Project Notebook
The Last Iteration

Baran Gürsoy
Mahdis Sabzevarzadeh
Bilgehan Savgu
Julie Vondráčková

Master Practical Course Games Engineering
Chair of Computer Graphics and Visualization
Technical University of Munich
Munich, Germany

January 6, 2026

Contents

Milestone 1: Formal Game Proposal

Our goal is to create a 2D, single-player, story-driven puzzle-platform game. We draw inspiration from games such as Limbo, Little Nightmares, The Swapper, and Machinarium. We aim to incorporate the topic of recursion and time travel into both the story and the game mechanics. The environment of the game takes place inside a time machine, and the goal is to fix the broken parts of this time machine. The game will be designed with several interconnected rooms featuring puzzles that must be solved to progress the story. The environment will feature a hub that connects multiple rooms to one another. Solving one puzzle in one room will open the door to another room. It is also possible that one puzzle in one room needs to be reused in another room. Our goal is to create engaging puzzle designs that immerse the player and provide hints through in-game notes about the story and the origin of the time machine and its creator.

1 Game Description

1.1 Story Overview

The year is far in the future. Humanity has mastered the impossible: Time Travel. But progress comes with a cost. The player awakens trapped inside a metallic chamber of a time machine, disoriented and alone. The player is a robot, a creation of the renowned scientist who built this machine. The player doesn't remember how they got here, only that something is terribly wrong. The lights are flashing, vital components lie broken, and the only way out appears to be through repairing the damaged systems. Guided by cryptic commands transmitted from an unknown source, the player begins repairing the system's failing components, one by one. As they explore the interior of the machine, they uncover notes scattered across the floor, written by the scientist himself and other robots before him. At first, they seem like instructions. But soon, they begin to reveal a darker truth. The time machine doesn't simply transport travelers through time; it duplicates them, sending a copy of the user into another timeline. Each use leaves the original behind, and with every cycle, something inside the machine breaks. To repair it, the scientist created the robot to fix, and many before the current player have had the same destiny. Every time the machine fails, a new robot is built, placed inside, and ordered to fix it. None of them has ever escaped. The robot wasn't the first and might not be the last. Deeper inside, the player discovers the corpse of a previous robot, broken in pieces and forgotten, confirming the worst suspicions. Through fragmented notes, the scientist himself admits the truth: this machine, his greatest invention, has become a recursive prison. A loop without end. Now, only one choice remains. If the player repairs the final piece, the time machine will reset. Upon next usage, the loop will begin again, and another robot will awaken to repeat the same fate. Alternatively, the player can refuse to finish the final piece of the broken device, sacrificing themselves to keep the machine broken forever, thereby trapping themselves inside and ending the cycle once and for all. Either way, someone is lost: the machine and the scientist's legacy, or the player.

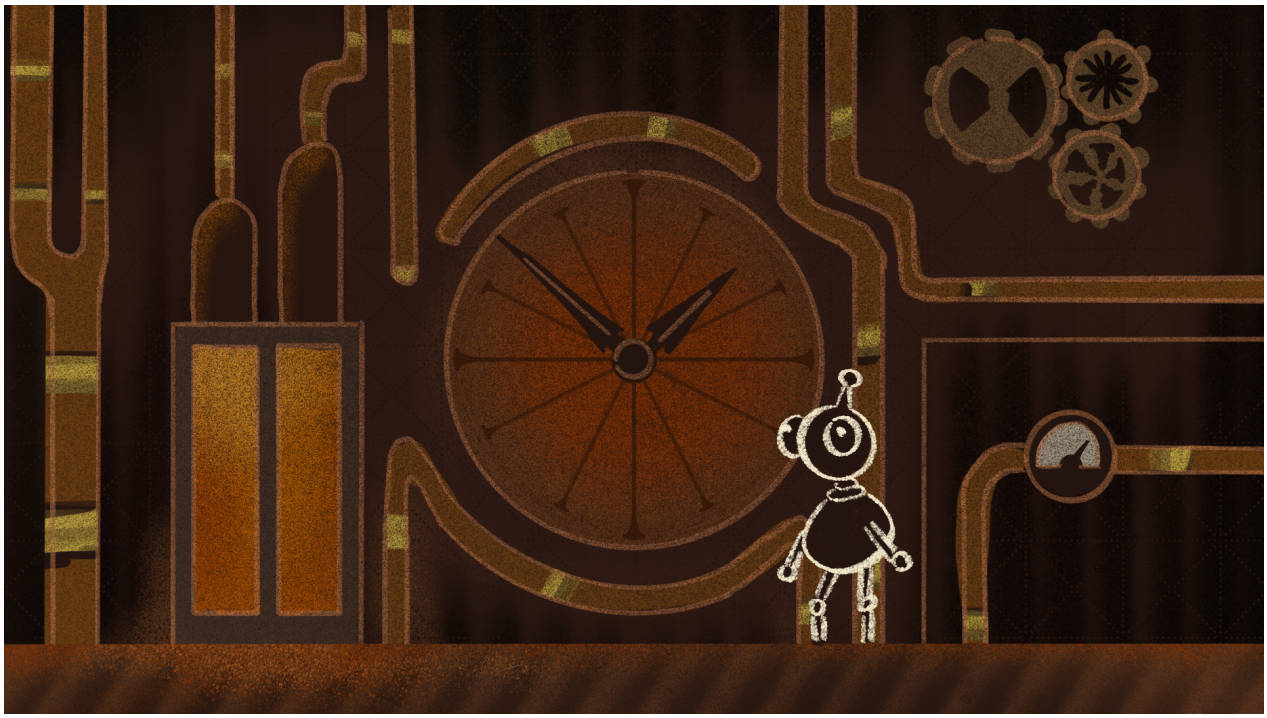
1.2 Relation to the Theme

Our project is built around the theme of "Recursion & Time Travel." We connect to this theme in both our main story and our core gameplay mechanics.
The "Time Travel" aspect is central to our game's setting and plot. The entire game takes place inside a large, malfunctioning time machine. The story is not told in the present, but is discovered by the player as they find notes and logs left behind by the scientist. These notes are messages from the past, and they slowly reveal the dark truth about the machine's true function.

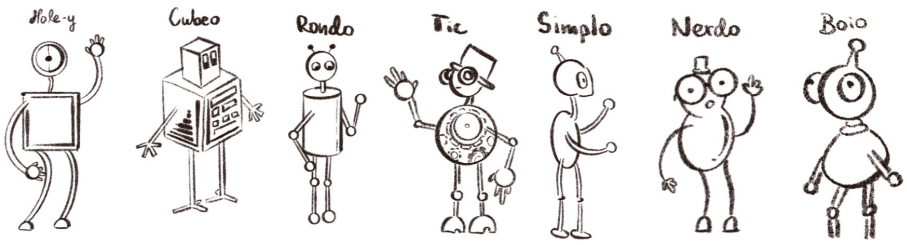
We also show "Recursion" and "Time Travel" in our core cloning mechanic. The "Time Travel" part is shown when the player freezes time to "record" their future actions. The "Recursion" is shown in the puzzle design: the player must perform an action in real-time that *depends on* the action they *already recorded* in the "frozen" time. This creates a time-loop paradox in miniature, where the player must solve a puzzle by co-operating with a version of themselves from a few seconds in the future (the recording). Finally, the narrative itself is a recursive loop, with the player discovering they are just one of many robots in an endless cycle.

1.3 Artistic Vision

The artwork shows the inside of a time machine designed with a strong steampunk style. The scene uses dark browns and rusty oranges, creating the feeling of old metal and heavy machinery. Pipes, gears, and gauges fill the background, giving the space a busy but organized look. The use of contrast is an important part of the design. Brighter elements, like the robot, key pipes, and important machines, stand out so players can easily spot what matters. This helps guide them through puzzles, showing where to focus and what to interact with. The overall vision is a warm, mechanical world that feels old but full of life, inviting players to explore and solve challenges within the time machine.



We wanted to make the game simple, so the main character animation shouldn't be extensive. The main robot designs are therefore simpler, but still hand drawn and with personality. The main character was chosen from these options.



1.4 Gameplay and Core Mechanics

Our game is a 2D puzzle-platformer. The player controls a robot inside a large, mysterious time machine. The core gameplay focuses on exploration and solving environmental puzzles. The player has several main abilities.

