Design Document (Team - Draft)

3D Escape Room Video Game Challenge

Sean Brandon, David McVittie, Robert Avery, Matt Wood

CS 461-400 Fall 2023

Oregon State University

Abstract

Escape rooms have become a staple of interactive entertainment, challenging users to find clues and solve puzzles under time pressure with a blend of storytelling, spatial reasoning, problem solving, and teamwork. Our group hopes to capture this spirit in the digital realm through a realistic simulation of an escape room in video game form. In this document, the planned design for this game is outlined across four functional areas: Sound, Level Design, Scripting, and Player Mechanics. Together, these functional areas compose a cohesive and immersive experience bringing the thrill and challenge of escape rooms to a computer screen.

Table of Contents

Revisions

ROW #	REVISION #	REVISION DATE	REVISION DESCRIPTION	REVISION TRACKING NOTES
1	v1.0	11/9/2023	Initial Specification	

Overview

This document covers the initial design for an immersive escape room video game. The scope of the document includes Phase I development and targets the Minimum Viable Product (MVP) as outlined in the Requirements. The MVP is considered to be a version that will allow the development team to make definitive design decisions for story, mechanics and content. Future development phases will undergo similar requirements gathering and design processes.

The purpose of this document is to outline the steps necessary to develop Phase I of the game. Game developers will use this outline as an example of phase definitions and workflow. Future phases will contain more detailed information for specifics for the game and will determine the overall presentation and content of the game.

The intended audience for this document is primarily the development team and department leads. Secondary stakeholders include our mentor and others.

Definitions

The following terms are defined for the project:

- **DAW —** A Digital Audio Workstation, an application designed to create and edit audio and music. Virtually all new music is mixed and edited in a DAW.
- **Layering —** A technique for dynamic sound design in interactive media using multiple sound elements played simultaneously to create a “richer” (more varied and context-sensitive) audio experience.
    - **Layering sound effects —** When layering sound effects, different sounds are played simultaneously to create variety. Layered sound effects are typically simply randomized; if they reflect in-game info it is at most at the level of position in-game, remaining ammo on a weapon, etc.
    - **Layering music —** When layering music, different tracks created to sync up with each other (usually just exported track-by-track from a DAW) are played simultaneously, with the volume of each track varying depending on game state. For example, in a racing game, an intense-sounding “layer” might fade in and out in accordance with the car's speed; or in our case, more complexity can be faded in each time the player solves a puzzle within a room. Layering music can enhance immersion even more than layering sound effects.
- **Effects —** Modifiers of audio such as reverb and low-pass filters. These can also enhance a player's sense of immersion: sound may reverberate large rooms and be low-pass filtered (i.e. muffled) if heard through a wall.

- **Spatialization —** The general process of creating the illusion of a 3D “soundstage”, where sounds seem to originate from different locations, using a limited number of audio outputs/speakers.
    - **Localization —** The psychoacoustic phenomenon by which the human brain can determine the direction from which a sound originates, based on subtle cues such as inter-ear delay and differences in filtering by the outer ear.
    - **Stereo Panning —** This is the most basic form of spatialization, where sounds are distributed between two speakers or headphones to give the impression of directionality from left to right. However, it is also one of the weakest.
    - **Reverb —** A type of effect in which sounds echo. Tweaking the characteristics of this echo can lead reverb to simulate different types of environment. For example, there should be little reverb in a flat empty plain, whereas in a cave there should be a lot. Reverb depends not only on the size of the room, but the room's shape and one's position in it: one can shout at a faraway wall or a cliff face and have it come back perfectly legible after a second or two, but this will not happen with rougher, jagged surfaces at the same distance.
    - **HRTFs —** Head-Related Transfer Functions, a type of effect which filters incoming audio according to the impulse response of an idealized listener's body. (A simple example: sounds to the left of someone will reach the left ear before the right. A complex example: sounds above will reflect off the outer ear ever-so-slightly differently than sounds below) HRTFs simulate how in-game sounds would have been filtered and reflected by a player’s ears, head, and body before reaching the eardrum. This filtering occurs in a manner that depends on the direction of the incoming sound, providing some of the strongest cues the brain uses to localize a sound. HRTFs characterize these filters parameterized by angle, encoding how sound from any point in space should be altered before it is

perceived as coming from that point when heard over headphones. By filtering sounds with in-game locations through an HRTF provided the angle of the sound relative to the player, the brain is “tricked” into localizing the sound to its in-game location.

