

Developing narrative oriented systems for a point-and-click adventure in Unreal Engine 4

Jorge Montolio Conde and Wendy Despain

Executive Summary— the traditional point-and-click games from the 90s elevated narrative in games to a new level. Their witty dialog, amusing characters, and unique worlds, influenced gamers and developers alike. With these games in mind, this thesis aims at adapting the classical point-and-click narrative elements to a modern 3D engine. As a starting point, the researcher has created a game design of his own for a point-and-click adventure called “Rewind”. “Rewind’s peculiarity comes from the fact that players witness the game events in sections out of chronological order, making the level a narrative challenge. Considering Rewind’s narrative and story needs, the researcher has developed a series of narrative systems using Unreal Engine 4. An inventory, a notebook, and a dialog system, are just some of the narrative systems that the researcher has implemented over the duration of this project. These systems create a framework that designers can use to develop a whole game. The researcher also provides a short demo of Rewind, that shows the functionality of the systems, and their potential to be part of any narrative-oriented title for Unreal Engine 4.

Index Terms—point-and-click, narrative, game systems

I. INTRODUCTION

The point-and-click adventures developed during the 90s by companies like LucasArts, marked a before and after in the design of narrative games. Today, the point and click genre has faded away, but the influence of these games remains. This thesis takes a look at some of these old titles, along with more recent adventure games, and identifies some of the systems that they used to tell their stories. From their inventories to their dialog systems, the researcher aims at adapting these elements to the Unreal Engine 4.

In order to accomplish that, the researcher has created a game design for a sci-fi game called “Rewind”. In Rewind, players take the role of a detective that receives an emergency call from a research facility. Upon arriving at the facility, the researcher finds that nobody is there, and starts investigating the events that led to the call. To help him with his quest, the researcher has a unique tool known as the Rewinder, which allows him to enter other characters’ memories by analyzing their DNA. Using blood samples and other DNA that he finds during his investigation, the detective has to enter some of the scientists’ memories and solve the mystery of the empty facility. Rewind’s narrative is unique because players witness the game events out of chronological order. In the level, players start by finding memories that happened the night of the call, right before they arrived at the facility. As they progress, the memories become more distant, and players start to learn about the midday and morning events. Hence, Rewind’s narrative poses the challenge of not only telling a story, but also telling the story out of order.

For this project, the researcher uses Rewind as the model to implement a series of narrative systems, which include an inventory, and object highlight system, a series of narrative-oriented menus, and a dialog system, among others. The researcher also provides a demo of Rewind, that shows the systems’ functionality, and their potential to be part of other narrative-oriented games for Unreal Engine 4.

To create the systems, the researcher started by analyzing old and new adventure games, as well as looking into guidelines for creating narrative-oriented experiences. Taking this analysis into consideration, the researcher looked into different narrative-friendly game engines to use for the project, and ultimately decided to use Unreal Engine 4. After choosing an engine, the researcher made a list of the narrative systems to implement, along with their purpose and narrative goal. Finally, the researcher implemented the systems in engine, focusing on usability, ease of use for designers, and adaptability to other narrative games.

II. RESEARCH REVIEW

Point-and-click adventure games are one of the best exponents of the use of storytelling in games. Their use of dialog, peculiar characters, and unique stories, led to the creation of some of the most memorable adventures games from the 90s. The Mystery of Monkey Island, Syberia, or Grim Fandango, are just some examples of the success of narrative oriented adventures.

For the research review of this paper, the researcher analyzed best practices for the development of narrative games. In addition, the researcher looked into the evolution of point-and-click games, and some of the reasons that led to the near disappearance of the point-and-click genre. Finally, the researcher looked at games with a strong narrative component, including some of the point-and-click classics that helped shape the genre. The researcher also looked into some games that tell their story out of order, just like it happens in Rewind.

The researcher found the information by looking at books, Google scholar, as well as game developer blogs and websites like Gamasutra. For the games, the researcher used his personal experience with the point-and-click genre, and selected the games that inspired him during the design of Rewind and the planning of this thesis.

A. Literature review

Before creating a point-and-click system, it is important to know how to develop a good narrative. The following articles start by introducing best practices when creating a story-driven product. The review then moves on to point-and-click adventures, analyzing the things that made them great, as well

as the reasons behind the point-and-click genre's loss of popularity in recent years. The researcher found most articles through game developers' websites, as well as specialized game sites like Gamasutra.

1) *Creating better narrative games*

Former Blizzard developer B. Schwabb created the engine Storybricks to overcome artificial seeming game worlds, where the player is the center of all the stories. Storybricks takes into consideration all the characters in the world and their relationships, and creates an emergent narrative with branches that do not necessarily involve the main character. For that, the engine considers the character's goals, their personality traits, and the cost of their decisions. The result are stories that always have a reason to exist, even if the main character is not in them. That way, the engine avoids creating uninteresting stories, regardless of who its protagonists are.

Although Storybricks is no longer in development, its principle is still valid when creating narrative for games. By considering all of the characters' goals and motivations independently, developers can create a rich and more-natural narrative [1].

2) *Mixing gameplay and narrative*

On the topic of developing a good story, author Shirinian addresses the issue of merging gameplay and narrative. According to Shirinian, it is important to be careful when adding cinematic techniques to games, such as the popular Quick Time Events. According to her, extremely cinematic techniques create dissonance in the game, and affect the player's perception of the story. To solve this problem, Shirinian suggests not differentiating between gameplay and narrative. Developers, Shirinian says, should never let the player lose control of the character. Instead, the interactivity of games should always prevail, and players should be able to control their character regardless of the situation. This comes with its own risks, such as players not paying attention to important story events, or missing them completely. Shirinian suggests reaching a compromise between freedom and cinematography. According to her, players should have freedom of movement during cinematic moments, but the actual space where they can move should be limited [2].

3) *Dissonance in games*

On the topic of gameplay and narrative, author T. Lee points out dissonance as one of the main causes of failure when merging both elements. Like Shirinian, Lee recommends not separating narrative and non-narrative moments, saying that usually leads to a loss of immersion. Lee goes on to describe two types of narrative: the explicit story, and the player's story. The explicit story is the story that the game is trying to tell through its art, visuals and sounds, while the player's story is the story that the player gets to experience from the controlled character's perspective. In a perfect narrative, Lee says, both stories are equal. Lee explains that it is important not to draw a line between those two kinds of story, by making the narrative moments interactive. That way, the player gets to participate in all of the events, and consequently the story that the game is trying to tell and the story that the player is living become the same [3].

4) *Why people still remember the old Lucasarts point-and-click adventures*

In his article for Gamasutra, author Frank Cifaldi analyzes the legacy of the old Lucasarts point-and-click adventures. Cifaldi turns to professional developers, and asks them their opinion on the classics that shaped the graphical adventure genre. The creator of Dragon Fantasy's, Adam Rippon, mentions Day of the Tentacle as the game that taught him how to create funny scenes in a game. Chris Charla, from Microsoft, adds that the LucasArts games' influence can go beyond the game development field. Knowing a developer's favorite Lucasarts title, Charla says, can reveal details about the developer's personality. Many of the interviewees agree on citing funny dialog as one of the best characteristics of the old point-and-click games. Author Rusel DeMaria cites Loom, and Grim Fandango, as the pinnacle of the graphic adventure genre, citing their unique and funny story as the reason behind its greatness. Game Designer Jordi Fine attributes the old adventure games' success to a combination of funny dialog, memorable characters, and excellent writing. Finally, Alistair McNally from Bioware argues that it is the dialog branching, and the players' choices, that make these games excel. Old graphic adventures, Alistair says, triggered players' imagination, and forced them to experiment to find the solutions to the puzzles. The possibility to use creative solutions to solve the game challenge's, gave life to the games' universes, and fueled the player's interest and eagerness to explore.

5) *History of the Adventure Game Interface*

In her paper "Shaper Playing Experience in Adventure Games: History of the Adventure Game Interface" PhD Clara Fernandez-Vara performs an in-depth analysis of the history of adventure games' interfaces.

Fernandez-Vara starts by defining adventure games as those games that use the same entities for gameplay and storytelling. In adventure games, Fernandez-Vara says, story and puzzles have the same relevance. Fernandez-Vara mentions that adventure games face a constant struggle, between giving players multiple choices, and communicating those choices properly. As Fernandez-Vara points out, early text adventure interfaces gave players complete freedom to type whatever command they wanted. However, one of the disadvantages of these games was that they used text as the only communication channel. Later games fixed this problem by adding images to the text on screen, which took some of the narrative weight away from the text. The later addition of the point-and-click interface introduced a completely new system, where players could directly manipulate the game with a cursor. However, the point-and-click mechanics came with a reduction on the number of players' choices. Fernandez-Vara concludes her article by talking about the classic Myst, one of the first games that excelled at direct player communication and immersion. It's first person perspective, Fernandez-Vara says, allowed players to identify with the main character, while its lack of NPCs avoided the need for a dialog system that could have broken immersion. Finally, the lack of object descriptions enhanced realism, and forced players to investigate the world through the reading of books and notes.

6) Implementing branching dialog systems

Former Bioware's lead writer, Alexander Freed, talks about the topic of branching dialogue systems in his article for Gamasutra. According to Freed, branching dialogue systems are ideal for games where the player can customize their own character. By choosing their character's answer in every situation, players can craft their character's personality. Branching storytelling also supports games with complex storylines, because it introduces a new game mechanic based on discovering details about the story through the dialog.

Freed also points out that branching dialogue forces developers to overcome certain interface-related challenges. In a good interface, Freed says, players need to understand their dialog choices as quickly as possible. Developers also need to think about the potential need for "forced character lines", which are dialog lines that the main character says automatically. An absence of forced lines can make the main character seem artificial, while an excessive amount can alienate players from the character. The number of dialog options is also a key element in a dialog system. Adding too many options slows the game down, while adding too few takes control away from the player. Freed finishes his analysis with a mention to "timers" and "interrupts" as ways to add spontaneity to the dialog. "Timers" that force players to give quick dialog answers can create tension, but can also lead to hurried and unfulfilling decisions. "Interrupts", on the other hand, allow players to interrupt any NPC whenever the NPC is talking. Like "timers", "interrupts" spice up the dialog, but if abused, they can end up becoming an irrelevant gimmick.

7) Seven Deadly Sins of Adventure Games

Regarding the current state of adventure games, designer Adrian Chmielarz points to seven common mistakes that developers make when creating graphic adventures. According to Chmielarz, one of the biggest mistakes that developers make is adding unnecessary cinematic techniques to games. Techniques like movie-like credits, Chmielarz says, don't add anything to the game experience and can be tedious for players. Another typical mistake, according to Chmielarz, consists on including low-quality writing in the game. Bad writing not only breaks immersion, but can lead to misleading dialog and mixed signals to the player. Regarding immersion, Chmielarz also points to the puzzle design as a risky area in adventure games. Players, he says, should always be able to solve the puzzles through logic and creative actions, but never by trying to put themselves in the designers' shoes. Chmielarz also mentions a lack of internal logic as a source for player frustration. Whenever players cannot perform an action, he says, they must clearly understand why they are unable to do so. If the reason seems arbitrary or not consistent, players quickly give up on the game's puzzles. On the topic of players' choices, the author also talks about story branching, and how it is important to use it wisely. Games with excessive and useless branching, he says, don't add anything to the experience and may even hurt narrative and gameplay. Finally, Chmielarz talks about extrinsic rewards and how they are unnecessary in adventure games. According to him, things like point counters are

unnecessary additions, and lead to players trying to click everything to obtain the biggest reward possible.

B. Field Review

1) Syberia

Syberia is a 2002, 3D adventure game for PC. It tells the story of lawyer Kate Walker, as she arrives at an isolated town in France, to seal a deal for her law company. Unable to find the person who must sign the contract, Kate starts a trip around France and Russia, trying to get the signature she needs, so that she can go back home.

One of *Syberia*'s most remarkable elements are its HUD and menus. *Syberia*'s HUD is completely empty, except for the cursor that players use to move Kate. The clean screen lets players focus on the game, while adding to the realism of the experience. Regarding menus, navigating *Syberia*'s UI is an effective and satisfying process. The inventory serves as a HUB for all of the other menus, providing a single point of access to the UI. Inside the inventory, the game makes an important distinction between gameplay-relevant items and storytelling-related items, by putting them in different submenus. If players want to look for a gameplay-related item, they can stay in the regular inventory, while if they want to learn more about *Syberia*'s story, they can access the secondary "Documents Menu". All of *Syberia*'s systems try to relate to real life one way or another. *Syberia*'s inventory, for instance, looks like a metal case, where Kate keeps all of her items. Inside the case, Kate has a cellphone that players can use to call other characters. Players use this cellphone as they would use a real-life phone, by pressing the number keys and the call button. Even the dialog's UI has a real-life inspiration. When Kate is talking to another character, she writes all of the topics that she wants to talk about in a notepad. Players can then click on each note to select the topic that they want to talk about.

Regarding communication of important gameplay elements, *Syberia* uses a cursor that lights up when players hover over an Interactive Object. This system prevents players from being overwhelmed by the amount of objects on the screen. It also makes sure that not all objects are trivial to find, since smaller objects require more pointing precision from players.



Figure 1: *Syberia*'s inventory (left menu) and document menu (right menu)[9]

2) *Grim Fandango*

Grim Fandango is a 1998 adventure game for PC, and one of the classic adventures from LucasArts. The game follows the misfortunes of Manny Calavera, a skeleton salesman from the underworld, who sells trips to the afterlife for the deceased.

One of *Grim Fandango*'s most distinctive features is its unique setting, a strange underworld with Mexican motives and Art Nouveau architecture. However, *Grim Fandango*'s excellence comes from its characters and witty dialog. The main character, Manny, is a skeleton with a strong sense of humor, who always responds to players' input with a joke. His best friend is a strange orange monster; whose ignorance constantly puts the pair in danger. They are just a couple of characters in a world filled with giant cats, fire beavers, and pink one-eyed monsters. Regarding systems, *Grim Fandango* shines thanks to its simplicity. The game tries to avoid menus and interfaces as much as possible, so that players get into the game's universe and story. Players can see this simplicity in the inventory. Instead of having the traditional menu, *Grim Fandango*'s inventory is just a first person view of Manny's jacket. In this view, Manny personally holds the items out of his jacket for players to see. *Grim Fandango*'s control scheme is also unusual when compared to most adventure games, since it lets players move the character with the arrow keys instead of the mouse. The scheme gives players a more direct way of controlling Manny, which potentiates immersion and character identification. The only place where *Grim Fandango* remains traditional is in its dialog system. During a dialog with another character, players can select one of several options, which show up on the screen as text lines. After talking about a certain topic, the option for that topic may disappear, giving a sense of progress to players.



Figure 2: *Grim fandango's unique art style contributes to its uniqueness.* [11]

3) *The Room 3*

The Room 3 is a recent adventure game released for mobile devices in 2015. In the game, players take the role of an unknown character who is in search of the null element, a strange artifact that brings madness to those who encounter it.

If there is something that *The Room 3* shines for, that is its power for immersion. *The Room 3* uses the first person view, along with great sound effects and remarkable music, to create an unsettling and eerie atmosphere. The main character in *The Room 3* never speaks, which helps players identify themselves with the person that they are controlling.

The Room 3 is a narrative heavy game, where players constantly encounter written documents and notes directed at them. Since players spend the whole game alone, these documents are the only access that players have to the minds and motivations of the other characters in the game. Regarding game elements, *The Room 3* has a simple HUD that shows all of the player's available items on screen at all times. *The Room 3* can get away with showing the whole inventory on screen, due to the fact that players never get more than 4 items at the same time. Despite the simple inventory, players can still get detailed information about their items by clicking on them, which opens an auxiliary screen that lets players zoom on the objects. Finally, *The Room 3* is unusual in the sense that it doesn't rely on highlights to show important gameplay elements. Instead, finding relevant items becomes part of its gameplay. This mechanic is never frustrating or boring, since gameplay elements are usually visible enough for the observing player to find them. Players can also use a special "lens" in some instances. With the lens on, players can see glowing marks and symbols in the environment, which direct their attention towards important gameplay areas.



Figure 3: *The Room's simple HUD.* [13]

4) *Amnesia*

Amnesia is a first person horror game for PC, released in 2008 by Frictional Games. In *Amnesia*, players control a character who has lost all of his memories. Not knowing who they are or where they are, players embark in a quest to find out what has happened to them.

Amnesia bases most of its gameplay on exploration and hiding from monsters, mixed with adventure-style puzzles. In *Amnesia*, players have an inventory where they keep all of the items that they find throughout the game. Players have the option to use these items in the world in order to progress. For instance, at some point in the game, players have to use a hammer and a chipper in order to open a tunnel that brings them to the next area. The game offers the possibility to even craft new items by merging other items. As an example, players can

create a hand drill by mixing several drill pieces that they find in the game.

Although *Amnesia* is considered a horror experience, it has many elements characteristic of 3D graphic adventures. Unlike many other horror games, *Amnesia* is a very narrative heavy experience, and players constantly find notes written by the game's antagonist, Alexander, and by the main character's past self. Like many adventure games, *Amnesia* makes a UI distinction between the gameplay UI and the story-related inventory. To keep consistency and realism, all story relevant elements are inside a menu called the "Notebook". The "Notebook" is just another item in the players' inventory, and once players open it, they gain access to the main character's notes, diaries, and mementos.



Figure 4: *Amnesia's* inventory clearly separates gameplay objects from storytelling objects. [15]

5) *The Legend of Zelda: Majora's Mask*

The Legend of Zelda: Majora's Mask, plays with the idea of a non-chronological story through causality and the passing of time. In the game, the player gets to relieve the same three days repeatedly. Once the game reaches the third day, the player can use their Ocarina to go back in time to the first day. Because of this unique mechanic, the player gets to see the end of some subplots before viewing the beginning. For example, when Link talks to a farmer on the second day, she reveals that her cows have disappeared. The player can then go back in time to the first day, and find a group of aliens trying to abduct the cows. By presenting the consequences of an event first, the game triggers the player's curiosity. That way, the player travels back in time voluntarily, with the hope of finding out what caused the events in the first place. This method also provides positive reinforcement: once the player has solved a problem, he can see how his actions have changed the world. In the case of the farmer, the game rewards the player with a different ending to the story, where the farmer is happy because she gets to keep her cows, and rewards link with a bottle of milk. Like *Hotline Miami*, *The Legend of Zelda: Majora's Mask* uses environmental storytelling to reveal the passing of time. As time goes by, the player can look at the moon's size and get an idea of how close it is to the end of the third day. While the moon is normal sized in the first day, in the last day its threatening face covers the whole sky.

Because of its use of the environment and its play with causality, *Majora's Mask* is another good example of how playing with non-chronological structures can spike the player's interest and create an interesting narrative [16][17].



Figure 5: The size of the moon in *Majora's Mask* shows how much time has passed since the first day [18]

C. Summary

One of the main challenges of narrative-oriented games is merging gameplay and narrative in a way that doesn't detract from the game experience. Using quick time events, creating unnecessary branching of story, and taking over the player's character during cinematic moments, may not add value to the narrative, and can get in the way of gameplay. For some genres, it is important to understand and overcome these challenges. Point-and-click adventures use the same gameplay for puzzle solving and for storytelling, so it is essential for them to merge gameplay and narrative properly. Over the years, point and click adventures have adapted their interfaces and gameplay in order to tackle this problem. As a result, adventure games' interfaces have undergone substantial changes, from being text-based, to depending on direct manipulation of the interface through cursors.

Some point-and-click adventures have successfully engaged players with their narrative-oriented gameplay, simple UI, and high quality dialog. Thanks to its characters, its underworld setting, and its Mayan-inspired universe, *Grim Fandango* became an example of how to create a narrative-heavy game successfully. Other games, like *Syberia*, have also stood out thanks to their unique stories and user-friendly UIs. In *Syberia*, whenever the main character wants to talk, she goes through her talking points by reading from her notepad. *Syberia's* main character also has a cellphone, and a "Notebook" where she keeps her documents. This real life based UI, along with a simple HUD and unique story, results in an example of good integration of narrative-oriented interfaces with point-and-click mechanics.

Although point-and-click adventures are not as common today as they were in the past, many of their elements are present in games from different genres. *The Room 3*, a mobile game developed in 2015, provides a first person point-and-click adventure that focuses on immersing the player into its horror-based narrative. On the other hand, the horror game *Amnesia* has completely redefined the narrative genre, by adding strong storytelling techniques to an unnerving horror experience. While players run away from terrifying monsters, they still have

to gather notes, craft new items, and solve puzzles. All while they try to solve the mystery of its character's memory loss, and unveil his dark past life inside a mysterious medieval castle.

III. METHODOLOGY

This thesis takes some of the narrative elements of the classic adventure games, and implements them in the Unreal Engine 4. To do that, the researcher created a game design that serves as the basis for the system's designs. This game design, for a game called "Rewind", adds a twist to the classical adventure game, by including a mechanic where players have to enter other characters' memories in order to solve a mystery. In addition to this unique mechanic, players witness the game events in sections out of sequence, adding a new level of complexity to Rewind's storytelling.

For the project, the researcher started by analyzing the narrative elements needed to develop Rewind. With these elements in mind, the researcher looked at several 3D engines with strong storytelling potential, and chose Unreal Engine 4 as the engine that best adapted to Rewind's requirements. The researcher then made a list of the narrative systems needed to build Rewind. With the systems decided, the researcher started implementing them in engine. In addition to the systems, the researcher developed a demo of Rewind, with the goal of proving that the systems work as intended.

While building the project, the researcher focused on narrative, usability, and ease of use. Level designers can take the artifact created for this thesis and use it to develop "Rewind" in its entirety. For some of the most generic systems, this thesis has focused on flexibility. With some blueprint work, developers can easily adapt the inventory system, the highlighting system, the dialogue system, or the state system, to their own adventure games for Unreal Engine 4.

A. Rewind's Design Overview

For the purpose of this thesis, the researcher created a game design for an adventure game called Rewind. All of the systems developed for this thesis come as a result of Rewind's narrative and gameplay needs.

1) *High Concept*

Rewind is a first person graphic adventure, developed for Unreal Engine 4. In Rewind, players take the role of Secret Agent Cooper as he arrives at a seemingly empty research facility, to investigate a mysterious emergency call. As a secret agent, Cooper has the chance to use the Rewinder, a tool that allows him to enter other people's memories by analyzing their DNA. Using the Rewinder on the blood samples that he finds, Agent Cooper manages to enter the DNA owner's memories. These memories allow him to see the events of the day through the eyes of the now-vanished laboratory's scientists.

To help him with the investigation, Agent Cooper has an inventory where he can keep keycards and other artifacts that he finds in the laboratory. As any good detective would, Agent Cooper also has access to a notebook, where he takes notes

about his discoveries. By using the Rewinder, as well as gathering items to unlock the laboratory's machinery, Agent Cooper must get to the bottom of the emergency call that brought him to the laboratory.

2) *Players' goal*

- Main goal: Finding out what happened in the facility.
 - Secondary: filling up the timeline inside the Notebook menu.
- Main goal: Solving puzzles in order to progress
 - Secondary: Using different items, like keycards, to access new areas of the facility.
 - Secondary: Entering other characters' memories to gather information needed to progress.
 - Secondary: talking to other characters to get extra information and items.
- Main goal: Rescuing any person that may still be alive inside the facility.

3) *Rewind's story*

Upon entering the facility, Agent Cooper realizes that something terrible has happened. All lights seem to be failing, the air is not breathable, and the dead body of a scientist is sitting in one of the facility's rooms. After making his way through the debris, the investigator manages to collect some DNA, and accesses the memories of a scientist called Dr. Anderson. Inside the memory, the player discovers that the laboratory filled with toxic gases a few hours before the call. For some unknown reason, Dr. Anderson shut down all electrical systems in the laboratory right before he died; trapping everyone inside.

After learning Dr. Anderson's name, Agent Cooper gets a gas mask from Dr. Anderson's locker, which allows him to venture further into the laboratory. There, he finds the other scientist, Dr. Cook, who's trapped in the main control room. Dr. Cook assures the player that he does not know what triggered the tragic events in the laboratory, and asks the investigator to free him. In order to do that, Dr. Cook asks the player to help him restore electricity in the laboratory. Dr. Cook then tells players to access the scientists' rooms and get his keycard, which he can use to free himself. Although Agent Cooper believes Dr. Cook at first, he soon starts suspecting that the doctor is not telling the truth. At the same time, players start to unveil the kind of experiments that were taking place in the laboratory. In a vision that brings Cooper to midday, he takes the role of Dr. Anderson as he is incinerating some kind of human-like being. More research brings Cooper to the conclusion that the laboratory was specializing in creating human-like AIs, that the scientists sacrificed if they did not meet certain standards.

The level ends when players find the dead body of Dr. Cook. Players soon realize that the person trapped in the laboratory is not Dr. Crook, but an imposter. Using Dr. Crook's DNA, the investigator is finally able to witness the events in the morning of the tragedy. Seeing the past through Dr. Crook's eyes, the player unveils a love story between Dr. Cook and one of the AIs in the laboratory known as Eve. Knowing that the scientists were about to incinerate her, Eve tricked Dr. Cook

into freeing her that morning. After Dr. Cook freed her, she immediately killed him, and proceeded to release toxic gases in the laboratory, with the goal of killing as many of her captors as possible. However, Dr. Anderson turned off the electricity before the AI could carry out her plan, trapping her inside the control room. Hence the person to whom players have been speaking is not Dr. Cook, but the AI known as Eve.

4) *Gameplay Overview*

Rewind is a first-person, point-and-click adventure, and as such, it shares most gameplay elements with the graphic adventure genre. In Rewind, players explore a seemingly empty laboratory and interact with the objects that they find. Players can observe objects or try to use them by clicking on them. Using the left-mouse button also lets players pick up objects, that they can use later in order to progress.

Despite having many point-and-click elements, Rewind's main mechanic is a unique tool called the "Rewinder". With the "Rewinder", players can enter another character's memories whenever they find DNA of that character. This is useful for cases when players need to know information that only the laboratory's scientists would know. For instance, if players run into a console that requires a password, they can enter a scientist's memory and find out his password. Then, players can come back to the real world and use the password on the console, which allows them to progress.

5) *Gameplay elements*

Rewind has several types of objects that the player can interact with:

- Regular objects, that the player can use or observe, such as consoles, doors, computers, etc.
- Items, that the player can pick up. Once players pick up an item, the item goes to their inventory.
- Logs that the player can read. Logs give players insight into the thoughts of the other characters in the level. Once players pick up a log, it goes to their log menu.
- DNA samples. Players can use the rewinder on DNA samples. The Rewinder then sends them to a new level, that contains a memory of one of the game's secondary characters. Players get to play through that character's memory as if they were the character himself.

To communicate with these game elements and with the environment, players have several available actions:

- Walking with the WASD keys.
- Interacting with objects by using the left-mouse button. Whenever players interact with an object in the environment, the main character makes a comment about the object, which gives players a hint about the object's purpose. Interacting with the same object several times results in different comments from the character, which contributes to the feeling of realism. Some objects may also react to the players' interaction. For instance, a locked door may open after players interact with it.

- Starting a dialog with an NPC by using the left-mouse button.
- Using items from their inventory, by the 1-4 number keys. If players use the correct item on an object, the main character makes a comment and the object reacts to the interaction.
- Using the "Rewinder" on a DNA sample by pressing R. Once players use the "Rewinder" on a DNA sample, they teleport inside the memory of the character whose DNA they analyzed. Players then play as that character, until they reach the end of the memory and automatically go back to the real world.

6) *Menus*

Players also have access to several menus, most of which take inspiration on traditional adventure games:

- A HUD, that shows the players' information, as well as information about the game's controls, and the first four items of the inventory.
- An inventory menu, where players can see the items that they have picked up and their descriptions.
- A timeline menu, where the character writes down the events that took place in the research facility, in chronological order.
- A Note Menu, where the main character takes notes about his discoveries.
- A Log Menu, where players can read the text logs that they have picked up in the level.

B. Choosing an engine

1) *Engine requirements*

One of the first steps in the project was to choose the best engine to develop Rewind's narrative elements. The researcher started by creating a list of requirements that the engine should be able to fulfil, based on Rewind's level design:

- The engine must allow players to travel between different levels. Entering memories means making big changes in the map, so different levels are necessary to tell Rewind's story.
- The engine must allow for dynamic lighting. Some elements of Rewind, like the research facility's alarms, require dynamic lights.
- The engine needs to have some kind of NPC. At the end of Rewind, players meet one of the secondary characters, which requires NPC functionality.
- The engine must have a dialog system, or allow the researcher to implement a dialog system. Dialogs are an essential part of any adventure game.
- The engine must have a customizable inventory, or allow the designer to implement an inventory. Inventories play a big part in adventure games progression.
- The engine should offer the possibility to customize the UI, i.e., add overlays to the screen or show custom menus.

Based on this priorities, the researcher decided to look into several modern engines that have a strong narrative component: G.E.C.K. (Fallout 3), Creation Kit (Skyrim), HPL2 (Amnesia), Source Engine (Half-Life 2), Dragon Age Origins Toolkit (Dragon Age Origins), Unity, and Unreal Engine 3.

2) Results

The following is a list of the advantages and disadvantages of each engine, after the researcher created a small project in all of them:

Engine	Advantages	Disadvantages
G.E.C.K.	<ul style="list-style-type: none"> • Very powerful dialog system. • Includes inventory. • Many NPC models available. 	<ul style="list-style-type: none"> • Very combat oriented. • Medieval lore. Would require big changes to the story.
Creation Kit	<ul style="list-style-type: none"> • Easy to travel between different levels 	<ul style="list-style-type: none"> • Limited control over lighting.
Source Engine	<ul style="list-style-type: none"> • Allows travelling between maps. • Allows for lighting control. • Very versatile scripting thanks to logic entities. • Available futuristic assets that fit Rewind's story. 	<ul style="list-style-type: none"> • Very limited control over menus and overlays. • Requires inventory implementation from scratch. • Only four text channels to communicate with players. • Requires building a dialog system from scratch, with only four text channels.
HPL2	<ul style="list-style-type: none"> • Gives control over lighting. • Has some useful storytelling elements, like the notes and the notebook. • Has a voice-over system. • Allows for level switching. 	<ul style="list-style-type: none"> • No friendly NPCs. All NPCs are monsters. • Small amount of tutorials, which could be difficult development.
Dragon Age Origins Toolkit	<ul style="list-style-type: none"> • Good dialog system. • Allows for character customization. • Inventory already implemented. • Good amount of assets 	<ul style="list-style-type: none"> • Documentation and tutorials are scarce.
Unity	<ul style="list-style-type: none"> • Prefabs and C# facilitate creating any systems from scratch. • Many available assets. 	<ul style="list-style-type: none"> • Requires coding for almost any system.

	<ul style="list-style-type: none"> • Meets all other requirements. 	
Unreal Engine 4	<ul style="list-style-type: none"> • Blueprint facilitates creating systems from scratch. • Little to no coding required. • Level streaming allows for the creation of similar maps with small differences. • Researcher is very familiar with the engine. 	<ul style="list-style-type: none"> • Researcher would need to buy assets.

With the previous table in mind, the researcher decided to use Unreal Engine 4 for the project. Although Unreal Engine 4 doesn't provide any of the systems that Rewind needs, it fulfills all of the requirements set by the researcher, while other engines did not. What's more, Unreal engine 4 has an extensive library of tutorials and documentation, which assured that the researcher would be able to progress without major engine problems. Although the same could be said about Unity, the researcher favored Unreal Engine 4 over Unity due to his familiarity with the former.

C. Storytelling elements in Rewind

In Rewind, players spend most of their time alone. Although there is dialog in the game, most of the narrative comes from other sources, such as the environment or the main character. A big part of the design process for Rewind, was exploring the different narrative channels that would help players understand Rewind's story. Based on Rewind's story and the capabilities of Unreal Engine 4, the researcher decided to use the following narrative elements.

1) Character's comments: conveying the main character's personality

Because the player is alone for most of the game, a big part of the narrative weight lays on the main character. The main character is the only character that stays with players at all times. However, since Rewind uses a first person perspective, players don't know how the character looks, how he acts, or how he moves. As a result, the researcher had to find a different way of communicating the character's thoughts, personality, and motivations.

To address this problem, the researcher decided to create the comment system. Like in many point-and-click adventures, players can click on many of the objects in the environment. Every time they do so, the main character makes a comment about the objects that players have selected. From his comments, players can get snippets of Agent Cooper's personality and train of thought. For instance, players may find a dead body and a gas mask close to the body. If they click on the mask, the main character may say "I'll borrow this, I don't think he'll need it anytime soon". From that line, players can imply that the character has a dark sense of humor, or that he is slightly cynical.

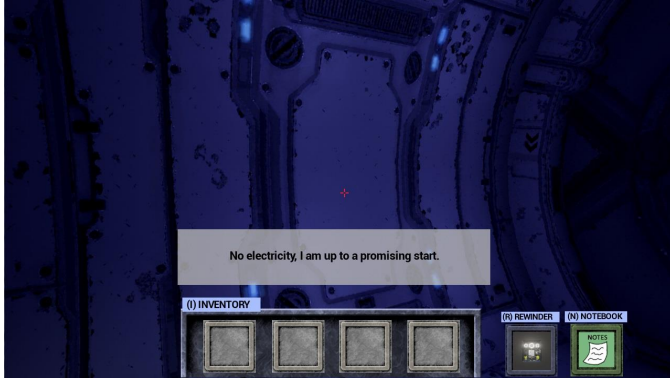


Figure 6: The main character's comments give hints about his personality.

2) Environmental storytelling through memories

Environmental storytelling is also a big part of Rewind's narrative. Rewind tells its story in out of order sequences, so players usually see the end of an event before they witness how it started. In order to accomplish this unusual storytelling structure, Rewind resorts to the secondary characters' memories. Entering secondary characters' memories allows players to see events from a different perspective, revealing new details that players did not know. For example, during the first part of Rewind, players enter a room that is filled with toxic gas. They then enter a character's memory, where they see how the laboratory's security systems released the gas as part of their "Purge Protocol". Hence players first witness the effects of the Purge Protocol, only to find out about the origin of the protocol later in the level. In another instance, players go into a room where the door is completely destroyed. After investigating, players enter another characters' memory, where they see the character destroying the door in order to escape from the toxic gas that is filling the room.

The environmental storytelling is not limited to the memories, and some environmental cues tell a story of their own. For instance, in some areas of the level players see blood on the floor, and they can instantly tell that someone is hurt and probably in current danger. In some other areas players may find that the lights are not working, or that doors don't open, so they immediately know that there was some kind of electric problem in the facility.

3) Telling the laboratories background story through the consoles

Environmental storytelling is useful for present and recent events. However, players also need to learn about story details that happened before the beginning of the game. What is the laboratory's purpose? Who founded the laboratory? How many scientists worked there and what was their background? To answer some of these questions, the researcher decided to create the consoles. The consoles are electronic displays that players can see all around the laboratory. In the game's fiction, the purpose of these consoles is to keep scientists updated about the most recent events happening in the laboratory. For instance, one console may say "Dr. X from Research Facility 3 has

received the Brainias award. After a breakthrough in his neural-networks research...[etc]". Reading these consoles gives players information about the laboratory's day-to-day, its scientists, and the research that they conducted there.



Figure 7: Players can find consoles in several locations.

Not all of the consoles are purely informative, and some of them are also interactive. For instance, at some point in the level, players have to use a console to restore the laboratory's electricity. Although this console has a gameplay purpose, it can also give story hints like the other consoles do. For instance, the message "Purge protocol activated. In order to restore electricity insert Dr. Y's code" may show up in the console when players interact with it. This message tells players that something activated the "Purge Protocol" in the laboratory. What's more, players may even deduce that Dr. Y is actually someone important in the laboratory's hierarchy, since his code can restore electricity to the facility.

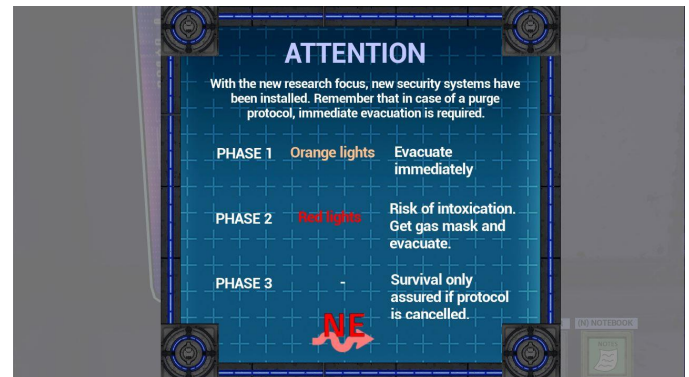


Figure 8: Players can read the console's content by clicking on it.

4) Text logs: conveying the secondary characters' story and personality

Conveying the personality of Rewind's secondary characters was essential to the story, since the events in the laboratory unfold as a result of the secondary character's actions. However, some of the characters are already dead or missing when the game starts. To address this narrative challenge, the researcher decided to create the text logs. Text logs are small tablet-like objects that contain pieces of text written by secondary characters. Scientists use the text logs for all kinds of

purposes, from recording details about their research, to sending emails or writing diary entries.



Figure 9: Text logs are pda-like objects that players find around the level.