1.4.1 Basic Interactions

The robot has a set of basic physical skills. These are the foundations for all puzzle-solving:

- **Movement:** The player has standard platformer controls, including moving left and right, and jumping.
- **Push and Pull:** The robot can interact with heavy objects like large boxes or crates. This allows the player to move items to create new platforms or block traps. We want to make this feel realistic by using rigid body physics.
- **Pick Up and Drop:** The player can pick up smaller, key items like power cells or keycards. As a design choice, the robot has no inventory and can only hold one of these items at a time. This makes the player think carefully about which item they need and where they need to take it.
- **Swinging (As an extra if we have time):** Some areas will have ropes or cables that the robot can use to swing across large gaps.

1.4.2 Cloning (The Core Mechanic)

The robot's most important ability is to create a "Temporal Clone." This is our key mechanic for solving complex, time-based puzzles, and it works in two distinct phases:

- **1. Recording Phase:** When the player activates the ability, the main game's time freezes. The player then takes control of a "temporal echo" of the robot for a limited time (e.g., 10 seconds). During this phase, the player can move, jump, and interact with objects as this echo. The game records all of these actions.
- **2. Playback Phase:** When the time runs out (or the player stops recording), the game's time resumes. An "actual" clone appears at the starting location and replays the *exact* recorded actions in real-time.
- **Co-op Puzzles:** While the clone is playing back its actions, the player has full control of their original robot. This allows for cooperative puzzle-solving, such as the player and the clone pulling two different levers at the same moment to open a door.

1.4.3 Narrative Interactions

The story is not just in the background. The player must interact with notes left by the scientist.

- A simple UI will appear on the screen to show the text from these notes.
- These notes are necessary to understand the story, the function of the time machine, and the final choice the player must make.
- We read all the feedback on our idea presentation, and we'll try to make sure this notes/journals part is not excessive or boring; and we'll try to put it naturally in the game.

1.5 Puzzle Design

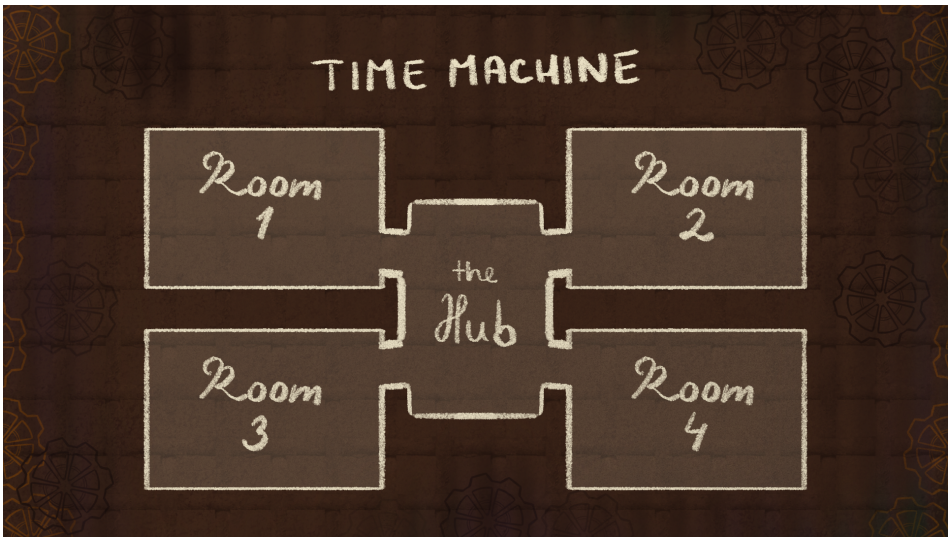
The game's world is structured like a large, connected escape room. The time machine is divided into several sections, or levels, which are all connected through a central hub.
The puzzles are designed to make the player use all of their mechanics in combination.

- **Connected Rooms:** A puzzle in one room might provide an item or activate a machine needed in a different room. This forces the player to move back and forth between levels and understand the layout of the machine.
- **Multi-Stage Puzzles:** Puzzles will have multiple steps. For example, a "mechanic" part of a room (like pushing a box) might be in a different section from the "electric" part (like connecting a cable).
- **Cloning Puzzles:** Many puzzles will require the clone. A key item might be visible but trapped in a small, inaccessible space. The player must create a clone, guide it into the space to pick up the item, and then destroy the clone to retrieve it. Or some of the puzzles might need to do an action at the same time in 2 different places, hence needs to use the clone's temporal recording ability and play the main character at the same time with the clone's recorded actions to solve the puzzle.
- **Connection Puzzles:** Some rooms will have "cable connecting" puzzles, where the player must correctly route power to a door or machine to activate it.

Our design approach for these puzzles will be modular. Instead of hard-coding each solution, we plan to build our different puzzle types (like the cable connections, clone interactions, and cross-room triggers) as flexible, reusable components. This will allow us to easily combine them to create more complex, multi-stage challenges. This also makes our design scalable, so if we have extra time, we can add more puzzles.

1.6 Levels

The game will consist of four main puzzle rooms (or sections) and one final room for the game's ending. These sections are all part of the same time machine and are connected, giving the feeling of one large, continuous world. Players will move between these sections by reaching their edges. Each room will blend exploration with puzzle-solving.



2 Technical Achievement

Our core technical achievement is the "Temporal Clone" system. This is not just a simple AI, but a full action recording and playback system that allows for complex, time-based puzzle design.
The technical challenge is divided into three parts. First is the **action recording**. We will need to create a system that can reliably capture the player's inputs or transform data (movement, jumping, and interactions) in a "time-stopped" state. These actions must be stored in a way that can be replayed.
Second is the **deterministic playback**. In the "real-time" playback phase, the clone's controller must execute the recorded actions perfectly to ensure its actions line up with the player's intentions and the game's physics.

Finally, the most complex challenge will be **desynchronization and conflict resolution**. As our team discussed, if the clone is recorded jumping on a box, but the real player moves that box during playback, the clone's action will fail. We plan to use this not as a bug, but as a core puzzle feature. The player must ensure their real-time actions do not create a "paradox" that invalidates the clone's recorded path. This creates a much deeper and more interesting puzzle dynamic.

3 "Big Idea" Bullseye

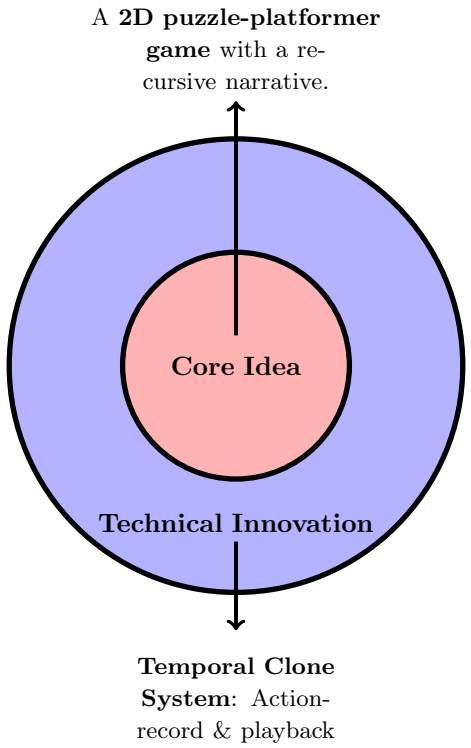


Figure 1: Our "Big Idea" Bullseye.

4 Development Schedule

4.1 Layered Development Description

This section provides a detailed description of the functionalities targeted for implementation in each iteration.

4.1.1 Functional Minimum

The minimal objective is to develop a playable puzzle-platformer prototype featuring basic room structures and limited interactive elements. The initial phase emphasizes establishing the core mechanics to ensure a stable and functional foundation.

- Player movement and jumping
- Player can push and pull items
- Player can pick up and put down items
- UI to show the notes

- A core layout showing how the hub and rooms connect and where puzzles occur.
- First iteration of 2D character design

4.1.2 Low target

At this stage, the goal is to refine gameplay mechanics and integrate new features to enrich the overall experience.