- **Heads Up Display (HUD) —** Method of displaying information visually to the player as a part of the game’s user interface. Often includes a map, health, selected powers, etc.
- **Ray Tracing —** Rendering technique in computer graphics that simulates the behavior of light by tracing the path of rays as they interact with virtual objects, producing realistic lighting, shadows, and reflections in three-dimensional scenes.

Introduction

Escape rooms have become a staple of interactive entertainment, challenging users to find clues and solve puzzles under time pressure with a blend of storytelling, spatial reasoning, problem solving, and teamwork. The software in development is an attempt to capture this spirit in the digital realm through a realistic simulation of an escape room in video game form.The Escape Room Challenge software has a variety of different requirements that need to be addressed such that their inner workings will function cohesively together and present an exceptional product. These requirements have been categorized into four functional areas that each team member will handle. These four areas are;

- Sound Design
- Level Design
- Scripting
- Player Mechanics

Problem

Escape rooms have seen a rise in popularity within the public sphere, as a result it can be observed that this rise of popularity has extended into the digital sphere as well. This has created a market for escape room simulation games software. As such there is a demand for this kind of software that has been extended to the development team.

Proposed Solution

System Architecture

The system architecture required for a video game developed with Unreal Engine 5 involves a multi-tiered approach that encompasses both the client and server sides. On the client side, the architecture should support high-end graphics rendering, taking advantage of Unreal Engine 5's advanced features such as Nanite virtualized geometry and Lumen global illumination. This requires modern GPUs and sufficient RAM to handle complex scenes seamlessly.

On the server side, a scalable and efficient architecture is needed to handle multiplayer functionalities, including player matchmaking, game state synchronization, and network communication. Cloud-based solutions, such as AWS or Azure, can be leveraged to provide the necessary infrastructure for handling the computational demands of large-scale multiplayer experiences.

Functional Area Details

In order to fully specify the requirements for an immersive and engaging game, our design specifications are divided into four functional areas: sound design, level design, scripting, and player mechanics.

Sound Design

Design Concerns

The most obvious question in this area of the game is what specific sounds to use in-game. However, this list will be difficult to compile until after other work such as level design is done. (Of course, sound and level design can be done in parallel once there is a “buffer” of designed, but sound-less, levels.) As we don't have such a specification, what is left to design is essentially the “meta-level”:

- How and/or from where will we acquire in-game sound effects and music?
- What game engine features should we leverage to play and spatialize sound?
- What unique features can make the game's soundscape stand out?