Depending on the log, the author may be talking about personal matters, or science related issues. Players may find a log where a scientist talks about his frustration after his research experiments fail. They may also find another log that a scientist used as his personal diary, where he talks about his feelings towards one of the laboratory's AIs. Whatever the content of the log may be, logs always convey a secondary character's feelings and thoughts to the players.

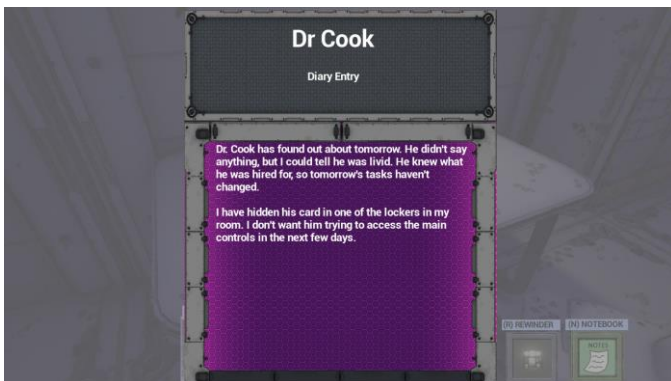


Figure 10: Picking up a log opens the log screen. Players can go back to this screen later.

5) Conveying the difference between “real world” and “memory world”

Rewind's mechanic is not common, since players constantly switch between the real world and characters' memories. Successfully conveying the difference between a memory and the real world was one of Rewind's biggest challenges. The researcher decided to use redundancy to convey this idea to the player, by communicating it in several different ways:

- Every time players use the Rewinder on an object that is not DNA, the main character makes the comment “I can't use the Rewinder here. I need DNA to enter someone's memory”. The comment reminds players about the Rewinder's purpose and use.

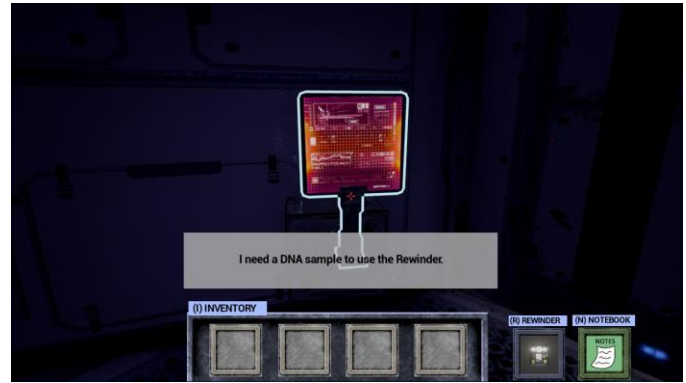


Figure 11: A message reminds players how to use the Rewinder, when they use incorrectly.

- When players use the Rewinder on a valid DNA sample, a screen opens and displays the message “Retrieving character X's memories”. After some time, the message changes to “Connection successful”. The message tells players that they are connecting to someone else's memory.

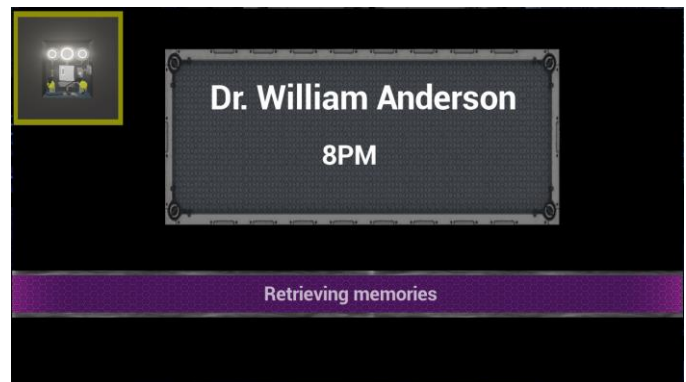


Figure 12: A message on screen tells players that the Rewinder is retrieving another characters' memories.

- On the screen that opens when players use the “Rewinder”, players can also see the memory's owner name and the time of day when that memory happened.
- Once players enter a memory, a post-process effect helps them understand that they are not in the real world anymore. The post-process effect is similar to the visual effects used in movies to indicate that a scene is happening in the past. The effect serves as a hint to tell players that they are inside someone's memories.

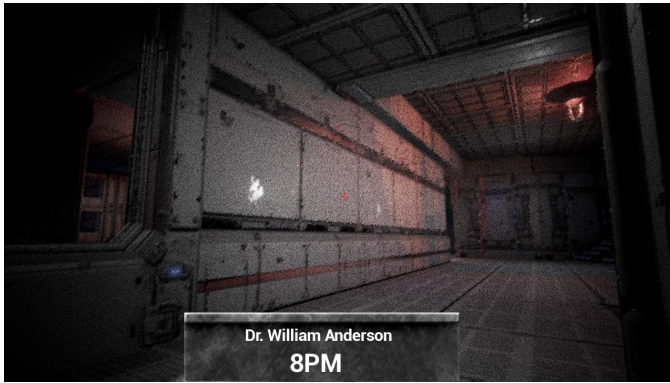


Figure 13: A post-process effect helps players differentiate between the real world and a memory.

- When a memory ends, a message saying “Losing connection to memory” shows up on screen.

D. Systems Overview

The final step before creating Rewind’s demo was to list all of the system that the game required. The researcher used his knowledge about Unreal Engine 4, as well as the narrative requirements of the game, to make a list of all the systems that Rewind requires.

1) System requirements for Rewind

Requirement	Solution
The main character can make comments about any Interactive Object.	Comment system
Players can interact with objects in the environment.	State system
The same object can react to the players’ interaction in different ways, based on the players’ progress and inventory.	
Players can see key information on screen at all times.	HUD
The state of every object in the level needs to get saved when players travel to another level.	Saving system
Dialog needs to get saved when players go to another level.	
Once players come back from a memory, the main level needs to load and be in the same state that it was when they left.	
When players come back from a memory, they need to spawn on the same location that they were at when they left.	Spawning system
When players come back from a memory, they have to be able to make comments on the memory they just visited.	

When players come back from a memory, some events may be added to the timeline, based on what players learned inside that memory.	
Players need to be able to know which objects they can interact with.	Highlight system
Players must be able to access a timeline, that shows them the events that happened in the facility, in chronological order.	Timeline menu
Players must be able to access the notes written by the main character.	Note menu
The level designer must be able to create logs easily.	Log menu
Players must be able to read the logs that they pick up as many times as they like.	
Players must be able to have a conversation with NPCs.	Dialog system
Players’ conversations with NPCs needs to feel real.	
Players’ conversations with NPCs need to adapt to the players answers and follow a critical path.	
Only one widget should be visible on the screen at the same time, including menus, consoles, and panels.	Widget controller
There should be a way for external level designers to control the system’s easily.	Easy Access Library

2) *Most relevant entities in Rewind*

All objects that the player can interact with belong to the blueprint class “Interactive Objects”. Based on their functionality, the objects in the level also belong to different children classes that inherit from “Interactive Object”. That means that all objects have common properties and functionality, coming from their “Interactive Object” parent class, and some additional properties that come from their child class, if they have one.

The following are the most common and relevant classes in the level:

Class	Parent Class	Description	Properties
Interactive object	-	Any object that the player can interact with.	Reacts to player interaction.
Info Panel	Interactive Object	TV screens that contain announcements, and information about the research facility.	A user widget pops up on the screen whenever players interact with info panels. Players can close info panels by clicking anywhere on the screen.
DNA sample	Interactive Object	DNA samples, usually blood drops, that are scattered around the level.	Players can use the “Rewinder” on these objects to enter someone else’s memory.
Doors	Interactive Object	Normal doors in the laboratory.	They are able to open and close when the player gets close to them.
Text logs	Interactive Object	Text logs, written by secondary characters.	When players pick up a log, a widget displays the logs content on the screen.
Items	Interactive Object	Items that the players can pick up.	Items go to the players’ inventory, and disappear from the world after players pick them up.

E. The highlighting and commenting system

The need for a highlight system showed early in development, as players became overwhelmed by the amount of possible Interactive Objects in the world. To solve this, the researcher took an approach similar to the one found in *Syberia*. Whenever players are close enough to an Interactive Object, the object shows a highlight around it, indicating that the players can interact with it.

Only one object can be highlighted in the level at the same time, and players can only interact with that object.



Figure 14: Highlight of an Interactive Object, as seen in-game

1) *Action triggers*

The highlight system uses what the researcher called “Action Triggers”. Action Triggers are in charge of activating and deactivating the highlights, based on the players’ position and camera’s normal vector. An action trigger has three main components, as seen in the following caption:

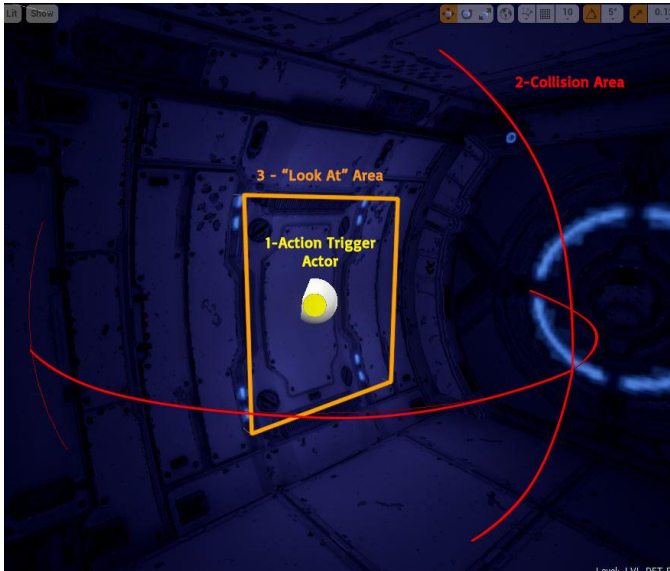


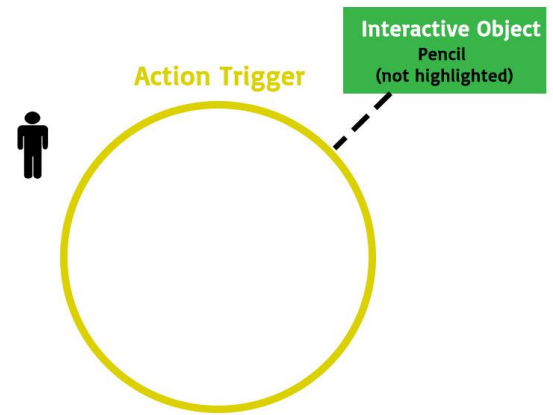
Figure 15: View of an action trigger in the engine

1. The researcher used a small yellow sphere to represent Action Triggers in the world. This is a purely aesthetical decision, aimed to help designers identify the action triggers in the level.
2. The collision area is the area where players need to be for the highlight to activate. If players are outside of this area, the object connected to the Action Trigger doesn't highlight.
3. Players need to be looking at the "Look At" area for the highlight to activate.

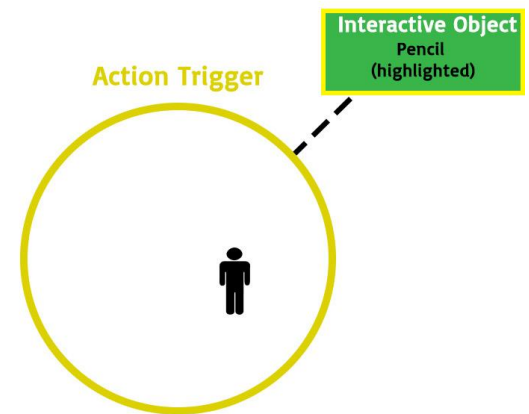
The different components of the Action Triggers allow designers to completely customize the highlight for every object. By changing the "Collision Area", designers can decide how close a player needs to be to an object, in order to interact with it. With the "Look At" area, designers can force players to look at a certain spot in order to interact with an object. That way, if designers want to hide an object, they can create an Action Trigger with a small "Look At" area. If they want to make an object obvious to players, they can use an Action Trigger with a "Look At" area that is difficult to miss.

2) Uses of the highlight system

Action triggers are linked to the Interactive Object that they highlight. As such, they control when a player can interact with an object. If players are inside the area of an Action Trigger, then they can use the Interactive Object connected to it. If they are outside of the area of the trigger, then they cannot interact with the linked object.



Player cannot interact with Pencil



Player can interact with Pencil

Figure 16: How action triggers control object interactions.

Several Action Triggers can be connected to the same Interactive object.

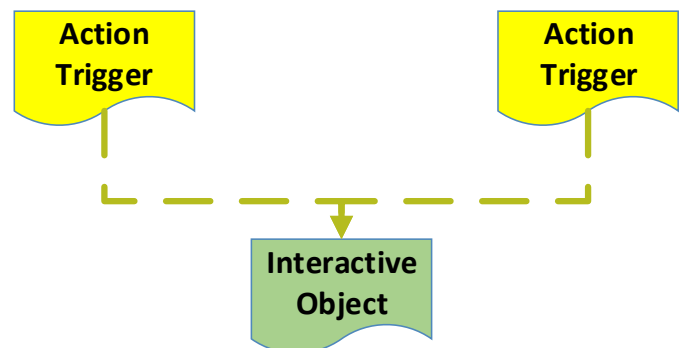


Figure 17: How action triggers connect to Interactive objects.

Thanks to that, designers can link the same behavior to different areas of the level. For instance, in a room with twenty coffee mugs divided among several tables, the designer may want all the mugs to behave the same. To do that, he can create an Interactive Object that contains the mug behavior. He can then create twenty Action Triggers, one per mug, and connect all of the triggers to that Interactive Object.

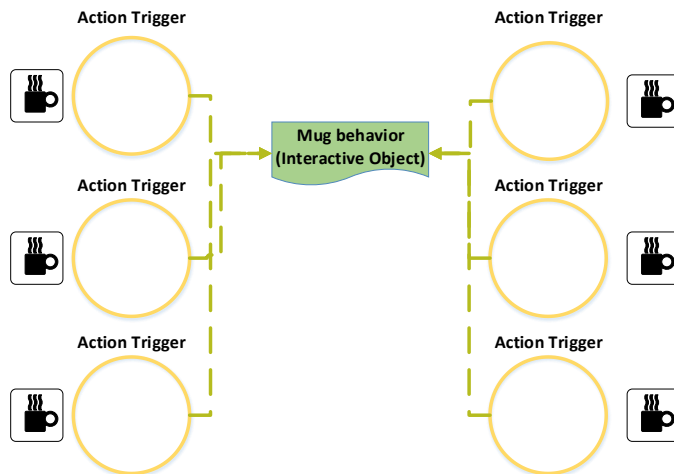


Figure 18: Designers can assign the same behavior to several objects by using the action triggers.

Using this method, designers can also centralize the players' reactions to the objects. Let's say, for instance, that designers want the main character to say "This is an ugly mug" when the player interacts with any of the mugs. However, if the player interacts with two different mugs, the designer doesn't want the main character to say "This is an ugly mug" twice. Instead, the designer wants the main character to say "This is an ugly mug" for the first mug, and "Yeah definitely a hideous group of mugs" for the second mug. With the use of action triggers, this becomes a trivial task, as the following diagram shows.

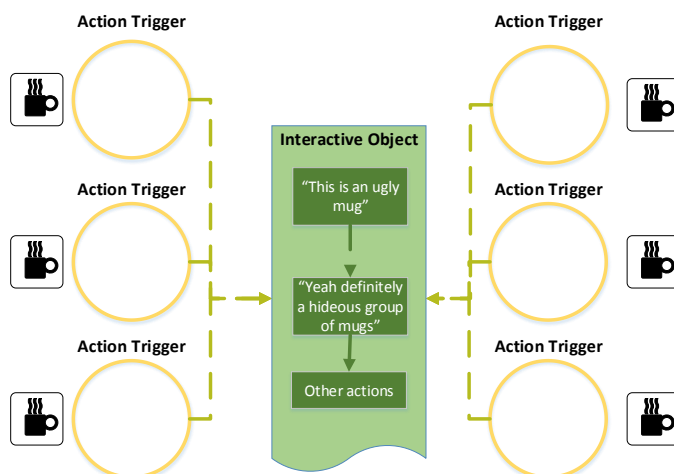


Figure 19: Visual representation of the mug example.

In the game demo, the researcher used this method to control the gas mask containers. All of the mask containers are in different positions, but their behavior comes from a central, invisible, Interactive Object. That way, the containers highlight separately, but they behave as one entity.

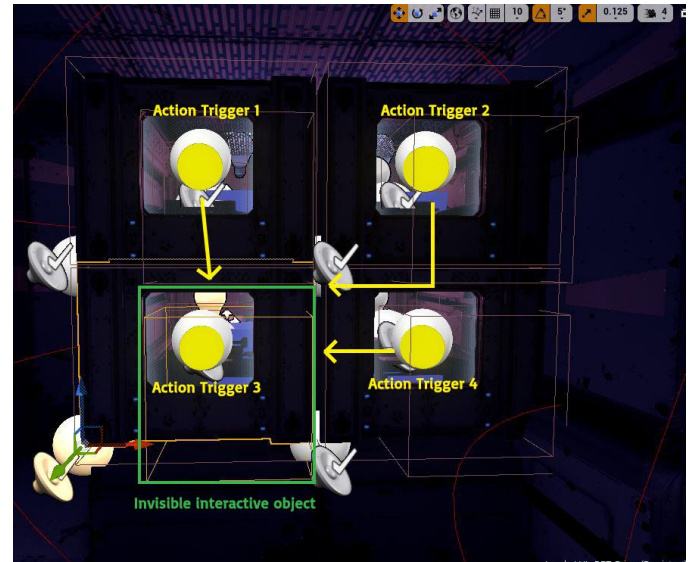


Figure 20: Use of multiple Action Triggers in the demo

The Action Triggers also allow designers to highlight additional non-Interactive Objects, such as static meshes and skeletal meshes.

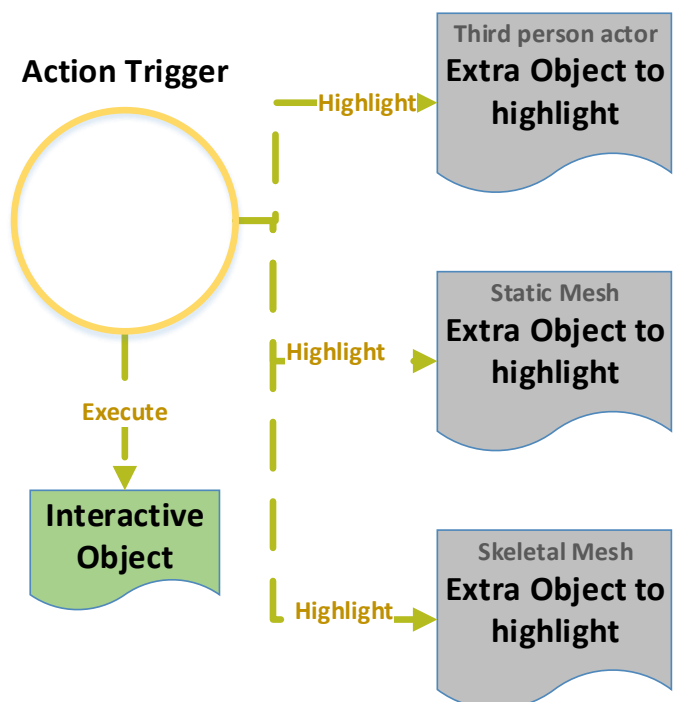


Figure 21: Action Triggers can show a highlight on objects that are not interactive.

This functionality is useful when designers want to highlight an actor that is not interactive. In the previous mug example, the designer would not want the invisible Interactive Object to be highlighted, but instead he would want to highlight a specific mug. He can accomplish this by using the “Extra Object to Highlight” property.

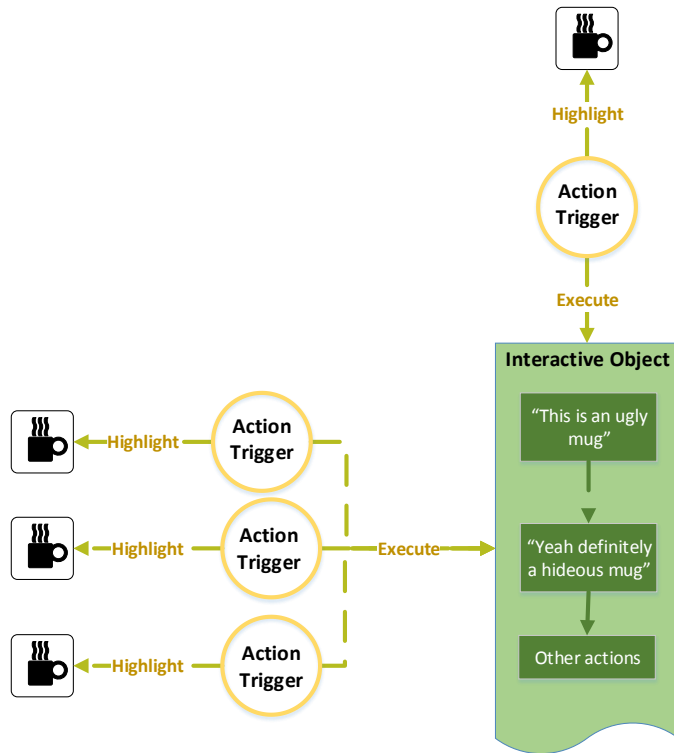


Figure 22: Example of highlighting external objects

Finally, action triggers can be disabled after an interaction. This is useful for objects like the pickup items, which disappear after players interact with them.

3) Advantages of the highlight system

Thanks to the Action Trigger structure, the highlight system gives designers a lot of freedom when working with Interactive Objects. Some of the advantages of the system include:

- Designers can “hide” objects by making their action triggers small.
- Designers can make objects more visible to the player, by increasing the size of their Action Triggers.
- Designers can control the behavior of several objects with the same entity, while still highlighting the objects individually.
- Designers can highlight gameplay-relevant objects that are not interactive, like static meshes and skeletal meshes.

4) Current limitations and possible solutions

Despite its flexibility, the highlight system has certain limitations. The Collision Area of the Action Triggers is always a sphere, so using other shapes requires changes in the Action Trigger. To use cubes, cylinders, or combinations of shapes, the

designers would have to make small changes in the Action Trigger blueprints.

Additionally, Action Triggers connect with Interactive Objects through a simple object reference variable. If a designer links an Action Trigger to an Interactive Object, and then moves the Interactive Object to another position, the designer has to move the Action Trigger manually. With more time, the researcher could make the Action Triggers inherit from Actor Components, so that designers could add the Action Triggers as a component in other actors.

F. The HUD

When developing the HUD, the researcher focused on balancing information conveyance with screen cleanliness. The HUD went through several iterations before the final implementation. The researcher used player feedback to know which information was useful for players, as well as which areas of the screen were best to display that information.

1) Considerations when building the HUD

The researcher set the following priorities, to use as guidelines when building the HUD:

Consideration	Goal
Many adventures games show the inventory in the HUD, for players convenience.	Let players see some of their inventory items in the HUD, so that they don't have to go to the inventory every time they need to use an item.
In Rewind, players can control different characters. When players enter a character's memory, they start playing as that character.	Display information about the character that the player is currently controlling.
Players may forget what the Rewinder is, and how to use it.	Show the Rewinder on the HUD at all times.
Access to the Notebook should be straightforward, otherwise players might not use it.	Show the Notebook icon on the HUD.
The game controls involve both the keyboard and the mouse, so players may forget some of them.	Display the controls on the HUD, in a subtle way.

2) First iteration of the HUD

Following the guidelines listed in the previous section, the initial HUD had the following layout:

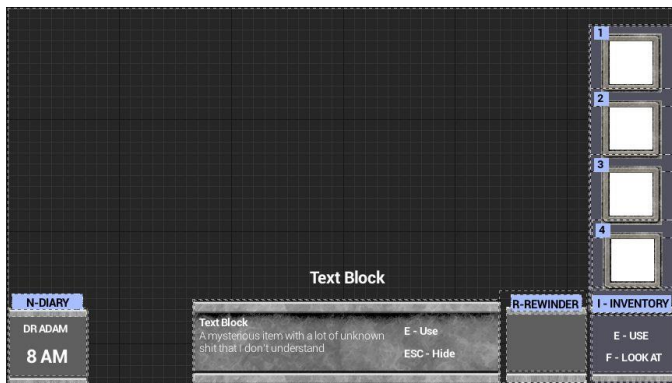


Figure 23: First iteration of the HUD with visible inventory

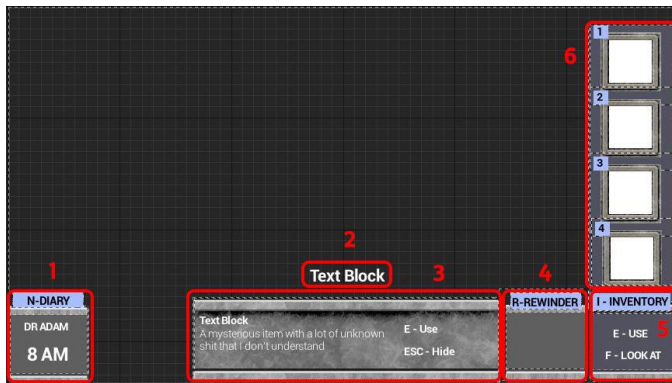


Figure 24: First iteration of the HUD, with hidden inventory.

1. Information about the character that the player is currently controlling. It also reminds players how to open the “Diary” (early name for the [Notebook](#)).
2. The main character’s comments appear on this text entity.
3. When players select an item from their inventory, the name of the item and a description appear on this bar.
4. A panel to remind players about the Rewinder.
5. A panel to remind players about the basic interaction controls, as well as how to open the inventory.
6. The inventory was closed by default, to avoid cluttering the screen too much. To open the inventory (panel 6) players just needed to press “I”. If players pressed “I” again, the panel disappeared again. Once the inventory opened, players could use any items in it by pressing a number between 1 and 4.

The initial implementation of the HUD had several issues, that became evident after playtesting.

Issue	Solution
Because the inventory was usually hidden, players forgot about it frequently.	Keep inventory visible at all times.
There was too much information on the screen, in several different places that were far from each other.	Condense information by removing some of the HUD panels.
Players found panel 3 to be confusing and not that useful.	Remove the panel from the HUD.
Even with the HUD information, players struggled using the controls.	Simplify game controls.
When players pressed R to use the Rewinder, there was no visual feedback to show them that the Rewinder was active. The same happened when players pressed a number between 1 and 4, in order to use an item from their inventory.	Implement a highlight for the Rewinder and for the inventory items.

3) Improvements to the HUD

After analyzing playtesters’ feedback, the researcher created a new iteration of the HUD with several improvements, as shown in the following caption.

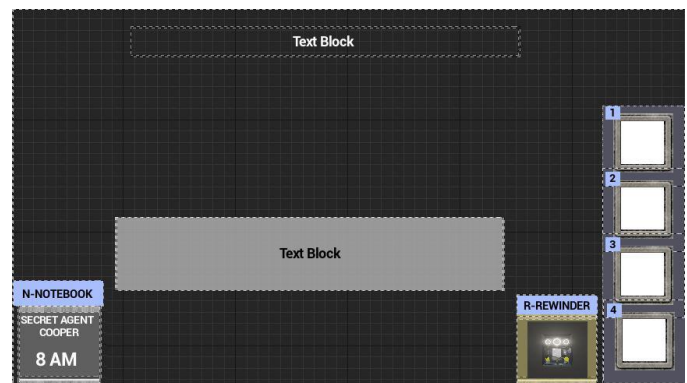


Figure 25: Intermediate iteration of the HUD

In this new version, the description bar (panel 3 in previous figure) disappeared, clearing some space on the screen. The inventory became permanent, so that players could see which items they had at all times. The researcher also decided to simplify the controls for this iteration of the HUD. To do so, he removed the distinction between “Using” and “Observing” an object. Instead, players could only “Interact” with an object by

pressing the left-mouse button. This change in the control scheme made panel 5 obsolete, and the researcher decided to remove it from the HUD. Finally, the researcher added a highlight for the Rewinder panel. The highlight would activate when players tried to use the Rewinder or an item, giving players immediate visual feedback on their actions.

The changes improved the players' experience, but playtesting still revealed flaws in the layout:

Issue	Solution
Information was scattered around the screen. Players seem to focus on one of the corners of the HUD and ignore the information on the other corner.	Move all HUD elements to the same area of the screen.
Players didn't realize that the information in the bottom-left panel changed, whenever they entered another character's memory.	Remove the panel with the character's name and the time of day. Find a better way to convey character switching.
Players found it weird that the HUD didn't change at all for different characters.	Make the HUD visually different when players enter a memory.
Players did not know when they could use the Rewinder,	Add an icon that pops up in the HUD whenever players are close to a DNA sample.

4) Final HUD implementation

The researcher decided to create two HUDs for the final implementation of the system: one HUD for the main level, where players control the main character, and another HUD for memories, where players control secondary characters.

- *Main HUD*

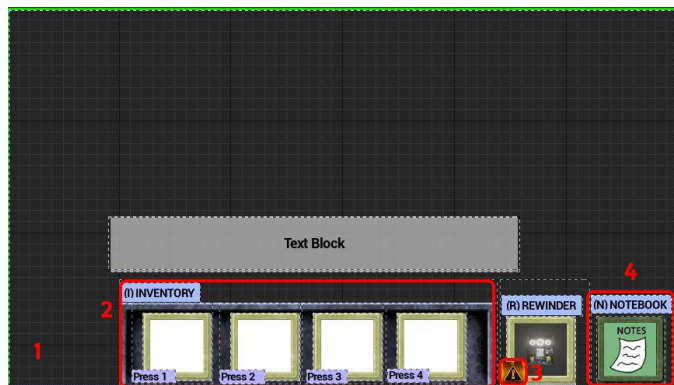


Figure 26: Final implementation of the main HUD

1. The researcher decided to remove this panel from the final HUD. With the addition of the secondary HUD, the

panel that contained the character's name and time of day became obsolete.

2. The researcher decided to move the inventory to the bottom of the screen, in order to concentrate all of the HUD's information in the same area.
3. The researcher added an icon that would pop up whenever players were close to a DNA sample. This icon let players know that they are in an area where they can use the Rewinder.
4. The researcher added a notebook icon, that reminded players of how to open the notebook.

- *Memories' HUD*

Since some memories only last for a few seconds, the secondary HUD needed to be straightforward and clear. Players had to be able to understand that they were controlling a secondary character with a quick glance.

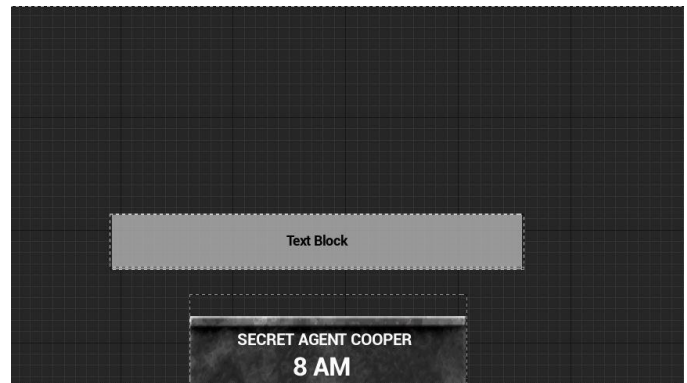


Figure 27: Players' HUD when they are in a memory

The layout only contains the essential information that players need to know: the name of the current character, and the time of day. This layout proved to be very effective during playtest, and most playtesters immediately understood that they were controlling a new character after they entered a memory.

5) Advantages of the final layout

The final layout worked well with playtesters, and proved to be effective for several reasons:

- It concentrated all the information into one area of the screen.
- It contained all of the controls for the different menus.
- The game's critical information was on the HUD at all times.
- It gave visual feedback on player actions, by highlighting items when players tried to sue them.
- Provided a way for players to interact directly with certain objects, such as the Rewinder, the Notebook, or the inventory items, without having to open any menus.

6) Current limitations of the HUD

One of the limiting factors of the HUD is that it uses elements that are specific to Rewind. Removing the Rewinder from the HUD is an easy task, but designers have to make sure that they disable the Rewinder functionality as well. To do that, they need to access not only the HUD blueprint, but also the First Person Character blueprint. Given more time, the researcher could add a boolean in the HUD, which could be used to enable/disable the Rewinder functionality and UI elements altogether.

The HUD also lacks a notification icon to notify players every time the main character writes a new Note or Timeline event. Although implementing the icon should be straightforward, designers would have to do it themselves. In the same way, designers can change the number of items in the HUD's quickbar easily, but it still requires some small widget modifications.

G. The State System

Like many other point-and-click games, Rewind required a character comment system. In order for the game to feel real, the player had to be able to look at an object and make comments about it. In the case of Rewind, this system was especially important because it was the only way for players to learn about the main character's personality. Realism also implied that the character's comments should change if players interacted with the same object several times.

To address this issue, the researcher implemented an interaction system, that controlled the comments that the main character made about the world's objects.

1) Initial implementation: The Text Array system

The initial implementation of the system used "Text arrays". Each Interactive Object had a Text Array assigned to it. This Text Array contained all of the possible comments that the player could make about that object.

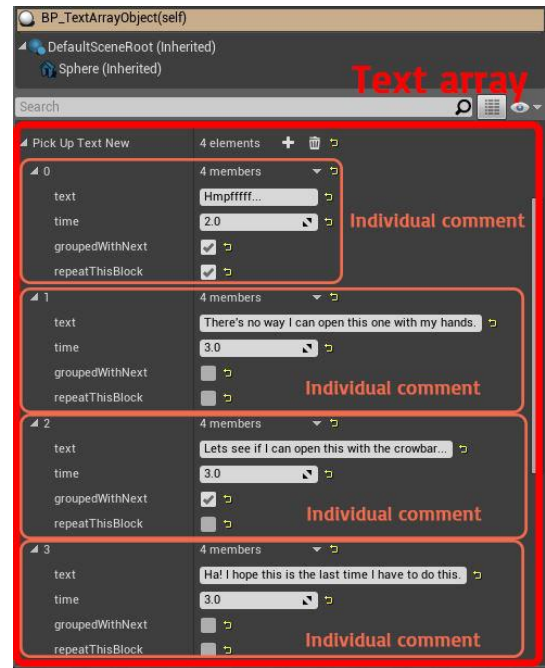


Figure 28: Structure of a Text Array

A counter would keep track of the last comment that the player made about the object. If the player interacted with the object again, the counter would increase and print the next comment in line. The arrays also offered the possibility to print the same comment over and over, until some event happened in the level.

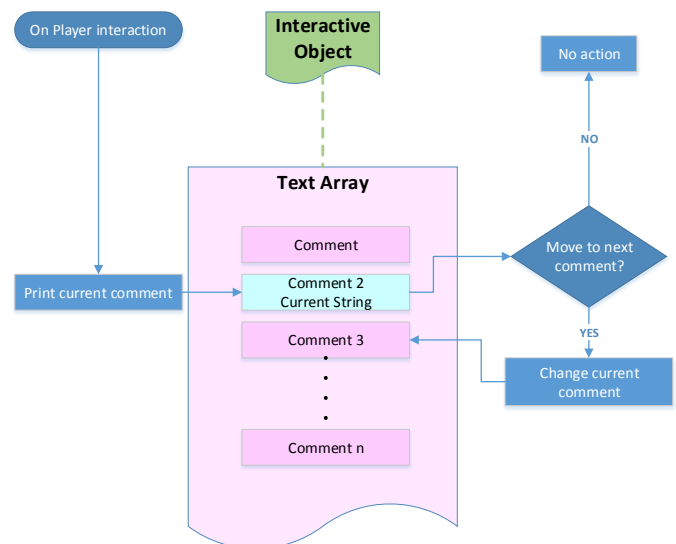


Figure 29: Graph of the Text Array functionality

Some objects could also have more than one Text Array assigned to them. The use of having two Text Arrays was that, if an object underwent a radical change, the player could use a completely new set of comments for that object.

The Text Array system also distinguished between two types of interactions:

- Pick up action (Right mouse button): the player tries to use an object, or pick it up.
- Observe action (Left mouse button): the player observes an object and makes a comment about it, but doesn't interact with it.

Each type of interaction had its own array of comments. If the players "picked up" an Interactive Object, the main character would comment something different than if players were "observing" the object.

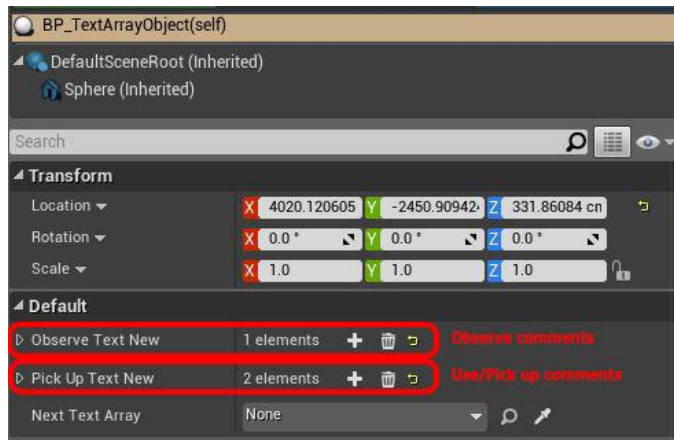


Figure 30: Text arrays distinguished between "observing" an object and "picking up" an object.

2) Advantages of the Text Array system

The Text Array system was useful for the following reasons:

- It let the level designer add character's comments to an Interactive Object easily.
- Whenever players travelled between levels, the game saved the current state of all Text Arrays objects. When players went back to the main level, the Text Arrays loaded again. That way, the character wouldn't repeat comments that he had made before.

3) Limitations of the Text Array system

Early playtester feedback showed that the Text Array implementation was lacking several features:

Limitation	Consequence
Text arrays only controlled the player's comments about an object. They did not save any of the other properties of the object.	If players travelled to another level, the state of the object wouldn't get saved. For instance, a door that the players unlocked, would be locked again when players came back from a memory.

The distinction between pickup text and observe text was confusing to playtesters.	Playtesters didn't enjoy having to use two different keys to interact with objects. They found the differentiation between "using an object", and "observing an object" confusing.
Comments could only flow linearly. Non-linear flow required a customization of the Interactive Object.	To arrive at a comment, the system had to go through all the previous comments first. Using Text Arrays to create a non-linear flow, that allowed jumping between comments, was cumbersome and time-consuming.

4) Final implementation: State System

To address of all the problems of the Text Array system, the researcher created the State System. Every object in Rewind has a series of states assigned to it. At any given point in time, an object can only be in one specific state. Based on its state, objects react differently to players' interactions. For instance, a door might have the following states:

State	Pre-Conditions	Reaction to player interaction
Door locked	<ul style="list-style-type: none"> • Player doesn't have the door's key. 	Player comments "This door is locked"
Door locked 2	<ul style="list-style-type: none"> • Player doesn't have the door's key. • Player has already interacted with the door once. 	Player comments "Still locked. Maybe I should try to find the key."
Door locked & player has key	<ul style="list-style-type: none"> • Player has the key in his inventory, but hasn't used it on the door yet. 	Player comments "I have the key, I should use it on the door"
Door is opening	<ul style="list-style-type: none"> • Player uses the key on the door. 	The door opens Player comments "The key works perfectly"
Door is opened	<ul style="list-style-type: none"> • Players has already used the key on the door. 	No reaction.

Interactive objects can move from one state to another, but the states themselves never change. This makes level saving a straightforward process. In order to save an object, the game only needs to remember the current state of the object. To reload a level, the game saves the id of the states for all the Interactive Objects in the level. When the level loads, the game simply moves the Interactive Objects back to their corresponding saved states.

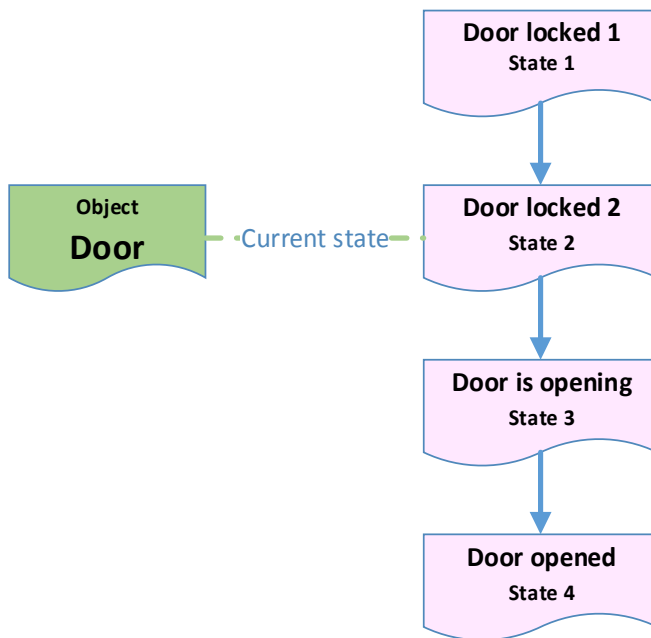


Figure 31: An object can have several states, but it can only be in one state at the same time.

Objects can transition between states based on player interaction, or other events. The following diagram shows an example of a state flow for an Interactive Object:

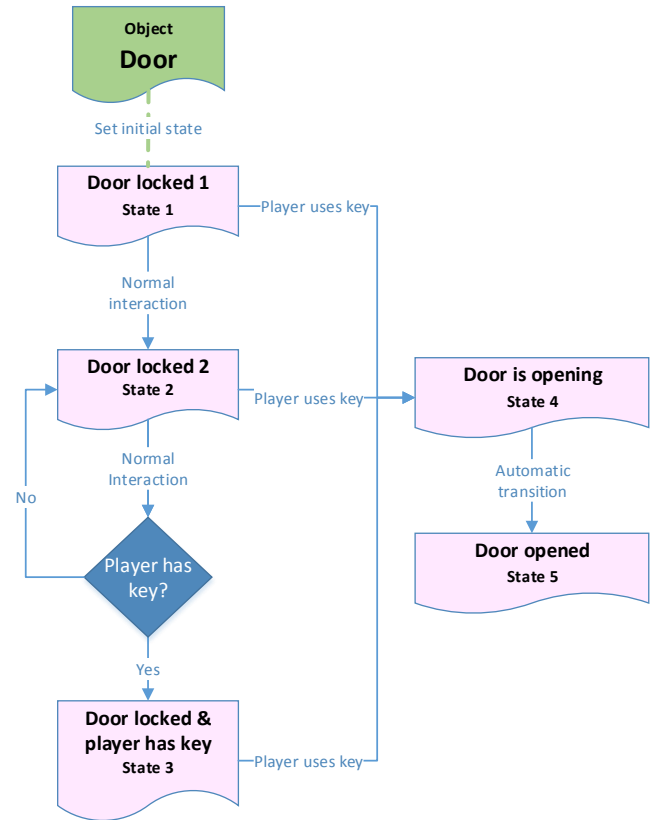


Figure 32: Example of flow between the states of an Interactive Object

Finally, unlike the old Text Arrays, the states don't differentiate between "Observe Text" and "Pick up" text. Both types of text have been merged into a "Text Block".

5) States' structure

States have the following overarching structure:

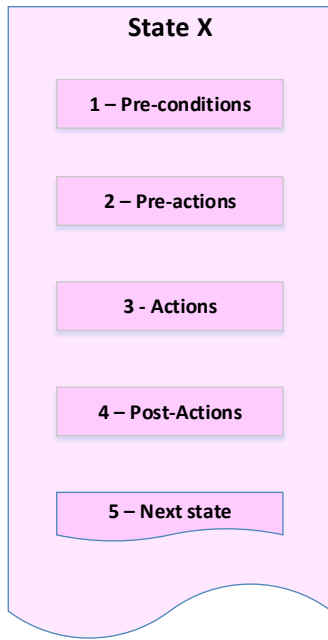


Figure 33: General structure of a state

1. Pre-Conditions

Pre-conditions are conditions that need to be fulfilled before moving to this state. For example, in the case of the state “Door locked & players have key”, a pre-condition could be “Players have the key in their inventory”.

2. Pre-Actions

Actions that are performed as soon as an interactive object enters the state. For instance, a pre-action for the state “Door is opened” would be “Open the door as soon as this state is loaded”.

3. Actions

Actions that are performed when the player interacts with the object, if the object is in this state. For example, an action for any of the door states could be “Show a comment from the main character saying ‘This is a door’”.

4. Post-Actions

Actions that are performed when the Interactive Object leaves the state. For example, a post-action for the state “Door is opening” could be “Remove the key from the players’ inventory”.

5. Next State

The state that loads after the current state, provided that the Next State’s pre-conditions are fulfilled.

6) Object State actors

To implement the states in the editor, the researcher used invisible actors called “Object States”. Each “Object State” actor represents a certain state for a certain Interactive Object. The “Object State” is the parent class for all states in the level. Some Interactive Objects only use the basic Object States, while others use child classes that add extra functionality.

Creating the final version of the Object States was an iterative process, where the researcher focused on creating actors that were adaptable to any adventure games. The

following caption shows the final implementation of the Object States, followed by a description of the limitation that led to the inclusion of each property:

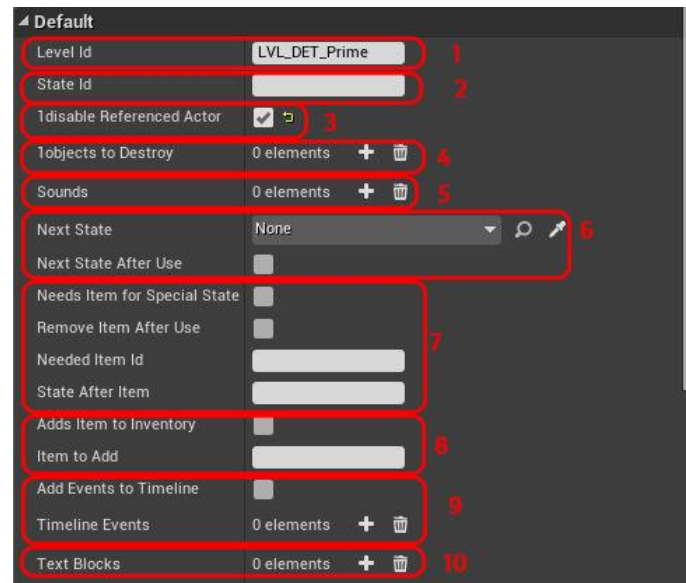


Figure 34: Properties of an Object State actor

1. Level Id

- *Description*

The state only activates if the player is in the level named “LevelId”

- *Why it was added*

Different levels can share the same Interactive Object, but the object’s behavior may need to be different based on the level.

- *Example of use*

Players fix a computer in the main level. Players then enter a memory that sends them to the past, when the computer was still broken, so they cannot use the computer while inside the memory. The state “Console working” would have LevelId=MainLevel, while the state “Console Broken” would have LevelId=MemoryLevel.

2. State Id

- *Description*

An Id that identifies this state. It doesn’t need to be unique, but non-unique ids can cause problems.

- *Why it was added*

Some objects may need to jump to a specific state, after the player collects an object, or after an important event in the level.

- *Example of use*

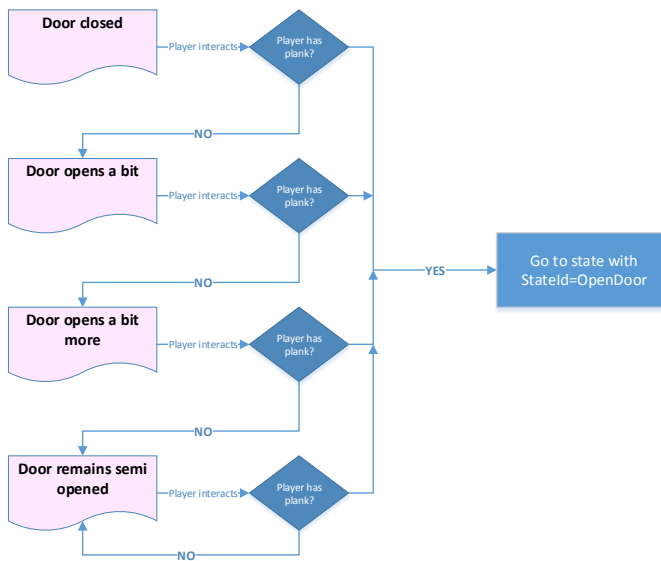


Figure 35: Example of use for "StateId"

3. Disable Referenced Actor (Pre-action)

- *Description*
Hide the Interactive Object whenever it reaches this state.
- *Why it was added*
Some objects, like the pickup items and the logs, need to disappear from the level once the player picks them up.
- *Example of use*
The player picks up an item. The item then moves to a state with "1DisableReferencedActor=true" and disappears from the level.

4. Objects to destroy (Pre-action)

- *Description*
The state removes the actors in this list.
- *Why it was added*
Some non-interactive meshes need to disappear from the level.
- *Example of use*
The player uses a console to remove the toxic gas from the laboratory. The console then enters a state called "No toxic gas", that removes all of the particle emitters that are generating the gas.

5. Sounds

- *Description*
Some objects need to play sounds on player interaction.
- *Why it was added*
Objects like broken consoles need to play a "broken sound" whenever the player tries to use them. Other objects, like doors, need to play an "opening sound" on interaction.
- *Example of use*
See previous bullet point.

6. Next State variables

- *Description*
Control which state is next in the state queue.
- *Why it was added*
Objects need to be able to move to a different state after the player interacts with them.

• Example of use

The player picks up an item. The item then moves to a state that removes it from the level.

7. Item requirement variables

- *Description*
These variables allow designers to move an Interactive Object to a new state, only after the player uses a certain item.
- *Why it was added*
Some objects require players to have a specific item before progressing to further states.
- *Example of use*
A door is in a "Door locked" state. The player can interact with it, which results in a comment by the main character ("This door is completely locked"), but simple interaction doesn't move the door to a new state. However, if the player uses a plank on the door, the door moves to a new state called "Door opened".

8. Item giving variables

- *Description*
After players interact with an object, they might receive an item.
- *Why it was added*
For pick up items. When players interact with a pick up item, the item needs to go to their inventory.
- *Example of use*
A player interacts with a locker. The locker has a keycard inside, that goes to the players' inventory.

9. Timeline-related variables

- *Description*
Some interactions can lead to new information added to the player's timeline.
- *Why it was added*
Players have to be able to make discoveries about the level's story by inspecting the environment. If they make a discovery, the timeline should update accordingly.
- *Example of use*
Players discover the dead body of a researcher. This adds a new event to the timeline, "10PM – Researcher died in the facility."

10. Text blocks

- *Description*
Text blocks contains all the comments that the character can make when it interacts with an item.
- *Why it was added*
Every player interaction should trigger a response from the main character. This is traditional for all adventure games.

7) Text States

While building the level demo, the researcher found the need to create states that would trigger automatically. Sometimes the design requires "spontaneous" comments from the character, which are not triggered by any input. For instance, the player might enter a room where all lights are off, and immediately make the comment "This room is really dark".

The need for this automatic states led to the creation of the Text States. What makes Text States special is that they don't

require player input to execute. When an Interactive Object reaches a Text State, it immediately executes the actions in that state, and automatically moves to the next state on the list.

8) Advantages of the State System

The final implementation of the state system considerably improves the Text Array implementation. It fixes all of the issues found during playtesting by the researcher, and provides the following advantages:

1. States don't need to flow linearly. An object can jump from any state to any other state at any time.
2. States never change. Loading an Interactive Object is a matter of connecting it to the right state.
3. States contain all the necessary information to completely restore an Interactive Object on level loading.
4. States are easy to understand and intuitive to set up.
5. Setting up a simple Interactive Object becomes an easy, straight-forward process, thanks to the state system. Level Designers only need to worry about correctly setting up the transitions between states.

9) Working on ease-of-use: State's implementation in engine

One of the biggest challenges of the state system was making it easy to organize. The internal logic of the states is intricate, and some Interactive Objects have many states that are connected to one another

To solve this issue, the researcher decided to use empty actors to implement the states. Level designers can drop these actors in the world, and arrange them to their liking.

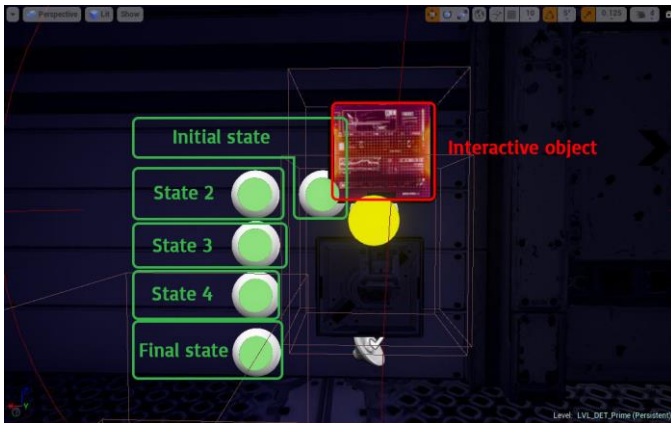


Figure 36: States have a visual representation in the world, so that level designers can organize them as they prefer.

With a simple glance, designers can immediately know the number of states of an object, and even the order of the different states. The researcher found this arrangement very easy to work with.

10) State System limitations and solutions

One of the main limitations of the state system is its semi-linearity. In the state system, states need to be connected to one another, through an object reference variable. Each state only has space for one object reference, which means that each

state can only link to one other state. Therefore, the states of an interactive object form some kind of "Linked List".

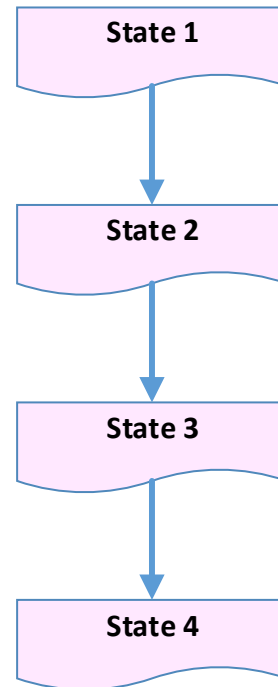


Figure 37: The states for a certain Interactive Object form some kind of "Linked List"

Despite this structure, the flow between states is not completely linear, since states can bypass other states depending on certain conditions that designers can set. For instance, designers might decide that if players have "Object X", then "Interactive Object 1" jumps to State 4, whatever its current state is. Although this gives designers some freedom when organizing states, the system does not allow for complex branching. If a designer wanted to connect one state to four different "Next States" he would have to do considerable blueprint work on the system.

Like it happens with the Action Triggers, moving an object in the editor doesn't move the states linked to it. Hence if designer move an object, they need to manually move the states that belong to that object. With more time, the researcher could make the states an "Actor Component", would automatically move them along with their actor owner.

Finally, states don't organize themselves automatically in editor, and designers need to organize them manually. That means that if an object requires drastic state changes, the designers have to spend some time relocating the states to their correct positions. Creating a script inside the Interactive Object, that automatically organizes the states based on their connections, could help designers with this issue in the future.

H. Menu hierarchy

To create the menu hierarchy, the researcher looked at other point-and-click games, taking inspiration on Syberia's menus. Syberia makes the inventory a central HUB that allows players to access any other menus. This structure results in a clearly

organized user interface, since players can access almost all of the game's information from the same menu. It also provides a more realistic inventory, where the main character keeps all of her objects (story or gameplay related) in a jacket or bag.

Rewind uses a similar system, by having an item called the Notebook inside the inventory. The Notebook gives players access to all the narrative-oriented menus of Rewind. Players can also open the Notebook from the HUD, and save the extra step of opening the inventory. The Notebook has three submenus, each with a different story-related purpose. The researcher explains these submenus in detail in the following sections.

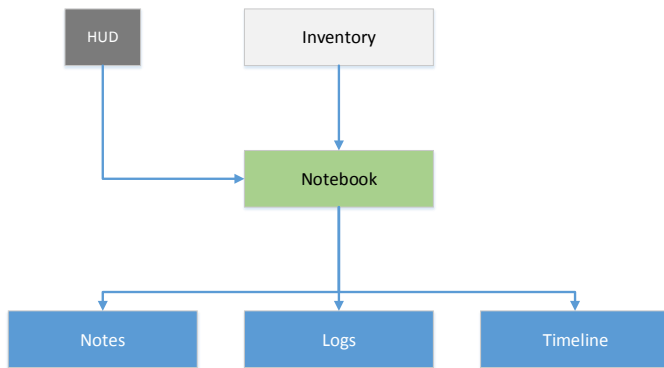


Figure 38: Menu hierarchy in Rewind

For more details about the Notebook, visit the [Notebook](#) section.

I. The Inventory

Like in many adventure games, Rewind players need to collect and use items in order to progress. The researcher aimed at providing the following inventory functionality:

- Players can use inventory items easily
- Players can see information about the items they have picked up.
- Players can rearrange their items inside their inventory.

1) Initial inventory

One of the researcher's priorities when building the inventory, was to provide players with a direct way of using items, that wouldn't force them to enter any menus. Initially, the inventory was part of the screen (see [HUD section](#)), so players could see and use their items at all items.

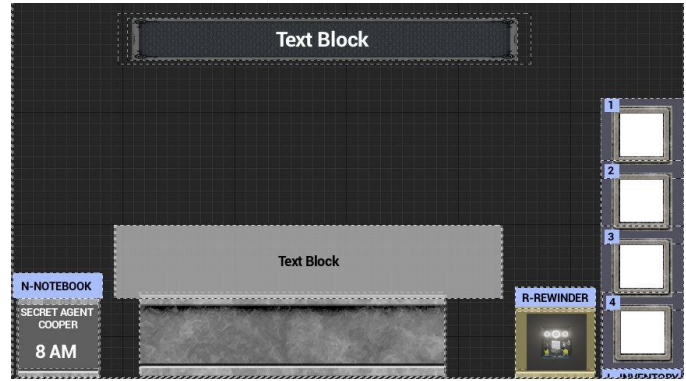


Figure 39: Initially the inventory was part of the HUD.

Although the layout let players use objects quickly, it had big drawbacks. The most obvious one was scalability. The inventory only allowed for four items, which are not usually enough for most adventure games. At the same time, trying to add more slots to the inventory resulted in a cluttering of the player's screen. The layout also lacked item information, since players could not see a description of the items they have picked up. In some instances, players would pick up an item and then forget what the item was. Finally, the inventory did not let players rearrange their items.

2) Final inventory

Using Syberia as a guideline, the researcher decided to apply the following solutions to the inventory problem:

Limitation	Solution
Allow for quick access to some of the inventory items.	Keep a small inventory on the screen at all items, which players can access directly.
Allow players to have more than four items.	Create a bigger inventory, that players can access by pressing I.
Let players read descriptions of their items.	
Let players rearrange the items in their inventory.	

The researcher decided to create two inventories: a small one that was always visible on the HUD, and a bigger one in the form of a menu. The smaller inventory served as a shortcut to the big inventory, and still gave players direct access to their items without having to open the bigger menu.



Figure 40: Final version of the small inventory.

If players wanted to learn more about an item, or relocate their items, they could go to the bigger menu to do so. The bigger inventory still offers players the option to use items by double clicking on them. A single click gives players information about the item, as well as the item's name.

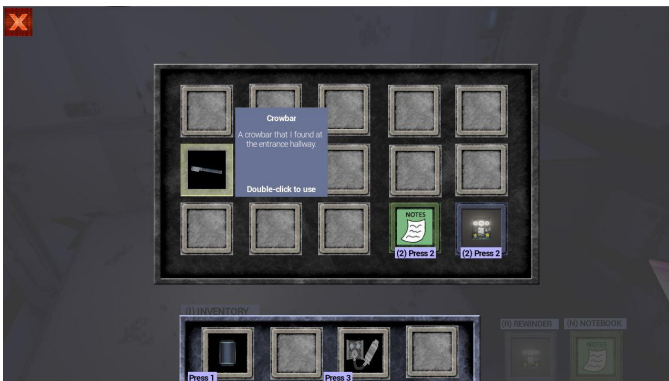


Figure 41: The big inventory, where players can see more details about their items.

Finally, as seen in the caption, both menus communicate with each other, so players can move items between the small and the big inventory without a problem. This lets players customize their shortcuts to items to their liking.

3) Item Storage: ease of use for designers

One of the biggest problems that the researcher had to solve while building the inventory, was related to consistency. The researcher wanted to make sure that references to the same item were always consistent throughout the level. For that, the researcher decided to build an Item Storage actor, which contains the information of all the pickup items in Rewind.

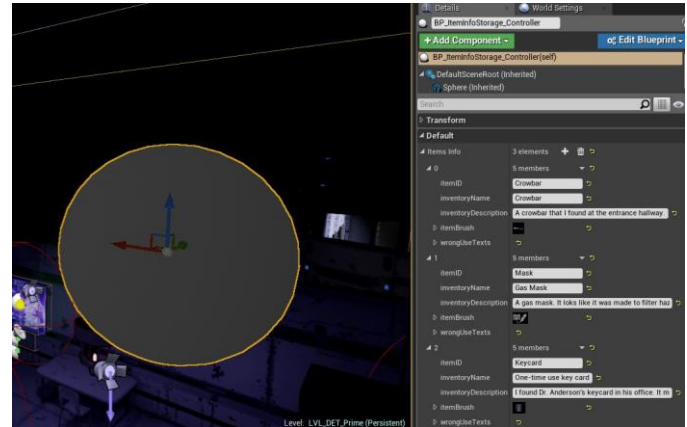


Figure 42: The Item Storage contains the information of all of the level's items.

Whenever a blueprint needs information about an item, the blueprint asks the storage for the item's information. This structure facilitates the designer's job, who can access consistent item information from any blueprint.

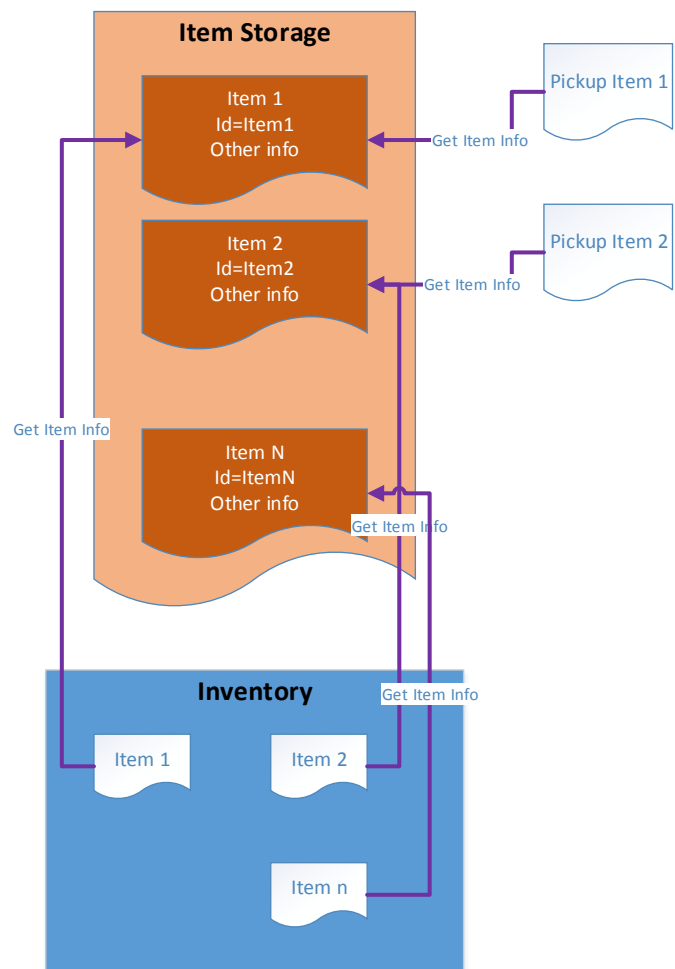


Figure 43: Whenever the designer needs an item's information, he can make a simple request to the item storage.

Hence designers only need to operate with item ids in most blueprints, calling the Item Storage only when they need full extended information about an item. The item storage also facilitates adding items to the inventory. Designers need a simple call to the storage to get the item's information, which they can add to the inventory afterwards.



Figure 44: Designers can add an item to the inventory by just using its id.

Finally, the item storage provides a centralized management of all of the level's items. If the designers want to change any of the items properties, they can just do so on the item storage, without having to worry about changing the information in any other blueprints.

4) Current Inventory limitations

The Inventory contains a fixed amount of slots that are always visible in the Inventory widget. In a game with a small number of items, many of the Inventory slots may remain empty throughout the whole game, making the Inventory look oversized. On the other hand, the Inventory limits the amount of items that a player can have at the same time. Scaling the inventory down is fairly easy but scaling it up requires work on the Inventory widget. A designer wanting to allow players to have more than seventeen items at once would have to transform the inventory into a scrollbar.

Regarding slot size, all slots in the Inventory have the same size. This limitation can affect games that use items with big differences in size, as some items' thumbnails may be too big/or small for an Inventory slot. Again, creating an inventory that allows for items with different sizes would require a big time commitment for designers and a considerable amount of widget and blueprint work.

In addition to the limitations on the design side, there are also some issues affecting players. Players only have limited control over the items once the items enter their inventory. For instance, once players pick up an item, they cannot put it back in the world.

Finally, in the current Inventory implementation, when players want to move an item from the inventory to the HUD, they need to drag it from one slot to the other. The dragging mechanic might be bothersome for some players, who might want to be able to send an item to the HUD by just pressing a number from 1 to 4. Creating this shortcut wouldn't be difficult but the implementation is not currently in the artifact.

J. The Notebook

Players witness Rewind's story in out of order sequences, starting from the end and ending at the beginning. At the same time, players witness many story moments while they are inside

another character's memory. Due to the complexity of Rewind's storytelling structure, the researcher decided to create the Notebook. The Notebook contains all the information that players need in order to understand the story. When players forget a story detail, a character, or an event, they can enter the Notebook and easily find the information they forgot.

The Notebook contains three submenus, each one of them having a different purpose:

Menu	Purpose	Example
Note menu	Contains the most relevant discoveries that the main character makes. These discoveries can include gameplay-relevant information.	The main character finds the code to a safe written on a wall. The main character adds the code to the Note menu, so that players can remember it at any time.
Timeline menu	Shows the story events in chronological order, to help players reconstruct the game's events.	Players enter Dr. Anderson's memory, where they witness his death. A new event appears on their timeline: "8PM – Dr. Anderson died".
Log menu	Shows the text logs that the player has collected so far.	A player picks up a log that another character wrote. The log becomes available for the player to read, inside the log menu.

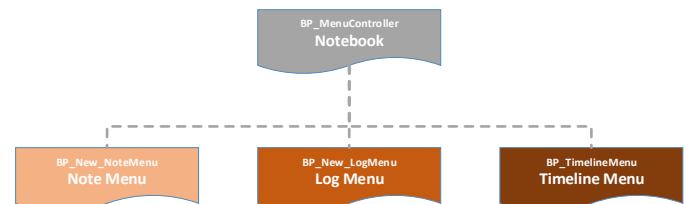


Figure 45: Diagram of menu hierarchy

1) Evolution of the Notebook

In early iterations, the Note Menu, the Timeline Menu, and the Log Menu, were all separate menus that players could open with different keys. This was a cumbersome process, since players had to be opening and closing menus constantly to look at different story-related information. It also increased the amount of keys that the players had to memorize, which was not desirable since it slowed down gameplay. The researcher decided that the best approach was to create the Notebook, which provides a single point of access for the three menus.



Figure 46: In the final iteration, all storytelling menus are part of the Notebook

K. The Timeline Menu

Players don't witness the story of Rewind chronologically, which initially made it hard for players to piece the story together. The researcher decided to create the timeline, so that players had a menu that they could access to see the level's story laid out in chronological order.

From a design standpoint, the timeline menu allows designers to get creative with the chronology of their story, without having to worry about players getting confused. As it happened with the HUD, the process of creating the timeline was a process of simplification through iteration.

1) Limitations and evolution

The first version of the timeline menu contained four different timelines, one for each character. Playtesting showed that by the end of the demo most timelines were almost empty, as there were not enough events to fill them. The timeline also had an unusual layout, so players had trouble understanding what they were seeing on the screen.

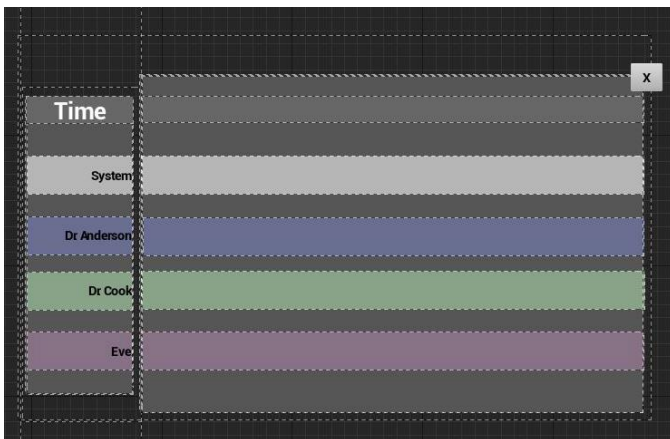


Figure 47: The first version of the timeline menu had one timeline per character.

To solve the initial timeline problems, the researcher merged all timelines into one, and gave the menu a more traditional design. The second iteration of the timeline had a vertical design, which was a layout that players could understand easily.

In addition, the menu only had one timeline, instead of one per character. Both changes helped players get a better idea of the level's story.



Figure 48: The second version of the timeline used a vertical layout

The problem with this new timeline came from a design standpoint, since the timeline only allowed for one event per hour. If the designer wanted to list two events at 10PM, he was unable to do so. For the final iteration, the researcher focused on making the timeline more flexible for designers. The final iteration of the timeline is similar to the mid iteration, but the timeline is horizontal instead of vertical.

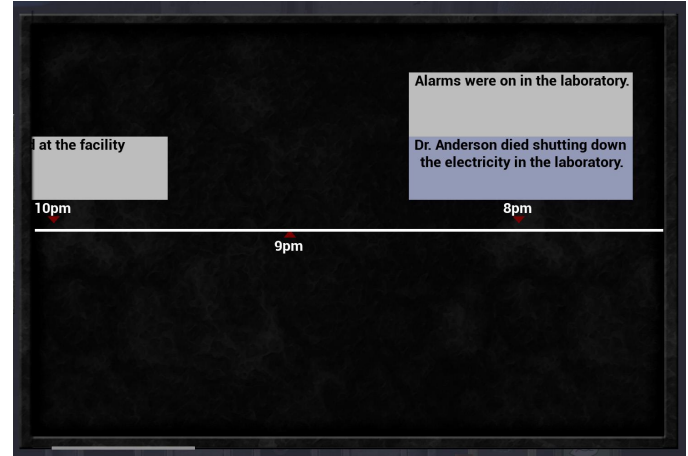


Figure 49: Final iteration of the timeline menu

The horizontal layout allows for more events per hour, giving designers freedom to make their story more complex. The horizontality helps players identify the menu as a timeline, since most timelines are usually horizontal.

2) Timeline customization

While building the timeline, the designer tried to make it as flexible as possible, so that other designers could customize it based on their needs. The timeline has two main customizable elements, the start/ending times, and the number of items per hour.

- Starting time and ending time.

Designers can choose at what time of day the timeline starts, and what is the last hour showed on the timeline. Designers can customize this value from the BP_TimelineMenu blueprint.

- Number of events per hour.

Designers can change the maximum number of events per hour, by accessing the blueprint BP_Timeline_Item_Big. However, due to the size of the timeline widget, only three elements fit in the screen at the same time. To go around that problem, designers need to adapt the widget if they want to fit more items in it.

3) Adding events to the timeline

Blueprint-wise, adding events to the timeline is very easy and doesn't require much work. The LIB_EasyAccess provides a function AddItemToTimeline, which adds an event to the timeline.

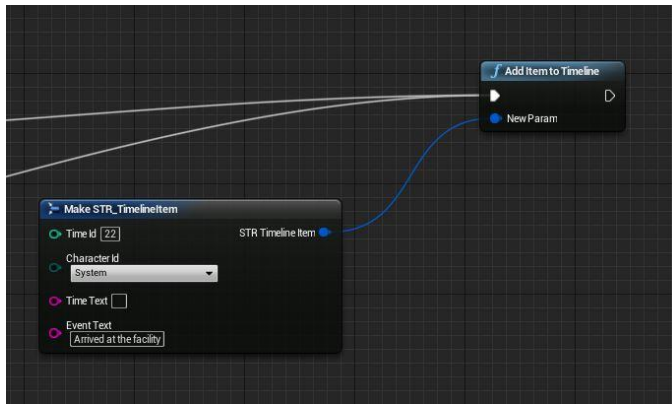


Figure 50: Adding events to the timeline is a simple process

When adding an event to the timeline, the designer has to create a struct item of type STR_TimelineItem with the event's information. When creating this struct, the research focused on simplicity. The researcher decided the following information was the minimum necessary in order to accomplish both good player communication and simplicity:

- TimeId
The time of the event, in military time. Example: 10pm has TimeId 22, 1pm has TimeId 13, etc. This field only allows entries from 0 to 24.
- CharacterId
If the event involves a character, the Id of the character, if not, nothing. This field color codes the events in the menu, based on the character that lived the event. That way, players can quickly differentiate between the different characters in the timeline.
- TimeText
For internal use, designers don't need to fill it.
- Event text
The text that is going to appear in the event.

4) Current limitations of the Timeline

Although designers can customize the maximum amount of events per hour in the timeline, there is space for only three events per hour in the Timeline widget. If designers want to have more events per hour, they need to expand the Timeline widget, or reduce the size of the event widgets, so that more of them can fit on the screen. With more time, the researcher could fix this problem by allowing scrolling in the vertical axis, making the timeline scrollable in all directions.

L. The Note Menu

Rewind hides clues and gameplay-relevant information inside secondary characters' memories. That means that in some instances, players must enter another character's memory in order to get information that they need to progress. This approach makes memories gameplay-useful, and encourages players to use the Rewinder. Memories become part of the gameplay core loop, instead of being a separate storytelling tool.

However, early playtesting showed a major problem with this method. Clues like passcodes or names were very easy to forget for players. When players forgot a code that they had seen inside a memory, they had to go back to the memory in order to remember it. This problem led to repetition, made the game frustrating, and resulted in players concentrating too much on remembering the memories' small details.

To address this issue, the researcher created the Note Menu. Every time the main character gets a relevant piece of information, he writes it on the Note menu. That way, if players forget something, they can use the Note Menu to remember it. What's more, the menu gives players insight about the thought process of the main character, since he writes the notes himself.

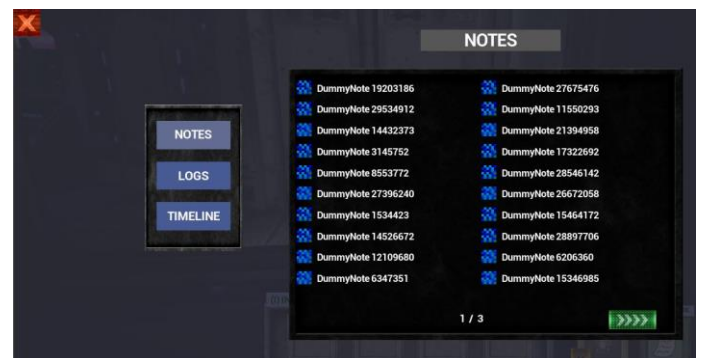


Figure 51: The main character writes down some of his thoughts on the note menu.

1) Past limitations and evolution

The note menu had a simple layout from the beginning, and layout changes weren't necessary for design purposes. The menu underwent some art-related changes to match the other menus in the game.

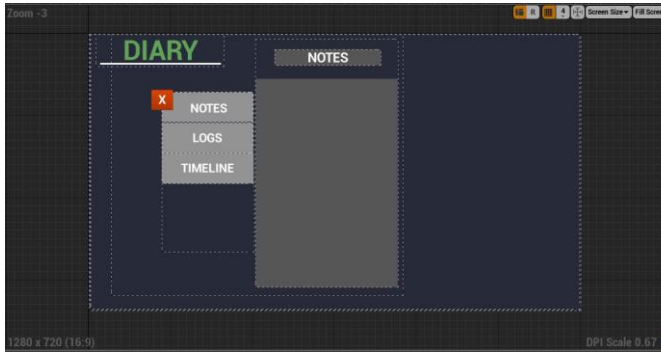


Figure 52: Initial iteration of the Note Menu

The biggest limitation in the Note menu came from a player communication perspective. Playtesting showed that players didn't usually open the menu, and that in some cases they were not aware of the menu's existence. The reason behind this problem was that the game didn't inform players about new notes. Consequently, many players were not aware that the Note Menu had new information inside.

With this issue in mind, the researcher decided to force the main character to make a comment whenever he wrote a new note in the Note Menu. Anytime Agent Cooper is about to write a note, he makes a different comment that looks like the following: "I need to write this down in my notebook, so that I don't forget". Using this approach, most players instantly go to the menu to double check that the note is there. This method also adds realism to the level, by presenting the Note Menu as a notepad that the character writes on.

M. Logs and the Log Menu

In Rewind, the player spends most of the time alone, and it only learns about the secondary characters' actions through their memories. Rewind called for a communication channel that would show not only the secondary characters' actions, but also the secondary characters' thoughts and feelings. Understanding the secondary character's personality is essential in Rewind, since they trigger the events that happen in the laboratory.

1) Text Logs

To give extra information about the secondary characters, the researcher decided to create the text logs. The text logs are documents written by secondary characters. Players can find them as physical items in the level and pick them up in order to read them. After that, players can still access the logs through the Log Menu, where they can reread them if they need to. Like the notes, the logs can also be gameplay relevant, and they are used as a way to teach players about certain story events. For instance, in Rewind it was important for players to understand that Dr. Anderson and Dr. Cook didn't get along. In order to do that, the researcher added a log in the level, where Dr. Anderson talked about his animosity towards Dr. Cook. In the log, Dr. Anderson also casually mentions that he has hidden a gameplay-relevant object inside his locker. Hence if players

wanted to find that object, they had to read the log first, and learn about Dr. Anderson and Dr. Cook's fight.

Playtesting showed that players found the logs useful, and that they were interested in the logs contents.

2) Log Menu

Initially, players would find the logs in the world, but they were not able to pick them up. If players wanted to reread a log, they had to go back to the area where they found it. This was not a specially entertaining process, and didn't add anything to the gameplay.



Figure 53: Players can find logs in the world.

To avoid unnecessary backtracking, the researcher decided to create a log menu, so that players could go back to previous logs from the Notebook. Like it happened with the timeline, the log menu's first iteration was too crowded and over-engineered. The first pass of the Log Menu included an option to sort logs by author, as the following capture shows:

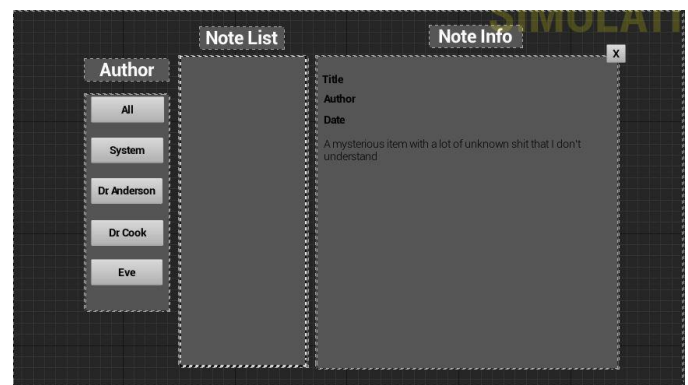


Figure 54: First pass of the Note Menu

Once players clicked on a character's name, thumbnails of all of the logs written by that character would appear on the "Note list" column. Players could then click on a thumbnail in the "Note List", and the full log info would appear on the "Note Info" column. This three column layout proved to be anti-intuitive, with players having trouble figuring out where to click

in order to read a log. The small amount of logs in the level also made the menu feel empty.

For the final iteration, the researcher decided to eliminate the “sort by author” column. That way, the menu only had a list of all of the logs gathered by the player. Players could still know who the author of a log was, by looking at the thumbnail of the log.

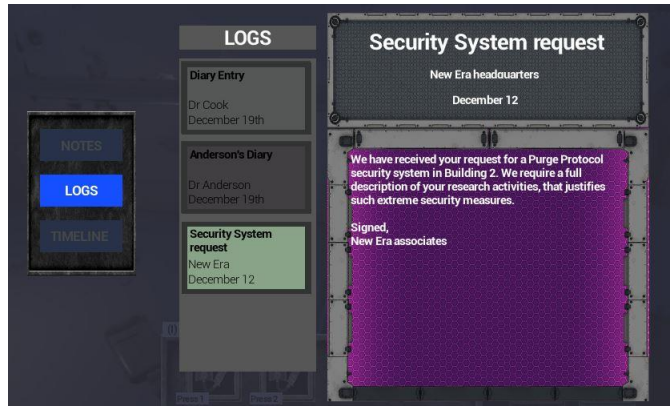


Figure 55: Final iteration of the log menu

3) Logs for designers: Creating a log and adding it to the menu

Although the logic behind log processing is complex, the researcher focused on developing an easy-to-use system for designers. Designers can follow this simple process to add a log to the world:

1. To create a log, designers need to start by creating a text document with the format seen in the following caption.

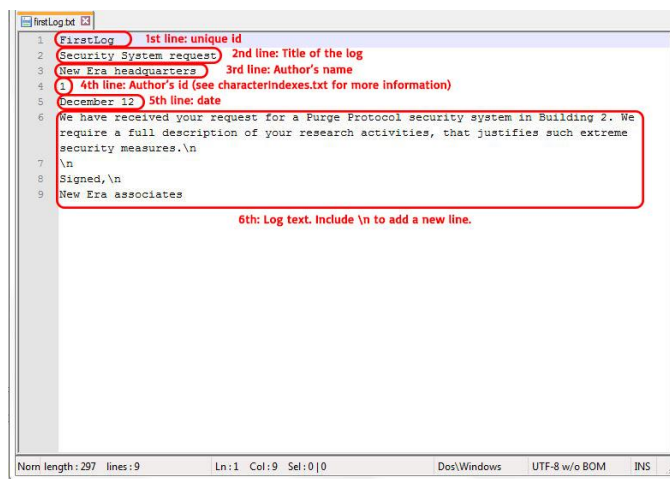


Figure 56: Log text file format

2. After creating the file, designers need to save it in the “LogFiles” folder, under the game directory.
3. The next step is creating a BP_Log blueprint in the world, and write the name of the text file in the “LogPath” variable.

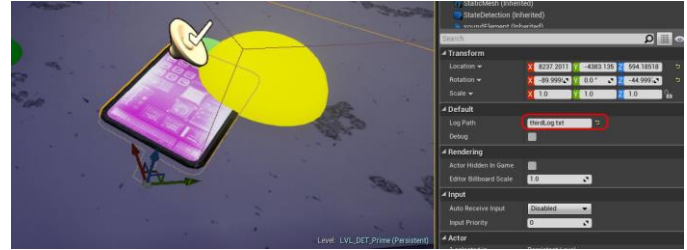


Figure 57: After setting the LogPath variable, the log is almost ready.

4. The log is almost ready for use, and the final step is setting up the states so that the log disappears after players pick it up. The underlying logic performs all of the other steps automatically.

4) Log System's limitations and solutions

To create a log, designers need to create a text file with the log content, and then write the file's name inside the editor. In point-and-click adventures with many text logs, this can mean having hundreds of separate text files. Since writers and designers might need to edit several files at a time, using separate files can be time consuming. Ideally, the system would use a .csv file that would contain all of the texts from all of the logs in the game. Each text would have an ID, and that ID is what designers would introduce in editor, rather than a text file name. The functionality is not present in the current artifact, but the researcher could implement it with some time by changing the C++ code that controls the logs.

Another limitation of the Text Logs derives from its C++ implementation. In the current system, designers can get the log's author, the log's date, the log's title, and the log's content. However, adding any extra information to the logs would require C++ coding. Given more time, the researcher could add a string array to the Log Info structure, so that designers could add as many fields to it as they wanted. The researcher would have to implement this in both blueprints and code.

N. Dialog system

Despite spending most of his time alone, the main character in Rewind encounters one other character during the level. Players can talk to this character, who gives them an item that they need in order to progress. Most importantly, dialog is almost a requirement in any adventure game.

When designing the dialog system, the researcher looked at traditional adventure games, and identified their common traits of their dialogs. Most adventure games offer players the chance to choose between several answers, with each answer triggering a response in the other party. Players also have the possibility to start side conversations that are not part of the critical path, in order to get more information about a certain topic. Voice over is also an essential part of dialog in most adventure games, and it adds personality to main characters and secondary characters alike. Finally, it is common for adventure games to provide a way for players to skip through the dialog.

With these properties in mind, the researcher set the following goals for the system:

Goal	Decision
The dialog should offer players several answers from which they would be able to choose.	Allow players to have a maximum of 6 answers and a minimum of 1, at any time during the dialog.
Players' answers should be able to have an effect in future conversation topics.	Develop a state-based system like the one implemented for Interactive Objects. Each state represents a line of dialog from the main character. Some states may have preconditions, so that they only become available after a certain event.
The dialog system should offer designers the chance to unlock dialog options only after a certain event happens.	
The game should save the dialog, so that dialog doesn't repeat when players come back from a memory.	
Players should be able to speed up the dialog.	Move to the next state if players left-click with their mouse.
The dialog should include the possibility for voice over, at least for the NPCs.	Add an audio variable to each dialog state.

1) General properties

Considering the goals listed above, the designer decided to create an interactive dialog system, where players have the chance to choose an answer for every NPC's piece of dialog. For the interface, the researcher decided to give players a maximum of six possible answers. Six answers provide enough freedom for players to feel like they were controlling the conversation, but not enough to make dialogs overwhelming.

The final dialog interface and its different sections can be seen in the following caption:

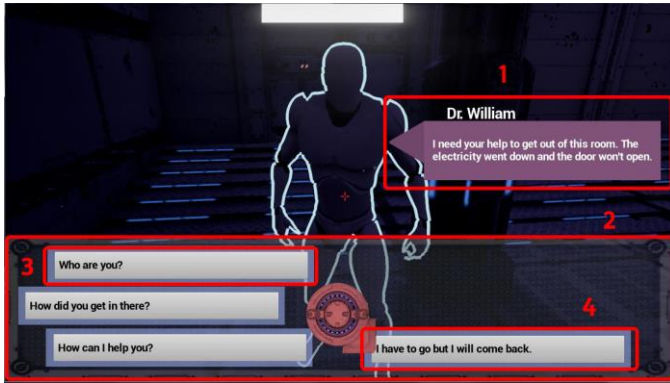


Figure 58: The final dialog interface

1. The NPC's bubble shows the last line said by an NPC. This bubble remains visible even it is the character's turn to speak.
2. The bottom part of the screen displays the available answers. Players usually get to choose between two to six options.
3. Some answers allow players to talk about a topic that is not part of the dialog's critical path. Once players have explored this dialog branch, the option may disappear or it may remain there for players to select it again.
4. The last option in the dialog is always a "goodbye line", so that players can leave the conversation at any time.

During a conversation, players can choose any of the visible options in the bottom panel. Once they choose an option, the NPC responds, and then players receive a new set of possible answers. Designers can make conversations as long as they desire. They also have the option to hide the bottom panel and create an NPC monologue, that ends when the NPC finishes talking.

2) The Dialog States

Considering all of the priorities listed above, the researcher decided to create a state system, similar to the one used for Interactive Objects. Each state in the dialog system represents a line of dialog from the main character, along with the NPC's answer. Dialog states have the following structure:

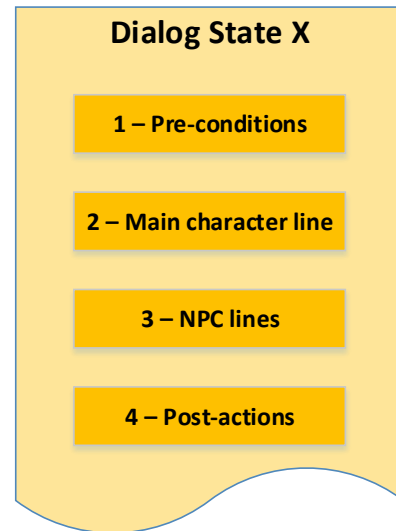


Figure 59: Structure of a dialog state

1. A dialog option only becomes available if the player meets its pre-conditions. If the player doesn't fulfill the requirements, the option doesn't show up in the dialog UI.
2. The "main character's line" is the line of dialog that shows up in the interface's buttons (see Figure 57).
3. The "NPC lines" are the dialog lines that the NPC says if the dialog moves to this state.
4. Post actions are actions that the game performs after the dialog moves from this state to another state.

Although all dialog states have this structure, the designers can choose to leave some of the properties empty. For instance, some states may not perform any post-actions.

3) Implementation of the dialog flow

The biggest difference between the dialog state system and the object state system, is that dialog states don't have a "Next State". Instead, the dialog flow is controlled by a single number called the "Dialog Status". The "Dialog Status" is an integer that changes as the dialog progresses. A dialog state only shows up in the answer panel if the "Dialog Status" is between a certain numeric range.

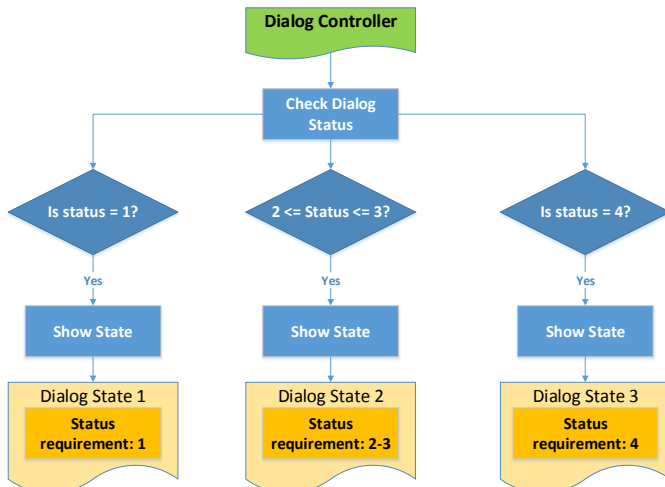


Figure 60: Logic behind dialog state availability

The “Dialog Status” changes based on the dialog options that the player chooses, allowing the dialog to progress. The designer also has freedom to change the Status manually whenever he considers it necessary. For instance, if the player obtains an important item, the designer can update the “Dialog Status” of a dialog. The new “Dialog Status” value may then activate some dialog options where the player can talk about the item they just collected.

The following diagram shows a potential dialog flow, including the change in the “Dialog Status”:

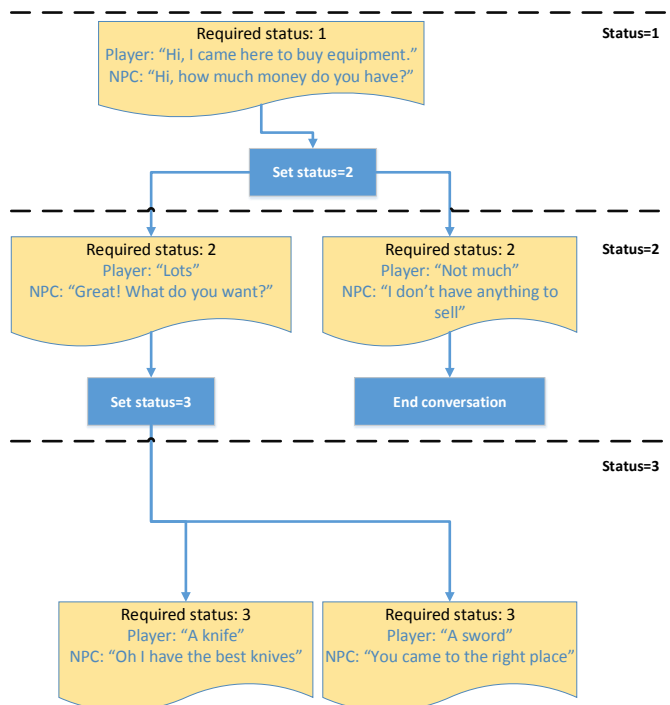


Figure 61: Example of dialog flow, based on dialog status.

4) The importance of side-conversations

Dialog systems usually offer side conversations, which are not part of the dialog’s critical path. For instance, in Rewind,

players have to talk to a scientist that is trapped inside a room. Freeing the scientist is the main topic of the conversation, and it makes the dialog progress. However, during the dialog, players may have dialog options such as: “By the way, how did you end up working here?”, “What do you guys do in this research facility”, or “Do you know who made the emergency call that brought me here?”. Although these options are not part of the main dialog topic, they offer insight into the game’s story and characters. Without these side conversations, the dialog would feel canned and unnatural. Hence adding side-conversations is an essential part of any dialog system.

The “Dialog Status”, however, doesn’t work when handling side conversations. Because these conversations don’t affect the main dialog flow, and the player can trigger them in any order, the “Dialog Status” number is not enough to implement them.

The researcher decided to add a flag system, that allows for side conversations to happen inside the same “Dialog Status”. Flags are essentially the same think as the “Dialog Status”, with the only difference that they are words instead of a number. This paper won’t go on extensive detail regarding their implementation, but the following diagram shows a simplification of the logic behind them:

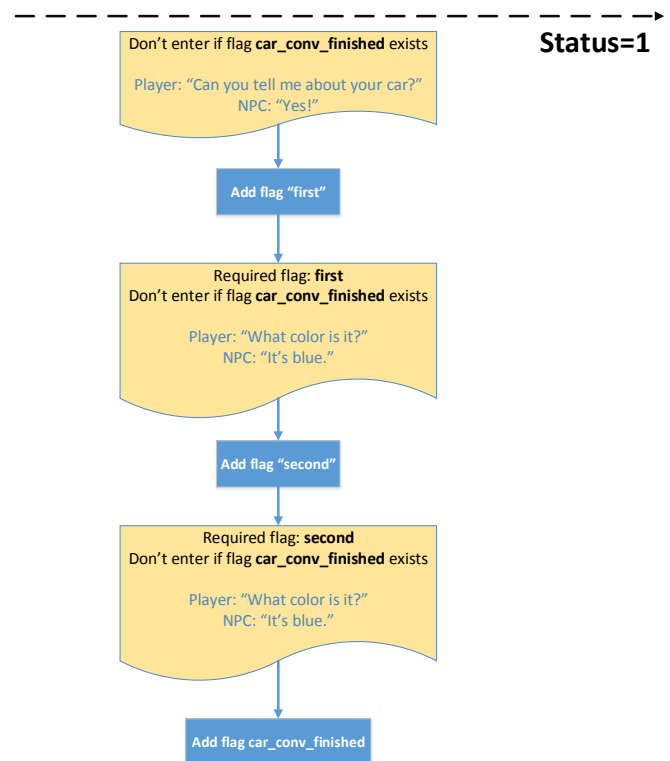


Figure 62: This side conversation happens inside status=1

5) Ease of use: the dialog’s implementation in Engine

Creating a dialog in the engine is similar to creating an Interactive Object. For dialogs, designers need to drop a “BP_MainDialogState” in the world. After that, designers just need to add references to all of the dialog states that they want to use in that conversation. When doing that, designers can

match states and UI slots, so that certain topics appear in specific UI slots. Designers don't need to do much else, as the system's logic takes care of showing the right text, based on the pre-conditions of the states and the dialog's status value.

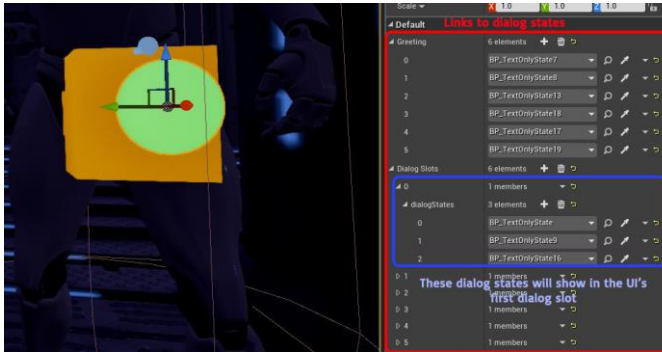


Figure 63: The main dialog state contains references to all the states in a dialog.

While developing the dialog system the researcher tried to make it as user friendly for designers as possible. Dialog states are visible in the world, so designers can organize them by status value. The following caption shows an example of how designers can lay the states in a comprehensible manner, that can help them while debugging:

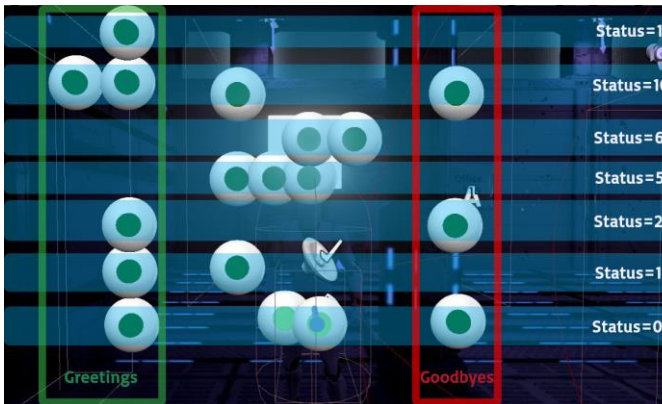


Figure 64: Visualization of a full dialog in engine.

6) Current limitations and potential solutions

Like it happens in the State System, organizing the states of a dialog object can become a task in itself for big enough dialogs. In situations where designers need to completely change a dialog, they also have to rearrange all of the dialog states manually, which can consume a lot of time. Additionally, long dialogs can clutter the editor with their states. One alternative to fix this issue would be to move dialogs to blueprints, instead of the editor. This, however, has its own risks since it could to potentially huge and incomprehensible blueprints. Another alternative would be to implement a script that organizes the dialog states automatically in editor, whenever designers make a change in a state. This system could be implemented inside the Dialog Object's constructor, so that it happens automatically

whenever designers make a change in one of the Dialog States.

The fact that dialog states are different than normal states, is another limiting factor of the Dialog System. Having different states meant that the researcher had to write separate logic to deal with normal states and dialog states. Ideally, both systems would use the same kind of states, so that designers would not have to make a distinction between state types inside blueprints.

O. Summary

For this thesis, the researcher developed several narrative systems for Unreal Engine 4. The researcher started by creating a game design document for a level called Rewind. While working on Rewind's Design, the researcher looked into different engines that he could use for the project, deciding on the Unreal Engine 4 based on its features and available documentation. The researcher then chose the narrative elements that the level needed in order to convey the story appropriately. These elements included comments from the main character, text logs, console panels, an inventory, and environmental storytelling, among others.

After choosing the game's narrative elements, the researcher decided which systems were necessary for Rewind. He decided to implement a level-saving system, a character's comments system, and inventory, a text log system, a notebook, and a dialog system, among others. During the implementation of these systems, the researcher also worked in a demo for Rewind. The researcher used this demo to test the systems, and to get playtesters' feedback on how to improve them. After iterating on the playtest feedback, the researcher finalized the systems and the demo. The final product shows the potential of the systems, that the researcher could use to fully develop Rewind. What's more, designers can also use the systems to develop other narrative-oriented projects in Unreal Engine 4.

IV. CONCLUSION

1) Unreal Engine 4 lessons

Creating the artifacts for this thesis yield several lessons related to both the Unreal Engine 4, and Rewind's narrative techniques. Regarding Unreal Engine 4, the researcher considers that it was the right choice to build the artifact for several reasons:

- It allowed for visual implementation of several systems, such as the state system or the dialog system. Designers can easily work on those systems thanks to the possibility of seeing the states as physical entities in the level editor.
- The visibility of actor's properties in the editor, makes the designers' job easier.
- Unreal Engine's level streaming made teleporting players between the real world and memories a lot easier.
- Overall, Unreal Engine's flexibility and available documentation resulted in an absence of show-stopping blockers during the project.

- Although widgets can be time consuming, their flexibility also helped when developing the inventory and the Notebook, among others.
- Blueprints made the scripting-side of the project go smoothly, and blueprint debugging was extremely helpful.
- Coding was a simple process in Unreal Engine 4.
- Some visual elements that would be hard to implement in other engines, like the post-process effects and the highlights, were straight-forward to implement in Unreal Engine 4.

However, the researcher also found that using Unreal Engine 4 had its drawbacks. Some of the issues that the researcher faced during development included:

- Inheritance in Unreal Engine contains bugs that can disrupt the development process. Some of the most notorious ones were:
 - In some instances, compiling an actor would erase the content of all of its public variables. Depending on the occasion, it led to hours of rework.
 - The researcher had limited control over overwritten functions. In double inheritance, making a call to a “parent function” doesn’t call the function in the immediate parent, it only calls the function in the base class.
 - Sometimes Unreal Engine overwrote public variables with their default values, after the researcher had given them custom values.
 - Debugging classes that inherited from another class did not work properly. Unreal Engine skipped many breakpoints that were inside child classes.
- If the researcher made changes to a struct, all the actors with that struct as a variable could lose the information contained in the struct.
- The researcher could not save object references by using Save Game blueprints, leading to a more complex level-saving logic.
- Deleting some actors could lead to problems in blueprints that referenced those actors.
- Playing in standalone and playing in the editor yielded different results. For instance, the first implementation of the level-saving system worked in editor, but did not work in standalone mode.

2) Other lessons

While building the artifact, the researcher learned several lessons related to narrative and UI implementation:

- The UI should be simple. Players do not read what’s in the UI when it contains too much text.
- Players expect to be able to get out of a menu with the same key they used to get in.
- In first person point-and-click adventures, Interactive Objects need a highlight, so that players can differentiate them from other objects.

- Players tend to miss dialog lines. Important information should be redundant, and appear on screen more than once.
- It is important to communicate terminology to players. In initial iterations of Rewind’s demo, players did not know what the “Main Control Room” was, even when it was obvious for the researcher,
- Players don’t enter menus unless they need to. In Rewind, players never entered the Notebook until it became part of the gameplay.
- Related to the last point, narrative should not be a separate thing from gameplay, not even in the UI.
- Simple controls are better, when possible. The researcher received negative feedback when the controls involved different keys rather than just a simple click.
- Guiding players with lighting is effective. In some occasions a simple lighting change affected the game experience considerably.

3) Additional changes to complete the point-and-click framework

This projects provides designers with a framework to create their own point-and-click adventures, but it doesn’t free them from doing blueprint work to accommodate the systems to their project. Hence designers that want to use the systems, need to know blueprints well and be willing to manipulate the internal logic of this project in order to use the framework that the researcher created.

Given more time, the researcher could work to complete the framework, and create a system that does not require much blueprint work from designers.

With this idea in mind, the researcher has looked into some of the changes that would make this system a complete point-and-click framework:

- State and Dialog Systems

Addition	Cost
Adding a “ State Editor ” to Unreal (different than the blueprint editor), where designers can drop states and connect states in the same way they connect blueprints. Requires adding new UI to Unreal Engine 4 and considerable C++ work.	Very high (1-2 months minimum)
Adding a “ Dialog Editor ”, that designers can use to control the dialog flow directly (drop dialog states, connect them, or change the “Status Number”). Similar to Skyrim’s dialog editor.	

- Interaction System

Addition	Cost
Make the Action Triggers an “Actor Component”, so that they move with the Interactive Objects.	Low
Allowing Collision Areas to have complex shapes, including combinations of different volumes such as spheres, cylinders, or cubes.	Low

- Inventory

Addition	Cost
Allow items of different sizes inside the inventory. Remove the slots and allow players to move items freely in the inventory.	Very high
Make inventory scalable, so that designers can easily change the number of items.	High
Make the inventory a scrollable list, so that there is no limitation on the amount of items that a player can carry.	Medium
Let players add their own descriptions to the items they pick up.	Medium
Let players move an item to the HUD by clicking on the item, and then pressing a number from 1 to 4, instead of having to drag the item.	Low
Let players drop items back in the world after picking them up.	Low

- Narrative menus

Addition	Cost
Let players add their own notes to the Note Menu.	Medium
Make the inventory a scrollable list, so that there is no limitation on the amount of items that a player can carry.	Medium
Let players move an item to the HUD by clicking on the item, and then pressing a number from 1 to 4, instead of having to drag the item.	Low

Let players drop items back in the world after picking them up.

Low

- Other menus

Allow different menus to be on screen at the same time.	Very High
Let designers/players rearrange the menus to their liking. For instance, let players choose to always see the timeline on top of the screen, or let designers merge the timeline and the inventory as a single menu.	Very High
Create other game UI, such as a Control Menu, an Options Menu, or a Pause Menu.	High

- Others

Addition	Cost
Let designers choose their own hotkeys for UI interactions and game interactions. For instance, designers might choose to link the keys 5678 to the HUD inventory items, instead of the keys 1234.	Very High. Requires C++ coding.
Create a general “Game Saving” system, so that players can save the game at any time.	Medium
Add an id tag to objects, that is always visible inside the editor, so that designers can label objects and find them easily.	Low

4) Upcoming changes

The researcher is constantly working on improving the narrative framework, by fixing bugs and adding functionality that may help designers and players alike. These are some of the changes that the researcher is going to implement in the near future:

- Flexibility for Action Triggers
 - Complex shapes for Collision Areas.
 - Make Action Triggers inherit from “Actor Component”
- Facilitating communication between different interactive objects.
 - Develop an easy-to-use system that easily allows designers to change the state of an object as a consequence of the players’ interaction with a different object.
 - Example: players fix the “Fuse Box” in a room, which moves the “Computers” in that room to the “working” state.

- Game saving system, so that players can save the game at any time.
- Tweaks and fixes to the state system to make it more reliable.
 - Some of the logic in the state system's blueprints might be difficult to understand for external designers. A refactoring would help make the system clearer.
- Implementation of .csv files for dialog and logs.

5) *Moving forward*

The researcher met all of the goals that he set at the beginning of the project. With more time, it would be possible to tackle some of the current limitations of the project, expanding its scope.

One of the project's limitations comes from the camera perspective. The demo uses a first person perspective, and Rewind's systems are not ready to be part of a third person narrative game. Hence, adapting the systems to a third person perspective would be a good way of expanding the project.

Another area with potential for expansion is the audio. Although objects can have sounds, and dialog can have voice over, music is not supported in the current state of the artifact.

The level streaming also has some limiting factors. Since a stream cannot have different static lighting, the researcher could not use streaming for certain parts of the levels that could benefit from it. Instead, he had to make identical copies of those level areas and create two separate level streams. This means that whenever the original level changes, the copy needs to change as well. Doing this was not excessively bothersome since those areas were small, but an expansion of the level would definitely require a rethinking of the system.

Another obvious way of expanding the game's scope would be to create Rewind from beginning to end. Although designers could do that with the systems created for this thesis, the expansion would require extra scripting work for specific events and wow moments.

Rewind's UI could also benefit from more iterations. The researcher created throwaway art for these menus, and an art pass would considerably improve their visual quality. Some UIs could also receive extra functionality that would help players. For instance, the Note menu could allow for players to enter their own notes, and decide what events they want to remember.

Finally, systems' flexibility and ease of use could always be beneficial going forward. One way of improving flexibility of the systems would be to make them more generic. That way, designers could use them in other narrative-oriented projects without having to enter the blueprints at all. As they are now, designers can use systems like the HUD for other narrative-oriented games, but they need to remove the Rewinder from it first. Regarding usability, some systems could be simpler, which would increase their ease-of-use. For instance, creating a pickup object is simple, but it requires a pickup blueprint and three states. With some time, the researcher could create a new blueprint that contains the three states inside of it, so that designers would only have to drop that blueprint into the world.

REFERENCES

- [1] E. Hera, "Heartstone dev invents stories that tell themselves," 2014. Polygon. [Online]. Available: <http://www.polygon.com/2014/7/28/5929187/hearthstone-storybricks-storytelling-engine-ai-director-blizzard>. [Accessed 12 September 2015].
- [2] A. Shirinian, "The uneasy Merging of Narrative and Gameplay," 2010. Gamasutra. [Online]. Available: http://www.gamasutra.com/view/feature/132641/the_uneasy_merging_of_narrative_.php. [Accessed 10 September 2015].
- [3] T. Lee, "Designing Game Narrative," October 2013. [Online]. Available: <http://hitboxteam.com/designing-game-narrative>. [Accessed 11 September 2015].
- [4] F. Cifaldi, "Why are We Still Talking about LucasArts Old Adventure Games?", 2013. Gamasutra. [Online]. Available: http://www.gamasutra.com/view/feature/189899/why_are_we_still_talking_about_.php. [Accessed 10 March 2016].
- [5] C. Fernandez-Vara, "Shaping Player Experience in Adventure Games: History of the Adventure Game Interface" in *Structure, analysis and design of computer game player experience*, 1st edition, Lapland University Press, 2008, pp. 210-223.
- [6] A. Freed, "Branching Conversation Systems and the Working Writer", 2014. Gamasutra. [Online]. Available: http://www.gamasutra.com/blogs/AlexanderFreed/20140909/225281/Branching_Conversation_Systems_and_the_Working_Writer_Part_2_Design_Considerations.php. [Accessed 12 March 2016].
- [7] A. Chmielarz, "Seven deadly sins of adventure games", April 2014. *The Astronauts* development blog. [Online]. Available: <http://www.theastronauts.com/2014/04/seven-deadly-sins-adventure-games/>. [Accessed 15 March 2016].
- [8] *Syberia* (PC). USA: Benoid Sokal, Microids, 2002.
- [10] *Grim Fandango* (PC). USA: LucasArts, 1998.
- [12] *The Room Three* (iOS). United Kingdom: Fireproof Games, 2015.
- [14] *Amnesia* (PC). Sweden: Frictional Games, 2010.
- [16] *The Legend of Zelda: Majora's Mask* (Nintendo 64). USA: Nintendo, 2000.
- [17] Several authors, "*The Legend of Zelda: Majora's Mask* Official Website", 2015. IGN. [Online]. Available: <http://zelda.com/majoras-mask/> [Accessed: 03 October 2015]

TABLE OF FIGURES

Figure 1: Syberia's inventory (left menu) and document menu (right menu)[9]	3
Figure 2: Grim fandango's unique art style contributes to its uniqueness. [11].....	4
Figure 3: The Room's simple HUD. [13]	4
Figure 4: Amnesia's inventory clearly separates gameplay objects from storytelling objects. [15]	5
Figure 5: The size of the moon in Majora's Mask shows how much time has passed since the first day [18].....	5
Figure 6: The main character's comments give hints about his personality.	9
Figure 7: Players can find consoles in several locations.	9
Figure 8: Players can read the console's content by clicking on it.	9
Figure 9: Text logs are pda-like objects that players find around the level.	10
Figure 10: Picking up a log opens the log screen. Plays can go back to this screen later.....	10
Figure 11: A message reminds players how to use the Rewinder, when they use incorrectly.....	10
Figure 12: A message on screen tells players that the Rewinder is retrieving another characters' memories.	10
Figure 13: A post-process effect helps players differentiate between the real world and a memory.	11
Figure 14: Highlight of an Interactive Object, as seen in-game	12
Figure 15: View of an action trigger in the engine	13
Figure 16: How action triggers control object interactions..	13
Figure 17: How action triggers connect to Interactive objects.....	13
Figure 18: Designers can assign the same behavior to several objects by using the action triggers.....	14
Figure 19: Visual representation of the mug example.	14
Figure 20: Use of multiple Action Triggers in the demo	14
Figure 21: Action Triggers can show a highlight on objects that are not interactive.	14
Figure 22: Example of highlighting external objects.....	15
Figure 23: First iteration of the HUD with visible inventory.....	16
Figure 24: First iteration of the HUD, with hidden inventory.....	16
Figure 25: Intermediate iteration of the HUD.....	16
Figure 26: Final implementation of the main HUD	17
Figure 27: Players' HUD when they are in a memory	17
Figure 28: Structure of a Text Array.....	18
Figure 29: Graph of the Text Array functionality.....	18
Figure 30: Text arrays distinguished between "observing" an object and "picking up" an object.	19
Figure 31: An object can have several states, but it can only be in one state at the same time.	20
Figure 32: Example of flow between the states of an Interactive Object.....	20
Figure 33: General structure of a state.....	21
Figure 34: Properties of an Object State actor	21
Figure 35: Example of use for "StateId"	22

Figure 36: States have a visual representation in the world, so that level designers can organize them as they prefer.	23	Figure 67: A post-process effects helps players differentiate between memories and the real world.	53
Figure 37: The states for a certain Interactive Object form some kind of "Linked List"	23	Figure 68: Players learn about the main character's personality through his comments.	54
Figure 38: Menu hierarchy in Rewind.	24	Figure 69: A new screen opens when players interact with a console.	55
Figure 39: Initially the inventory was part of the HUD.	24	Figure 70: Screen that opens when players interact with a log.	56
Figure 40: Final version of the small inventory.	25	Figure 71: HUD concept	57
Figure 41: The big inventory, where players can see more details about their items.	25	Figure 72: Real world HUD	58
Figure 42: The Item Storage contains the information of all of the level's items.	25	Figure 73: Memory HUD	59
Figure 43: Whenever the designer needs an item's information, he can make a simple request to the item storage.	25	Figure 74: Rewind's Inventory	60
Figure 44: Designers can add an item to the inventory by just using its id.	26	Figure 75: The Notebook gives players access to three submenus	61
Figure 45: Diagram of menu hierarchy	26	Figure 76: Menu hierarchy in Rewind.	62
Figure 46: In the final iteration, all storytelling menus are part of the Notebook	27	Figure 77: The main character takes notes about his discoveries.	63
Figure 47: The first version of the timeline menu had one timeline per character.	27	Figure 78: Initial concept for the log menu.	64
Figure 48: The second version of the timeline used a vertical layout	27	Figure 79: Final iteration of the log menu, as it is in game.	65
Figure 49: Final iteration of the timeline menu	27	Figure 80: Players can read a log again by clicking on the buttons on the left sidebar.	66
Figure 50: Adding events to the timeline is a simple process	28	Figure 81: In the initial concept, the timeline was part of the HUD.	67
Figure 51: The main character writes down some of his thoughts on the note menu.	28	Figure 82: Timeline menu in the final game.	68
Figure 52: Initial iteration of the Note Menu	29	Figure 83: Incinerator concept	71
Figure 53: Players can find logs in the world.	29	Figure 84: Chronological events for the player.	74
Figure 54: First pass of the Note Menu	29	Figure 85: Chronological events for Dr. Anderson	76
Figure 55: Final iteration of the log menu	30	Figure 86: Chronological events for Dr. Cook	77
Figure 56: Log text file format	30	Figure 87: Area breakdown of the level	78
Figure 57: After setting the LogPath variable, the log is almost ready.	30	Figure 88: First iteration in Area 1 (Investigator)	79
Figure 58: The final dialog interface	32	Figure 89: Second iteration in Area 1 (Dr. Anderson 7pm) .	81
Figure 59: Structure of a dialog state	32	Figure 90: Third iteration in Area 1 (Investigator)	82
Figure 60: Logic behind dialog state availability	33	Figure 91: Fourth iteration in Area 1 (Dr. Anderson 6pm-7pm)	84
Figure 61: Example of dialog flow, based on dialog status.	33	Figure 92: Fifth iteration in Area 1 (Investigator)	86
Figure 62: This side conversation happens inside status=1	33	Figure 93: First iteration in Area 2 (Investigator)	88
Figure 63: The main dialog state contains references to all the states in a dialog.	34	Figure 94: Second iteration in Area 2 (Dr. Anderson 5pm-6pm)	90
Figure 64: Visualization of a full dialog in engine.	34	Figure 95: Third iteration in Area 2 (Investigator)	92
Figure 65: Rewind concept [1]	40	Figure 96: First iteration in Area 3 (Dr. Anderson 3pm-6pm)	94
Figure 66: An icon (bottom right) shows up every time players are in front of a valid DNA sample.	50	Figure 97: Second iteration in Area 3 (Investigator)	96
		Figure 98: Third iteration in Area 3 (Dr. Cook 9am-10am) .	98
		Figure 99: Fourth iteration in Area 3 (Investigator)	100

V. APPENDIX

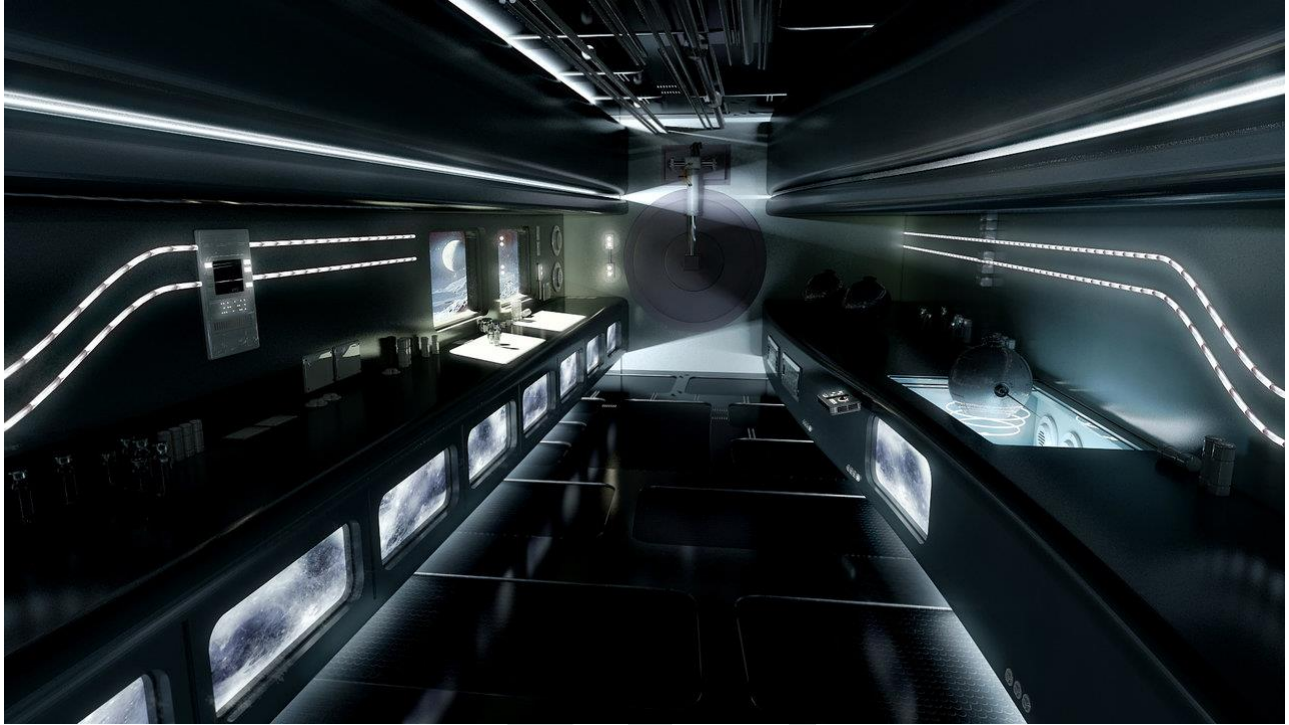


Figure 65: Rewind concept [\[1\]](#)

Level Design Document: Rewind

Unreal Engine 4

Version 1.0

Designer:

Jorge Montolio Conde

Document Date:

11/17/2015

Intended Level Delivery Date:

02/22/2015

A. Document Revisions Table

[illegible]

B. TABLE OF CONTENTS

Table of Contents	42
Table of Figures	44
Quick Summary	45
Hook(s) 45	
Level goals 45	
Gameplay Minute 45	
Gameplay details	49
The Rewinder 49	
Interactions 51	
Gameplay Example 51	
Narrative elements	53
Memories 53	
Main character comments 54	
Consoles 55	
Logs 56	
The GUI	57
HUD 57	
Inventory 60	
Notebook 61	
Notes.....	63
Logs.....	64
Timeline	67
Dialog and player thoughts 69	
Story Summary	70
Backstory 70	
Levels story 70	
Aftermath 71	
Characters 72	
Level Summary 73	
Campaign 73	
Context	73
Backstory.....	73
Aftermath	73
Objective(s) 73	
Overview Maps 74	
Chronologically ordered events - Player	74
Chronologically ordered events – Dr. Anderson.....	76
Chronologically ordered events – Dr. Cook.....	77

Level Details	78
Level Areas 78	
Alarm levels 78	
Detailed Walkthrough 79	
Area 1: Entrance area	79
Area 2: The laboratory	88
Area 3: The Incineration room and the dormitories	94
References	101

CONFIDENTIAL

C. Table of Figures

Figure 1: Rewind concept [1]	40
Figure 2: An icon (bottom right) shows up every time players are in front of a valid DNA sample.	50
Figure 3: A post-process effects helps players differentiate between memories and the real world.	53
Figure 4: Players learn about the main character's personality through his comments.	54
Figure 5: A new screen opens when players interact with a console.	55
Figure 6: Screen that opens when players interact with a log.	56
Figure 7: HUD concept.	57
Figure 8: Real world HUD.	58
Figure 9: Memory HUD.	59
Figure 10: The Notebook gives players access to three submenus.	61
Figure 11: Menu hierarchy in Rewind.	62
Figure 12: The main character takes notes about his discoveries.	63
Figure 13: Initial concept for the log menu.	64
Figure 14: Final iteration of the log menu, as it is in game.	65
Figure 15: Players can read a log again by clicking on the buttons on the left sidebar.	66
Figure 16: In the initial concept, the timeline was part of the HUD.	67
Figure 17: Timeline menu in the final game.	68
Figure 18: Incinerator concept.	71
Figure 19: Chronological events for the player	74
Figure 20: Chronological events for Dr. Anderson	76
Figure 21: Chronological events for Dr. Cook	77
Figure 22: Area breakdown of the level	78
Figure 23: First iteration in Area 1 (Investigator)	79
Figure 24: Second iteration in Area 1 (Dr. Anderson 7pm)	81
Figure 25: Third iteration in Area 1 (Investigator)	82
Figure 26: Fourth iteration in Area 1 (Dr. Anderson 6pm-7pm)	84
Figure 27: Fifth iteration in Area 1 (Investigator)	86
Figure 28: First iteration in Area 2 (Investigator)	88
Figure 29: Second iteration in Area 2 (Dr. Anderson 5pm-6pm)	90
Figure 30: Third iteration in Area 2 (Investigator)	92
Figure 31: First iteration in Area 3 (Dr. Anderson 3pm-6pm)	94
Figure 32: Second iteration in Area 3 (Investigator)	96
Figure 33: Third iteration in Area 3 (Dr. Cook 9am-10am)	98
Figure 34: Fourth iteration in Area 3 (Investigator)	100

D. Quick Summary

“Rewind” is a single player, point-and-click adventure level for Unreal Engine 4. “Rewind” follows the investigation of a 25th century detective, Dale Cooper, as he arrives at a laboratory where a terrible accident has happened. To help him, Dale has a special tool known as the “Rewinder”, a device that allows him to enter people’s memories after analyzing their DNA. By using the rewinder, Cooper is able to witness the event of that day through the eyes of the laboratory’s scientists.

“Rewind” is a unique point-and-click adventure in that players witness the events of the day out of order, from the end of the day to the beginning. In addition to using the Rewinder, players can pick up keycards and other objects that help them progress through the facilities, as well as interact with some of the malfunctioning laboratory’s equipment. By using the “Rewinder”, gathering clues, and picking up objects, players need to enter the dark rooms of the laboratory, and shed light on the mysterious events that led to the current state of the facility.

1) Hook(s)

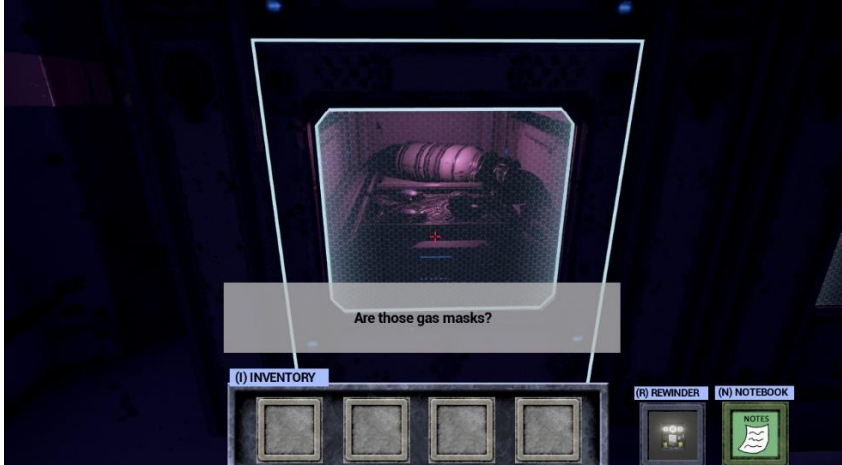


- Using the Rewinder to control different characters.
- Using the Rewinder to enter other characters’ minds and see the past through their eyes.
- Seeing the laboratory at different points in time.
- Seeing the end of a scene before seeing the beginning. In the laboratory, players find a different scenario in each area, and they must see the past in order to understand what has happened in that part of the laboratory.


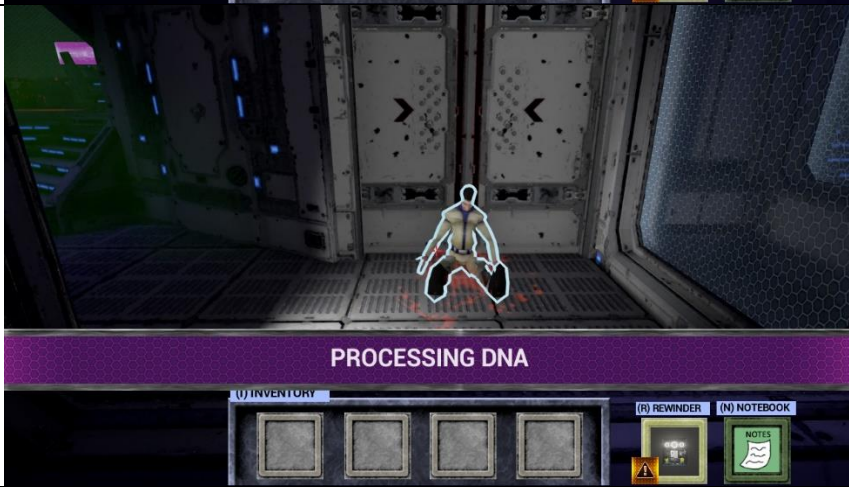
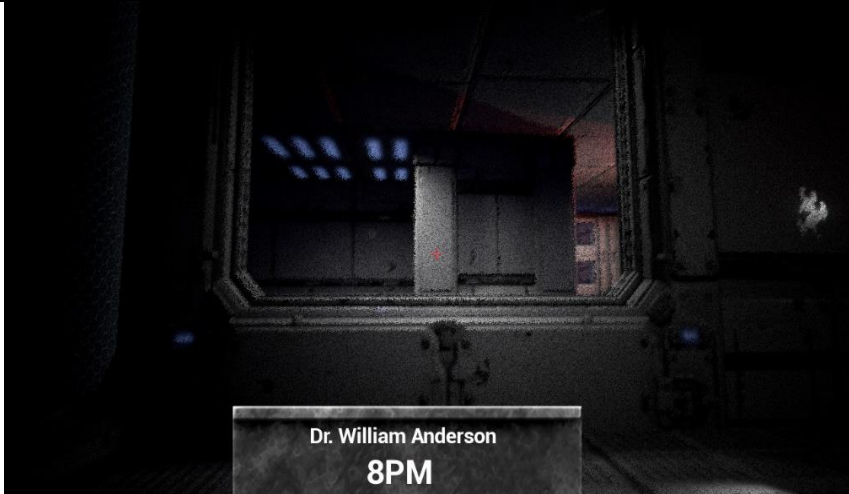
2) Level goals

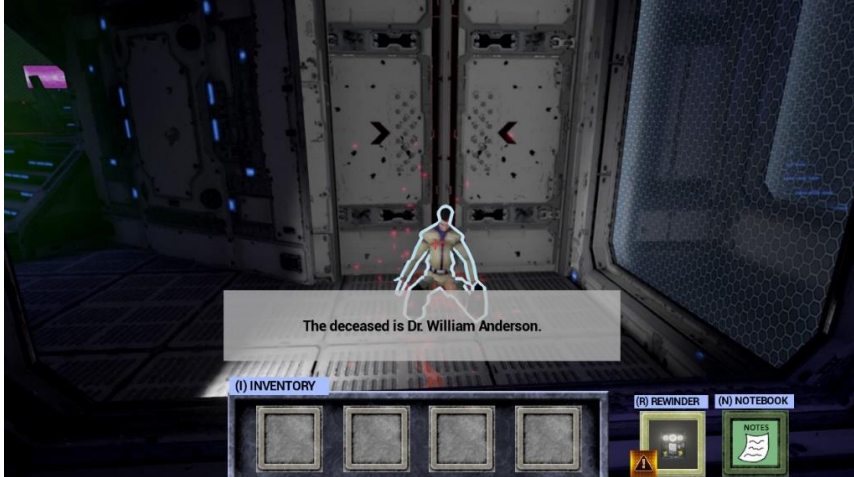
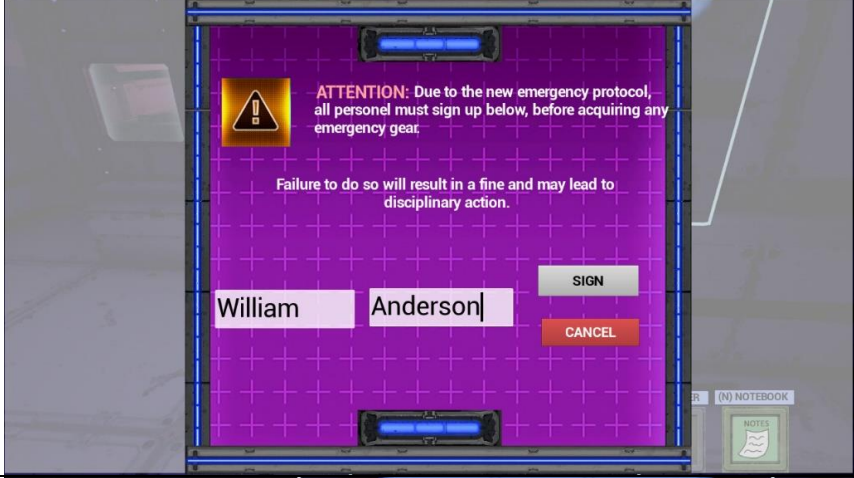

- Find out what happened in the laboratory.
- Find out who made the emergency call and why.
- Save any laboratory personnel who remains in the facilities.



3) Gameplay Minute

	<p>Players find a hallway filled with toxic gas. They need a gas mask to get through it.</p>
--	--

	<p>Players find a container with gas masks inside.</p>
	<p>Close to the container there is a console that players need to use to open the gas mask containers.</p>
	<p>To activate the console, players need the name of one of the laboratory's scientists.</p>

	<p>Players look around the room and find the dead body of one of the scientists.</p>
	<p>Players use the Rewinder to enter the character's memory.</p>
	<p>Once inside the character's memory, players find out about the character's name, thanks to the DNA analysis.</p>

	<p>Players go back to the real world after the memory finishes.</p>
	<p>Players use the character's name on the console.</p>
	<p>The mask container opens after players introduce the name.</p>

	Players pick up the mask.
	Players use the mask to progress to the next area.

E. Gameplay details

1) The Rewinder

The main mechanic of the level is “rewinding”. Whenever players find another character’s DNA, they can enter his memories and see the past through their eyes. For example, if the player collects DNA that Dr. Cook left after drinking from a cup at 9am, the player can then see the past through Dr. Cook’s eyes. The memory starts sometime before Dr. Cook used that cup, and ends right when he started drinking from the cup. DNA samples have particle effects around them, to help players identify them. In addition to the particles, an icon in the HUD pops up whenever players are looking at a sample, reinforcing the visual feedback and letting them know that they can use the “Rewinder”. Once this icon appears on the screen, players just need to left click with their mouse to start the teleportation to the other character’s mind.



Figure 66: An icon (bottom right) shows up every time players are in front of a valid DNA sample.

Inside a memory, the game functions exactly in the same way as if they were in the present. A light overlay tells player that they are inside some else's memory, and the HUD updates with the name of the character that they are controlling, as well as the time of day that they are seeing in the memory.

When players are inside a character's memory, they are taking the role of that character for the duration of the memory. However, memories are in no way a time travel, and players cannot change any present or future events while they are in them. Memories serve merely as a way to tell the story to the player through another character's perspective. Once players enter a memory, they stay in the memory until the memory finishes (i.e. they reach the part of the story where they need to go back to the present). In cases when players are under extreme circumstances (running away from a fire, trying to get out of a room that is about to explode, etc.), dying sends players back to the beginning of the memory. At the end of the memory, a simple fade to black and a "Losing connection to memory" message on the HUD, tells players that they are about to return to their original body.

2) Interactions

The level is a point and click adventure, where players progress by exploring the environment. Players can interact with environment objects, collect items, and use the items in the right places in order to unlock new areas. Sometimes, players have to interact with consoles in order to progress by introducing a password or a character's name to activate the console.

The way that players can interact with the world are:

Keys	Action	Description
WASD	Walk	Players can walk around the level with the WASD keys. Jumping and sprinting are not available in Rewind.
Left Mouse Button	Observe	Players can observe any objects with a highlight around them. Observing an object shows a comment by the main character about that object.
	Use	Players can use some objects in the environment, such as consoles or doors. For those objects, using the left-mouse buttons triggers a reaction in the object, along with a comment from the main character.
	Pick-up	When players interact with a pickup item, the main character picks up the item. The item goes to the players' inventory automatically.
	Opening console screens	When players interact with certain consoles, an auxiliary screen opens. This screens may be purely informative, or they may ask players for a password or code.
R	Rewind	When players are in front of a DNA sample, an icon on the HUD tells them that they can use the Rewinder. If players press R while the icon is on the HUD, they automatically enter the character's memory.
I	Inventory	Players can open the inventory by pressing "I"
N	Notebook	Players can open the notebook by pressing "N". The Notebook is a menu that contains all of the narrative-relevant information in the level (see Notebook section).
1-4	Using items	Players can use items from their inventory by pressing a number from 1 to 4. Players can see the items assigned to each number in the HUD. They can also change these numbers by entering the inventory.

3) Gameplay Example

The gameplay follows a simple pattern, where players switch between the present and other characters' memories. From the present, players can explore and find DNA, which they can use to enter a memory and see the past. Once in the memory, players can gather information (such as a key code or a name), that allows them to progress further once they go back to the present.

The following is an example gameplay scenario for the game:

- Players need to access the lower laboratory, but there is no electricity in the elevator that connects to the lower laboratory's floor.
- Players find some of Dr. Anderson's DNA and enter his memories.
- Inside Dr. Anderson's memory, players see Dr. Anderson entering a code in a console. With this code, he is able to turn the electricity off in the facility.
- Players go back to the present and use that code on the console to turn electricity back on.
- Players use the elevator and access the next area.

CONFIDENTIAL

F. Narrative elements

1) Memories

Memories are the main narrative element in Rewind. Every time players find another character's DNA, they can use the Rewinder to enter the character's memories. Inside the memories, players play as that character, which gives them a unique perspective of the character's actions during that day. Once players reach the moment in time when the character lost the DNA, the memory ends and players go back to the real world.

Memories are both a narrative element and a gameplay element. Although they give players more insight into the level events, they also provide information that helps players progress through the level.

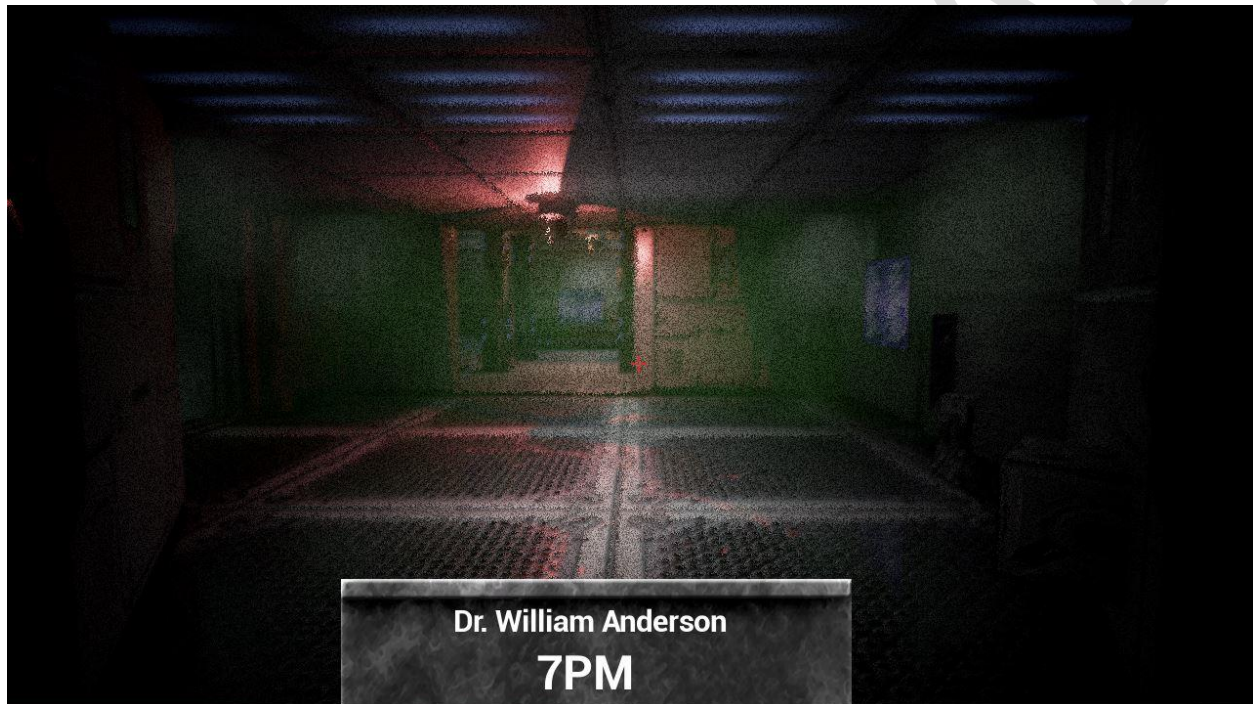


Figure 67: A post-process effects helps players differentiate between memories and the real world.

2) Main character comments

In Rewind, the main character makes comments every time players interact with an object. Since players spend most of the level alone, these comments are the only way of conveying the main character's personality and thoughts to players.



Figure 68: Players learn about the main character's personality through his comments.

3) Consoles

Players can find several consoles inside the laboratory. All of the consoles give players information about the live inside the laboratory, the laboratory personnel, or the laboratory's protocols. The consoles are a way of telling the laboratory's backstory to players, since most of the secondary characters in Rewind are dead or missing.

To interact with a console, players can just click on it. Clicking on it opens a new screen, where players can see the console's information. In some cases, players may have interact with the console by entering a password or a code, that in turn activates some mechanism inside the room.

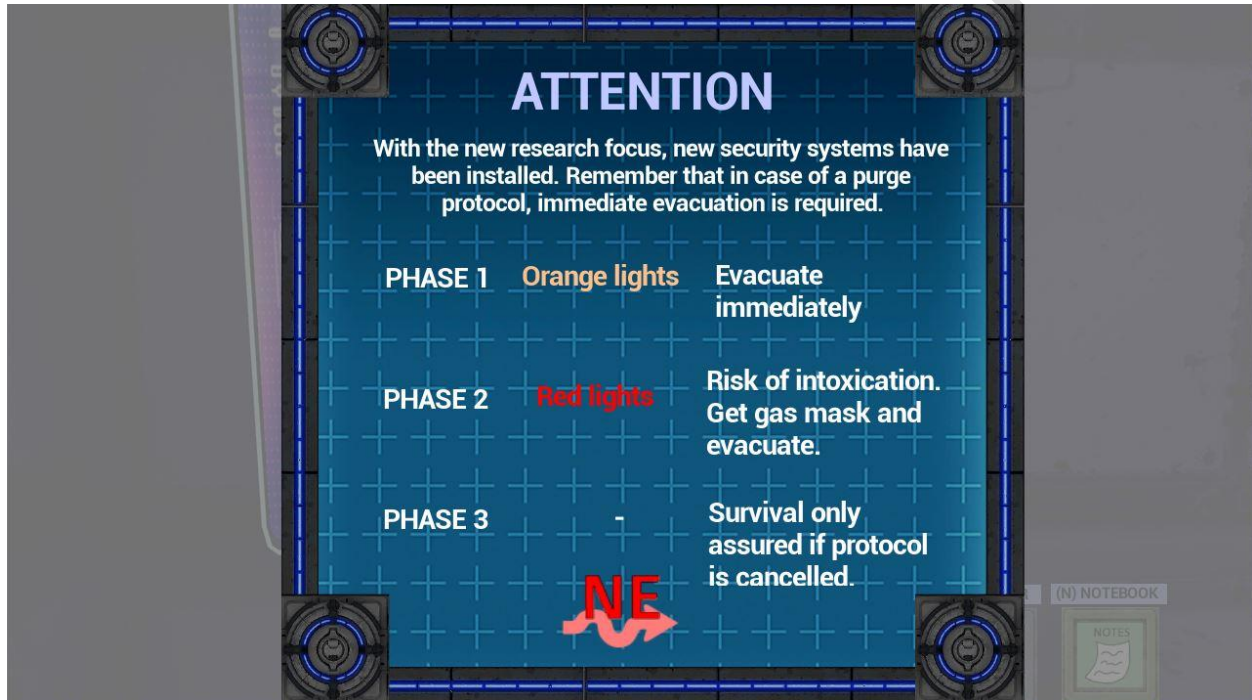


Figure 69: A new screen opens when players interact with a console.

4) Logs

In Rewind, players can find text logs written by the laboratory's scientist and other laboratory personnel. Some of these logs are part of the scientists' diaries, while some others are just scientific notes or comments about a specific research. The logs purpose is to tell players about the story of the laboratory's scientist, as well as their personality and motivations. Since players only gets to meet one of Rewind's secondary characters in person, logs are the only way for players to learn about all of the other secondary characters.

Once players interact with a log, an auxiliary screen opens, showing the log contents. After players close this screen, the log automatically goes to their Notebook, so that players can look at the log contents later.

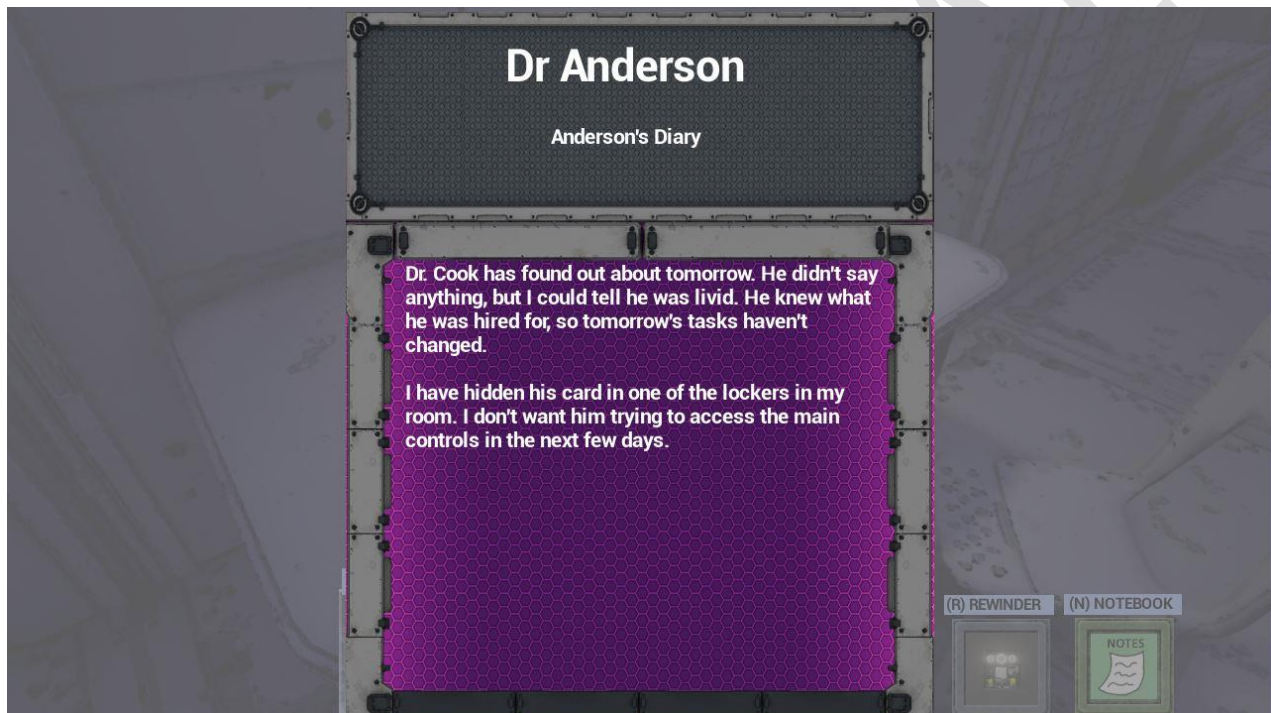


Figure 70: Screen that opens when players interact with a log.

G. The GUI

1) HUD

The game's constantly displays essential information through the HUD, which is always visible on screen. The following captions shows an early version of the HUD:

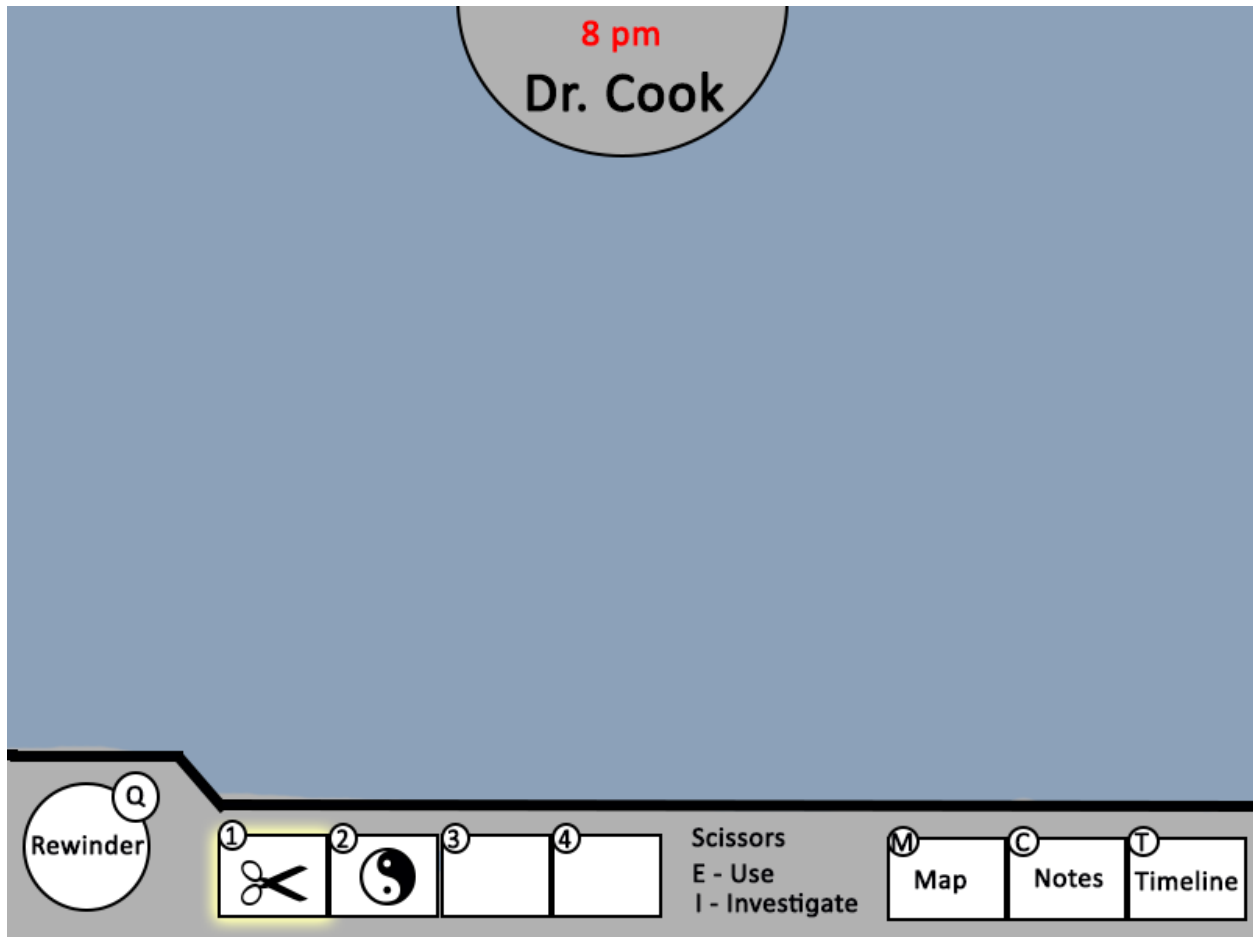


Figure 71: HUD concept

The final version of the HUD has a simpler layout, with only the essential information so that the HUD doesn't take a significant portion of the screen.



Figure 72: Real world HUD

The HUD always contains the following information:

- Item Bar
Four items from the inventory. Players can get into the inventory and change these four items at any time. New items go directly to the item bar, if there is a free spot. A small blue box also reminds players that they can open the inventory by pressing “I”.
- A “Rewinder” icon, along with the key to press to use the “Rewinder”.
- A Notebook icon, along with the key to press to open the Notebook.

The HUD changes when players enter a memory. In memories, players can not use items, the “Rewinder”, or the Notebook, so the HUD becomes a lot simpler:

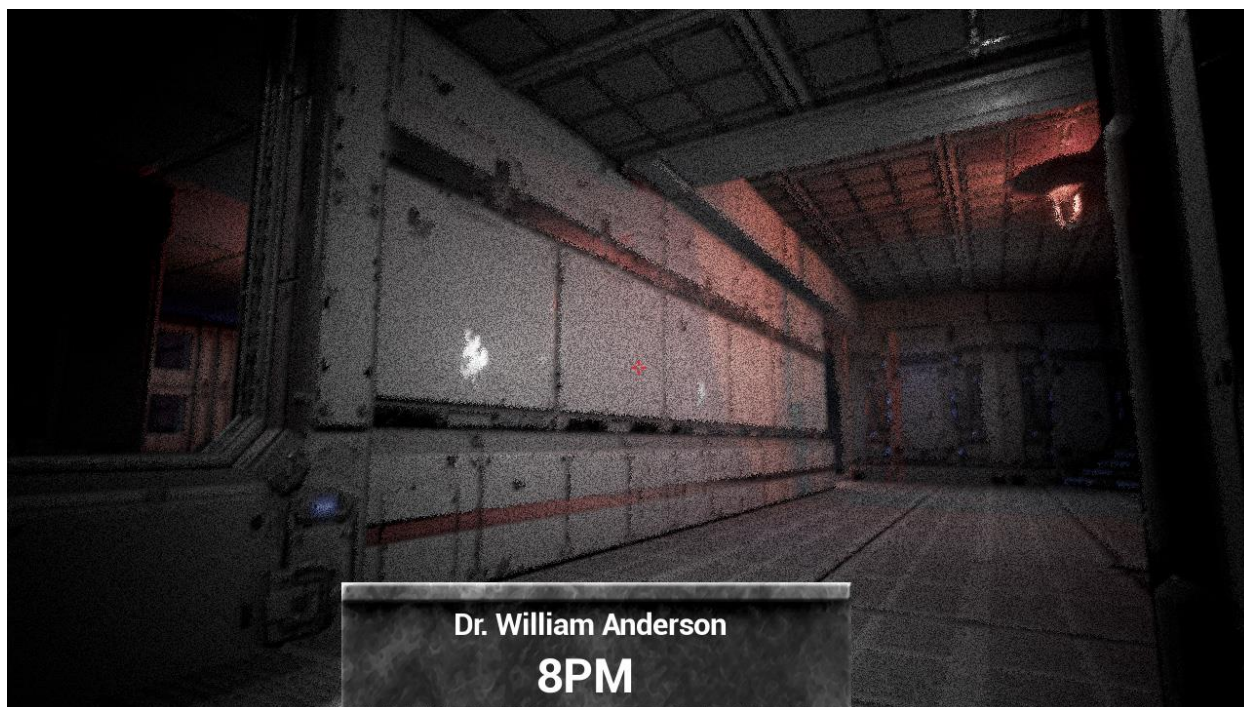


Figure 73: Memory HUD

Inside a memory, the HUD only displays the name of the character whose memory players have entered, and the time of day when that memory happened.

2) Inventory

Players can open the inventory at any time by pressing I. The inventory gives players access to the game's items, as well as to the "Rewinder" and the Notebook (although players can also access the "Rewinder" and the Notebook from the HUD).

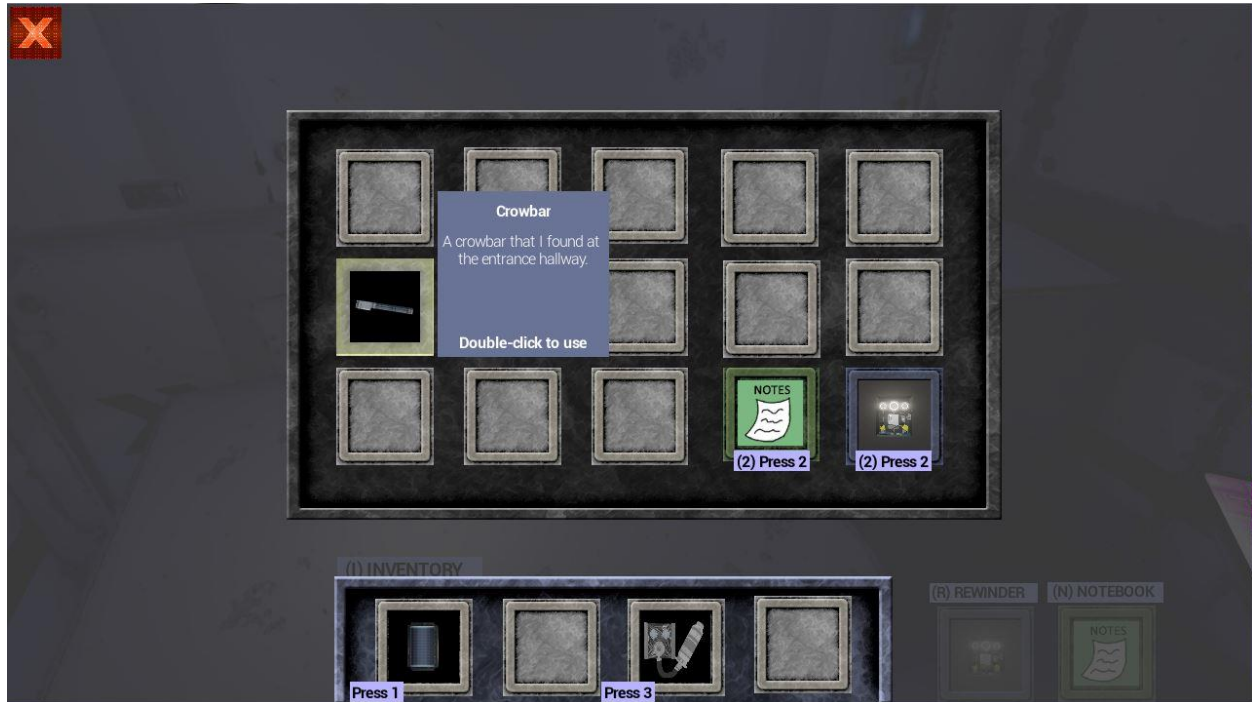


Figure 74: Rewind's Inventory

From the inventory, players can move items around by clicking and dragging them with their mouse. Players can also click on an item to see the item's name and a description of the item. Double clicking on an item or on the "Rewinder", closes the inventory and automatically uses the item/"Rewinder" in the world. If players double click on the Notebook, the Notebook menu opens.

3) Notebook

The Notebook is a menu that contains all of the narrative-relevant information that players gathered throughout the level. From the Notebook, players can access text logs, the main character's notes, as well as a timeline of the level's events. Players can open the Notebook from the HUD by pressing N, or from the inventory by double clicking on it.

The Notebook contains three submenus, each with a different purpose:

- The Note Menu, where the main character takes notes about his observations while exploring the level.
- The Log Menu, where players can access text logs after they collect them in the level.
- The Timeline, that displays the level events in chronological order.

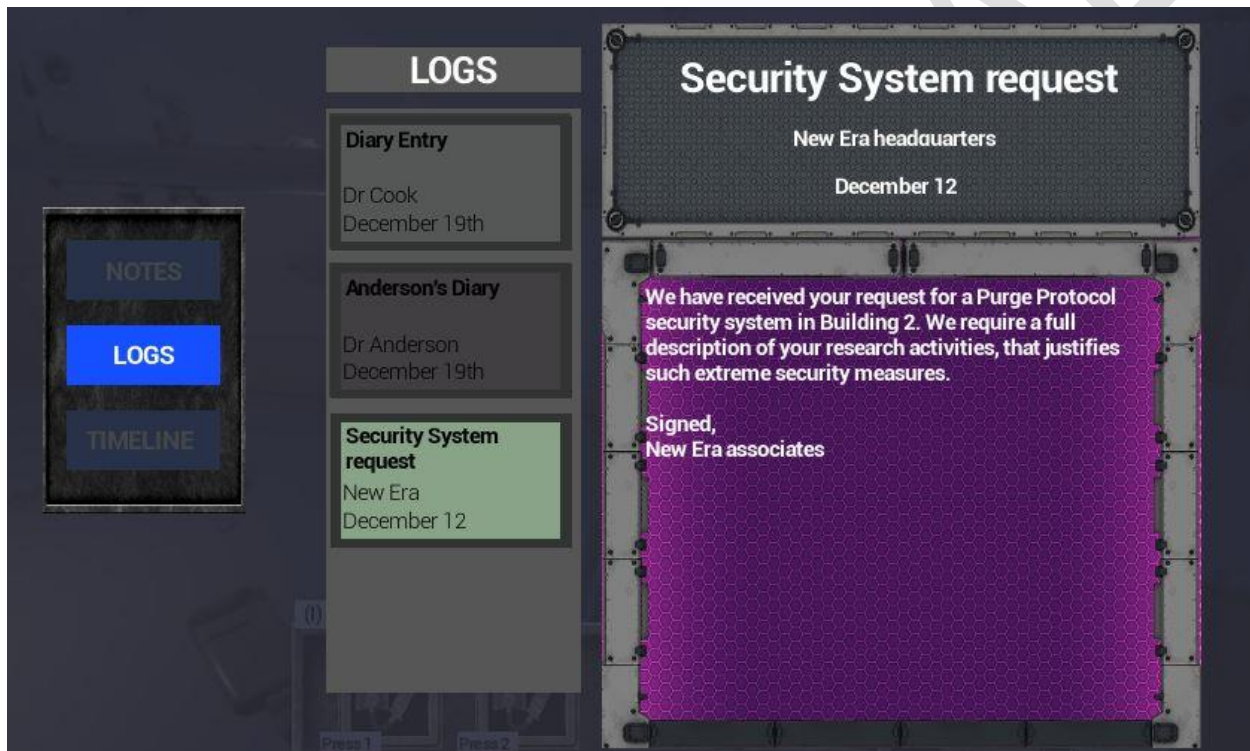


Figure 75: The Notebook gives players access to three submenus

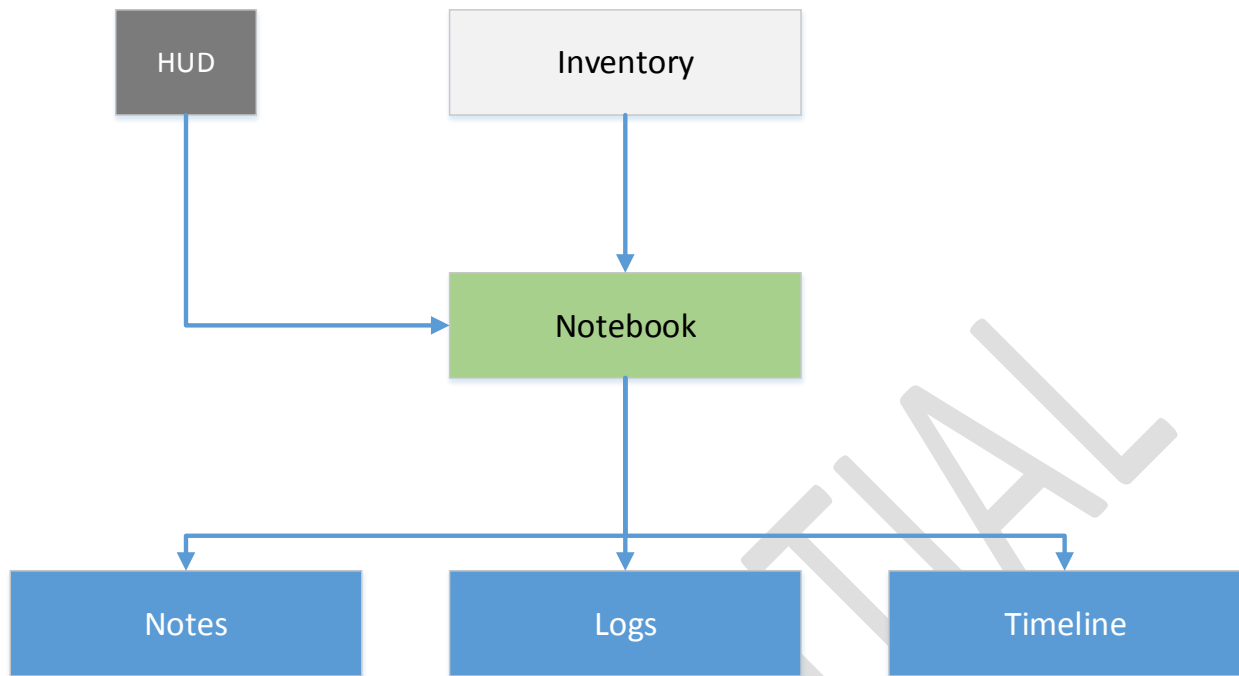


Figure 76: Menu hierarchy in Rewind.

a) *Notes*

Every time players find out about an important piece of information, the main character writes it down in the Note Menu. To inform players about the update in the menu, the main character makes a comment every time he adds a note. For instance, after discovering a character's personal keycode, the main character may say "I better write down this code in my notes, so that I don't forget".



Figure 77: The main character takes notes about his discoveries.

The Note menu helps player remember key pieces of information, some of them gameplay-relevant, such as passwords and codes.

The Note menu can have as many pages as needed, and as soon as a page fills out, another page starts. Players can navigate through the pages by using two green arrow buttons on the side of each page, which send them to the previous or next page in the Note menu.

b) Logs

In the game, players can collect logs written by other characters. Some logs contain codes for doors and machinery, so it is important for players to be able to go back and review the logs that they have gathered throughout the level. All of the logs that players pick up during the level, go to the Log Menu. Players can access the Log Menu from the Notebook, and review any of the logs in their possession.

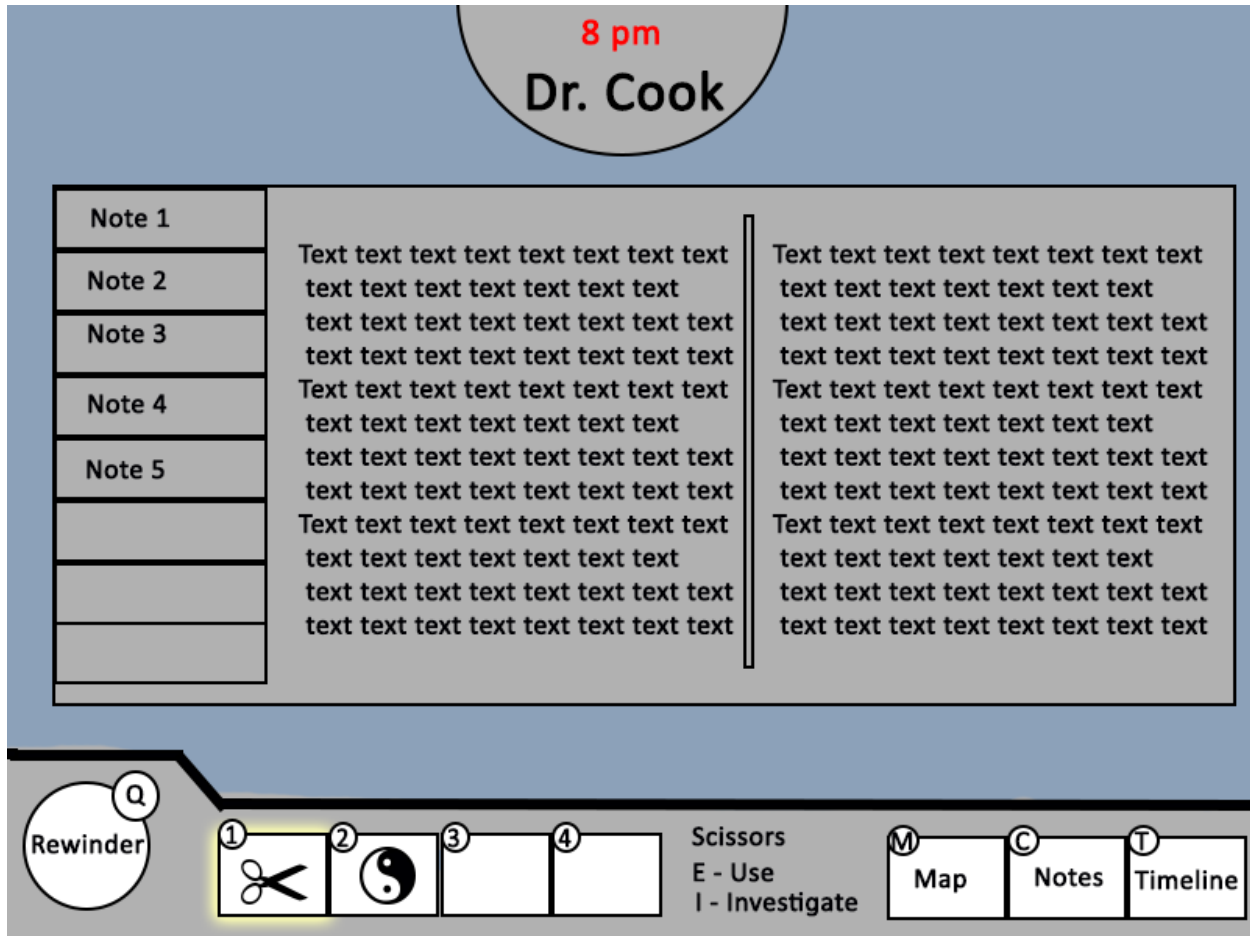


Figure 78: Initial concept for the log menu.

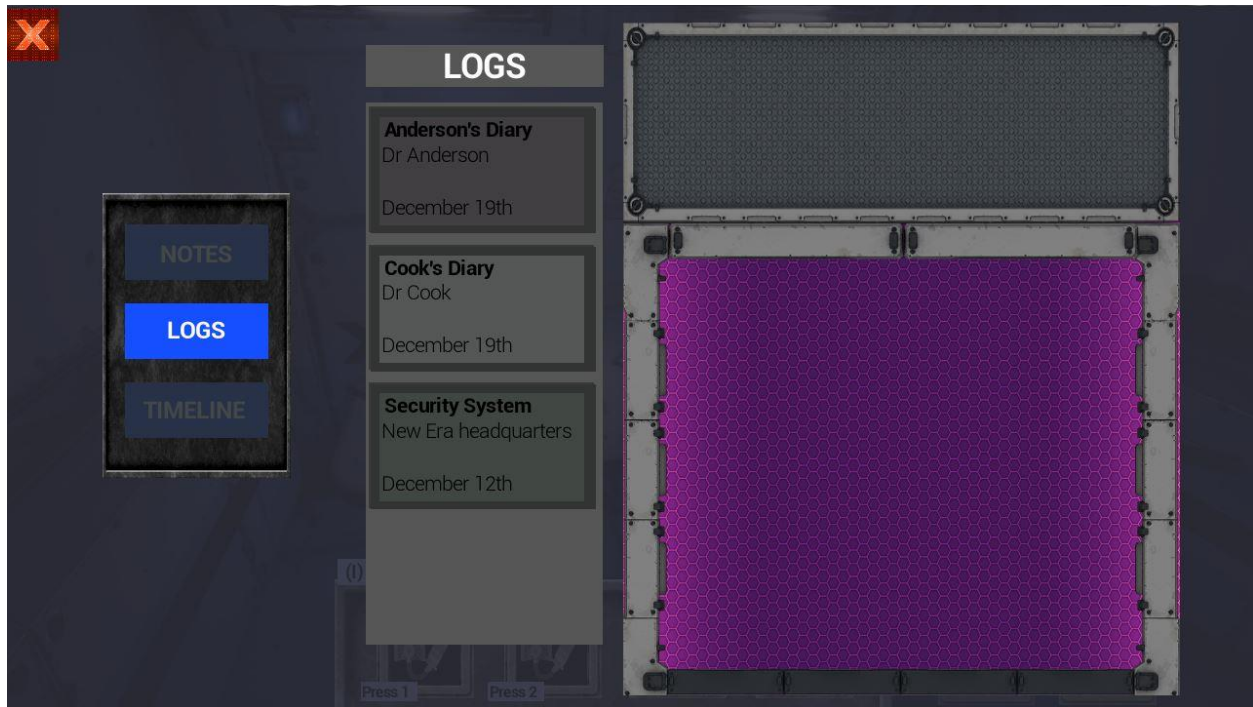


Figure 79: Final iteration of the log menu, as it is in game.

Once players click on one of the buttons on the left sidebar, the information contained in that specific log shows up on the screen. The buttons on the left sidebar have different colors based on the character that wrote the log:

Background Color	Character
Pink	Eve
Green	Dr. Cook
Blue	Dr. Anderson
Grey	Others



Figure 80: Players can read a log again by clicking on the buttons on the left sidebar.

c) *Timeline*

Players witness the events of the level out of order, which forces them to chronologically organize the level events in their mind. To help them with that, Rewind has the Timeline Menu, which shows them the events of the day chronologically ordered. Every time the player witnesses an important story moment, the event goes to the timeline, so that players can later look at the levels' story as a whole. Like it happens with the Log Menu and the Note Menu, the Timeline Menu is a purely informative menu. Players can access the Timeline from the Notebook (see [Notebook section](#)).

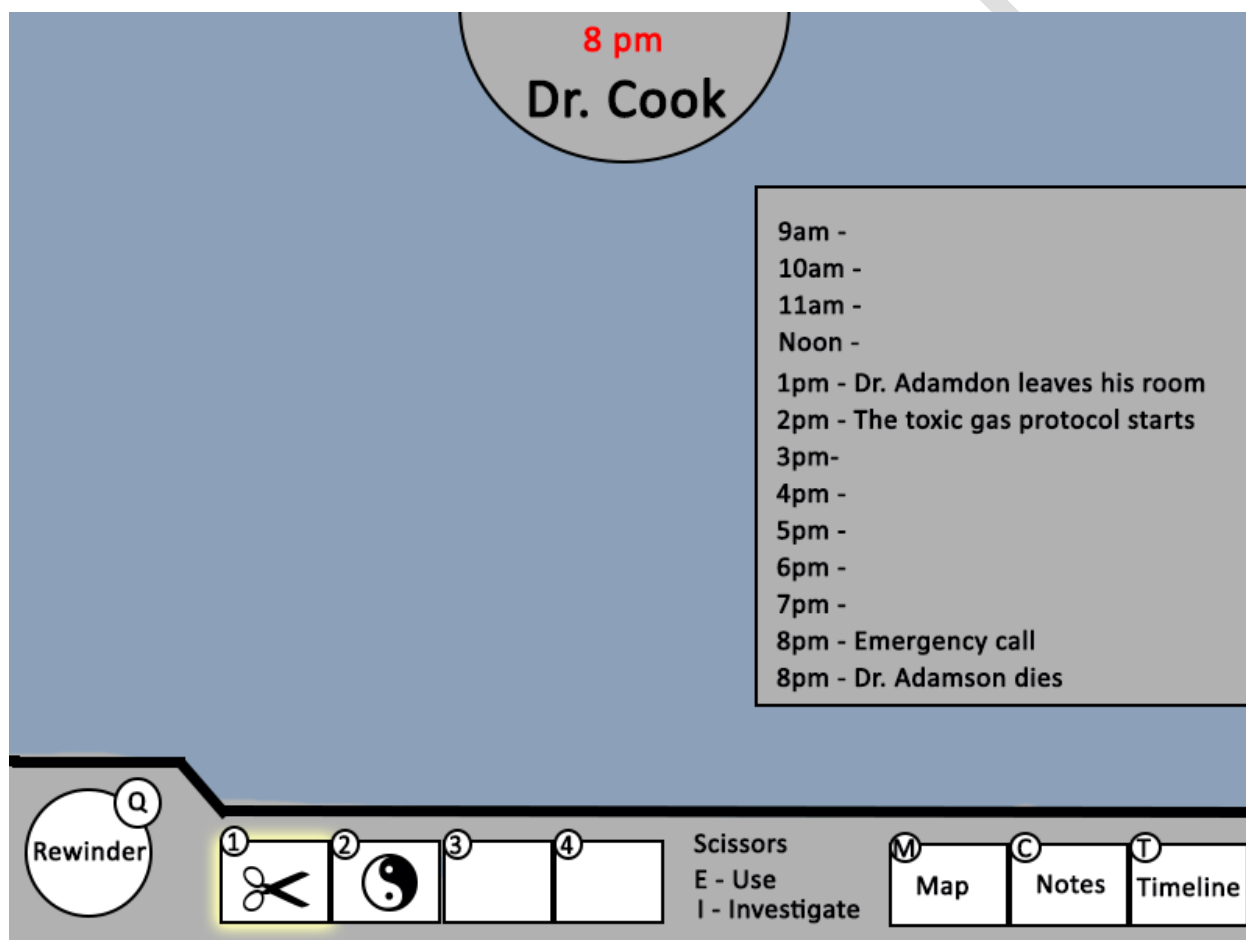


Figure 81: In the initial concept, the timeline was part of the HUD.

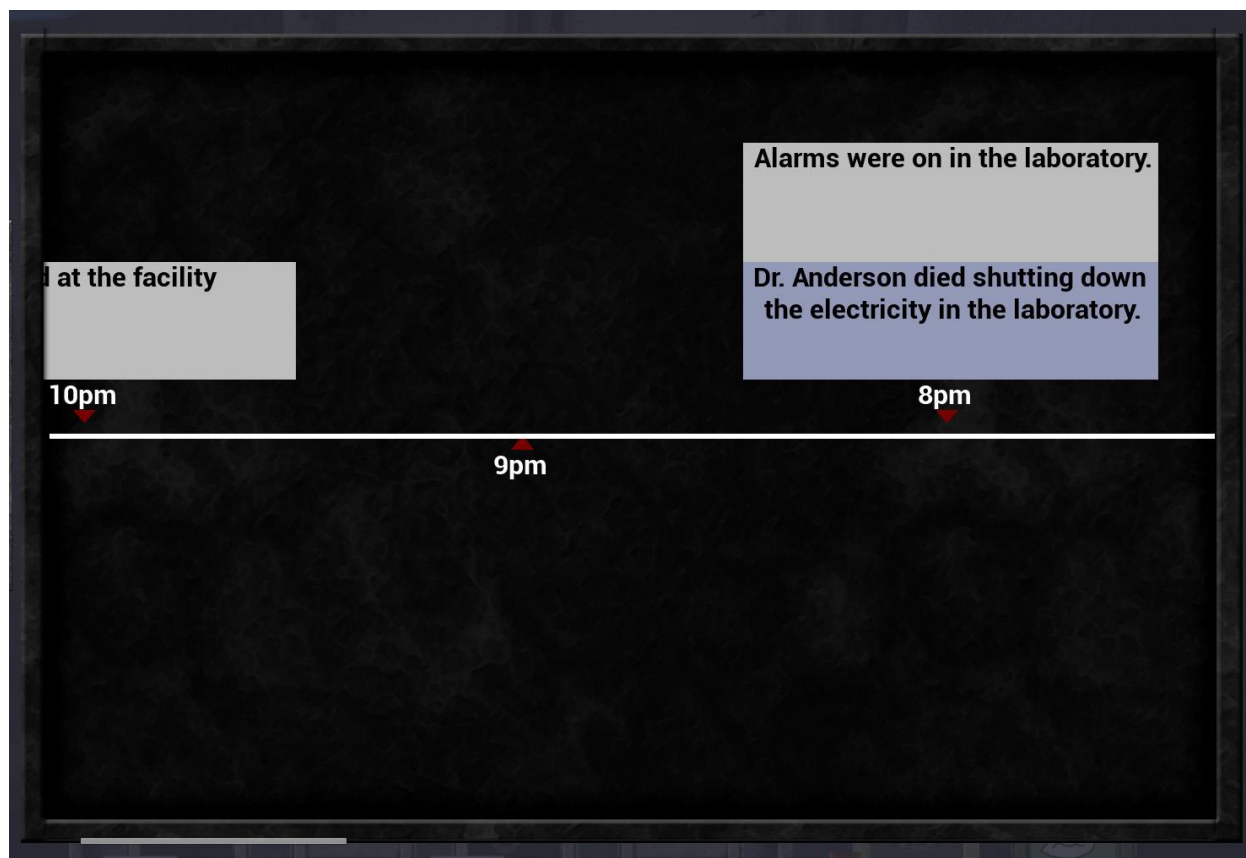


Figure 82: Timeline menu in the final game.

4) Dialog and player thoughts

Because of the nature of point-and-click adventures, the main character constantly makes comments about the environment, or the objects he picks up during the level. These comments appear on the screen in the form of subtitles, which stay on the screen for a few seconds before disappearing.

CONFIDENTIAL

H. Story Summary

1) Backstory

The level takes place in the year 2500, where it has become common for all investigators to use the devices known as “Rewinders”. With the Rewinders, investigators can take DNA samples, and the Rewinder instantly lets them see the past, through the eyes of the person whose DNA they have analyzed.

In the level, players take the role of one of these investigators, who is attending an emergency call coming from a research laboratory. The laboratory, which belongs to the corporation Meta Corp, stopped all communications at noon on December 21st 2500, and sent an emergency signal eight hours later. The players’ goal is to find out what happened in the laboratory, to track the origin of the call, and to rescue any possible survivors that might still be in the laboratory.

2) Levels story

Upon entering the facility, the investigator realizes that something terrible has happened. All lights in the laboratory are off, and there seems to be some kind of toxin in the air. Upon looking into a room, the investigator finds the body of Dr. Anderson, one of the laboratory’s scientists. After making his way through the debris, the investigator manages to reach the scientist body and collect some DNA, which allows him to see the past through the scientist’s eyes. Thanks to his vision, the player discovers that a “Purge” protocol started in the laboratory two hours before the call. For some unknown reason, Dr. Anderson shut down all electrical system in the laboratory right before he died, trapping the other scientist in the facility, Dr. Cook.

Thanks to Dr. Anderson’s key code, the player gets to venture further in the laboratory, where he finds Dr. Cook, who is alive and trapped in the Main Control room. Dr. Cook assures the player that he does not know what triggered the “Purge” protocol, and asks the investigator to free him. In order to do that, the player needs to access Dr. Cook’s room, get his key card, and give it to him, so that Dr. Cook can restore electricity to the facilities from the control room. Although the investigator believes Dr. Cook at first, he soon starts suspecting that the doctor is not telling the truth. As the investigator enters more visions, and sees the day through Dr. Anderson’s eyes, he starts to confirm his suspicions. At the same time, the player finds out more about that experiments that were taking place in the laboratory. In a vision that brings him to midday, the player takes the role of Dr. Anderson as he is incinerating some kind of human-like being. More research brings the investigator to the conclusion that the laboratory was specializing in creating human-like AIs, which the scientists sacrificed if the AIs did not meet certain standards.

The level ends when the player finds the dead body of Dr. Cook, which makes him realize that there is an imposter in the Main Control Room. Using Dr. Cook’s DNA, the investigator is finally able to witness the events in the morning of the tragedy. Seeing the past through Dr. Cook’s eyes, the player discovers that Dr. Cook had developed feelings for Eve, one of the AIs in the laboratory. Knowing that the scientists were about to incinerate her, Eve tricked Dr. Cook into freeing her in the morning of December 20th. After Dr. Cook freed her, she immediately killed him, and proceeded to release toxic gases in the lab, with the goal of killing as many of her captors as possible. However, Dr. Anderson turned off the electricity in the laboratory before the AI could escape, trapping her inside the lab.



Figure 83: Incinerator concept

3) Aftermath

The player finally accesses the Control Room, where Eve is. There, she finds Eve bleeding and almost unconscious. Eve is about to die, and the Investigator asks her about her reasons behind the killing of the scientists. Eve gives the doctor her reasons and ends the conversation wondering about the fairness of her confinement and final fate.

4) Characters

Character	Description
Agent Dale Cooper	The main character. He is a detective and his company has sent him to investigate the events in the laboratory, which led to the emergency call.
Dr. Anderson	One of the scientists in the laboratory, in charge of the engineering work. He considers the AIs to be mere research subjects, and is incapable of seeing them as human. Because of that, he clashes frequently with Dr. Cook's humanitarian views.
Dr. Cook	An expert in bioethics, Dr. Cook makes sure that the experiments in the laboratory meet a minimum ethical standard. After some time in the laboratory, he starts to feel bad for the AI's, and falls in love with one of them known as Eve. His views lead to frequent fights with Dr. Anderson.
Eve	One of the AIs created by Dr. Anderson. She prides herself as being the best AI created in the laboratory. When she discovers that Dr. Anderson is going to incinerate her, she cannot deal with the rejection, and tricks Dr. Cook to free her. She then proceeds to kill all of the scientists and AIs on site by starting the Purge Protocol.

J. Level Summary

1) Campaign

a) Context

Rewind is a level that takes place in the future, where physic detectives can access victim's memories through their DNA. In the level, players take the role of Dale Cooper, who is investigating a mysterious SOS call that came from a secluded research laboratory. Upon arrival, Dale finds the body of one of the lab's scientists, and realizes that a major accident has happened in the facility. His mission then becomes to find out what has happened, and to rescue anyone who might still be inside.

b) Backstory

The goal of the laboratory is to explore the creation of artificially made human-like droids of high intelligence, which can mimic human thought process and that have feelings of their own. In the laboratory, scientists create AIs every month, test them, and destroy them if they do not pass certain mental-health and logic tests. During the process of creating the ultimate AI, Dr. Cook, one of the scientists, starts feeling sympathy for the poor artificial beings. One day the oldest AI, Eve, fails one of the tests and she is subsequently marked for incineration. Eve then quickly becomes aware of her imminent death, and tricks Dr. Cook into freeing her. After hiding Dr. Cook's body inside her capsule, Eve proceeds to kill everyone inside the laboratory by starting an emergency Purge protocol, hence taking revenge on the people who decided she was not a good enough AI.

c) Aftermath

Dale Cooper finds out that the person inside the main laboratory is Eve. He finally gets to the main control room, but it is too late and Eve is already dying. Dale has a conversation with Eve before she dies, where she explains the reasons for her actions. Dale leaves the facility wondering about the legitimacy of the lab's experiments, and about how they drove a smart artificial being into insanity.

2) Objective(s)

- Main goal:
 - Find out the origin of the SOS message
 - Rescue any scientists locked in the facility
 - Player fails if he is dies as a result of the machinery or the environment
- Secondary goal: Access the main laboratory
 - Investigate Dr. Anderson's death
 - Restore electricity to the laboratory and Incinerator
 - Reach the laboratory
- Secondary goal: Investigate the facility's activities
 - Find out what kind of experiments took place in the facility
 - Find out the cause of the damage in the laboratory
 - Find out what happened to Dr. Cook
- Secondary goal: Free the scientist inside the main control room

- Obtain Dr. Cook's code to open the door to the room
- Free the scientist inside the main control room

K. Overview Maps

1) Chronologically ordered events - Player

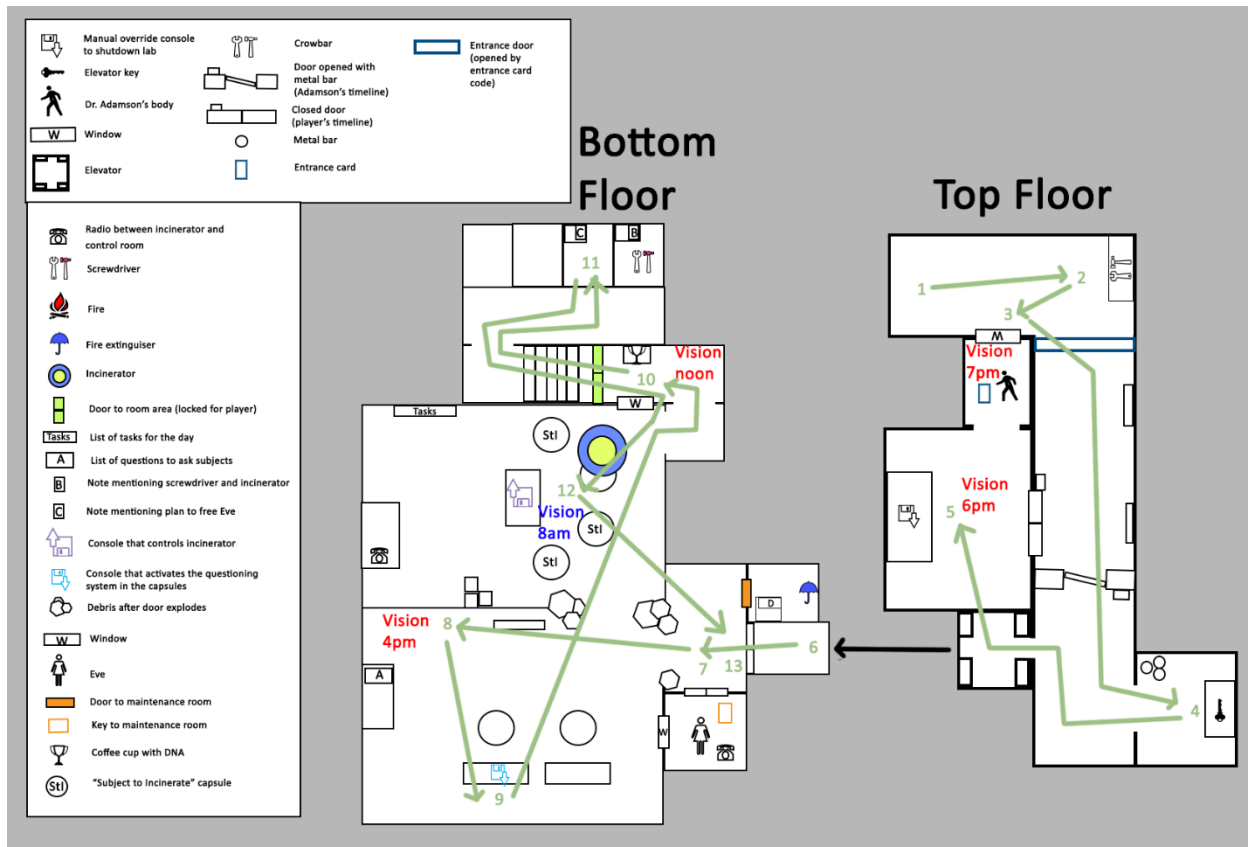


Figure 84: Chronological events for the player

1. The player arrives at the laboratory
2. The player gets a tool that is laying on top of a table
3. The player breaks a window with the tool. He does not fit through the window but he is able to gather Dr. Anderson's DNA. The player enters the first vision, which shows Dr. Anderson dying.
4. The player gets the key to the elevator
5. The player uses the elevator to reach the top window of the control room. The player enters the room through the window and is able to get more DNA from Dr. Anderson. The player enters the second vision, and after coming back, he reactivates the electricity in the laboratory.
6. The player reaches the bottom floor by using the elevator
7. The player talks to Eve, who tells him that he needs to find the code to open the control room. The code should be in the dormitories, in one of the rooms.
8. The player finds more of Dr. Anderson's DNA and enters the third vision, where Dr. Anderson is getting out of the laboratory and trying to reach the top floor.
9. The player reactivates the electricity in the Incinerator room, and enters the room.

10. The player finds DNA in a cup of coffee, and another vision begins. This time, the player sees Dr. Anderson waking up in the morning and starting his incineration routine. Suddenly, the “Purge” protocol alarms go off.
11. The player enters Dr. Cook’s room and finds the key to Eve’s capsule.
12. The player enters Eve’s capsule and sees Dr. Cook’s body. He enters the last vision, where he sees Dr. Cook freeing Eve earlier in the day.
13. The player confronts Eve, who explains her motivation behind the killing. She then dies in front of Dale.

CONFIDENTIAL

2) Chronologically ordered events – Dr. Anderson

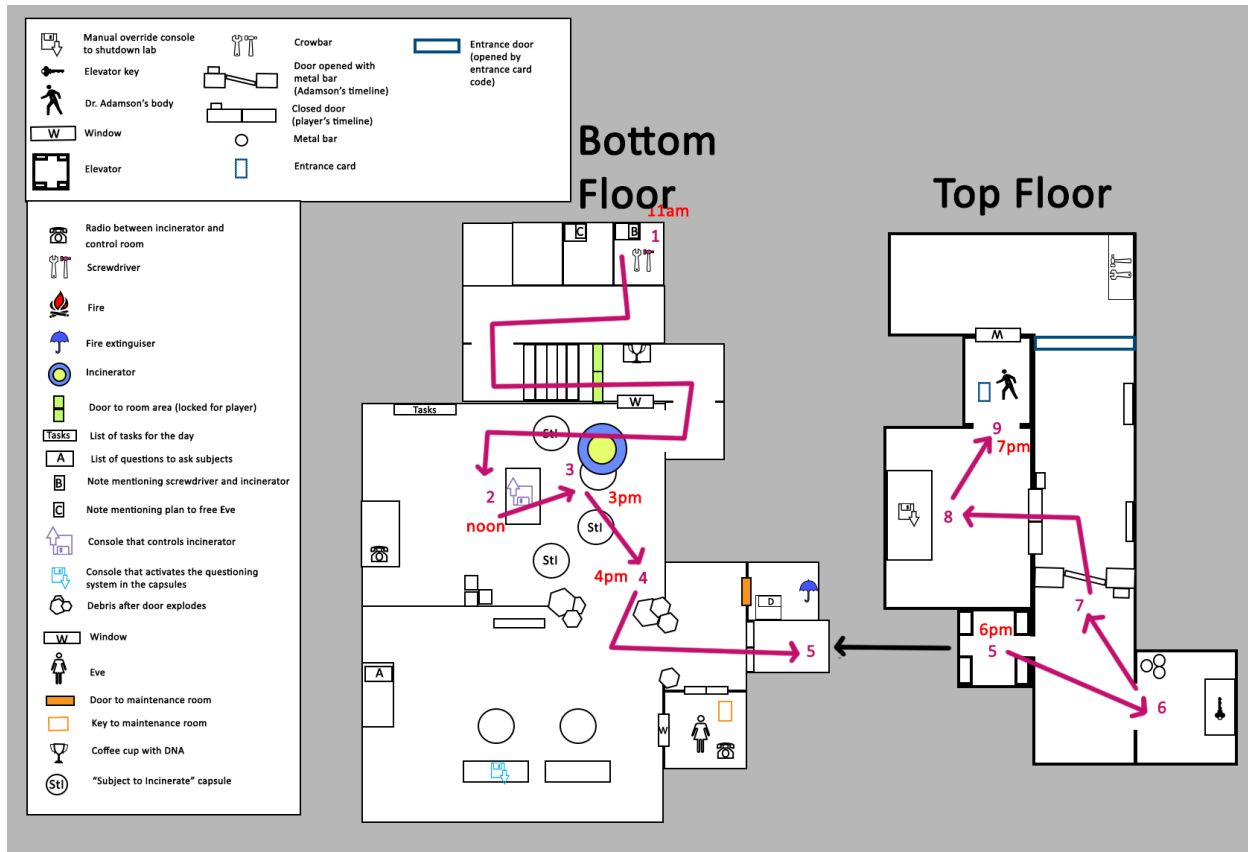


Figure 85: Chronological events for Dr. Anderson

1. Dr. Anderson wakes up
2. Dr. Anderson starts working on the incinerations for the day
3. When Dr. Anderson is about to incinerate Eve, he finds out that she is not in her capsule. Suddenly, the "Purge" protocol alarms go off.
4. Dr. Anderson destroys the Incinerator room's door, by making the Incinerator's laser drop, and then pointing it towards the door.
5. Dr. Anderson reaches the elevator to the top floor
6. Dr. Anderson gets a metal bar from the room
7. Dr. Anderson uses the metal bar to keep the door locked and he reaches the Emergency Control room
8. Dr. Anderson turns off the electricity in the laboratory, the Incinerator, and the Main Control Room.
9. Dr. Anderson dies in the small room close to the Emergency Control Room.

3) Chronologically ordered events – Dr. Cook

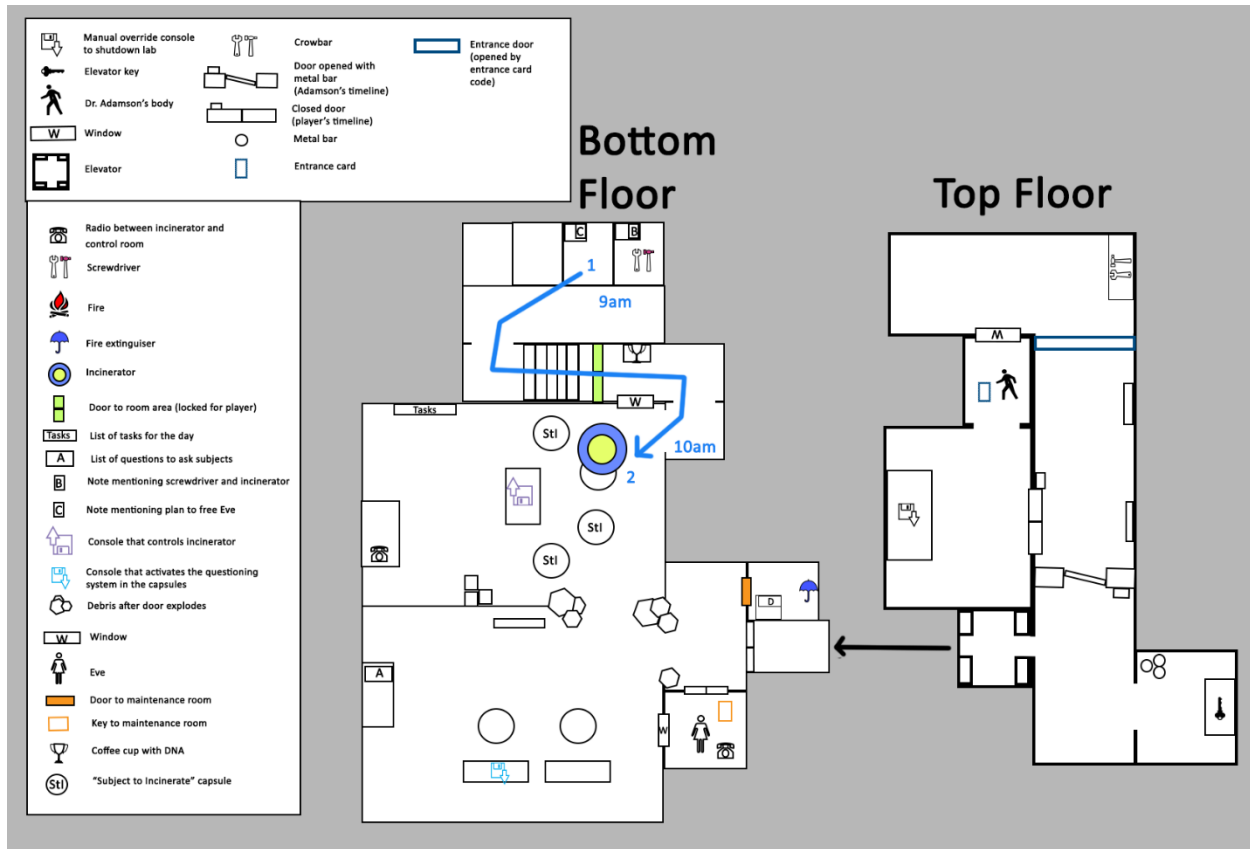


Figure 86: Chronological events for Dr. Cook

1. Dr. Cook wakes up and goes to Eve's pod.
2. Eve convinces him to open the pod, and kills Dr. Cook when he opens the door.

L. Level Details

1) Level Areas

The level has different areas, and in order to access them the player needs key codes, which he gets throughout the level. The player gets some key codes by collecting a key card, or by seeing the code numbers in a vision.

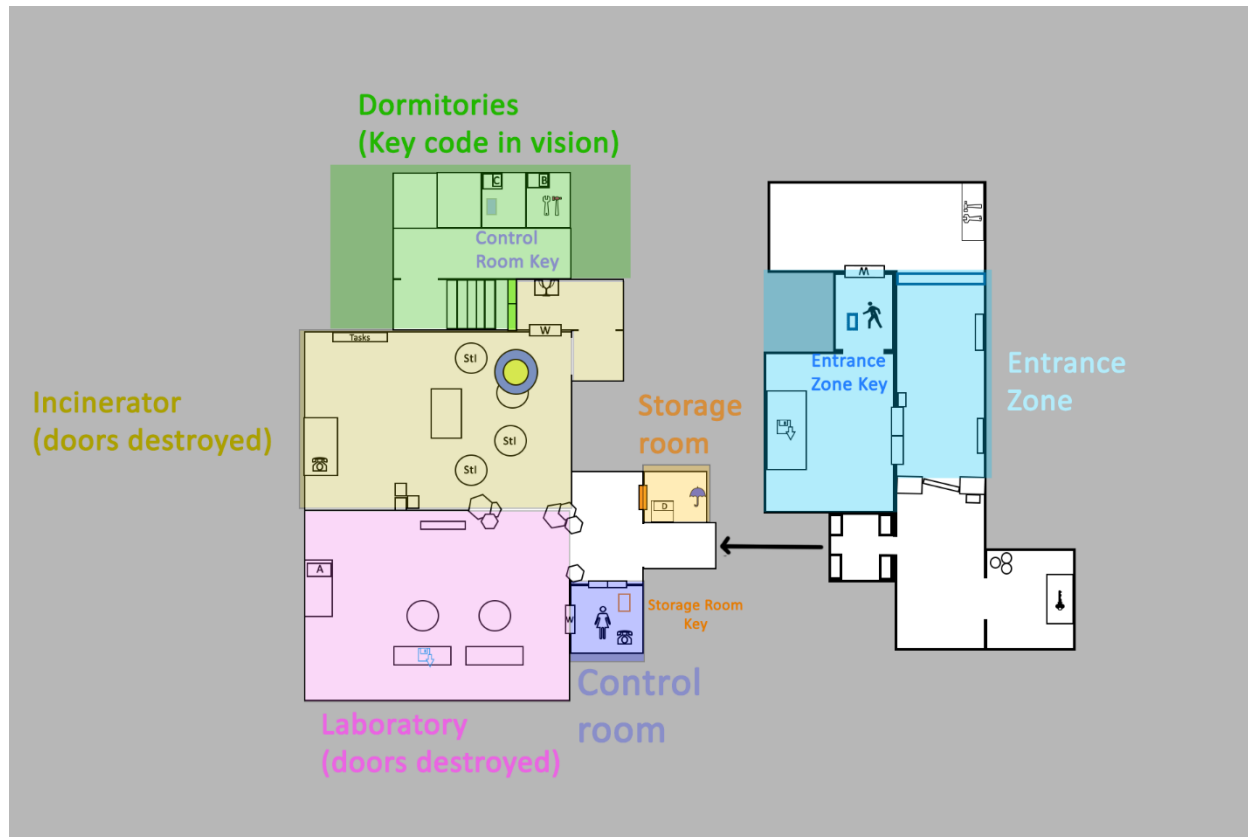


Figure 87: Area breakdown of the level

2) Alarm levels

The laboratory has three different kinds of warnings and alarms, which play through the speakers when something happens in the laboratory:

- Blue alarm: they are warnings, usually due to system malfunctions. Do not require an immediate response.
- Orange alarm: there is potential danger to the people inside the laboratory. Immediate response is required.
- Red alarm: emergency, laboratory evacuation is required.

M. Detailed Walkthrough

1) Area 1: Entrance area

a) First iteration: Investigator

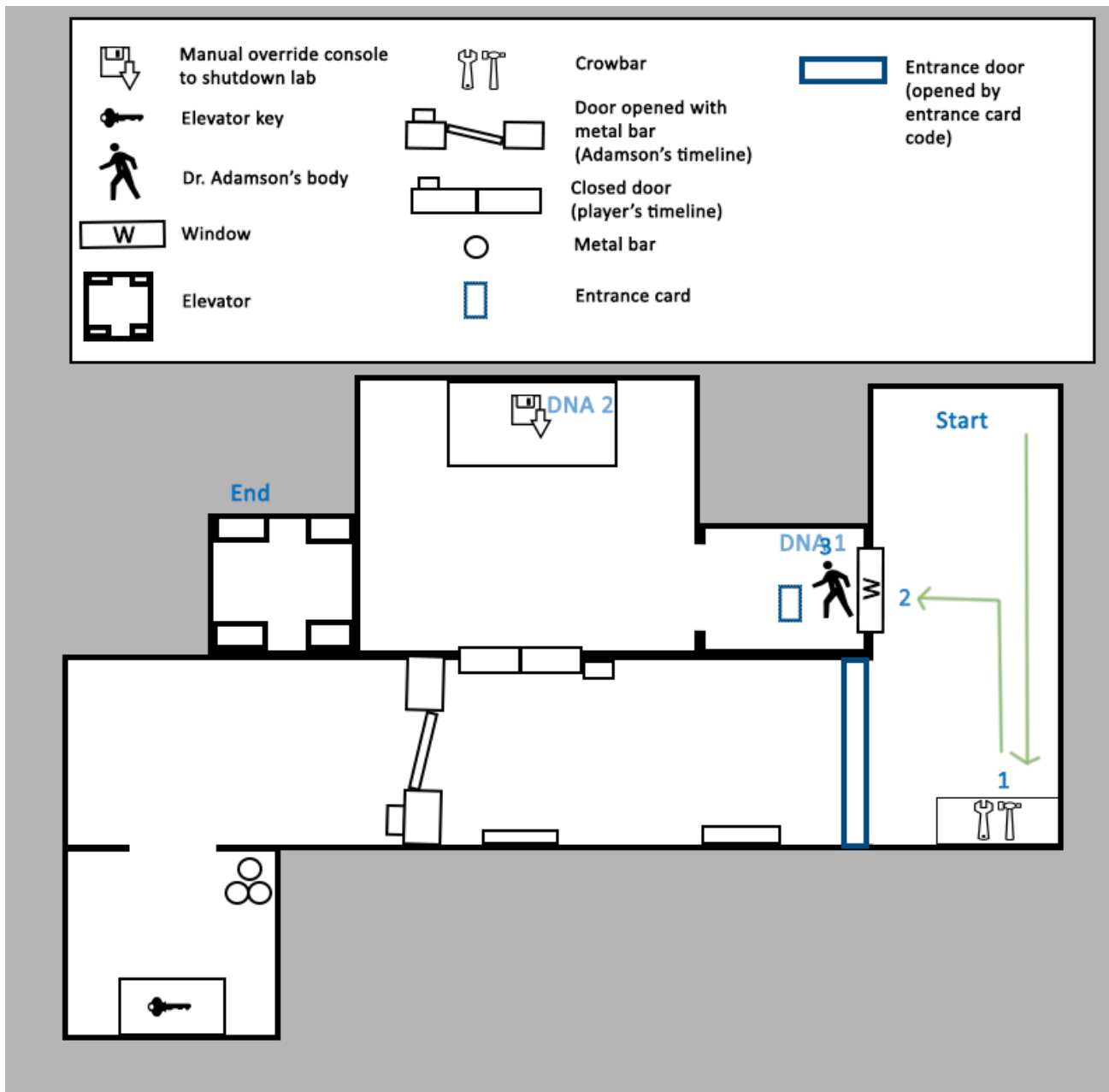


Figure 88: First iteration in Area 1 (Investigator)

- Initial Level condition
 - The player has just arrived at the facility
 - A locked door (blue in image) is blocking the player's path
 - The player can see the dead body of Dr. Anderson through a window.
- Story
 - The player finds out that Dr. Anderson is dead
- Goals
 - Open the entrance door
- New mechanics
 - Using the rewinder for the first time
 - Picking up an object
 - Using an object
 - Using a code seen in a vision
- Gameplay/Story
 1. Players go to a table at the end of the hallway and pick up the crowbar
 2. Players use the crowbar on the window, breaking it
 3. Players have now access to Dr. Anderson's body. They use the rewinder to enter the first vision of the level.

b) *Second iteration: First vision, Dr. Anderson (7pm)*

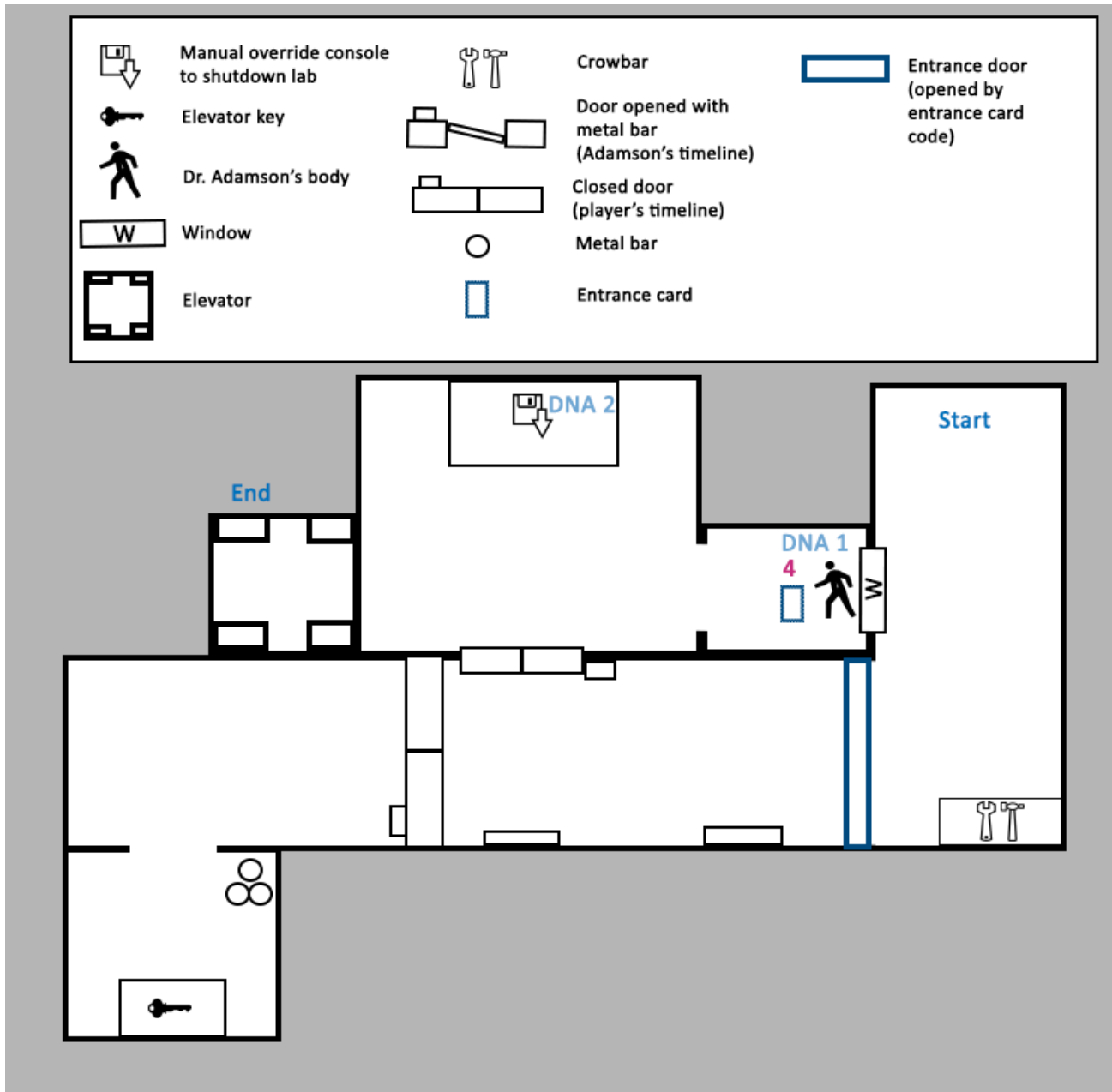


Figure 89: Second iteration in Area 1 (Dr. Anderson 7pm)

- Initial Level condition
 - Alarms in the laboratory warn the player about the high levels of toxicity in the air.
 - In this iteration, players spawn as Dr. Anderson as he is laying on the ground, unable to move. He has managed to start the shutting down sequence, but the toxic gases have entered his body and he is about to die. Due to the toxic gases, he cannot move, so players can only look around.
- Story
 - The player receives sporadic radio transmissions from Eve throughout this iteration. They are simply words of anger (“What have you done?”, “I will get out of here!”, etc.). The radio transmissions never mention the name Eve.
- Goals

- Read the entrance code written on the card that lays by Dr. Anderson. The card contains the code to open the door to the entrance area.
- New mechanics
 - Entering a vision for the first time
 - Using information from a vision in order to progress in the present
 - Using a code for the first time
- Gameplay/Story
 4. Players start the vision as Dr. Anderson, but they cannot move. They look around and see a code card. Upon reading it, they find out that it contains the code to open the entrance area's door. Dr. Anderson dies. As his eyes close, he can see the lights in the laboratory turning off.

c) *Third iteration: Investigator*

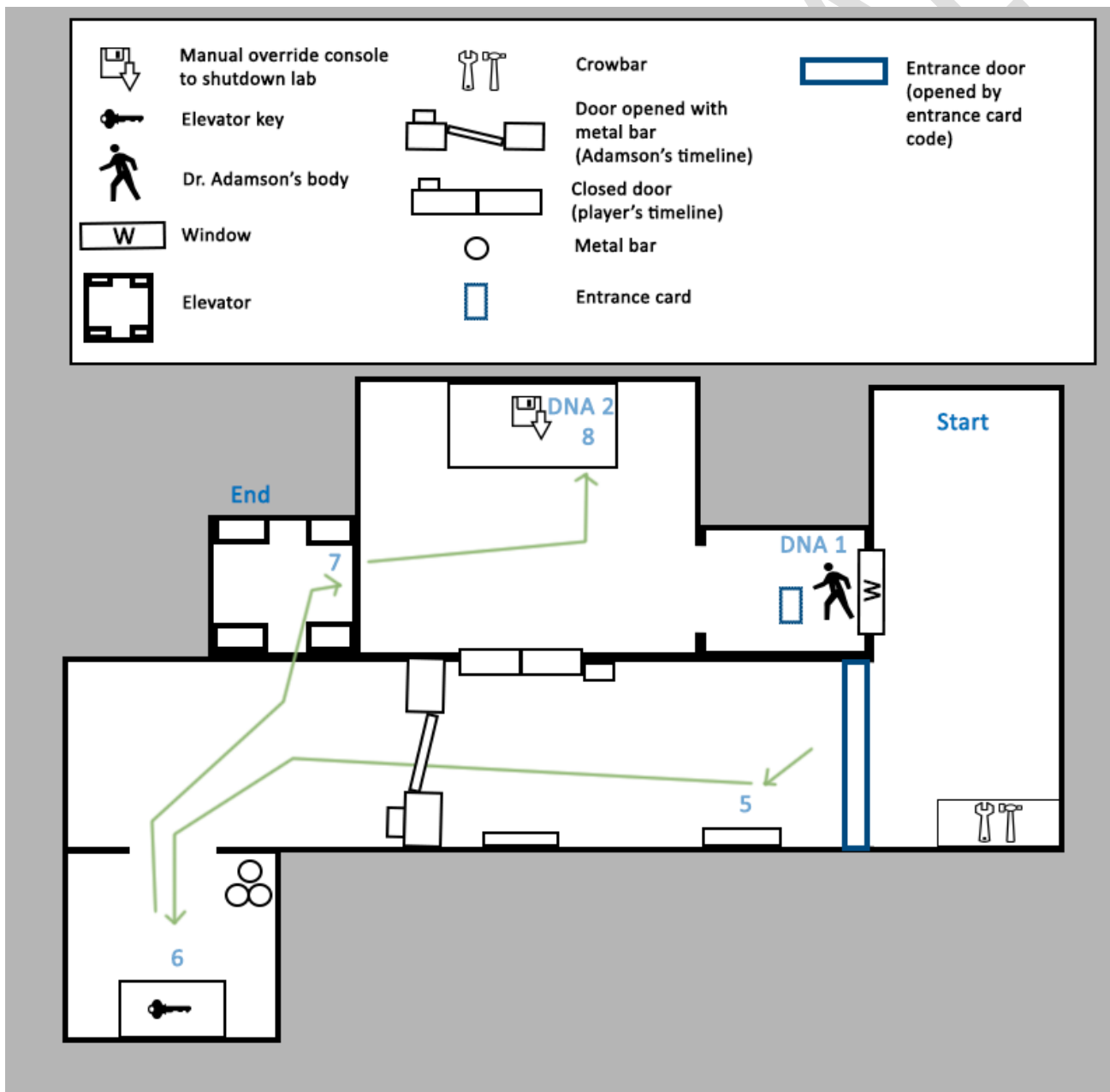


Figure 90: Third iteration in Area 1 (Investigator)

- Initial Level condition
 - Players enter the entrance area. The corridor has an elevator, but it requires a special key to function. When the elevator is working, it can go down one floor or up one floor. The bottom floor contains a door that does not open, because the electricity in the laboratory is down. The elevator gets stuck when it tries to reach the upper floor.
 - Only emergency lighting is working in the facility, illuminating the player's critical path.
- Story
 - Players learn about Dr. Anderson and Dr. Cook's background through some commemorative plaques in the hallway of the entrance area.
 - Players get a hint about the nature of the laboratory's experiments.
- New mechanics
 - Review of previously learned mechanics
- Goals
 - Get in the control room, where Dr. Anderson's body is.
- Gameplay
 5. Players read the two plaques that talk about the two scientists in the laboratory: Dr. Anderson and Dr. Cook. Players can get information regarding the scientist's degrees: neuroscience and computer engineering. The plaques also hints the nature of the laboratory's experiments.
 6. Players go to the storage room. There they can try to open the "Emergency tools" locker. The toolbox requires the name of the person who is going to borrow a tool, for logging purposes. It only accepts the names of the scientists in the laboratory. Players enter either Dr. Anderson or Dr. Adam's name (which they have read in the plaques), and they get the elevator's emergency key.
 7. Players get into the elevator and go up. The elevator malfunctions and stops in front of a window. Players break the window using the crowbar and enter the control room.
 8. Players gather some blood with the Rewinder, and enter the second vision as Dr. Anderson.

d) *Fourth iteration: Dr. Anderson (6pm-7pm)*

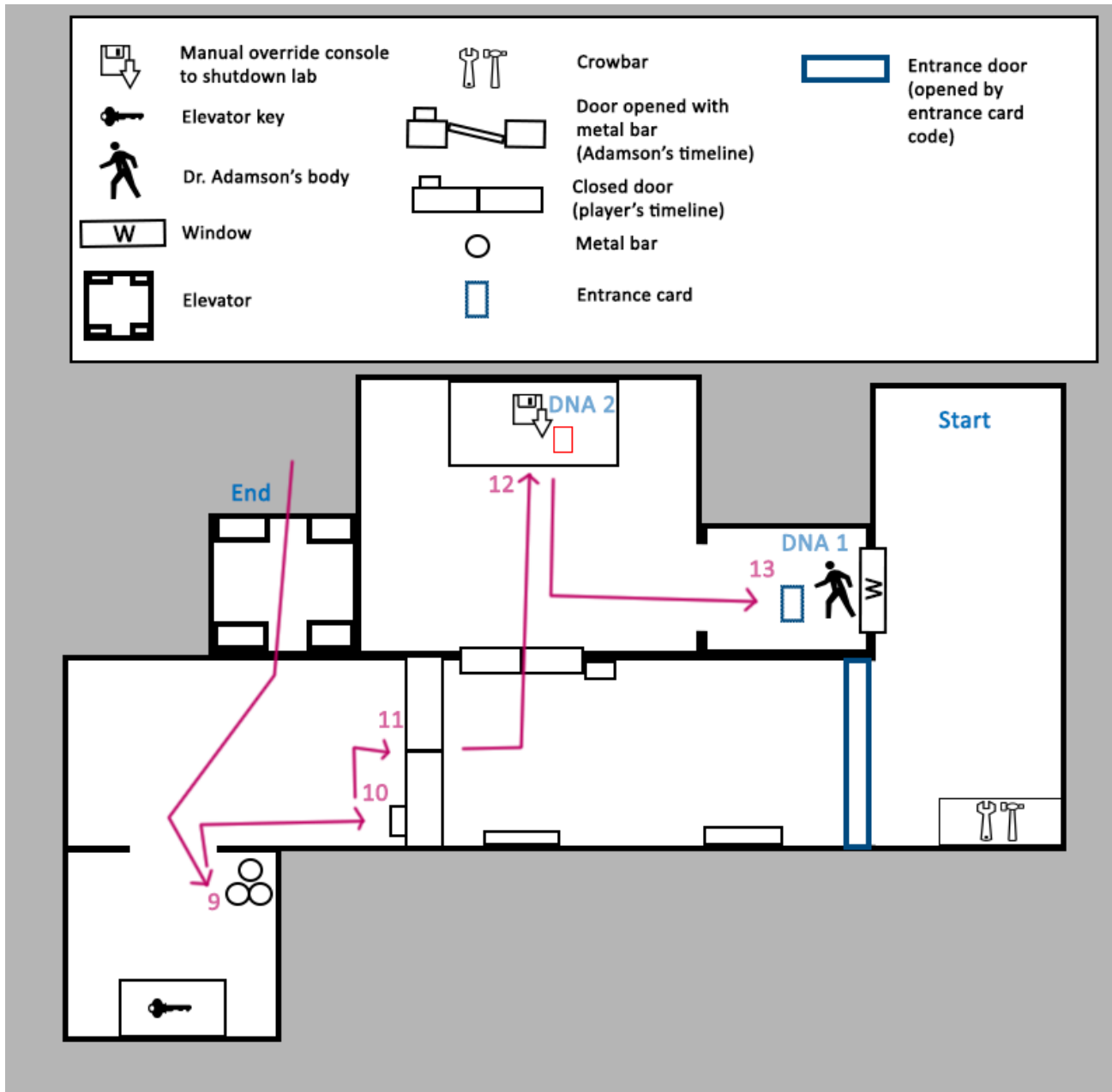


Figure 91: Fourth iteration in Area 1 (Dr. Anderson 6pm-7pm)

- Initial Level Condition
 - Red alarms in the laboratory warn the player about the recent release of toxic gas in the facilities.
 - Players move difficultly due to the toxic gas affecting their senses.
- Story
 - Players wake up as Dr. Anderson, as he is leaving the elevator. Eve has already released the toxic gases and Dr. Anderson is having trouble breathing. He walks slowly.
 - Players receive radio transmission throughout the whole level, coming from Eve.
 - She knows that Dr. Anderson is trying to shut down the lab, and she is aggressively trying to convince him not to do it.

- Goals
 - Shutdown the electricity in the laboratory and trap Eve inside.
 - New mechanics
 - Review of vision's gameplay
- Gameplay
 9. Players start playing as Dr. Benjamin. He is having trouble moving. Players enter the storage room, and grab a metal bar.
 10. Players press the button that opens the door and introduce their code.
 11. The door malfunctions and starts opening and closing. Players put the metal bar between the door panels, to hold it.
 12. Players turn off electricity in the facility by introducing their code in the main console. Since player's don't know the code, Dr. Anderson talks to himself and hints which numbers are the correct ones:
 - a. "My code doesn't start with a 2..."
 - b. "I'm feeling dizzy, was this number really a 3?"
 13. Once they insert the code, players start losing consciousness and the vision ends.

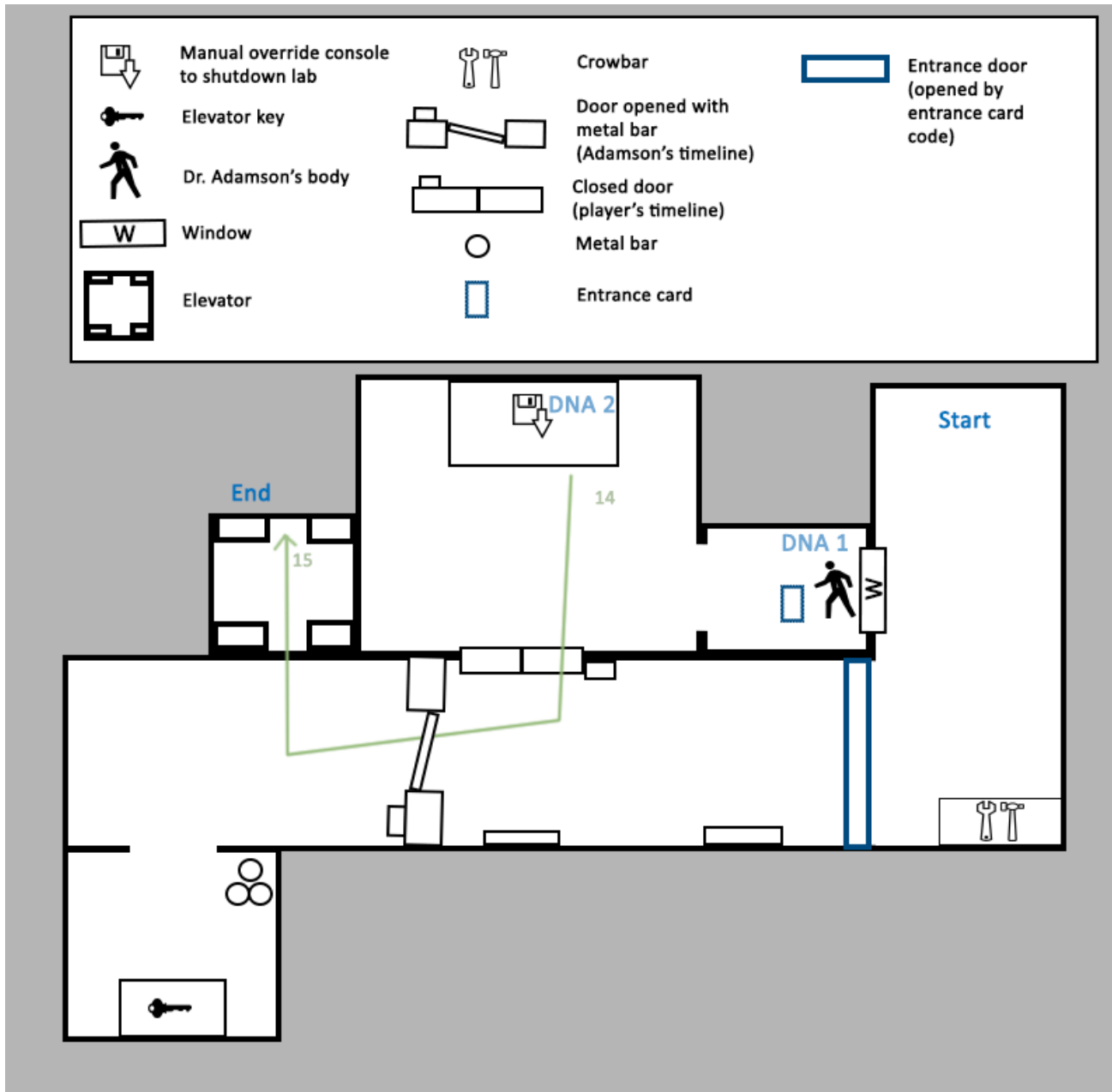
e) *Fifth iteration: Investigator*

Figure 92: Fifth iteration in Area 1 (Investigator)

- Initial Level condition
 - Nothing has changed since iteration 3
- Goals
 - Restore electricity to the laboratory area
- New mechanics
 - No new mechanics
- Gameplay
 - 14. Players use the code that they saw in the vision to restore electricity to the laboratory. If the player tries to restore electricity to the other areas, they receive a warning:

- “Connection to Incinerator lost, please restore electricity from laboratory”
 - “Connection to main control room lost, please use emergency protocol to restore electricity”
15. Players get into the elevator and go to the bottom floor. The electricity is back, so the door opens and they are able to access the laboratory.

CONFIDENTIAL

2) Area 2: The laboratory

a) First iteration: Investigator

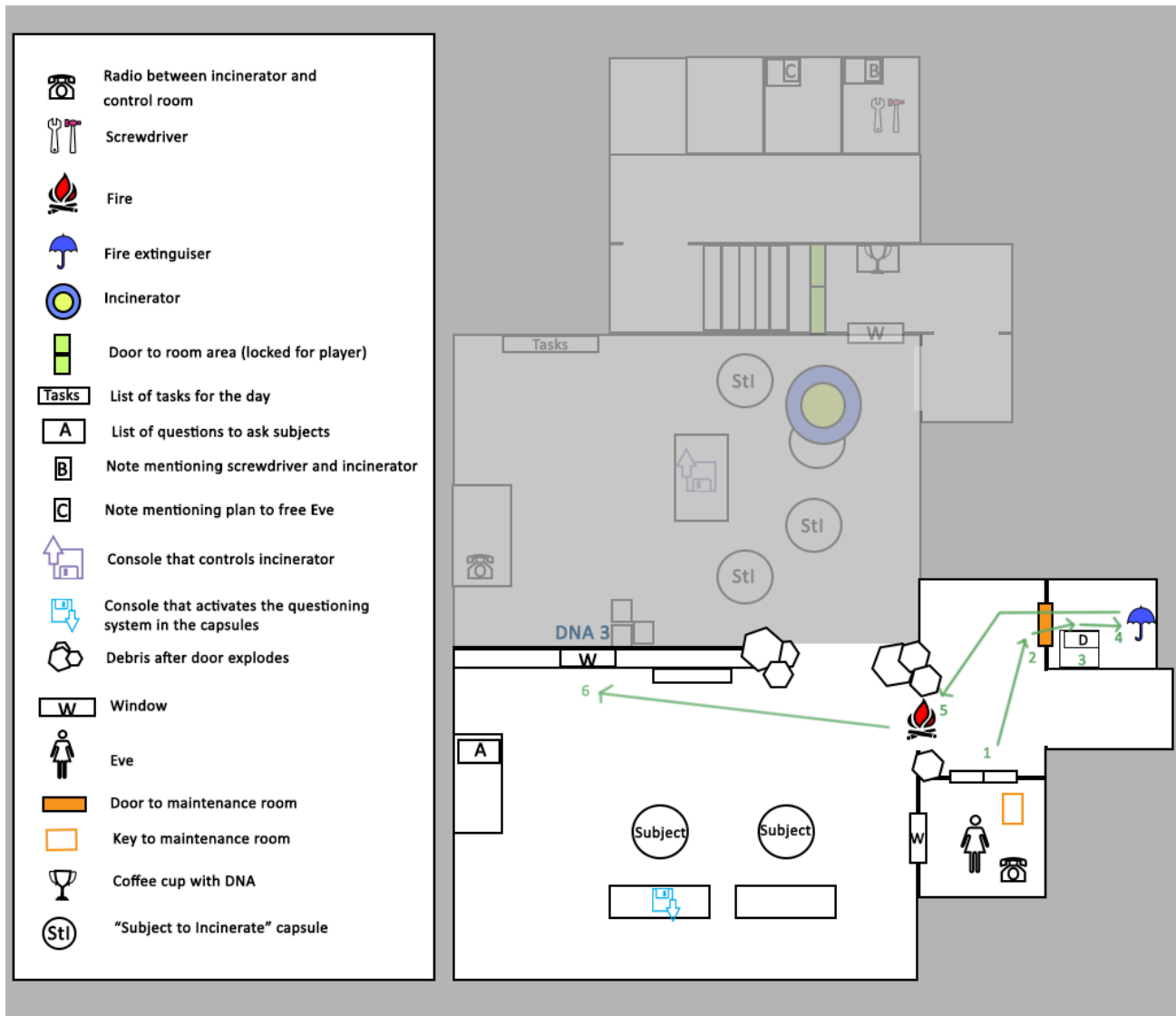


Figure 93: First iteration in Area 2 (Investigator)

- Initial Level condition
 - The player reaches the entrance corridor to the laboratory, but fire blocks his path. The fire originated from the explosion created by Dr. Anderson hours before.
 - Players see the Main Control room to the side of the corridor is the main control room. Eve is inside this room, and she calls the player over.
 - Area 3 (the incinerator) is in darkness, so the player can enter it but he cannot perform any actions while in it.
 - The subject capsule on the left (see map) is missing a battery
- Goals
 - Talk to Eve in front of the Main Control room
 - Keep exploring and find out where Dr. Cook is
 - Reach the scientists' dormitories

- Story
 - **Subject capsules:** The laboratory contains the “Subject capsules”. These are the capsules where the scientists keep the AIs and do research on them. Every month, the AIs have to pass certain tests. If they fail the tests, the scientists send the AIs to the incineration room.
 - The player cannot see what’s inside the capsules, they are opaque.
 - One of the capsules is the one where Eve used to be.
 - The other capsule contains the body of Dr. Cook, but the player cannot see it yet.
 - **Meet Eve:** the player meets Eve for the first time. She is inside the control room and introduces herself as Dr. Cook. The player cannot see inside the room so he falls for Eve’s lie. Eve then tells the player that she was the one that made the distress call, and that she got trapped inside the control room when the electricity went down.
 - **Receive a new goal:** Eve asks the player to find Dr. Cook’s key card. With Dr. Cook and Dr. Anderson’s key cards, Eve can restore electricity to the whole facility. Eve tells the player that her card (Dr. Cook’s card) is in the dormitories.
 - **AI tests:** During this iteration, the player finds notes that refer to the tests that the scientists were carrying out on the AIs. Most of these tests are questions that the scientists asked the AIs, to decide if the AIs should go to the incinerator or not.
 - **Experiments with live beings:** The player also sees the pods where the scientists kept the AIs. This is the iteration where players start to figure out that the scientists were experimenting with some kind of living beings.
- New mechanics
 - Put down fire
- Gameplay
 1. A fire is blocking the player’s path, so the only thing he can do is to talk to Eve. The player talks to Eve, who asks him to find the code to restore electricity in the control room. He gives him the key to the maintenance room.
 2. The player opens the maintenance room, where he sees a fire extinguisher.
 3. The player sees a note close to the fire extinguisher. The note mentions the Incineration room, and reminds the scientists that it is important to be alert because of possible fires. This is the first hint towards the subject incineration protocol.
 4. The player gets the fire extinguisher.
 5. The player puts down the fire using the fire extinguisher.
 6. The player enters the laboratory. In the window between the laboratory and the incinerator, the player finds some of Dr. Anderson’s DNA in the form of sweat. By using the Rewinder, the player enters another vision that shows him Dr. Anderson’s past.

b) Second iteration: Dr. Anderson 5pm-6pm

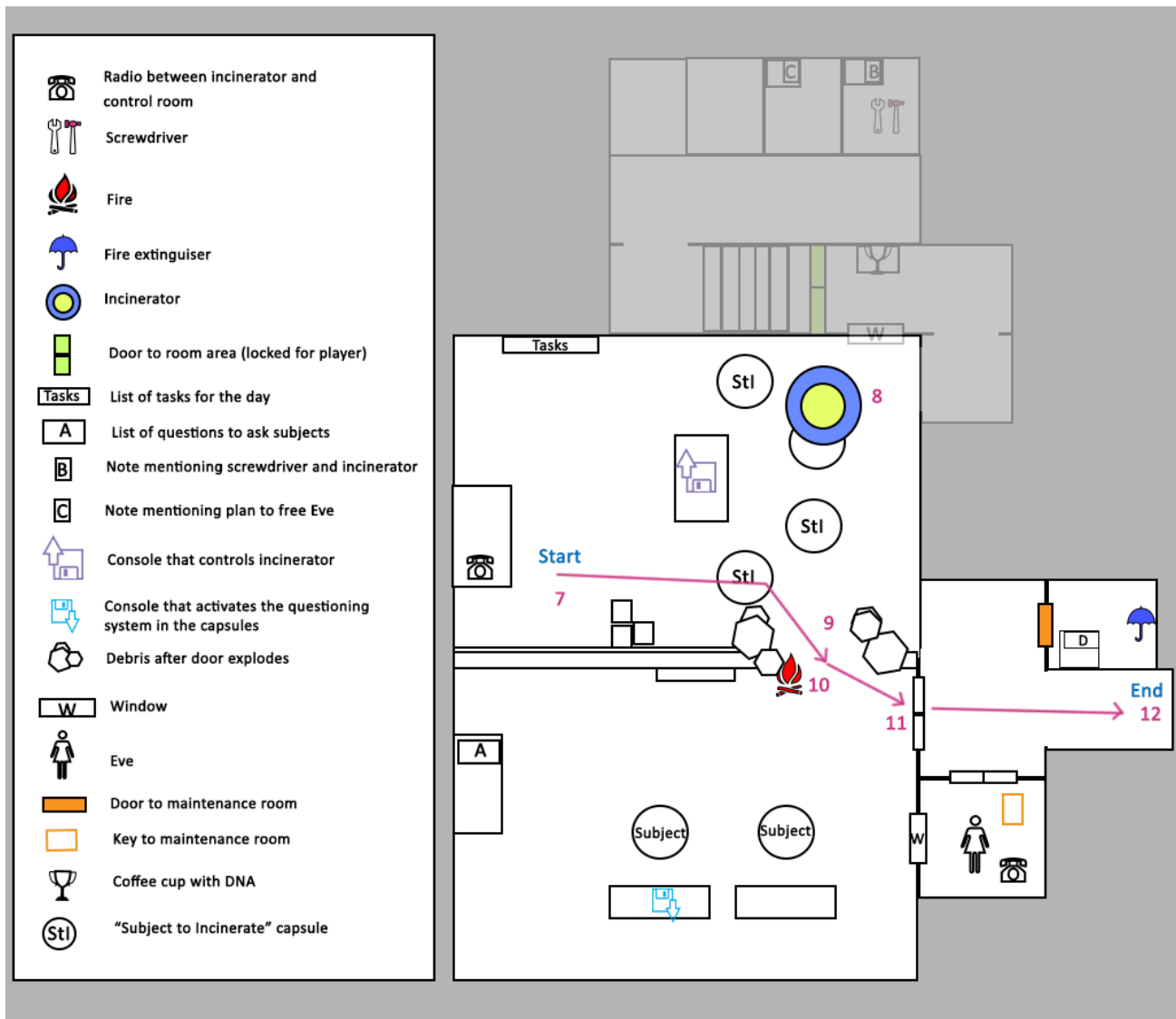


Figure 94: Second iteration in Area 2 (Dr. Anderson 5pm-6pm)

- Initial Level condition
 - The vision starts right before Dr. Anderson blows up the door using the Incinerator
 - The door's explosion creates a fire that hurts Dr. Anderson as he is exiting. This is the fire that the player just put down.
 - Orange alarm at the beginning of the level: "ALARM: Purge protocol activated. Release of toxic gases imminent"
 - After Dr. Anderson exits the laboratory doors, the toxic gases are released, and the alarm changes to a red alarm "ALARM: Toxic gases released. Evacuate immediately."
- Goals
 - Dr. Anderson is trying to get to the "Emergency Control Room" on the top floor to shut down the electricity in the laboratory.
- Story

- **Backstory:** Dr. Anderson starts incinerating some of the subjects, but halfway through his work, the orange “Purge” alarms go off. Dr. Anderson, worrying for his life, destroys the Incineration room’s door using the Incinerator.
- **Start:** The vision starts as Dr. Anderson hides behind a pile of boxes, while waiting for the incinerator to destroy the door.
- New mechanics
 - No new mechanics
- Gameplay
 7. The player starts hiding behind a group of boxes.
 8. The incinerator gun has dropped from the ceiling, and it is now laying on the floor, pointing at the door. As soon as players get leave their hiding place, the incinerator activates and destroys the Incineration Room’s door.
 9. The door explodes and debris covers the room. The explosion results in a fire around the door area.
 10. Players go through the door, and fire burns them in the process. Their walking becomes more difficult after this point.
 11. Players use the laboratory’s key code to open the laboratory’s door and get out. Like in Area 1, Dr. Anderson hints the number to the players. Players have to use this number after the vision.
 12. Players get to the elevator and the vision ends.

c) *Third iteration: Investigator*

Figure 95: Third iteration in Area 2 (Investigator)

- Initial Level condition
 - Same as iteration one.
- Goals
 - The player is trying to access the dormitories. In order to get to the dormitories, he needs to go through the incinerator room, but the room is in darkness.
 - The player needs to restore electricity to the Incineration Room, so that he can then access the dormitories.
- Story
 - Players find some of the AIs' tests for the first time. Players start to wonder if the "subjects" are indeed intelligent beings, since the scientists were using personality and logic tests on them.
- Gameplay
 - 13. The player wakes up after completing the previous vision.
 - 14. The players finds a note on the billboard:

- “Due to safety reasons, manual activation of the incineration room has been disabled. The Incineration Room will be activated automatically if an urgent incineration is required.”
15. The player searches in the desk's drawers. There, he finds one of the tests that the scientists usually use on the AIs. The scientists use these tests to select the AIs that need to go to the Incinerator, because of personality/logic malfunctions. Along with the tests' questions, the document indicates the adequate action to take if the scientists receives a certain answer:
- Question 1: Does the subject look happy/sad/confused?
 - Question 2: Is the subject responsive to exterior stimuli, including light and sound?
 - Question 3: On a scale from one to ten, what is the subject's reaction when exposed to constant auditory stimulation for more than one hour?
 -
 - Result evaluation 1: If all questions received a negative answer, imminent incineration required. Emergency protocol activates.
 - Evaluation result 2: If 50 questions received a negative answer, extensive tests needs to be scheduled for the following day.
 - ...
16. The player uses the laboratory's code to activate the testing console. He learned the code in the previous vision, when he was playing as Dr. Anderson.
17. The player starts a test in one of the capsules. The capsule then prompts the Subject Evaluation questions to the player, and the player has to introduce the AIs responses to those questions in the system, as if he was a scientist evaluating an AI. If the player manages to obtain an evaluation of “Imminent Incineration Required”, then the Incineration Room automatically activates. The player has to look at the document that he found in 15, and answer the questions so that he obtains an “Imminent Incineration Required” result.
18. The player introduces the right answers and the Incineration Room's electricity comes back. The player accesses the incineration room, where he then finds new DNA belonging to Dr. Anderson. A new vision then starts.

3) Area 3: The Incineration room and the dormitories

a) First iteration: Dr. Anderson

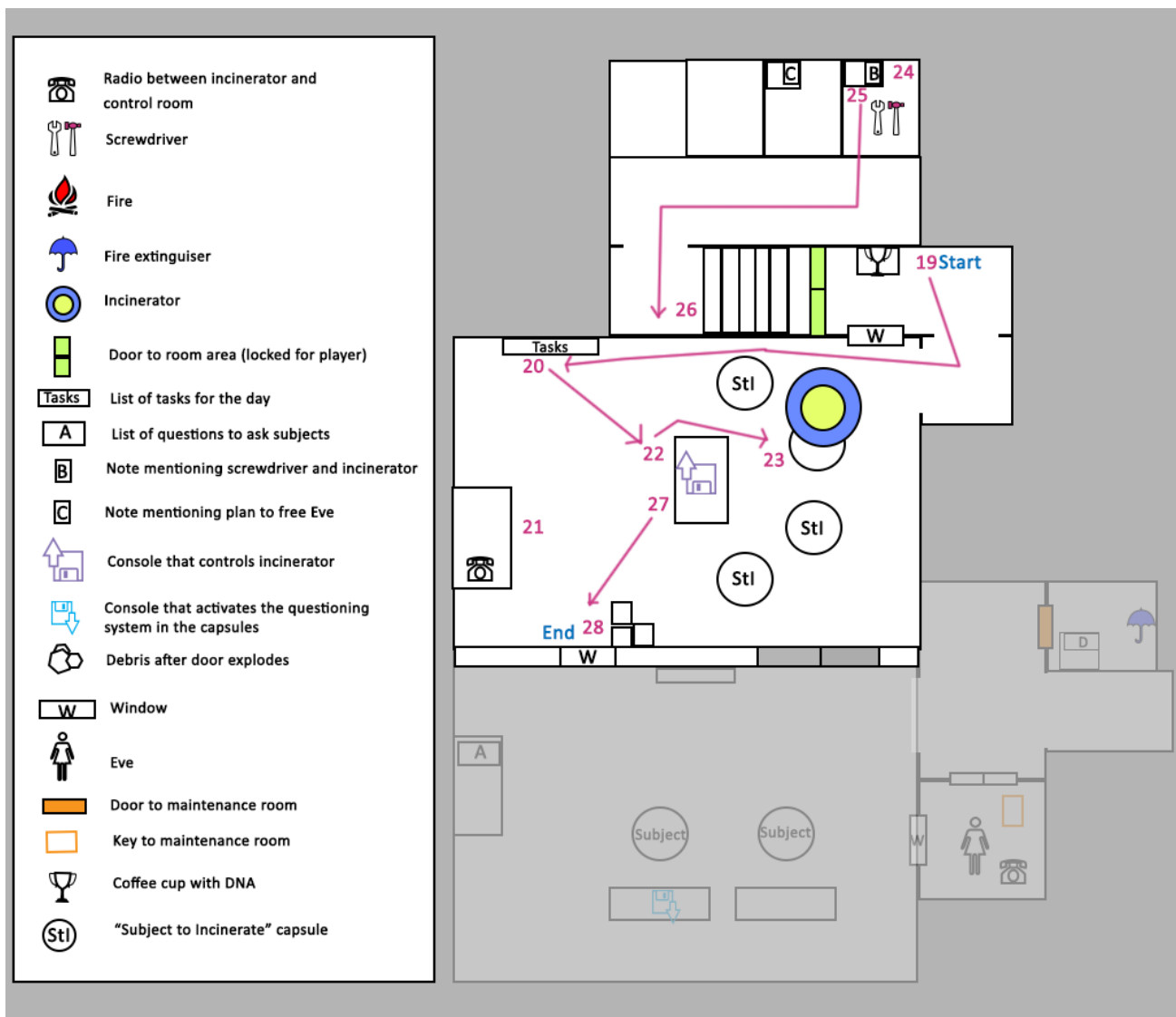


Figure 96: First iteration in Area 3 (Dr. Anderson 3pm-6pm)

- Initial Level condition
 - This iteration takes place in the morning, before Eve starts the "Purge" protocol.
 - The door between the Incinerator room and the Laboratory is intact.
 - A warning message can be heard through the speakers in the laboratory. It is not an emergency but just a warning: "WARNING: purge protocol evaluation started."
 - Halfway through the iteration, the "Purge Protocol" starts, and the laboratory's orange alarms go off.
- Goals
 - Goal before "Purge Protocol" starts: complete Dr. Anderson's tasks for the day (incinerations)
 - Goal after "Purge Protocol" starts: destroy door in order to reach the top floor and shutdown electricity in the laboratory, preventing Eve from escaping.
- Story

- The players take the role of Dr. Anderson at the beginning of the day. The blue alarm (a warning, no imminent danger) is already on. When he wakes up, Eve has already killed Dr. Cook, and has started her plan to kill the scientists. Dr. Anderson thinks that the blue alarm is just a system malfunction, and starts carrying out his tasks for the day.
- When it is time to incinerate Eve's capsule, he realizes it is empty, and then the orange alarm (imminent danger) goes off. Dr. Anderson realizes that there is something wrong, and tries to get out of the Incinerator in order to shut down the laboratory.
- Gameplay
 19. The player starts and hears the warning ("Purge protocol assessment started"). The player talks to himself and disregards the warning as a common malfunction.
 20. The player goes on to check the tasks of the day. In the list, there are three pending incinerations:
 - Eduard 1
 - Eduard IV
 - Eve
 21. The player can use the radio to communicate with the control room and ask about the warning. He does not receive any answer but still ignores the warning.
 22. The player starts incinerating AIs. The player incinerates two of the capsules, but the system gives him a "No subject in capsule" error when trying to incinerate Eve's capsule.
 23. The player looks in the capsule and realizes Eve is not there. The orange alarm goes off and the player starts looking for a way out.
 24. The player goes to Dr. Anderson's room, where he finds a screwdriver.
 25. The player also finds a note that talks about how the Incinerator Gun keeps falling to the ground. The note mentions it was necessary to tighten the screws using the screwdriver, because when they are loose, the Incinerator gun is prone to falling to the ground.
 26. The player brings the Incinerator Gun closer to the top floor. He then goes to the top floor, and from there he can reach the Incinerator Gun. The player uses the screwdriver to loosen the bolts and let the gun drop to the floor.
 27. The player activates the Incinerator Gun, which is pointing to the door.
 28. The player hides behind the boxes and waits until the Incinerator goes off and destroys the door.

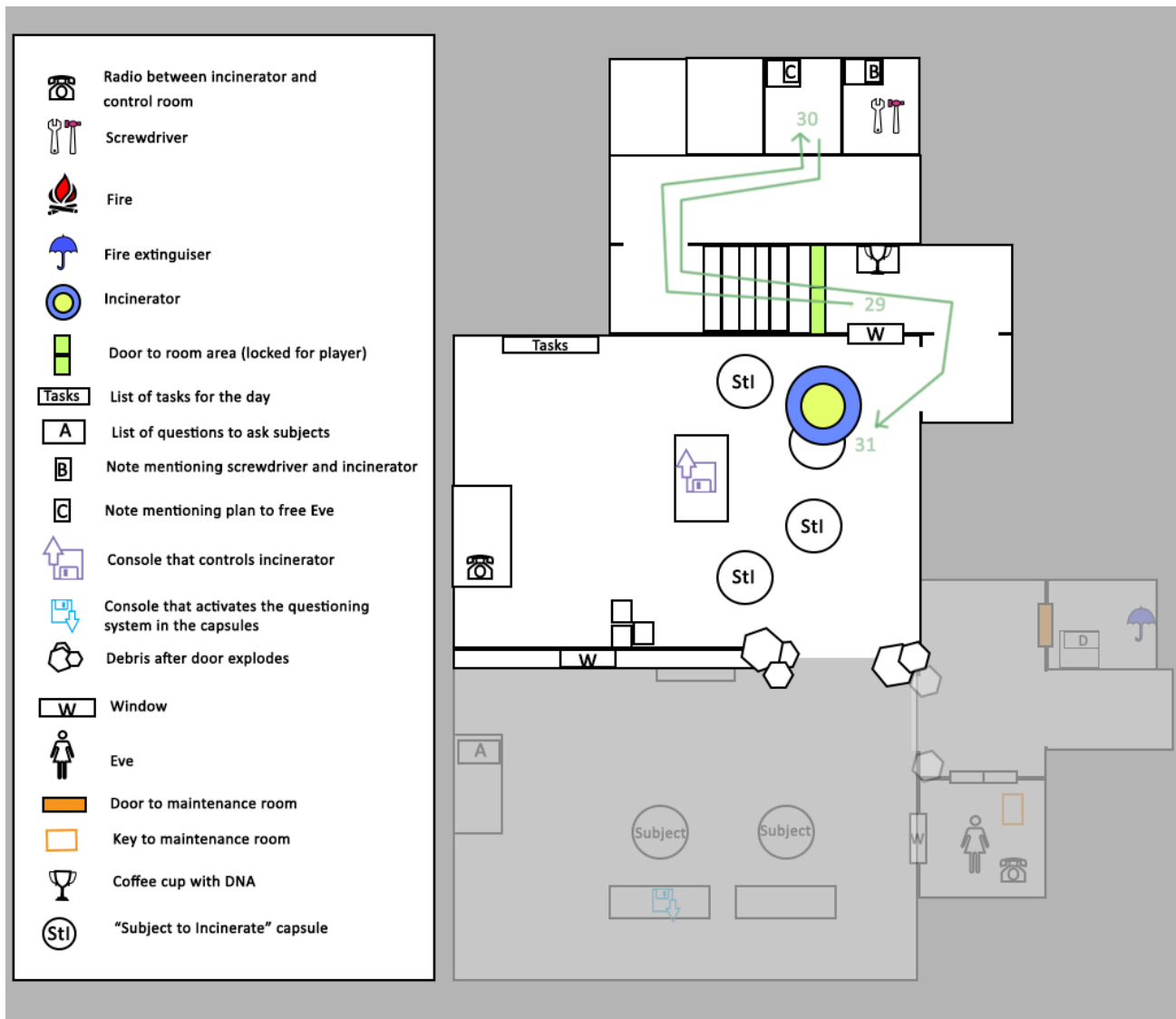
b) *Second iteration: Investigator*

Figure 97: Second iteration in Area 3 (Investigator)

- Initial Level condition
 - Same as last player iteration
- Goals
 - Getting the card for the Control Room in the dormitories
- Story
 - The player finally has the code to enter the dormitories.
 - When he enters Adam's room, he does not find the Control Room card there, but he finds the key that opens Eve's capsule.
 - When the player opens Eve's capsule, he finds Dr. Cook's dead body, and enters the last vision of the game.
- Gameplay
 - 29. The player reaches Dr. Cook's room and finds a note. In the note, Dr. Cook talks about his desire to free Eve before they incinerate her on December 20th. Attached to the note, there is the key to Eve's capsule.

30. The player opens Eve's capsule and finds Dr. Cook's body inside. He soon realizes that the person inside the control room is not Dr. Cook, but Eve. If the player tries to talk Eve then, he receives no answer. Eve has been losing blood and she is almost unconscious.
31. The player uses Dr. Cook's DNA to enter the last vision of the level.

CONFIDENTIAL

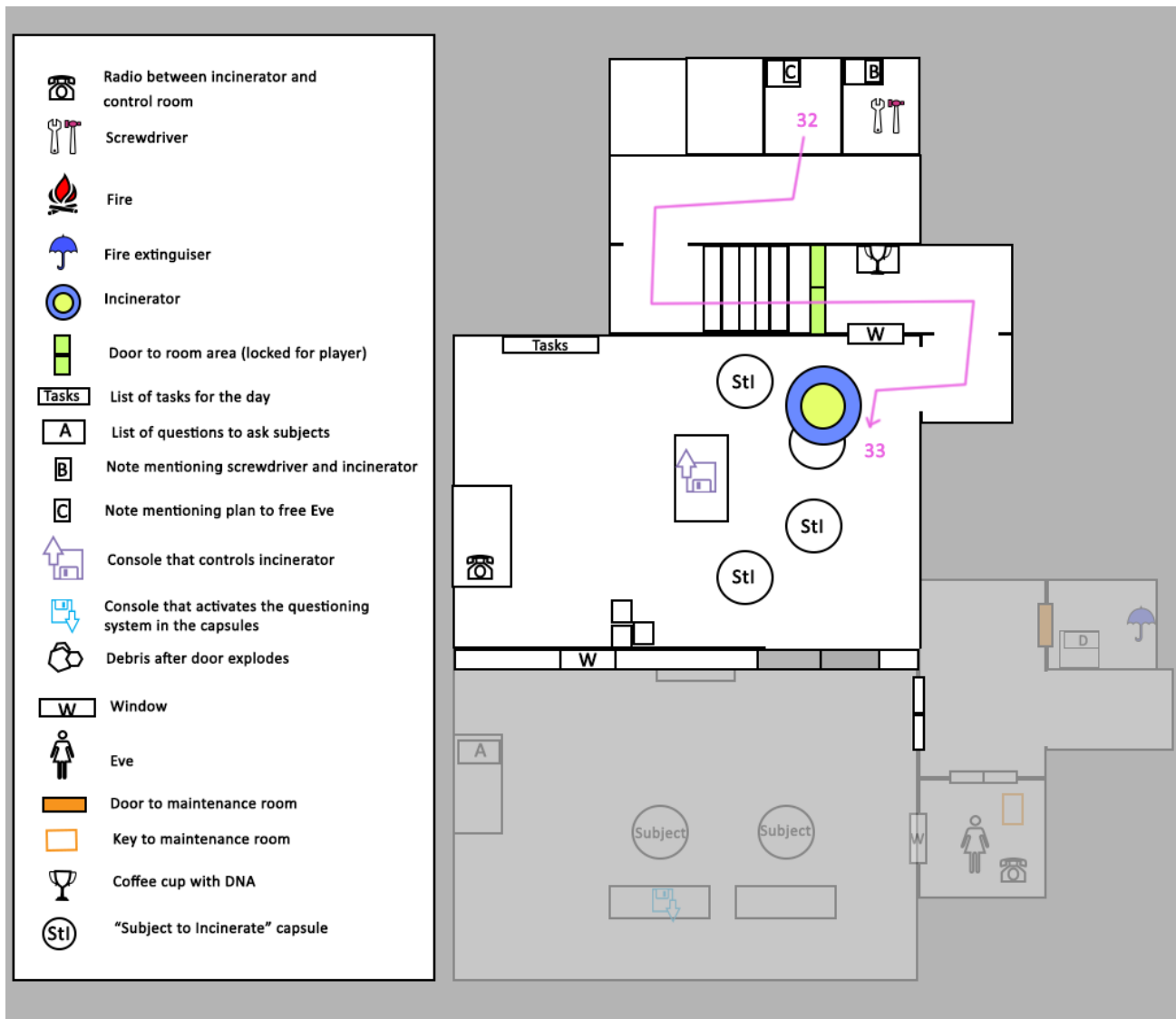
c) *Third iteration: Dr. Cook*

Figure 98: Third iteration in Area 3 (Dr. Cook 9am-10am)

- Initial Level condition
 - The laboratory is completely functional.
 - It is the early morning, before any of the events of the level happened.
 - The player plays as Dr. Cook, as he wakes up.
- Goals
 - Free Eve
- Story
 - The iteration takes place at the very beginning of the day, when Dr. Cook wakes up and decides to visit Eve one last time to say goodbye.
- Gameplay
 - 32. Dr. Cook wakes up in his room, and puts the Control Room key card in his inventory. He also Talks to himself:
 - “I should go say goodbye. I can’t believe I will never see her again”

- On his way to the capsule, the player can see more of Dr. Cook's thoughts: "Is it really necessary? I wonder how I would feel if I was one of them, and I knew what was going to happen to me". "She despises us...don't those feelings prove that she is as human as we are?"
33. Dr. Cook reaches Eve's capsule. For the first time, Eve is kind to him, and asks him to please let her go. Dr. Cook, filling guilty, decides to free her. As he is opening Eve's capsule, the screen fades to black and the player hears the sound of blood splattering. The player returns to the present.

CONFIDENTIAL

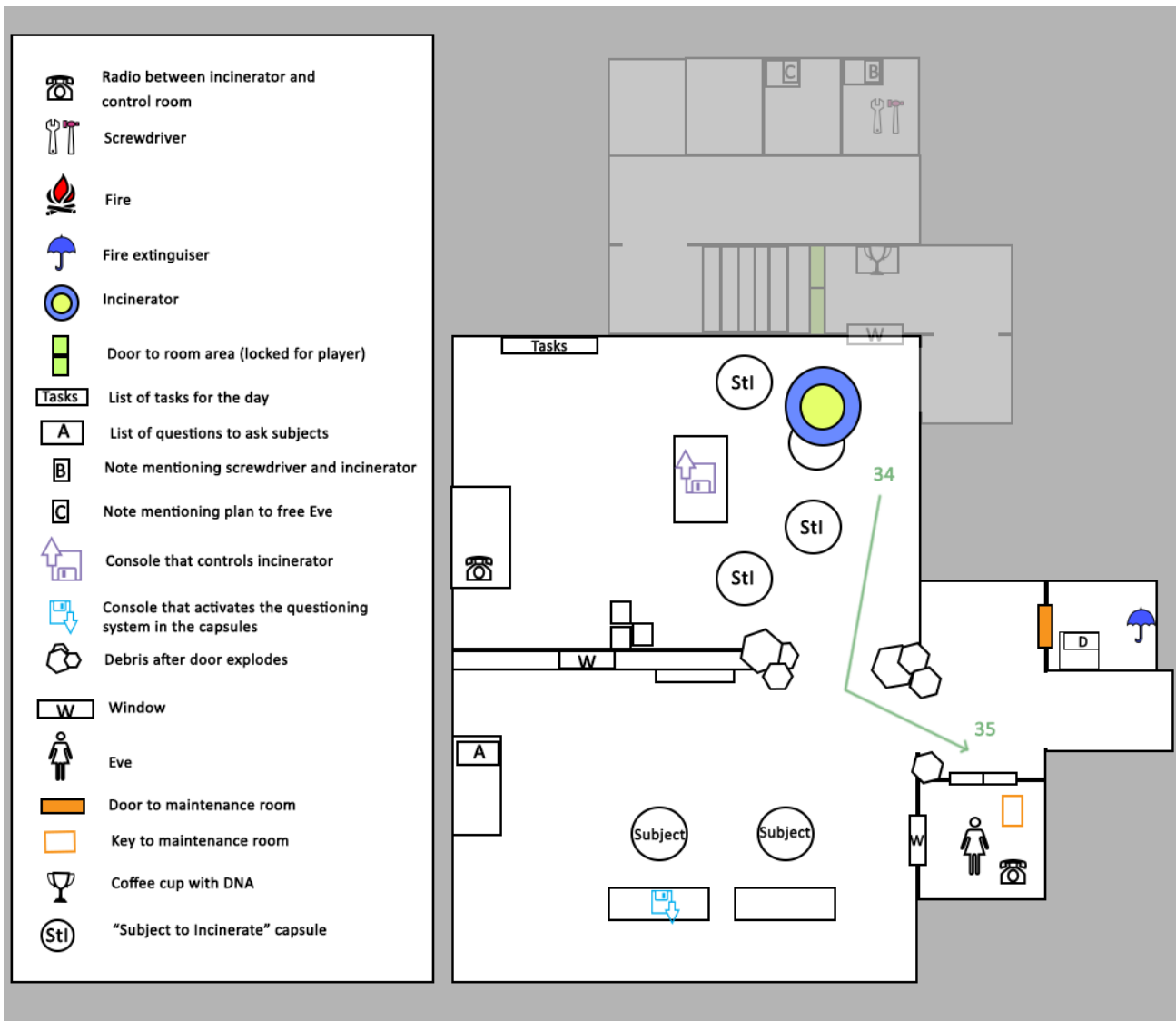
d) *Fourth iteration: Investigator*

Figure 99: Fourth iteration in Area 3 (Investigator)

- Initial Level condition
 - Same as previous player iteration
- Goals
 - Confront Eve
- Story
 - The player finally has the code to the Control Room, and enters it ready to confront Eve. When he enters, he sees she is almost dead. The investigator and Eve have a conversation about Eve's intentions, the origin of his hate, and the fairness of her death and vengeance. The level ends with a fade to black, where text explains that Eve died, and the Investigator wrote down a report of the tragic events that happened in the laboratory.
- Gameplay
 - 34. Players go back to controlling Agent Cooper.
 - 35. The player accesses the control room using the code that he saw in the previous iteration.

VI. REFERENCES

- http://vignette2.wikia.nocookie.net/half-life/images/c/c3/Depot_pods.jpg/revision/latest?cb=20090114004954&path-prefix=en
- <https://m1.behance.net/rendition/modules/24558673/disp/7619acb2e6465cb3bfec92f394cf4518.jpg>
- <http://i.ytimg.com/vi/O4-QYCKh6TM/maxresdefault.jpg>

CONFIDENTIAL