- More concrete level design of the environment and the connected rooms
- Implementation of puzzle designs for a minimum of two rooms
- Cloning mechanism without temporal recording (only playing the clone in real-time)
- 2D character design and animation
- Environmental design with some interactive objects for two rooms

4.1.3 Desirable target

We aim to build on previous stages to create a more dynamic and immersive environment.

- Implementation of puzzle designs for the remaining two rooms
- Cloning mechanism with temporal recording as discussed
- Enrich interactive elements in environments
- Add music and sound effects for a better user experience
- Implementation of notes and hints about the flow of the story

4.1.4 High target

- Choice-based cutscenes reflecting player decisions
- Additional puzzle mechanics to enrich gameplay
- New rooms expanding the overall map size
- Ability for the player to swing between platforms

4.1.5 Extras

- Item crafting system
- Voice acting for both the scientist and the robot when reading notes
- Flashback cutscenes to reveal story elements instead of relying solely on notes
- Expanded map with a dedicated UI and more interconnected puzzles

4.2 Task List and Timeline

Date	Milestone	Week	Layer	Task	Expected Time	Actual Time	Owner	State
November 05 - 11	Prototype	1	Prototype	Physical Prototype	4*5		All	TODO
November 12 - 18		2	Minimum	Player movement and jumping	5		Baran	TODO
				Push and pull objects	5		Baran	TODO
				Pick up and put down items	5		Mahdis	TODO
				UI to show the notes	5		Mahdis	TODO
				Core layout of level design	5		Bil	TODO
				2D character design 1/2	5		Julie	TODO
				Version Release v1	4*5		All	TODO
				Documentation v1	4*5		All	TODO

Figure 2: Time schedule [1/5].

November 19 - 25	Interim Demo	3	Low	Level Design 1/3	5			TODO
				Puzzle Design 1/3	5			TODO
				Cloning Mechanism 1/2	5		Mahdis	TODO
November 26 - December 02		4		2D character design 2/2	5			TODO
				Cloning Mechanism 2/2	5		Mahdis Baran	TODO
				Assets for the Environment 1/3	5		Julie	TODO
				Bug fixing before release	4*5		All	TODO
				Version Release v2	4*5		All	TODO
				Documentation v2	4*5		All	TODO
		5		Level Design 2/3	5			TODO
				Music and Sound Effects 1/3	5		Bil	TODO
				Puzzle Design 2/3	5			TODO
December 03 - 09				Assets for the Environment 1/3	5		Julie	TODO
				Story Implementation 1/3	5			TODO

Figure 3: Time schedule [2/5].

December 10 - 16	Alpha Release	6	Desirable	Puzzle Design 3/3	5			TODO
				Music and Sound Effects 2/3	5		Bil	TODO
				Story Implementation 2/3	5			TODO
				Level Design 3/3	5			TODO
				Assets for the Environment 2/3	5		Julie	TODO
December 17 - 23		7		Music and Sound Effects 3/3	5		Bil	TODO
				Assets for the Environment 3/3	5		Julie	TODO
				Story Implementation 3/3	5			TODO
				Bug fixing before release	4*5		All	TODO
				Version Release v3	4*5		All	TODO
				Documentation v3	4*5		All	TODO
December 24 - 30		8	Holiday (Dec 24 - Jan 06)		0			
December 31 - January 06		9	Holiday (Dec 24 - Jan 06)		0			

Figure 4: Time schedule [3/5].

January 07 - 13	Playtesting	10	High	Cutscene 1/2	5			TODO
				Map Extension v1 1/2	5			TODO
				Puzzle Extension 1/2	5			TODO
				Swinging Mechansim 1/2	5			TODO
January 14 - 20		11		Cutscene 2/2	5			TODO
				Map Extension v1 2/2	5			TODO
				Puzzle Extension 2/2	5			TODO
				Swinging Mechansim 2/2	5			TODO
				Bug fixing before release	4*5		All	TODO
				Version Release v4	4*5		All	TODO
				Documentation v4	4*5		All	TODO

Figure 5: Time schedule [4/5].

January 21 - 27	Final Release	12	Extra	Item Crafting 1/2	5			TODO
				Voice Acting 1/2	5			TODO
				Flashback cutscenes 1/2	5			TODO
				Map Extension v2 1/2	5			TODO
January 28 - February 03		13		Item Crafting 2/2	5			TODO
				Voice Acting 2/2	5			TODO
				Flashback cutscenes 2/2	5			TODO
				Map Extension v2 2/2	5			TODO
				Bug fixing before release	4*5		All	TODO
				Version Release v5	4*5		All	TODO
				Documentation v5	4*5		All	TODO

Figure 6: Time schedule [5/5].

5 Assessment

The main strength of The Last Iteration lies in its Temporal Clone mechanic, which fuses the narrative theme of time travel and recursion with interactive puzzle-solving. The most striking moments emerge when players must collaborate with their past selves to solve time-based puzzles. The game is designed for players who enjoy atmospheric and thought-provoking puzzle-platformers such as The Swapper, Limbo, and Inside, and who are drawn to philosophical sci-fi narratives about time travel and loops. Throughout the game, players explore interconnected rooms, repair the malfunctioning time machine, record and replay their actions, and gradually uncover the dark truth behind the recursive loop. The design will be considered successful if players can intuitively understand and use the clone mechanic, feel emotionally engaged by the final moral choice, and experience puzzles that are challenging yet fair.

Milestone 2: Game Prototype

6 Description

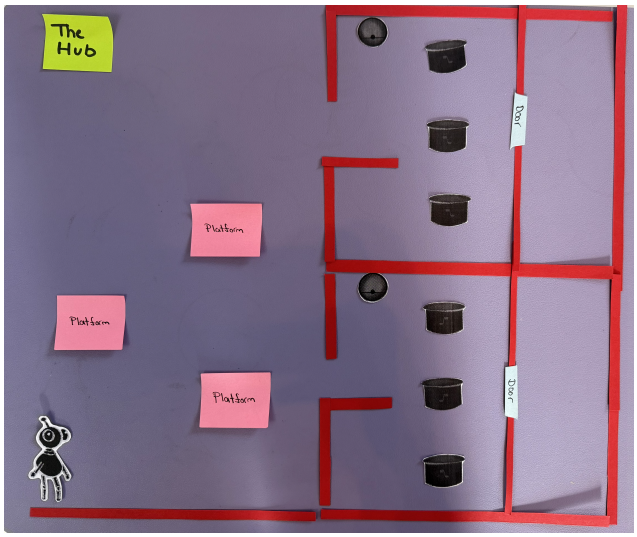
For our physical prototype, we created a tabletop, 2D paper version of our game world, as described in the course guidelines. Our goal was to test the flow of our core game loop and see if our main puzzle ideas were fun and understandable.

- **Materials:** The prototype was built using printed cutouts of our 2D character and game assets (gears, musical platforms, etc.). The levels themselves were built using paper blocks to represent platforms, with the layout based on our map sketches.
- **Process:** One team member managed the entire prototype session. This person acted as both the "computer" (managing the game's rules and puzzle states) and the "player" (moving the robot cutout). Each significant step was photographed to document the playthrough and to assemble into a demonstration video.

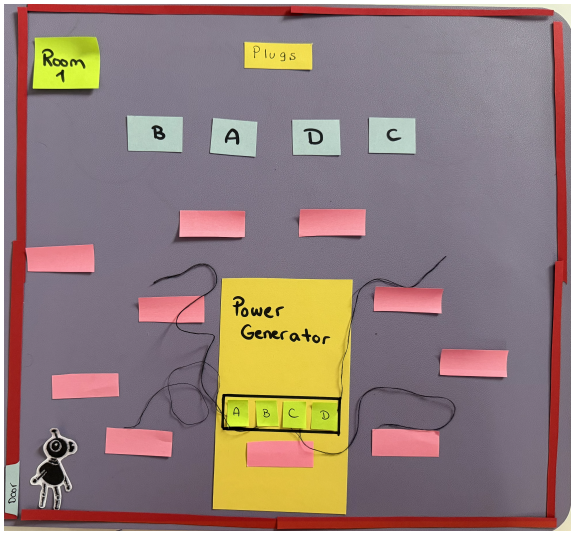
6.1 Modeling Core Gameplay

We focused on testing our main mechanics:

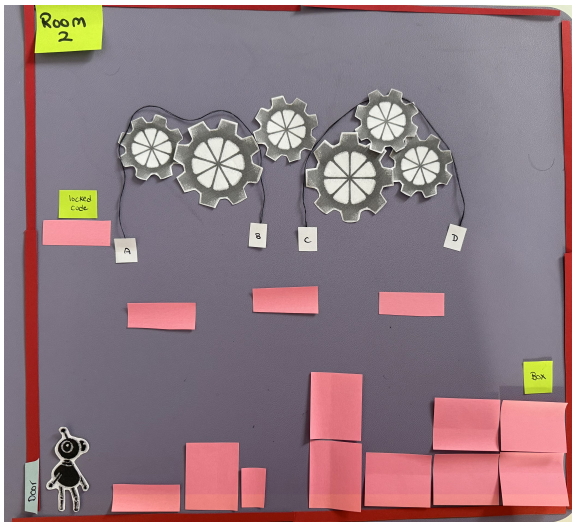
- **Exploration & Puzzle-Solving:** The player moved the paper robot through the "Hub" to "Room 1" (Electrical Vibe) and "Room 2" (Mechanical Vibe). The player can start exploring either rooms. The order does not matter.
- **Cable Puzzle:** In Room 1, we used pieces of string to represent the cables. The player has to physically test whether each string is long enough to connect the "Power Generator" to the correct plugs. The main constraint of this puzzle is cable length. At first glance, it seemed possible to connect any cable to any plug, but some wouldn't reach due to their limited length, while others were incompatible. Through trial and error, the player has to identify the correct connections to activate the generator. Once powered, a terminal appears, prompting the player to enter a code that has to be discovered in another room.
- **Cogwheel Puzzle:** In Room 2, we designed a cogwheel puzzle where the correct configuration depended on how the chains, again represented by strings, were pulled. The player must pull the chains in a specific sequence: first A, then B and C simultaneously (which requires utilizing the clone mechanic), and finally D. When the cogwheels are aligned correctly, a special item is unlocked. A code that the player would later need to solve the puzzle in Room 1.
- **Cross-Room Dependency:** We tested our plan for interconnected puzzles. The player had to find a paper token for the "Special item in Room 2" (a code) and physically bring it to the "Terminal" in Room 1. This tested our "Connected Rooms" design. This allows the player to start exploring their desired room, but prevents them from progressing unless certain dependencies are solved.
- **Temporal Clone Mechanic:** We modeled our core mechanic during the musical puzzle. The player (controlling the main robot cutout) would state their "recorded" action. The "computer" (the same player) would then place a second "clone" cutout on a musical platform. The player could then move their main robot to a different platform, simulating the co-op action needed to solve the puzzle. Additionally, we tested this mechanic for jumping to certain platforms that are unreachable with a jump.



(a) The prototype of "The Hub," showing the player character, platforms, and the musical platform puzzle.



(b) The prototype of "Room 1," showing the cable puzzles, platforms, and the player.



(c) The prototype of "Room 2," showing the gear-based puzzle, platforms, and the player.

Figure 7: Photographs of our physical paper prototype.

7 Experience

Our experience with the prototype was overall positive. However, physical prototypes come with certain limitations and do not allow us to explore all aspects of the game. They are, nonetheless, a great way to test whether our core ideas work. For us, playing with the prototype was especially enjoyable, as we imagined each room from the perspective of someone entering it for the first time, anticipating their thoughts, movements, and curiosity. It was exciting to experiment with the clone mechanic, watching the robot's recorded actions come to life in playback and creating a strong sense of cause and effect. Designing interconnected puzzles across different rooms added another layer of satisfaction, as actions in one area could influence outcomes in another. Of course, experiencing these mechanics as designers differs from how players would. Since we already know the puzzles and their solutions, we focused on creating challenges that weren't too straightforward and iterated on them to refine the gameplay experience.

What turned out to be harder than expected was the overall puzzle design process. Crafting puzzles that strike the right balance and are clever enough to be challenging, yet not so obscure that players feel lost or frustrated, proved far more complex than it first appeared. Translating ideas from the physical prototype to the digital version also introduced unexpected challenges: timing-based mechanics often lost their rhythm without instant input or sound feedback. Implementing the time-recording logic manually made it tricky to clearly distinguish between the recording and playback phases. Additionally, the lack of visual or audio feedback made it difficult for us to recognize when they had completed a puzzle, which can lead to confusion during this phase.

Our core mechanic, the cloning mechanism, however, made sense conceptually. We could demonstrate the core concept of record and replay, which served as the foundation of our puzzle design. We focused heavily on integrating this functionality into the gameplay in a way that felt logical and coherent, and in practice, it worked as intended. However, without real-time feedback, it is difficult for players to feel the timing and impact of their actions in a prototype. To make the mechanic more intuitive in the digital version, we plan to add visual and audio indicators, such as countdowns, echo effects, or timeline cues, that will help players better understand when recording or playback is occurring.

8 Revisions to Game Idea

With the prototype, we were able to confirm that our overall idea for the game is fun, and it works nicely. We had to iterate on our puzzle ideas multiple times, and we'll likely continue refining them in the future. One of our main focuses moving forward is implementing real-time visual and auditory responsiveness, as our puzzles rely heavily on timing and precise control, and this was not possible to test with the physical prototype. We also realized how strongly the quality of feedback can influence the overall game feel. To make the clone mechanic more straightforward, we plan to introduce visual indicators for the recording and playback phases, which will help players better understand the timing of their actions. As future work, we aim to first test existing puzzle mechanics with player feedback in an in-game prototype and, if time allows, design puzzles with more alternative paths, ensuring that solutions aren't too obvious and that players can experiment with different approaches to reach their goals.

Milestone 3: Interim Demo

9 Overview and Task Progression

Entering the Interim Demo phase, our primary objective was to transition from the physical prototype to a fully functional digital implementation. Our development schedule targeted the completion of the "Functional Minimum" layer and the "Low Target" layer.

We are pleased to report that we are **ahead of schedule regarding the core technical implementation**. The complex systems required for the "Temporal Clone" mechanic; input recording, deterministic playback, and state management, are fully operational and polished. We have effectively reached the "Desirable Target" for our gameplay mechanics.

However, regarding content creation and level design, we are currently utilizing "Tech Demo" environments. While we have implemented robust puzzle logic (pressure plates, musical drums, elevators), the final level layouts and environmental art are slightly behind the initial schedule. This was a conscious decision to prioritize a bug-free foundation. We believed that building levels on top of shaky mechanics would lead to wasted effort. Now that our core systems (Movement, Physics, cloning) are stable, we are confident that the level design phase will accelerate significantly, balancing out the schedule by the next milestone.

9.1 Completed Layers

We have finalized the following aspects of the game:

- **Core Physics & Movement:** A responsive character controller with acceleration, drag, and variable jump height.
- **The Cloning Mechanic:** A complete cycle of Hologram placement, Input Recording, and Deterministic Playback.
- **Puzzle Architecture:** A modular Signal/Receiver system allowing for pressure plates, doors, and moving platforms without custom code for every instance.
- **Camera System:** A dynamic Cinemachine 3.0 setup that handles room transitions and focus switching between the Player and the Clone.
- **Visuals:** Implementation of URP 2D lighting and post-processing effects to distinguish game states (e.g., Time Stop).

10 User Interaction and Gameplay Flow

The gameplay loop has been refined to emphasize "Solo Co-op." The player controls a robot inside a malfunctioning time machine. The core interaction loop is as follows:

1. **Hologram Phase:** Holding the clone button (R) freezes time and spawns a hologram. The player positions this hologram to select a starting point for their time loop.
2. **Recording Phase:** Upon releasing the button, control transfers instantly to the Clone. The main player character freezes. The game records the player's inputs (Movement, Jump, Interaction) for a set duration.

3. **Playback Phase:** Once the timer expires or the player cancels manually, time resets. A new Clone instance is spawned which replays the recorded actions. The player regains control of the main character and must cooperate with this playback to solve puzzles (e.g., passing a box to the clone, or having the clone hold a door open).

To improve usability, we added a "Watch" feature (Tab key) which allows the camera to pan over to the Clone during playback, ensuring the player can synchronize their actions even if the Clone is off-screen.

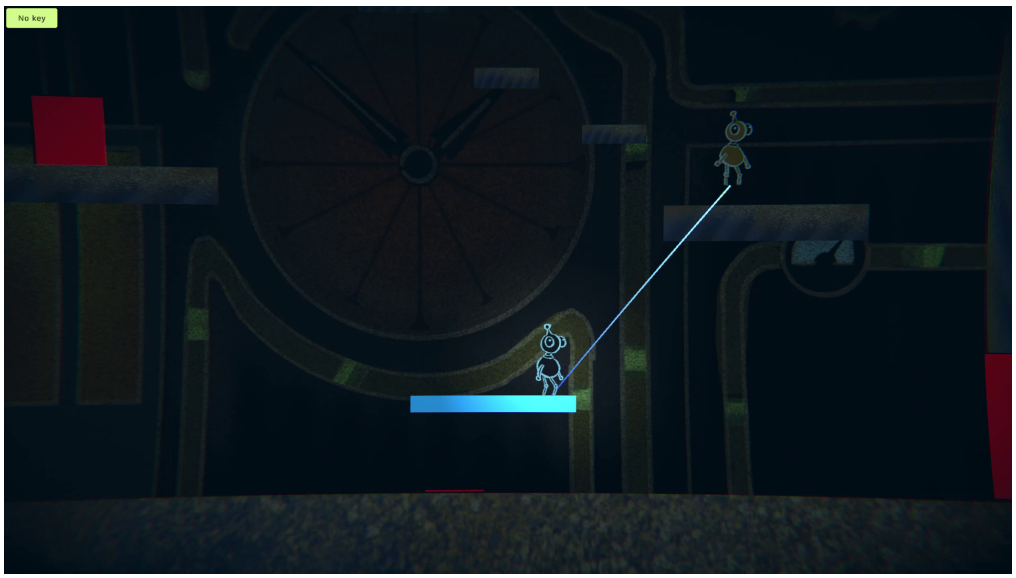


Figure 8: Creating and placing the clone (hologram phase)

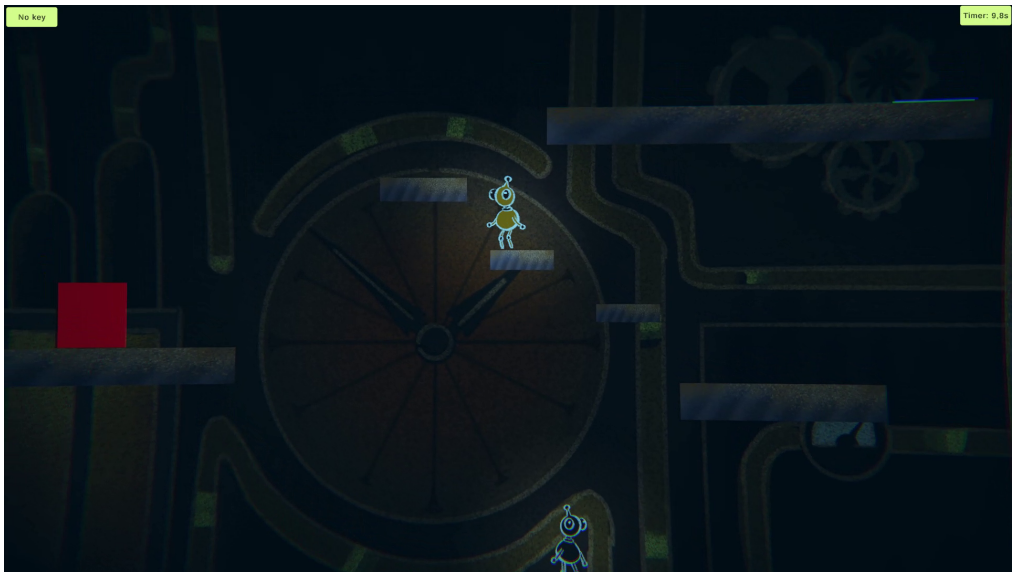


Figure 9: Clone in time stop & recording mode

Figure 10: Current state of the game showing the Player and the Clone cooperating to solve a puzzle.

11 Artistic Direction and Asset Production

Visuals are one of the core components of "The Last Iteration." We are aiming for a distinct hand-drawn Steampunk aesthetic that separates our game from standard tile-based platformers. Currently, the game utilizes a mix of assets and placeholders to test scale, readability, and level design.
The Steampunk environment will have multiple variants, depending mostly on the room. This will be done by editing the color curves and the hues of the art itself. As for the assets themselves, they are, and will be, hand-drawn in Procreate without the use of any AI.
Designing a steampunk environment brings many challenges, since it often needs a high level of detail. To keep the design readable and not to be disturbed by the game aesthetic, we intend to refrain from too detailed assets that are not important for the gameplay.

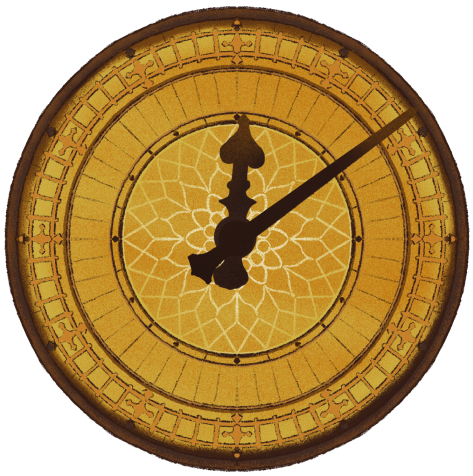


Figure 11: An example of a background element, a clock, to simulate the steampunk environment.

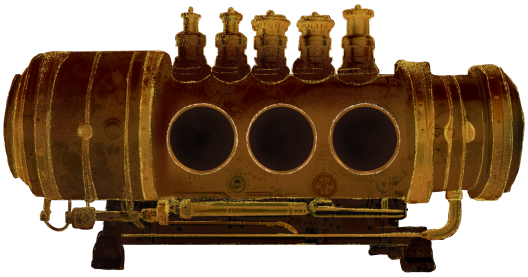


Figure 12: An example of an interactible element, a power generator.

The character animation pipeline has been established, with the main robot featuring distinct animations for Idle, Running, Jumping (split into launch, fall, and land for fluidity), and Pulling objects. The character itself is supposed to be a little bit contrasting against the background. Even further, the character animations should be more fluid and wavy - again, contrasting the environment.

Figure 13: Frame-by-frame animation of the clone walking.

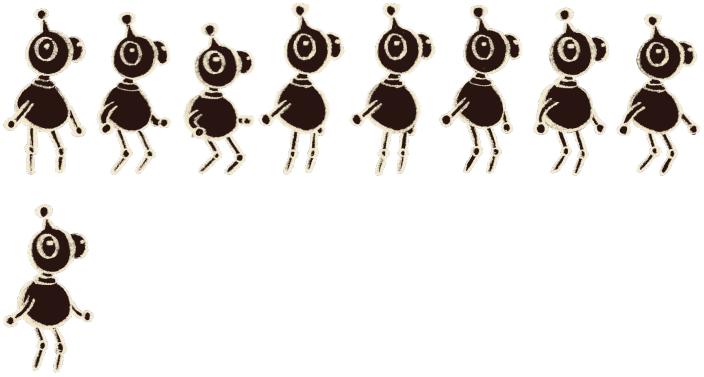


Figure 14: Frame-by-frame animation of the main character jumping.

The environment art focuses on a dark, industrial atmosphere using URP 2D lighting to highlight the interactive and important elements against the rusty background.

12 Technical Achievement: The Temporal Clone

The most significant technical hurdle for this milestone was the implementation of the recording and playback system. We iterated through several architectures to ensure stability and extensibility.

12.1 Architecture Refactoring: The Brain Pattern

Initially, our `PlayerController` script handled input, physics, and logic simultaneously. We realized this would make controlling the Clone impossible, as the Clone needs to be driven by data, not a keyboard.
We refactored the code into a **Puppet/Brain** architecture:

- **AgentMovement (The Puppet):** An abstract base class that handles `Rigidbody2D` physics, collision, and animation. It exposes public methods like `SetInputVector()` and `TryJump()`. It does not know where the input comes from.
- **PlayerInputHandler (The Brain):** Listens to the Unity Input System and routes commands to the `AgentMovement`.
- **ClonePlayback (The Robot Brain):** A separate script that reads saved data and routes commands to the same `AgentMovement`.

This separation allows the Clone and the Player to share 100% of the physics code, ensuring that a jump performed during recording behaves exactly the same during playback.

12.2 Data Recording and Drift Correction

To record the "Time Loop," we decided against recording transform positions every frame, as this ignores physics interactions (like pushing boxes). Instead, we record **Inputs**.
We created a `ReplayFrame` struct containing:

- `Vector2 MoveInput`
- `bool JumpPressed`
- `bool InteractPressed`
- `Vector2 Position` (For error correction)

During playback, the Clone applies the recorded inputs to its Rigidbody. However, Unity's physics engine is non-deterministic. Over 10 seconds, microscopic floating-point errors can cause the Clone to desynchronize (the "Butterfly Effect"). To solve this, we implemented a **Soft Sync** system. The Playback script compares the Clone's current position to the recorded position every frame. If a small deviation occurs, it gently nudges the Clone back on track. If a large deviation occurs (e.g., falling off a ledge due to a physics glitch), it snaps the Clone to the correct position.

13 Challenges and Solutions

13.1 The "Abstract vs. Concrete" Puzzle Design Challenge

One of the most significant hurdles we faced was the dependency between Puzzle Design and Game Feel. In our initial schedule, we planned to design complex puzzle rooms early. However, we discovered that designing time-loop puzzles on paper is vastly different from playing them.
Without the fully implemented mechanics, specifically the "feel" of the jump height, the speed of the clone, the clone recording system itself, and the friction of the boxes, it was nearly impossible to design tight, satisfying puzzles. Early designs often resulted in puzzles that were either trivially easy to "cheese" (bypass). This created a temporary bottleneck where Level Design had to pause while the Mechanics were refined. We realized that we could not effectively build the levels until the mechanics were solid. Now that we have implemented features like `Clone Recording`, `Jump Buffering`, `Coyote Time`, and the `Watch Clone` camera feature, we can empirically test puzzle timing. This has transformed our level design process from theoretical guessing into rapid, iterative testing, allowing us to catch up on the level design schedule quickly.

13.2 Character Animation and Physics Feel

One of the implementation challenges was making the movement feel "snappy" rather than "floaty," which is common in Unity 2D. We implemented a custom gravity system where the gravity scale increases when the player is falling or releasing the jump button early.
We also faced a visual issue where the character animation would "loop" incorrectly mid-air. We solved this by splitting the jump animation into three distinct states in the Animator State Machine: *JumpUp* (Launch), *Fall* (Looping), and *Land* (Impact). This state machine is driven by the Rigidbody's vertical velocity.

13.3 Physics Interaction (The "Heavy Box" Problem)

We implemented a physics-based Push/Pull mechanic using `FixedJoint2D`. A major challenge was balancing the mass. If the box was light enough to push, the Clone (who has the same mass as the player) would accidentally knock it over by walking into it.
Solution: We implemented a dynamic mass system. The box has a high mass (50) by default, making it immovable by simple collision. When the player grabs the box, we do not make the box lighter; instead, we make the Player heavier (Mass 50) via code. This allows the player to dominate the physics interaction while dragging, but prevents the Clone (Mass 1) from accidentally disturbing the puzzle pieces.

13.4 Input State Management

Managing the transition between controlling the Player, the Hologram, and the Clone was complex. We encountered bugs where the player would continue "sliding" after switching to the clone because the input vector was never reset. We resolved this by implementing a strict State Machine within the `PlayerInputHandler` that forces a `ForceStop()` command on the Rigidbody whenever control is transferred.

14 Design Revisions

As a result of our implementation experience, we made several revisions to the initial design:

14.1 Scalability Considerations

We shifted away from hard-coding puzzle logic. We implemented a **Signal System** using Unity Events. We created generic `PressurePlate` and `DoorController` scripts that can be wired together in the Inspector. This allows us to create complex logic gates (e.g., a door requiring two plates) without writing new code for every level.

14.2 Camera Mechanics

Originally, we planned a simple camera follow. However, testing revealed that players needed to see different parts of the level depending on where the Player/Clone was. We used **Cinemachine 3.0** and implemented an Event-Driven camera system. Triggers in the level broadcast events to the `PlayerInputHandler`, which dynamically prioritizes cameras based on whether the player is in the room or watching the clone via the "Watch" button.

14.3 Puzzle Design Adjustments and "Tech Demos"

We initially tested puzzles where the Clone would simply stand on a button to help the player. Dev-testing revealed this was boring for the recording phase. We tried to design the puzzle philosophy to be "Active-Active," where the recording phase requires the player to perform platforming challenges (fetching items, pulling levers) rather than passive waiting.
To facilitate this, we shifted our focus from building full levels to building "Micro-Puzzle Tech Demos." We have implemented and verified three distinct puzzle archetypes:

- **The Weighted Pressure Plate:** We implemented a filtering system (detecting "Heavy Objects Only" vs "Agents Only"). This prevents players from bypassing puzzles by simply spawning a light-weight clone on a heavy-duty exit button.
- **The Sequence Drum Puzzle:** A memory-based puzzle where the player and clone must jump on musical drums in a specific order. This was refactored from a hard-coded script into a flexible `SequenceManager` that allows us to drag-and-drop notes to create new songs instantly.
- **The Elevator Synchronization:** A co-op challenge where the player must hold a button to operate an elevator for the clone (or vice versa). We solved the issue of "waiting" by placing the elevator button in a location that requires either active traversal or giving the player early cancellation chance for the "clone recording phase" & "clone playback phase", ensuring the player is always moving.

These modular pieces are now ready to be assembled into the final map layout.

15 Next Steps

With the core technical risks resolved, our focus for the Alpha Release will shift to content creation. We will utilize our modular puzzle tools to build the full level layout, add the narrative elements like "Notes", and refine the visual aspects of the game.

Milestone 4: Alpha Release

16 Overview

The primary objective for the Alpha Release was to transition "The Last Iteration" from a collection of isolated mechanics into a cohesive, fully playable experience suitable for blind playtesting.
We are proud to report that we have not only met our **Layer 3 (Desirable Target)** goals but have also successfully implemented key **Layer 4 (High Target)** features, most notably the physics-based Swinging Mechanic. The game now features a continuous level structure, a complete narrative loop, and a robust tutorialization phase.
Furthermore, this milestone marks a significant visual evolution. We have replaced placeholder geometry with a complete set of custom assets, implemented new character animations, and constructed the full game map with a focus on environmental storytelling. The principal design is complete, and the focus has shifted toward system stability, audio-visual polish, and "game feel."

17 Technical Implementation & System Architecture

Since the Interim Demo, development focused on establishing a scalable architecture to support complex mechanics such as physics-based swinging and verlet-integrated cables. We moved away from standard Unity physics in key areas to ensure stability and determinism.

17.1 Physics-Based Chain Mechanics

A core gameplay feature (originally a High Target goal) introduced in this phase is the ability to swing from hanging chains. Initial prototypes using standard `FixedJoint2D` attachments resulted in significant instability due to the conflict between the player's collider and the discrete chain links. To resolve this, we implemented a custom state-switching approach in the `AgentChainSwing` system.

Figure 15: The player utilizing the kinematic chain swinging mechanic to traverse a gap.

17.1.1 Kinematic Mounting Strategy

We abandoned the reliance on continuous physics simulation for the player while attached to a chain. Instead, we adopted a Kinematic strategy:

- **Kinematic Transition:** Upon detecting a valid chain link via `Physics2D.OverlapCircle`, the player's Rigidbody is switched from `Dynamic` to `Kinematic`. This removes the player from the physics solver's force calculations, effectively eliminating erratic collisions or "jitter."
- **Manual State Management:** While kinematic, the player's position and rotation are manually synchronized to the active chain link in the `Update` loop.
- **Momentum Transfer:** To preserve the sensation of weight, we calculate the player's velocity vector at the moment of impact and apply it as an instantaneous `ForceMode2D.Impulse` to the chain link. This ensures the chain reacts naturally to the player's jump trajectory, transferring energy from the avatar to the environment.

17.1.2 Visual Decoupling

Directly parenting the player to the chain link caused visual harshness due to the rapid angular velocity changes of individual links. We solved this by decoupling the visual representation from the physics anchor. By using `Vector3.SmoothDamp`, the visual sprite interpolates smoothly toward the target link position (with a damp time of roughly 0.03 seconds), allowing the physics to remain rigid while the animation appears fluid.

17.2 Deterministic Time-Cloning System

The cloning mechanic requires the game to record the player's actions and replay them accurately. A major challenge was dealing with physics non-determinism over long recording durations.

17.2.1 Hybrid Replay Architecture

The `ClonePlayback` system utilizes a hybrid approach to maintain synchronization:

1. **Input Reproduction:** Instead of recording transform positions every frame (which consumes excessive memory), we record the inputs (Move Vector, Jump State) and simulate the movement logic via the same `AgentMovement` script used by the player.
2. **Drift Correction (StaticPrecision):** To combat physics drift, we record the player's world position at keyframes. During playback, if the clone deviates from the recorded position, a linear interpolation factor gently nudges the clone back to the correct path.
3. **Physics-Only Zones:** For interactions involving dynamic rigidbodies (e.g., elevators, pushable boxes), we implemented `ReplayZone` triggers that switch the clone to a purely physics-driven mode. This disables position correction to prevent the clone from clipping through moving platforms, ensuring they ride elevators naturally.

17.3 Event-Driven Input Architecture

To prevent input conflicts and "spaghetti code," we refactored the input layer into a centralized Event-Driven architecture using a `ScriptableObject`-based `InputReader`.
This acts as the single source of truth for all inputs. The `PlayerInputHandler` functions as a Finite State Machine (FSM), routing events based on the current context (e.g., `Normal`, `Swinging`, `ControllingClone`). This allows us to cleanly disable player movement during dialogue or cutscenes by simply muting the event channel, without needing to disable scripts on the player object itself.

18 Gameplay Content & Puzzle Engineering

With the core mechanics stabilized, the Alpha Release integrates four distinct puzzle archetypes. Each system was designed to be modular, allowing level designers to construct complex sequences without writing additional code.

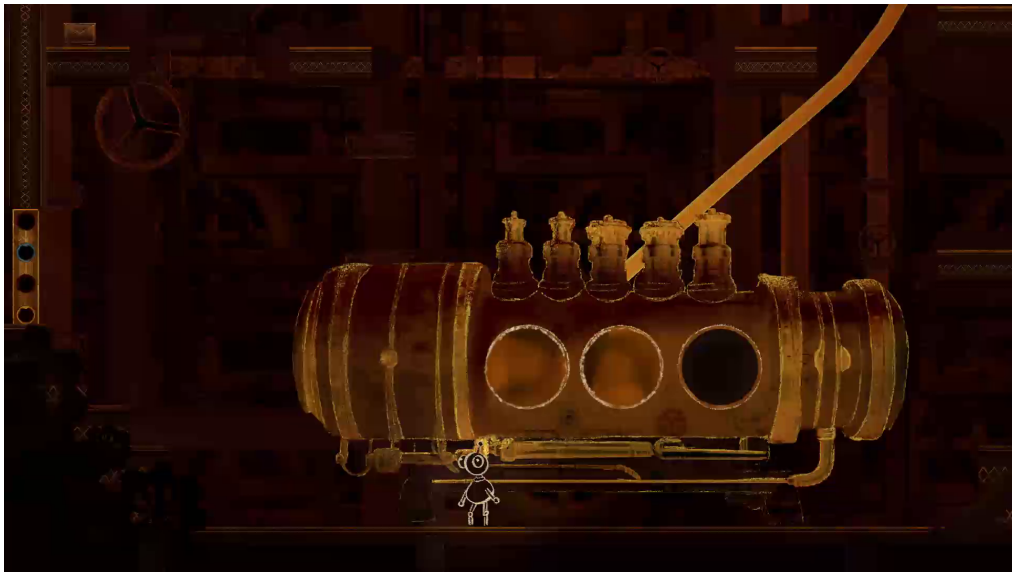


Figure 16: The Cable Puzzle system, utilizing Verlet integration for realistic rope physics.

18.1 The Cable & Socket System (Verlet Integration)

The "Cable Puzzle" represents our most complex physics interaction. Standard Unity joints proved too elastic for the precise "plug-and-play" feel we required.

- **Verlet Simulation:** To create realistic, flexible interactions for power cables and connections, we implemented a custom Verlet integration system `CableVerlet.cs`. Unlike standard transform-based ropes, this system simulates physics by calculating the movement of individual points (segments) governed by gravity, damping, and distance constraints. The script features real-time collision detection that allows cables to drape over environment geometry, bounce off surfaces, and dynamically react when grabbed or attached by the player. This provides a tactile, heavy feel to the game's mechanical puzzles, bridging the gap between functional gameplay objects and immersive environmental props.
- **Socket Logic:** To prevent physics glitches when "plugging in," the `CableSocket` script disables the cable's dynamic collider upon connection and switches the tip's Rigidbody to `Static`. This "locks" the cable into place, triggering the `CablePuzzleManager` to evaluate the circuit state.
- **Visual Feedback:** We implemented a `CableSplitEnd` procedural renderer that simulates "sparking" wires using particle systems when the cable is live but disconnected.

18.2 The Temporal Co-op Puzzles

These puzzles require the player to cooperate with their past self. The engineering challenge was ensuring that the "Clone" could interact with the world (pressing buttons, standing on plates) without causing physics conflicts with the live player.

- **Pressure Plates:** The `PressurePlate` script uses a filtering system ('FilterType') to distinguish between the Player, the Clone, and heavy physics objects. This prevents accidental triggering by debris while ensuring the Clone's mass is registered correctly.

- **Sequence Verification:** The `ButtonSequenceManager` and `MusicSequenceManager` systems enforce strict ordering of inputs. The "Bell Puzzle" requires players to memorize and replay melodies, while the "Button Puzzle" necessitates split-second timing.
- **Clone Synchronization:** These puzzles are designed to be unsolvable by a single actor. By integrating the `CloneRecorder`, players must "layer" their actions—recording themselves pressing the first set of buttons or bells, then playing along with their own recording to hit the remaining targets simultaneously. This effectively translates the "Co-op" genre mechanic into a single-player loop.

18.3 The Musical Bell Puzzles

The `BellPlatform` integrates gameplay with our audio engine.

- **Input Quantization:** When a player jumps on a bell, it triggers a musical note whose amplitude is scaled by the impact velocity. The `MusicSequenceManager` validates this input against a predefined melody, ensuring both rhythmic and tonal correctness.
- **Audio-Visual Redundancy:** To improve accessibility and reduce reliance on hearing alone, each bell note is associated with a specific color from the `GlobalLightingManager` palette. The solution sequence is displayed on the target door using `DoorVisualCue`, enabling the puzzle to be solved through visual pattern recognition as well as sound.

Figure 17: The Music Puzzle

19 Audio-Visual Architecture & Polish

19.1 FMOD Audio Integration

Audio production was approached as a systemic layer rather than a collection of isolated sounds. We migrated from standard Unity `AudioSource`-based playback to an FMOD-driven pipeline to support procedural audio behaviors, runtime mixing, and flexible iteration. The audio pipeline was intentionally split across tools:

- **Authoring (Ableton Live):** Music composition, sound design, and pre-mix decisions were performed externally, allowing precise control over tone, dynamics, and musical structure.
- **Middleware (FMOD):** FMOD was used to define runtime behavior, parameter-driven modulation, spatialization, and final mix balancing.

- **Engine Integration (Unity):** Gameplay systems trigger semantic audio events without direct knowledge of underlying assets.

This separation enabled rapid iteration and late-stage tuning without requiring gameplay code changes.

- **Decoupled Event Architecture:** An `AudioEventChannel` ScriptableObject acts as an abstraction layer between gameplay logic and audio playback. Gameplay scripts emit high-level requests (e.g., interactions, impacts, UI feedback), while the `AudioManager` resolves these into FMOD event instances.
- **Procedural & Physics-Driven Audio:** Continuous sounds attached to moving objects (e.g., elevators, draggable props) modulate pitch and volume in real time based on `Rigidbody2D` velocity. This reinforces perceived mass and motion without requiring multiple baked variations.
- **Non-Verbal Character Audio:** The robot character communicates through procedurally generated sound textures rather than traditional voice-over, supporting the abstract narrative tone while remaining expressive and reactive.

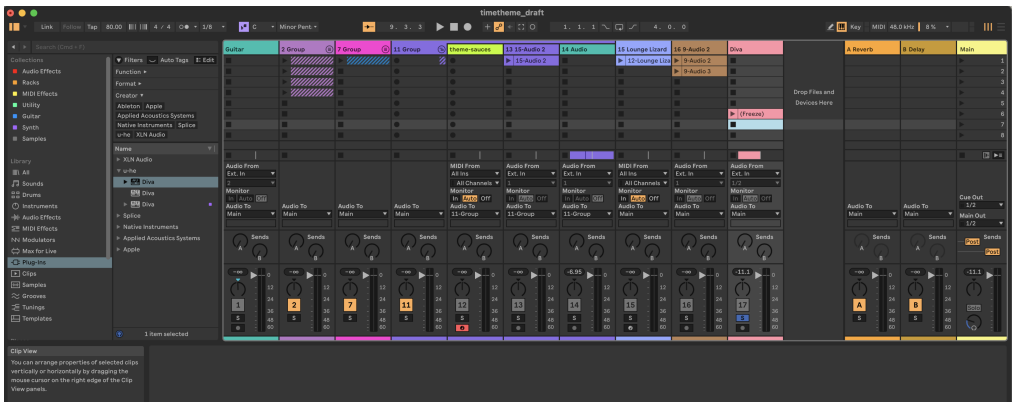


Figure 18: Ableton Live Project

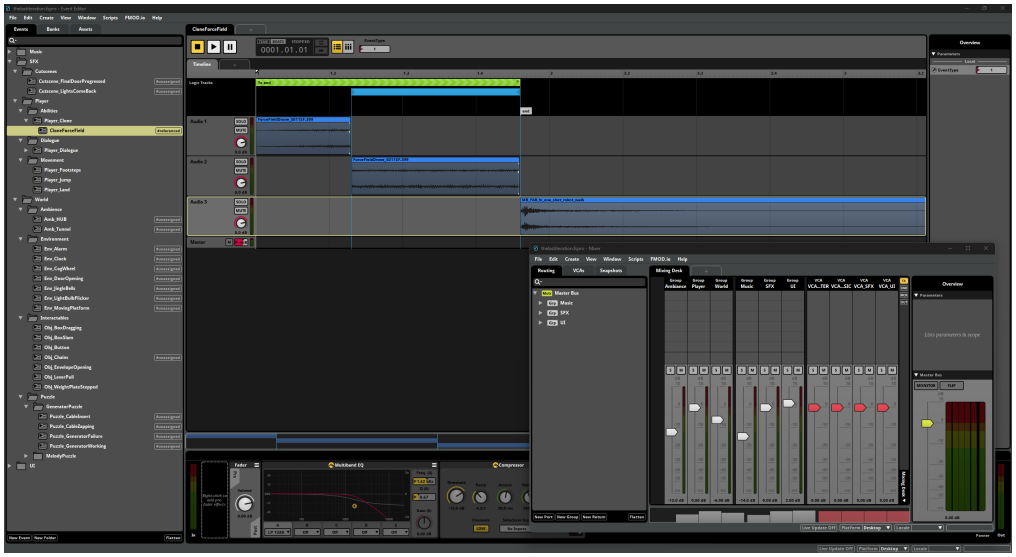


Figure 19: FMOD — Runtime Audio Middleware

19.2 Environmental Storytelling & Lighting

To guide the player without UI markers, we heavily utilized the Universal Render Pipeline (URP) 2D Lighting system.

Figure 20: The Global Lighting Manager shifting the world state from 'Blackout' to 'Powered', guiding the player visually.

- **Global State Management:** The `GlobalLightingManager` controls the ambient mood. It interpolates between "Blackout" (cold, dark blue) and "Powered" (warm, industrial orange) states using `DOTween` sequences. In the blackout mode, we subtly guide the player to the generator puzzle (the first objective) through the arrangement of emergency lights and closed gates.
- **Reactive & Atmospheric Props:** The visual polish serves two distinct functions. First, the `EmissivePulse` script provides atmospheric grounding, specifically applied to the "Giant Clock" in the central hub to create a rhythmic, living world state. Second, the `GeneratorAnimator` provides functional feedback; when a generator is repaired, it plays a "Startup Sequence" of tweened scales and particle emission, offering immediate positive reinforcement for puzzle completion.
- **Progressive Storytelling**: The system utilizes a sequence-based logic to synchronize narrative delivery with gameplay milestones. By tracking puzzle completion against the `requiredPuzzles` threshold, the GameManager dynamically retrieves localized content from Dialogue System to trigger correct dialogues and displays the correct content for the notes which are found in each room after a puzzle is solved. This ensures a linear narrative evolution where the world's "story state" shifts with the player's achievements, providing immediate narrative context to every solved puzzle, and makes the story progression independent from the order that the puzzles are solved.
- **Choice-Triggered Cinematics**: We used the DialogueRunner's OnChoicePicked event to catch the player's final decision in real-time. By identifying specific choice IDs—like "ending_fix_it" or "ending_break_cycle"—the GameManager seamlessly hands off control to the Unity Playable Director, playing the specific cutscene that matches the player's choice. This setup lets us jump smoothly from a dialogue choice straight into a nice cinematic to end the game, rewarding the player's journey.

19.3 Asset Integration & Map Design

In addition to the systems above, a significant effort was placed on the artistic pipeline. We integrated a full suite of custom assets to replace the grey-boxing of the Interim Demo. This included:

- **Full Map Creation:** A unified tilemap seamlessly connecting the tutorial area, the central hub, and the puzzle chambers.
- **Animation:** New sprite-based animations for the player character.

19.4 Tutorialization

Recognizing the complexity of the cloning mechanic, we implemented a context-sensitive tutorial system. The `TextProximityFader` smoothly fades in instruction text (e.g., "Press R to Clone") only when the player enters a specific trigger zone, reducing UI clutter and keeping the focus on the environment.

20 Design Changes since Interim Demo Phase

Since the Interim Demo, several key design pivots were made based on technical constraints and playtest data.

20.1 Unified World Structure

We transitioned from isolated test scenes (used for prototyping individual mechanics) to a single, unified gameplay scene containing the entire world map.

- **Scene Architecture:** To support the flow from the Main Menu to the Game, we implemented an asynchronous `SceneLoader`. This handles the initial transition into the gameplay scene without freezing the UI, but once loaded, the entire map exists in a single scene to minimize complexity and memory management overhead.
- **Persistent Managers:** A `ReferenceManager` was introduced to act as a central hub, allowing the other scripts to locate the Player and Input components immediately upon scene load without expensive `FindObjectOfType` calls.

20.2 Menu and UI Flow

We implemented a complete `MainMenuController` and `UIManager` to handle the game loop. This includes a functioning Main Menu with "Play" and "Quit" functionality, as well as an in-game Pause Menu. We added audio feedback (Hover/Click) to all UI elements to improve responsiveness.

20.3 Puzzle Complexity Adjustment

After adding the puzzles, internal testing revealed that the "Cable Puzzle" was initially too difficult due to the physics sensitivity of the cable. We iterated on the `CableVerlet` integration, adjusting the stiffness and collision layers to prevent the cable from getting tangled in the player's geometry. We also added visual feedback to the sockets to clearly indicate when a connection is successful.

21 Conclusion

The Alpha Release represents a "Feature Complete" state for *The Last Iteration*. The technical risks associated with the physics-based chain swinging and deterministic cloning have been resolved through the architectural patterns detailed above. All four puzzle complexes are implemented, the narrative loop is functional, and the audio-visual feedback is in place.
Our focus for the final milestone will shift entirely to:

- **Playtesting & Tuning:** Analyzing player behavior to refine puzzle difficulty and tutorial timing.
- **Bug Fixing:** Resolving edge cases in the whole gameplay loop.
- **More SFX:** Adding more sfx for what is currently missing.

- **Juice:** Adding final particle effects, screen effects, and polish to interaction feedbacks.

We are confident that the robust technical foundation laid during this Alpha phase will support a smooth transition to the final release.