Sources

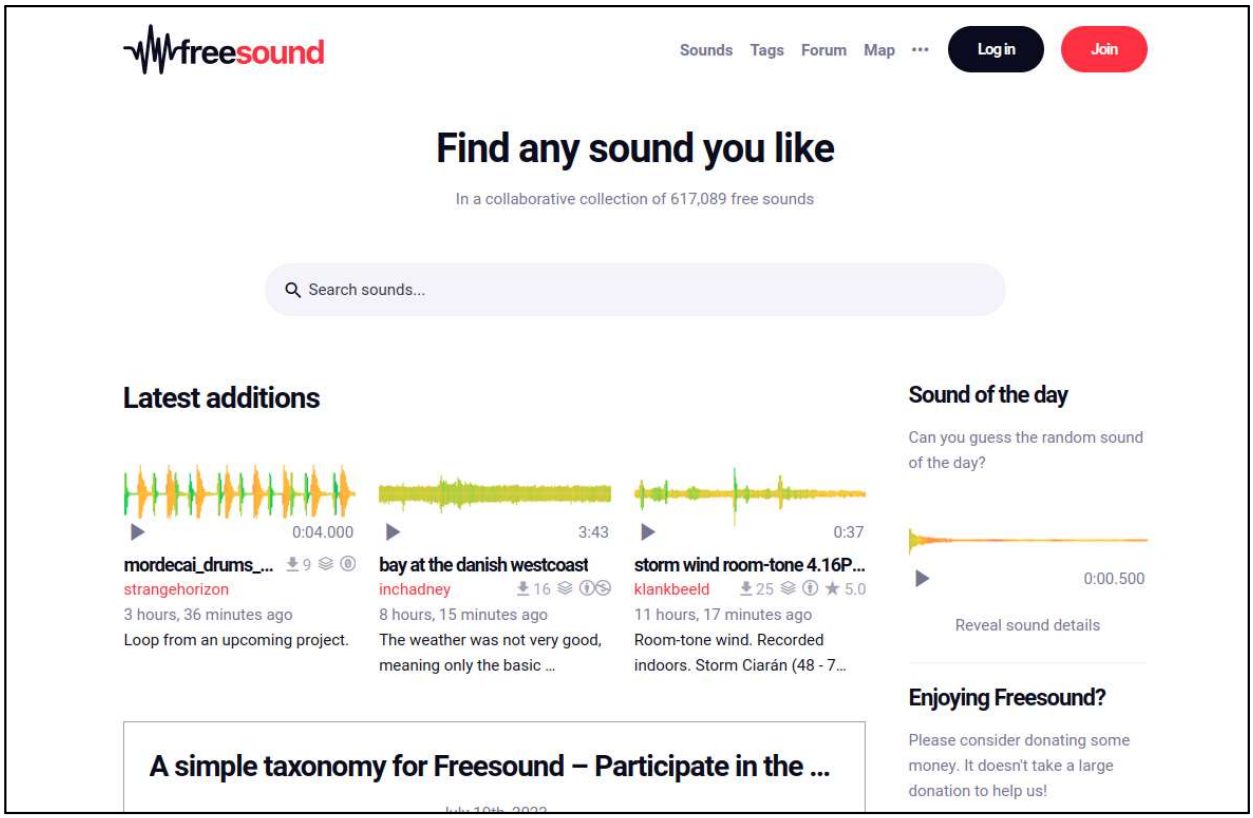

Figure 1: The Freesound website at freesound.org as of November 8th, 2023. Freesound has over 600,000 permissively licensed audio clips, making the resource ideal for our project.

In the past, it was difficult to find freely licensed sound effects online, let alone music. Fortunately, times have changed: there are now many sites offering public-domain or Creative Commons-licensed sound effects, the most prominent of which is Freesound [1] (seen in Figure 1). At this stage of the project, the specific sound effects required have not been determined, but the extensive catalog of over 600,000 sounds on Freesound virtually guarantees that any sound effects needed can be found.

Freely available and/or royalty-free music is nearly as abundant as freely licensed sound effects. However, royalty-free music may not be ideal for the project. Players often perceive royalty-free music as lacking authenticity: in some of team members' previous game projects,

multiple playtesters complained of music that “sounds royalty-free” (guessing correctly). In addition to royalty-free music, the use of recently released AI music generators such as Google's MusicLM [2] or Facebook's MusicGen [3] may be explored. There are open-source and hosted options, generating public-domain music, some trained exclusively on public-domain sources, approaching the quality of human composers for “simple” scenarios such as in-game music. A final option is to compose original music; the main benefit of this is to increase immersion via layering, as described in Definitions and Unique Features.

Engine Features

Unreal Engine 5 was chosen as the basis for the game rather early on, mostly on the strengths of its graphics, physics, and scripting systems (as well as a few team members' prior experience). Initial concerns that Unreal Engine would not have adequate support for advanced sound features were quickly found to be unfounded:

- Earlier versions of Unreal Engine had somewhat limited support for features such as layering (even the Unreal documentation admits “Sound Mixes can... become challenging to debug” [5]. However, Unreal Engine 5 overhauled the audio system, introducing “a high-performance audio system that provides... complete control over Digital Signal Processing (DSP) graph generation for sound sources.” [5]
- The new Audio Modulation feature introduced in the aforementioned overhaul allows easier control of parameters such as track volume, which should make layering much easier.
- The Unreal Engine documentation includes a comprehensive page describing audio spatialization techniques in general and the subset of which Unreal Engine supports [5]. While some advanced techniques that would essentially require audio raytracing are not

implemented (as they would be too convoluted), it appears all “reasonable” features such as panning and HRTFs are implemented and enabled by default.

Unique Features

The Escape Room Challenge game should go above and beyond the features provided by Unreal Engine “for free” to make its soundscape even more immersive. This will be achieved via layering of the game's music. As briefly discussed earlier, multiple audio tracks can be created and mixed at runtime, essentially creating several variations of a given piece of music. The player's sense of accomplishment will be bolstered when, after discovering a clue, solving a puzzle, or otherwise progressing through the story, another track is added to the background music, bringing in another instrument. This can continue until at the end of the level, the full song plays with all instruments.

An additional avenue towards a stand-out soundscape is audio-based challenges within an escape room. For the sake of accessibility, any audio-based challenges should be opt-in and not necessary to complete the game or advance the plot. There are interesting challenges which balance these two concerns: in one of the levels, somewhere along the trail of clues will be the combination to a safe. The player can find the combination clue, then use it to open the safe as usual... or simply crack the safe by sound, as one can with real safes. The puzzle is also thematically appropriate: many real-life escape rooms include safes. Indeed, upon further research this idea has already been implemented in The Last of Us [6]. Despite its apparent unoriginality, this would still be an interesting challenge to add to the game.

Level Design

Visual Fidelity and Realism

Unreal Engine 5 is renowned for its exceptional visual fidelity and realism. This is crucial for puzzle-based games that often rely on intricate environments and detailed level designs to immerse players in their worlds. The engine's Nanite virtualized geometry technology allows for the creation of highly detailed and complex environments, making it easier for developers to craft visually stunning puzzles and environments that players can explore and interact with. The stunning graphics offered by UE5 contribute to the game's overall immersion and the player's emotional connection to the world, which can be particularly important in puzzle games where players spend extended periods exploring and solving intricate challenges.

Dynamic Lighting and Atmosphere

The visual appeal and atmosphere of puzzle games are essential for immersing players in the game world. Unreal Engine 5's dynamic lighting and atmosphere systems allow for the creation of stunning and immersive environments. Real-time global illumination and ray tracing enhance the realism and atmosphere of the game, making puzzle environments feel more tangible and engaging. These capabilities empower developers to convey the intended mood and tone of their puzzle-based worlds effectively.

Scripting

Powerful Physics and Interaction Systems

Puzzle-based games rely on physics and interaction systems to create dynamic and interactive puzzles. Unreal Engine 5 excels in this aspect with its Chaos physics system. This

system allows developers to create realistic physics-based puzzles and interactions, which are vital for puzzle games that often involve manipulating objects, triggering mechanisms, and solving physics-based puzzles. The Chaos physics system is not only highly realistic but also integrated seamlessly into the engine, making it easy for developers to implement complex interactions without needing to develop extensive custom physics systems from scratch.

Blueprint Visual Scripting

Blueprint Visual Scripting in Unreal Engine 5 is a game-changer for puzzle game development. This node-based visual scripting system allows designers to create complex gameplay logic and level scripting without the need for extensive programming knowledge. In puzzle games, where intricate puzzles and intricate sequences of events are common, Blueprint simplifies the scripting process. Designers can easily create, modify, and debug gameplay logic through a user-friendly interface. This accessibility to scripting empowers level designers to focus on the puzzle design aspect, leading to more innovative and challenging puzzles.

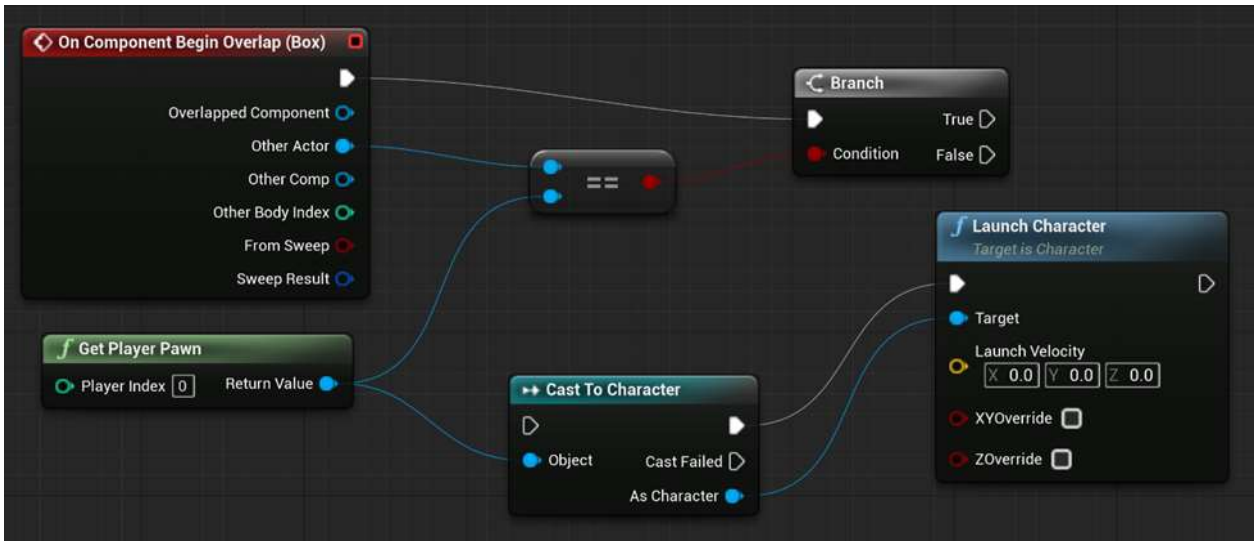

Figure 2: An example of an Unreal Engine 5 Blueprint.

Visual Scripting for Rapid Prototyping

Unreal Engine 5's visual scripting not only simplifies the scripting process but also accelerates prototyping. In puzzle games, prototyping is essential for experimenting with new gameplay mechanics and puzzle ideas. With Blueprint (shown in Figure 2), developers can create and iterate on prototypes rapidly, enabling them to test and refine their ideas without investing extensive time in coding. This agile approach to prototyping fosters creativity and innovation in puzzle design.

Slate UI

Unreal Engine also has its own user interface framework called Slate. This integrated framework is easy to procedurally build interfaces, which should be useful for the hint system, timers, and system menus. It is also generally easy to pull data directly from the code. The system is not without its cons, as it seems that it is more difficult to use uniquely made animations and styling [9].

Online Subsystem and Services

The Online Subsystems and Services may make it easy for a stretch goal of multiplayer to be implemented, as online interactivity is the main goal of this feature, and escape rooms are generally challenges to be solved by working with a team. This feature may also be used to further support the high score attribute with its leaderboard functionality [10].

Features for Both Level Design and Scripting

Collaboration and Accessibility

Unreal Engine 5 supports collaborative development, making it an ideal choice for puzzle games that often involve multidisciplinary teams working on complex projects. The engine allows multiple team members to work on the same project simultaneously, with each developer focusing on their specific areas of expertise, such as level design, scripting, or asset creation. This collaborative approach enhances efficiency and creativity within the team, as various elements of the puzzle game can evolve in parallel.

Community and Marketplace

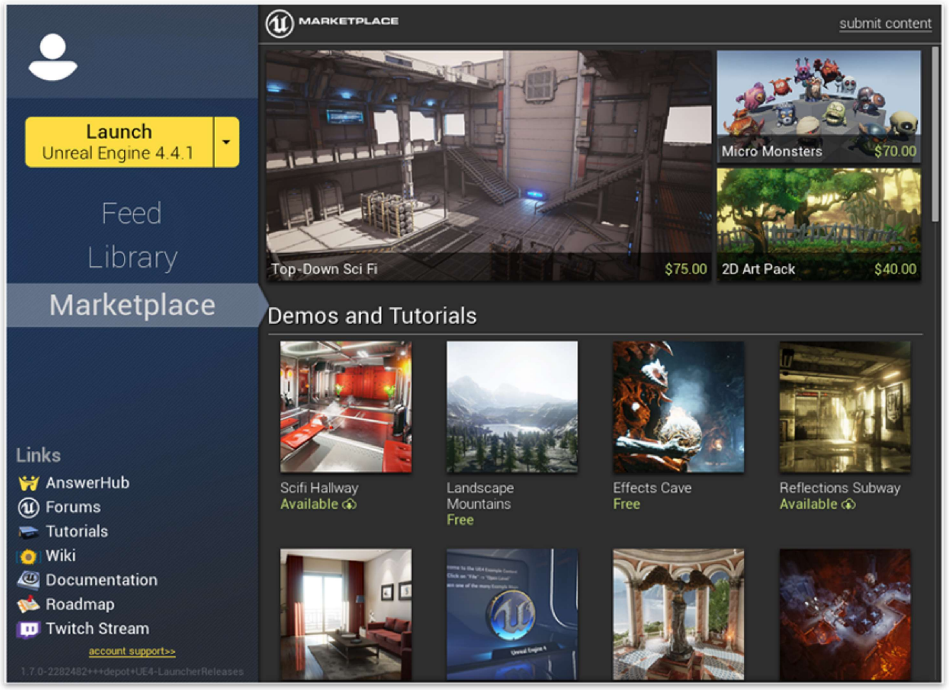

Figure 3: The Unreal Engine Marketplace.

Unreal Engine 5 benefits from a robust community and marketplace (seen in Figure 3). For puzzle game developers, this means access to a wealth of resources, assets, and plugins created by other developers. This significantly reduces the time and effort required to create assets and mechanics from scratch. Game developers can leverage the Unreal Engine Marketplace to find premade assets, materials, and scripts that are compatible with UE5, accelerating the development process and allowing more time for focusing on the unique aspects of their puzzle game.

Real-Time Feedback and Iteration

In puzzle games, the iterative design process is crucial for refining puzzles and ensuring they are engaging and challenging. Unreal Engine 5's real-time rendering capabilities facilitate rapid iteration, allowing developers to see changes instantly and gather immediate feedback. This real-time feedback loop enhances the development process, as designers can quickly test and adjust puzzle mechanics and level layouts, resulting in a more polished final product.

Integration of Third-Party Tools

Unreal Engine 5 supports seamless integration with a wide range of third-party tools and software. This is particularly beneficial for puzzle game developers who may want to use specialized software for creating puzzles, designing intricate 3D models, or implementing advanced AI systems. The engine's flexibility and compatibility with external tools ensure that developers can incorporate their preferred software into the development pipeline without major roadblocks.

Accessibility for Designers

Puzzle games often require a balance of creativity and logic in level design and scripting. Unreal Engine 5's user-friendly interface and Blueprint visual scripting system make it accessible to level designers who may not have extensive programming experience. Designers can visually create logic and level elements, allowing them to focus on crafting challenging and engaging puzzles rather than getting bogged down in code. This accessibility empowers designers to bring their unique ideas to life in a more direct and efficient manner.

Data Management and Version Control

In puzzle game development, version control and data management are crucial to maintaining project integrity and collaborating effectively within a team. Unreal Engine 5 offers integrated tools for version control, making it easier to track changes and collaborate with team members. This ensures that puzzles and level designs remain consistent, and that no critical assets or data are lost during development.

Player Mechanics

Heads-Up Display (HUD)

There are many important aspects to player mechanics. The first of which is the HUD or heads up display. This is essentially the part of the screen that tells the player what is happening in the world around them in terms of numbers and data. It is something that could take on an infinite amount of variety from being extremely complex and overwhelming to being completely non-existent. Through personal experience and research into popular opinions of video game players, it has been found that players tend to be more immersed in the world of the game when there is the barest amount of distractions from the HUD [7] (unlike the HUD shown in Figure 4.)

Figure 4: A HUD with too much information is only a distraction that will serve to annoy and take the player out of the in-game moment.

Only information that is necessary and cannot be accessed through the world should be stapled permanently to the player's screen. Because this is primarily a puzzle game, the only thing that should be on the HUD is the amount of time left that the player has to solve the challenge. This can be a small clock in the bottom right of the screen that shows how much time is left. There is no need to track a health bar, stamina bar, compass, mini-map or even what the player has equipped. There is no health or stamina, the environments will be small enough to not require the use of a mini-map or compass, and whatever the player has equipped can be shown in the players hand as it is planned on this software being from a first person perspective. The timer additionally will be easy enough to implement as all it requires is a timer and to display that in a static position in the player's view.

Player Inventory System

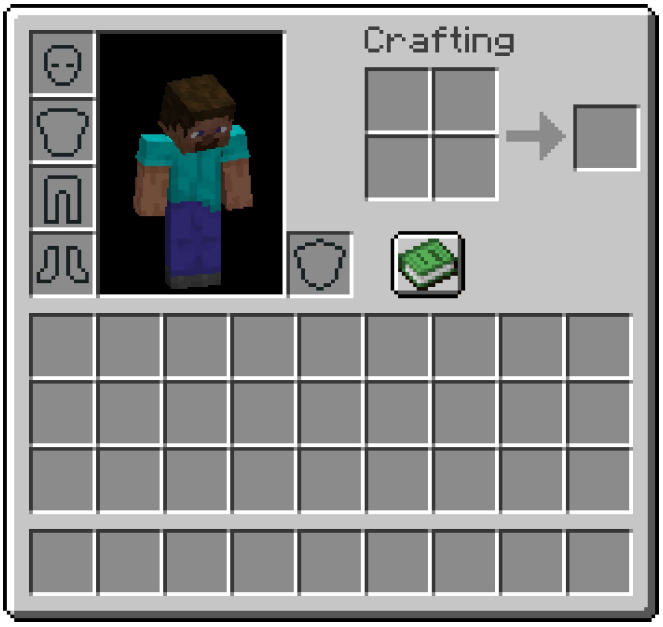

Figure 5: The Minecraft inventory screen is exemplary of player inventory systems.

The next important design consideration is the inventory system. This may be a bit more complex than the HUD because there will be many things that the player could interact with and pick up for use in the puzzles. The player will want to take these items with them as they move though the world; therefore we will need to track what items the player has. For the sake of both realism and simplicity, the player's inventory capacity should not be unlimited. An example of an inventory system similar to our approach is given in Minecraft (pictured in Figure 5).

Even this, though, is more complex than what is needed. All that will be needed is 10 slots for the player to be able to access from a key press (ideally the key 'i' as it is the most common inventory access key for video games that use a mouse and keyboard). The rest is unnecessary, meaning all that really is needed is the bottom row. The first slot can be designated as the active item holder for an intuitive mechanism for player equipment management. To encode the player's current inventory contents, an integer array of length 10

cna be utilized: if a slot is holding an item, it is filled with a non-negative “item ID”; if a slot is empty, it is assigned to ‘-1’ to indicate that it is available. To drop items from the inventory, an add slot at the very end of the row will be highlighted in red with a “drop” text underneath it such that whatever item is dragged and dropped to that slot will be dropped. Players will be able to pick these dropped items and new items up in the game when they move close enough to an object and left click on it. Lastly, the items in the inventory will be presented to the player as little icons that graphically depict the item which is being held.

Player Interaction

Another important design consideration to discuss is how the player will be able to interact with the world around them. Because this is a puzzle game the player will be interacting with the world a lot to solve the puzzles. As previously mentioned, a player will be able to pick up objects and have one object actively equipped. The object which is equipped is the object which can be used to interact with the outside world. For example, a key could be equipped and then used on a locked door to unlock the door. If the key were not equipped, the player would not be able to unlock the door. There are other examples of how the player could interact with the world such as picking up items and dropping them in different locations or even clicking on a static item to change its state such as to move the dial of a lock. These interactions will depend on the level design and the puzzles we choose to include. However, the ability to pick up items and interact with other items by clicking on them will prove to be more than enough to give us many puzzle possibilities.

Player Movement

The last thing to discuss is player movement. This will be a first person game where the player can walk around an environment and explore the world around them. It does not need to be complex at all and will not include things such as running, climbing, or even jumping as this is a puzzle game and is intended to test the puzzle solving ability of the player, not their in game agility. Simply using the ‘W’, ‘A’, ‘S’, and ‘D’ keys will suffice for the movement controls while the mouse will act as the control for where the player is looking. This is a simple thing to implement in a game engine as the physics of movement are already built in (no additional plug in or api is required as this is essentially boilerplate code for a video game) and Unreal Engine will be used as the development environment this won’t be a challenge. The only variable which will be needed to figure out is the movement speed of the character which can be tweaked once an environment to walk around in has been found and then test out speeds.

System Dependencies

The main system dependency which will be utilized in this project is Unreal Engine 5. It is a game development engine which is where all of the work for completing this project will be done. Through it graphical assets will be imported such as objects and textures to be used in the game. These will prove invaluable as they will allow for a quick development because time will not have to be spent on creating art for the game. Additionally, Unreal Engine 5 allows for collaborative synchronized development.

Release / Deployment Strategies

Test Plan

The testing process includes functional testing to verify core functionalities, compatibility testing across different platforms, and performance testing to assess smooth gameplay under various conditions. Usability testing focuses on the player's experience and should include accessibility testing to account for a variety of common conditions. Security and stability testing are incorporated to identify and address potential vulnerabilities. Regression testing is conducted iteratively to maintain the integrity of existing functionalities after bug fixes or feature additions. The plan concludes with user acceptance testing, bug tracking, and documentation, providing a systematic approach to quality assurance before final release.

Risk Assessment

There are many risks and potential challenges that could impact the development and release of the game. Several need special consideration, including resource constraints such as a limited budget or manpower, which could affect the overall development timeline and quality. Technical risks may manifest in compatibility issues across different platforms or unanticipated complications during the implementation of specific features, such as advanced graphics or multiplayer functionality.

Other potential risks include:

- **Lack of Testing:** Insufficient testing could result in the release of a product with bugs, negatively impacting user experience.

- **Intellectual Property Issues:** Legal challenges related to copyright or trademark infringement could arise, particularly if certain game elements resemble existing intellectual properties.
- **Dependency on Third-Party Tools**: Relying on external tools or libraries may introduce risks, such as compatibility issues or unexpected changes in third-party software.
- **Scope Mismanagement:** Poorly defined or constantly expanding project scope can lead to unrealistic timelines and resource overruns.
- **Team Dynamics:** Internal conflicts or communication breakdowns within the development team could hinder progress and creativity.

By acknowledging and addressing these potential risks early in the development process, the team can implement mitigation strategies and foster a more resilient and adaptive approach to the project. