 15: The Generator blueprint logic for counting levers.

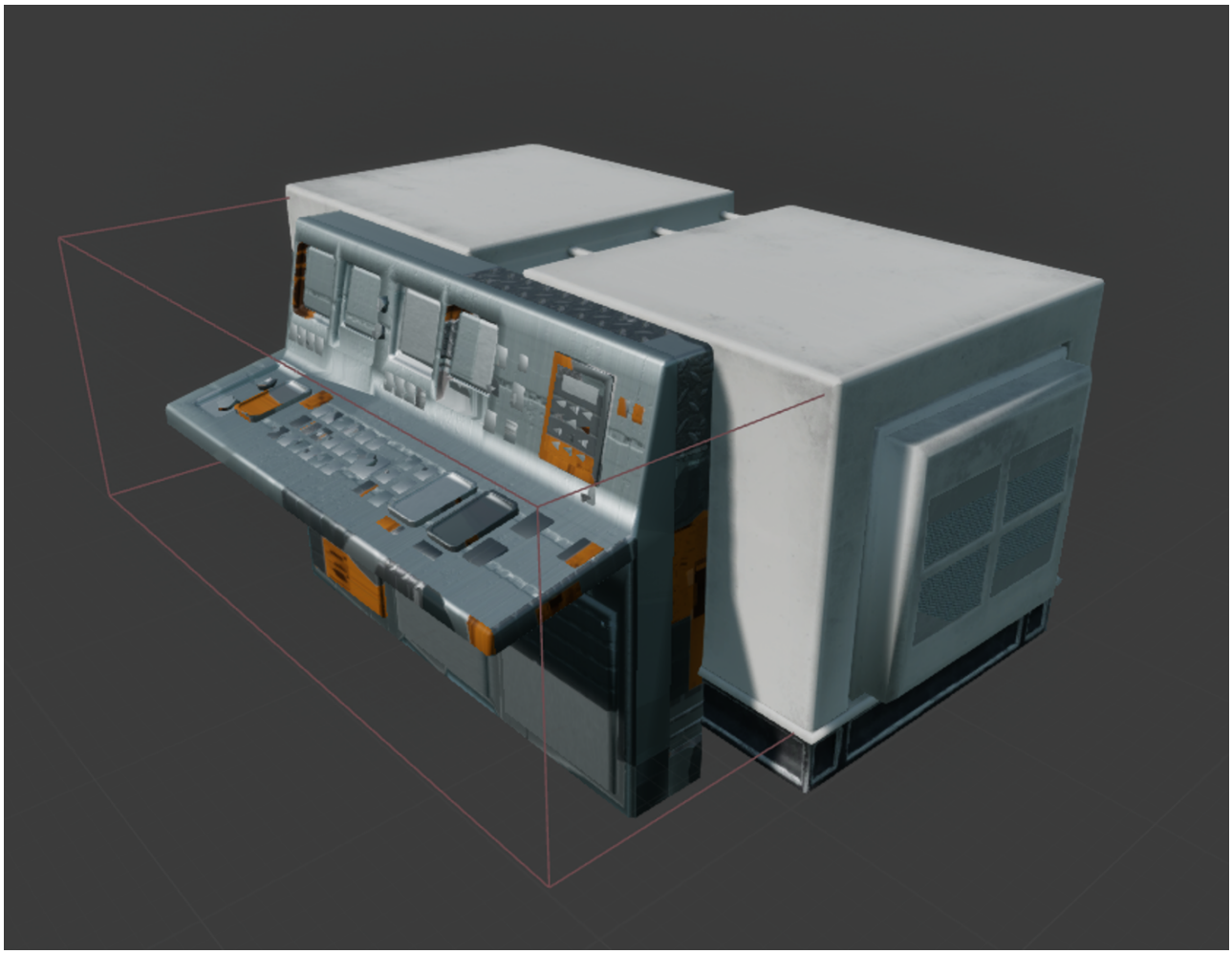

Figure 16: Viewport representation of the Generator actor.

- **Visual Feedback**: To communicate progress without UI, the blueprint toggles the visibility of static mesh components (`Lever1`, `Lever2`, `Lever3`) as the count increases. This diegetic feedback confirms successful placement to the player.

Figure 17: Visual feedback of the Generator with all levers placed.

- **Consistency**: Like `BP_Lever`, the Generator implements `BPI_Scannable`. Its `OnScanned` event applies the same `M_HighlightItem` overlay, establishing a consistent visual language for interactable objects.
- **The Crane (`BP_Crane`)**:
    - As soon as the generator has all three levers in place it will call the `StartRotation` event on the crane. Upon receiving the `StartRotation` call, it sets a boolean `bShouldMove`.
    - **Interpolation**: In `ReceiveTick`, it uses `RInterpTo` (Rotation Interpolation) to smoothly rotate the arm towards `TargetRotation` at a defined speed. This avoids jarring "snapping" movement,

preserving immersion.

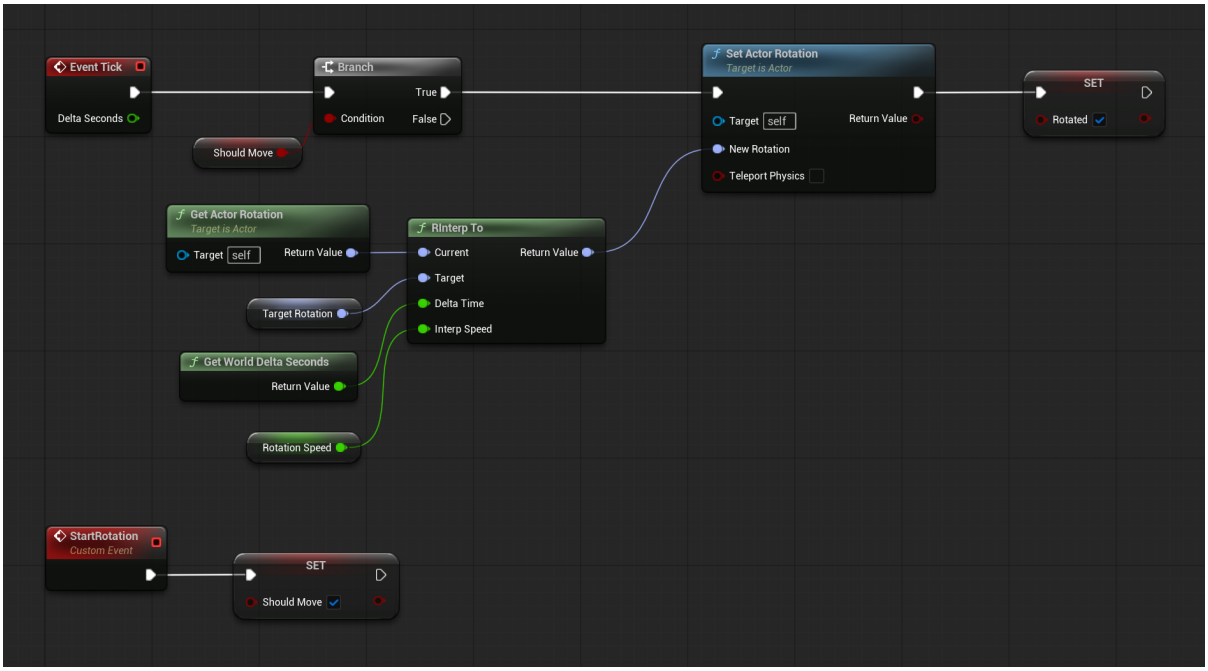

Figure 18: Rotation interpolation logic in BP_Crane.

4. UI

- **New Area Popups**: In Stray, when the player enters a new area a popup will appear in which it has the areas name using the games unique font as well as an English translation below. For most of the 4 weeks we did not have a popup like this and the game felt a bit empty but after adding it in, it really helped a sense of discovery and progression.

https://interfaceingame.com/screenshots/stray-new-zone/

Figure 19: New area popup example from Stray.

Figure 20: Our custom Area Popup implemented in-game.

- **Implementation**:
  - **Trigger (`BP_OfficeUI`)**: I created an actor that functions as a trigger. On `ReceiveActorBeginOverlap`, it checks if the overlapping actor is the player (`BP_ThirdPersonCharacter`). If confirmed, it creates the `WBP_Office` widget, adds it to the viewport, and immediately calls `DestroyActor` on itself. This ensures the popup only triggers once per playthrough.
  - **Widget Logic (`WBP_Office`)**: The widget handles its own lifecycle to keep the implementation clean.
    - **Construct**: When created, it enters a `Delay` of 5 seconds, keeping the text fully visible.
    - **Animation**: After the delay, it plays the `FadeOut` animation, which modulates the `RenderOpacity` from 1 to 0.
    - **Cleanup**: I used the `OnAnimationFinished` event to call `RemoveFromParent`, ensuring no invisible widgets remain in memory.

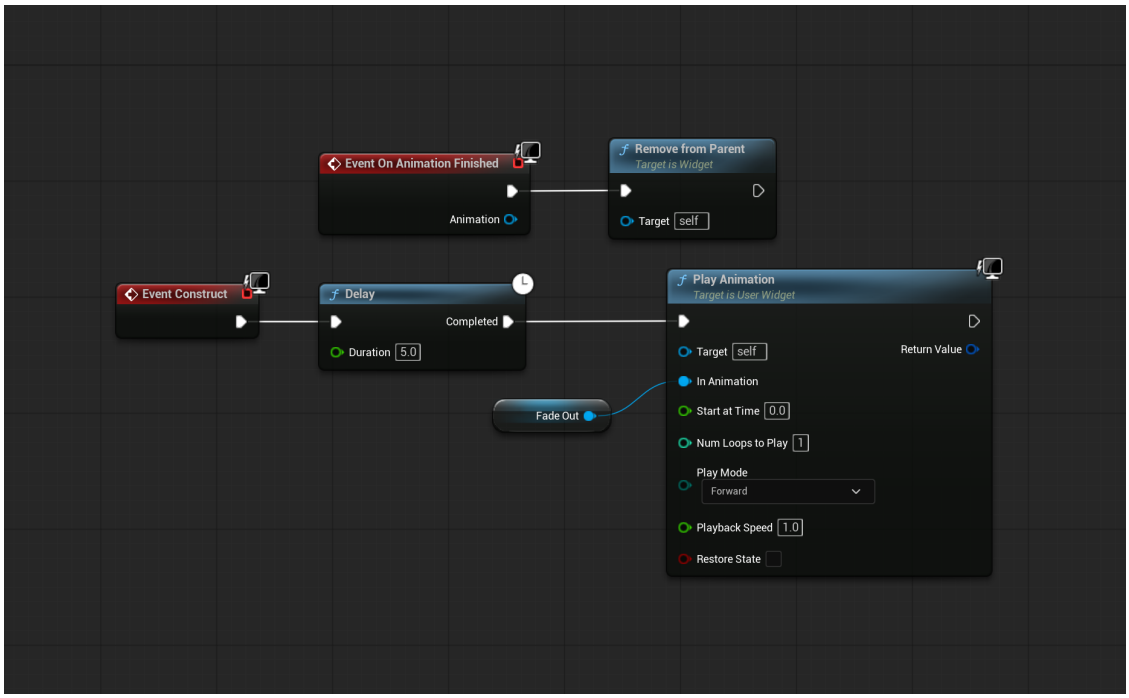

Figure 21: Blueprint logic for WBP_Office.

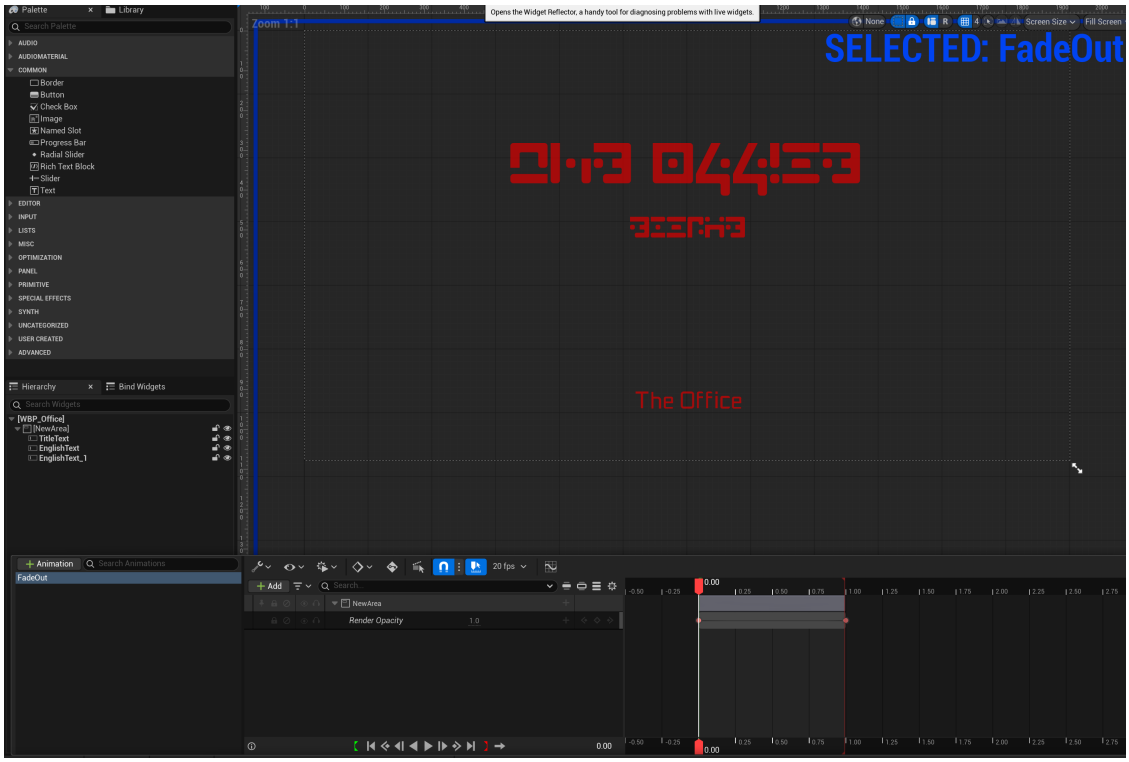

Figure 22: Fade-out animation for the popup widget.

- **Credits Screen (`BP_Credits`)**:
  - **Auto-Scroll**: I implemented a scrolling mechanic in the `Tick` event. It takes the current `ScrollOffset` of the main ScrollBox and adds a `ScrollSpeed` variable multiplied by `DeltaTime`. This creates a smooth, frame-rate-independent rolling effect typically seen in films.
  - **Concept Art Slideshow**: To make the credits visually engaging, I added a dynamic background.
    - **Logic**: On `Construct`, a Timer is set to fire every 3 seconds.
    - **Randomization**: The `CycleConceptArt` event picks a random texture from an `ImageArray` and updates the background image (`SetBrushFromTexture`). This

showcases the team's concept art alongside their names.

- **Navigation**: A simple "Back" button allows the player to close the credits via `RemoveFromParent`, returning them to the main menu.

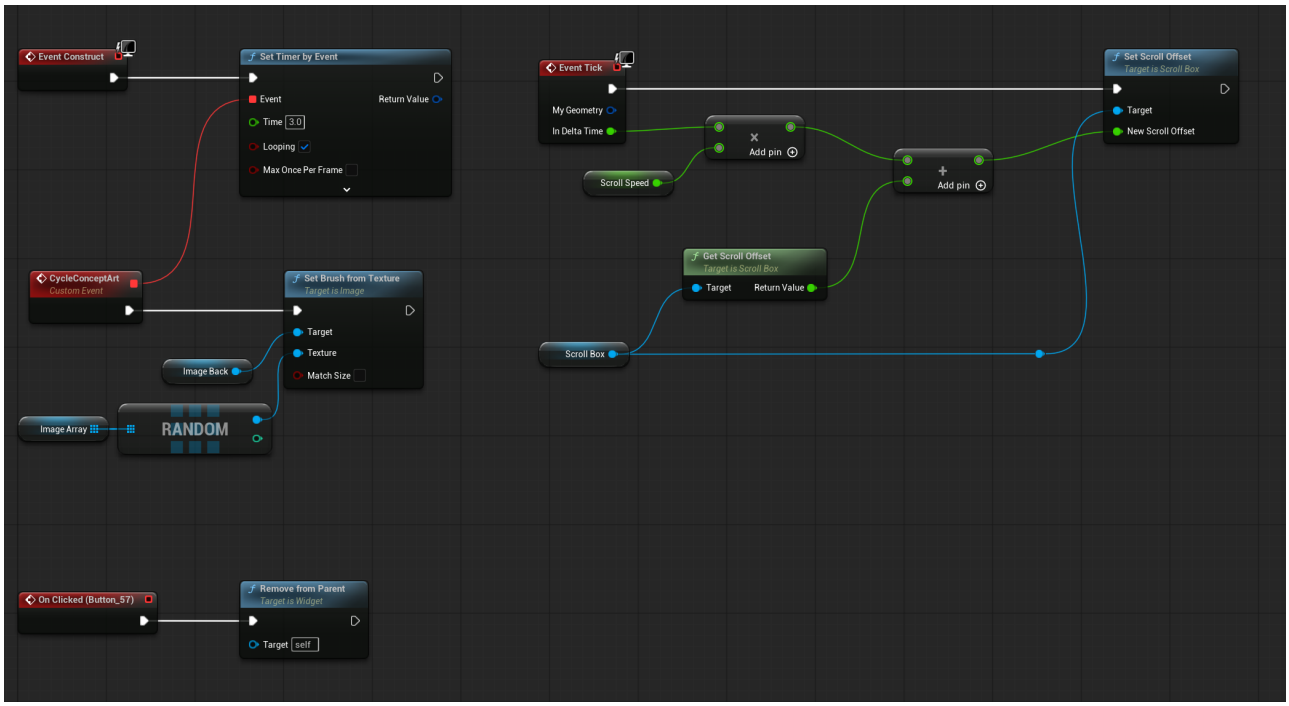

Figure 23: Blueprint graph for the scrolling Credits Screen.

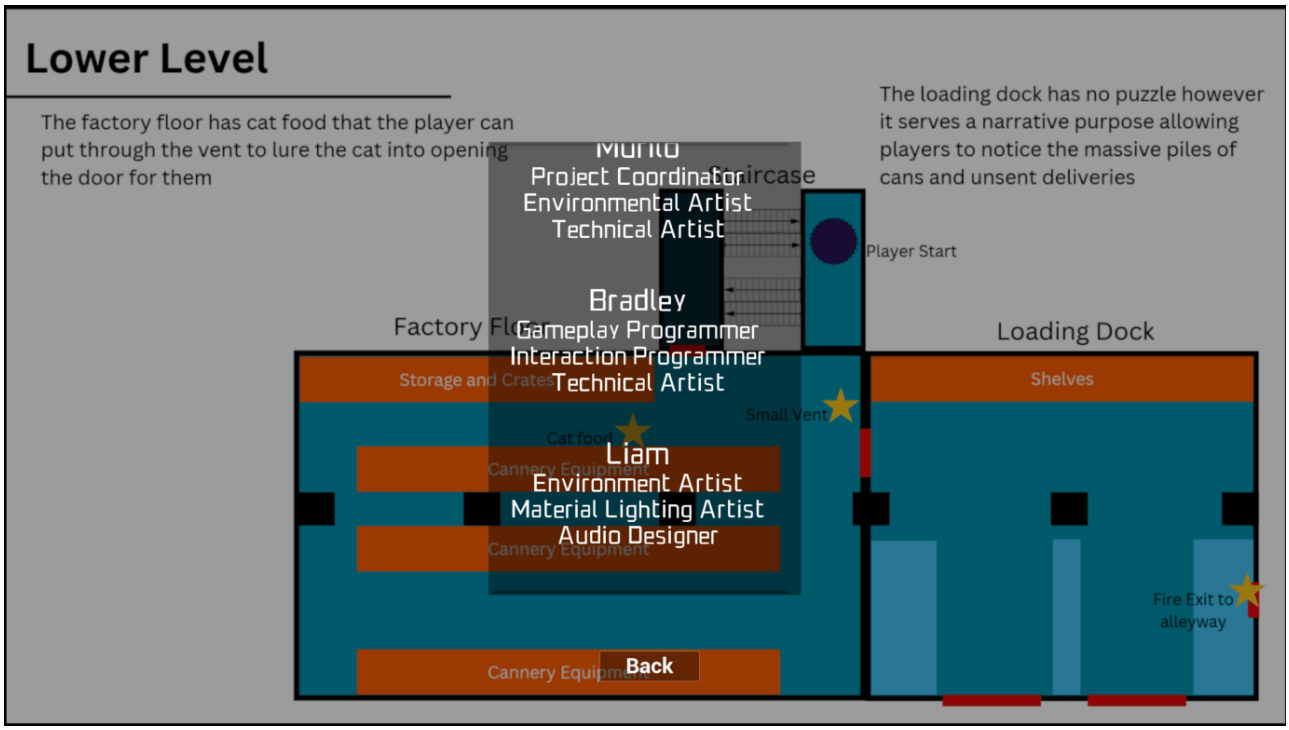

Figure 24: The final Credits Screen UI with scrolling text.

Connecting (Collaboration)

Teamwork & Workflow

- **Communication**:

- During the in-person lessons we would communicate needs for the upcoming week and tasks that urgently needed completion for others to progress too.
- Every Sunday at 5PM we had a Discord call where we could playtest the game and take notes on what went well for the week and what needs to be improved.
- I took on managing the Github repository for the project and ensuring that all team members were up to date with the latest changes. I also helped resolve any merge conflicts that arose. At the start of the project I asked everyone to send a message on the Discord server with the name of the branch they were working on and what they were working on. This helped me keep track of what everyone was working on and what needed to be merged.
- In the last 2 weeks of the project we started having an issue where people wouldnt say what they were working on and they didnt pull from the most recent main branch. This caused a lot of merge conflicts and issues with the project.

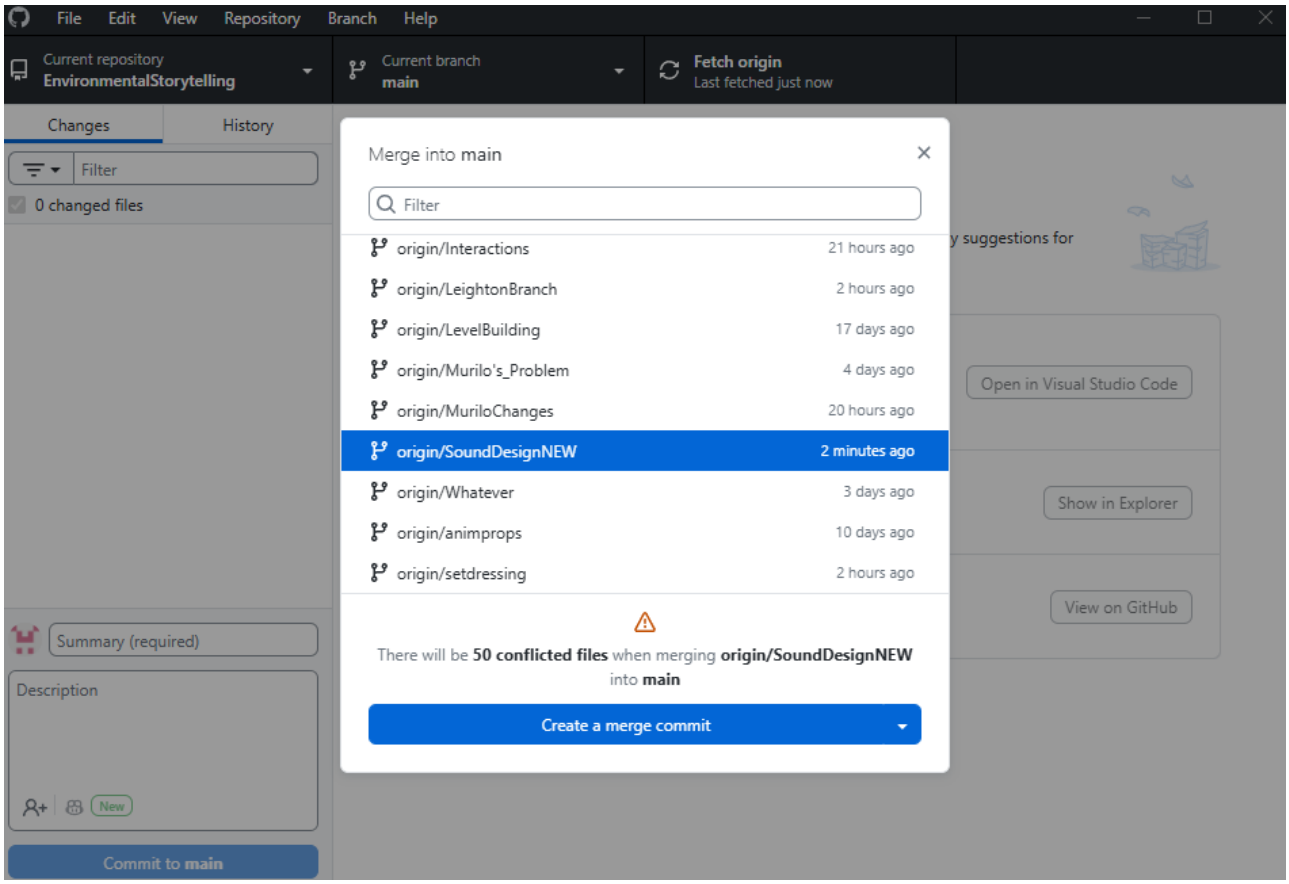

Figure 25: Example of a merge conflict faced during development.

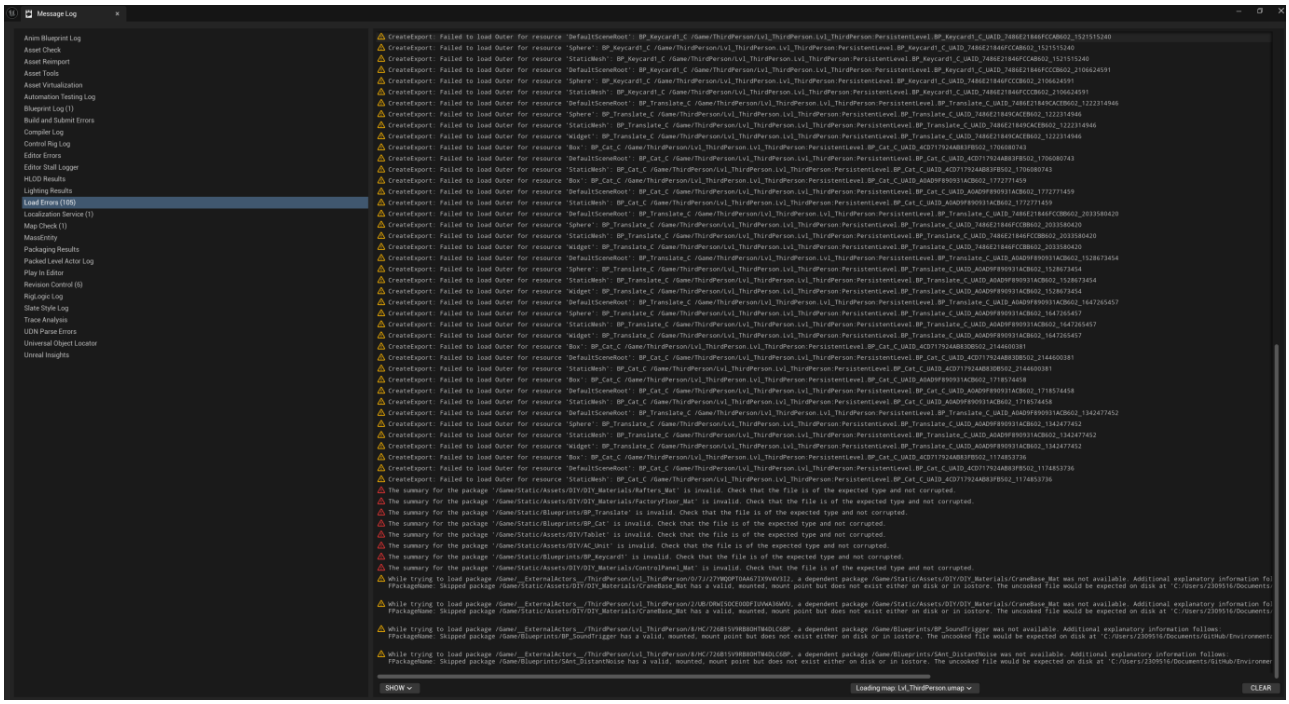

Figure 26: Build errors encountered in Unreal Engine.

- **Agile Methodology**:
    - We used Trello and Figma to plan and track our progress.

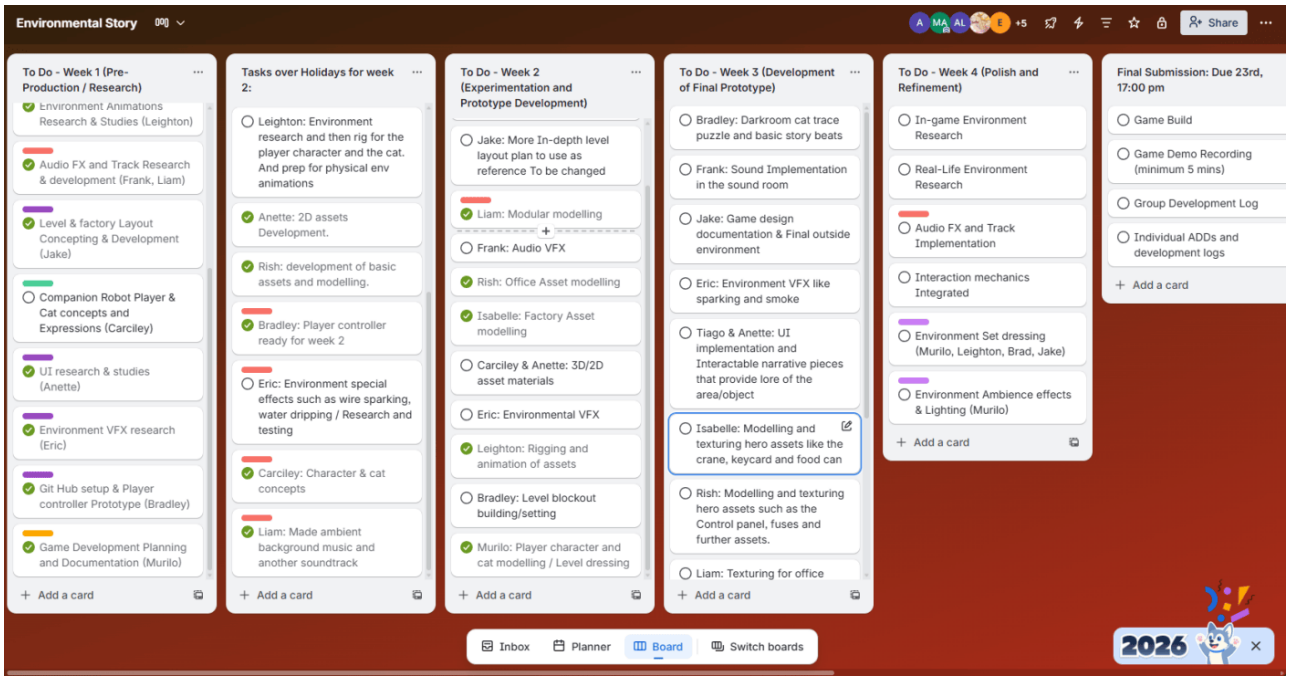

Figure 27: The Trello board used for Agile task management.

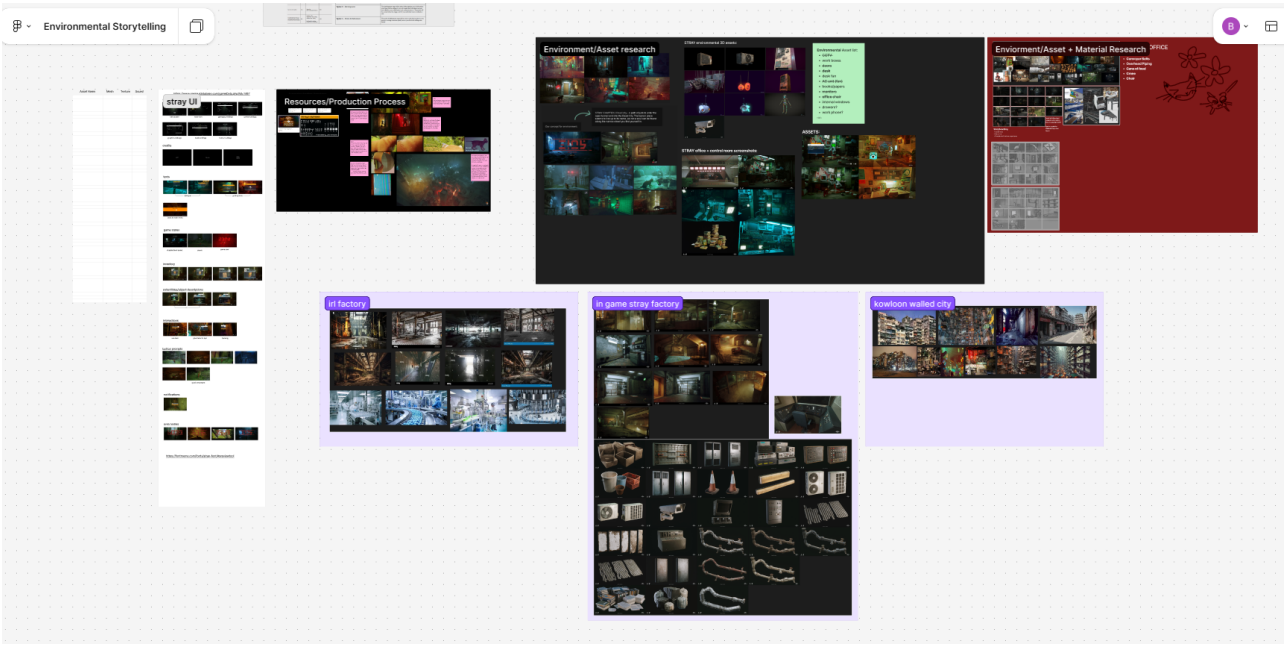

Figure 28: The Figma board used for planning and wireframing.

- **File Management**:
    - When I created the project I wanted to have clear file management standards to prevent clutter and confusion. We organized the main Content folder into clear subdirectories for Blueprints, Maps, Materials, Assets, and UI. I also tried to stick to Unreal Engine's standard naming conventions using prefixes like `BP_` for Blueprints (e.g., `BP_Generator`), `M_` for Materials, and `WBP_` for Widgets. This consistency combined with the structured folder hierarchy ensured that the project remained scalable and that files were correctly linked and easily accessible for everyone on the team.

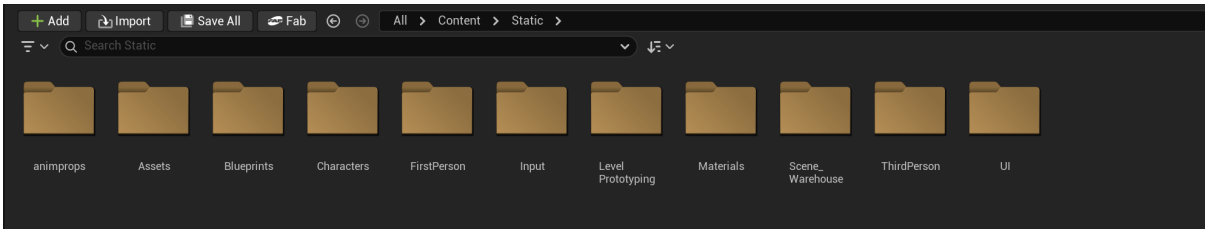

Figure 29: File structure organization in the Content Browser.

Situating & Synthesizing (Reflection - Context & Conclusion)

Reflection & Justification

- **Narrative Justification**:
    - I chose the scanner as the main mechanic because it felt like the perfect bridge between our robot protagonist and the stories of the humans who used to work here. It turns "revealing the past" into an actual game mechanic, letting players dig up hidden stories just by being curious, rather than forcing them to read big blocks of text.
    - For the visuals, we really wanted to nail that "cold, abandoned atmosphere" to make the player feel truly isolated. We stuck to harsh, cool lighting for the factory to make it feel dead and

unfeeling, which contrasts really nicely with the warm, safe glow of the scanner. It’s like the robot is literally shedding light on the dark, forgotten world.

- **Critical Evaluation**:
    - **Successes**: I’m really happy with how all the different parts of the project came together in the end. The UI elements, specific assets, materials, and game mechanics didn't just stand on their own, they worked together perfectly to convey our original idea. It was satisfying to see the project turn out exactly how we wanted, with every layer adding to that final feel.
    - **Challenges**: The biggest setbacks from the project were definitely issues with GitHub. We constantly had to deal with build errors that kept on coming back after each new merge happened to the git. It was incredibly frustrating trying to keep the project stable when random errors would pop up every time we tried to combine our work, and I ended up spending a lot of time just fixing the repo instead of working on the actual game.
    - **Future Improvements**: If I were to do this project again, I would definitely create a dedicated document outlining clear GitHub best practices for the team. Since the merge conflicts and build errors were such a huge setback, having a guide on how to pull, commit, and resolving conflicts properly would have saved us so much time and frustration. Also, if I had more time, I would have liked to jump in and help out more with the other roles. I noticed some people really struggled getting their assets and work into the game on time, and being able to support them directly would have made the whole production process smoother.

References

- Analysis Visor - No Man’s Sky Wiki (s.d.) At: https://nomanssky.fandom.com/wiki/Analysis_Visor (Accessed 18/12/2025).
- Control - Remedy Entertaiment (s.d.) At: https://www.remedygames.com/games/control (Accessed 17/12/2025).
- DEATH STRANDING Tips: Connecting the Eastern Region | Kojima Productions (s.d.) At: https://www.kojimaproductions.jp/en/DSTips (Accessed 17/12/2025).
- Edith Finch (s.d.) At: https://www.giantsparrow.com/games/finch/ (Accessed 17/12/2025).
- Keeley, P. (2019) ‘Control’: Game Review. At: https://www.hollywoodreporter.com/movies/movie-features/control-game-review-1234007/ (Accessed 17/12/2025).
- New zone - Stray (s.d.) At: https://interfaceingame.com/screenshots/stray-new-zone/ (Accessed 17/12/2025).
- Ramanan, C. (2017) 'What Remains of Edith Finch review: magical ode to the joy of storytelling' In: The Guardian 26/04/2017 At: https://www.theguardian.com/technology/2017/apr/26/what-remains-of-edith-finch-review-giant-sparrow (Accessed 17/12/2025).
- Scan results - Cyberpunk 2077 (s.d.) At: https://interfaceingame.com/screenshots/cyberpunk-2077-scan-results/ (Accessed 17/12/2025).

- Turbulent_Sun_9378 (2025) Scanning ability in 10 minutes! :). [Reddit Post] At: https://www.reddit.com/r/unrealengine/comments/1m9dzu9/scanning_ability_in_10_minutes/ (Accessed 06/01/2026).

This document was modified with the use of *Google Gemini 3 Pro*

(Google Gemini, s.d.) At: https://gemini.google.com/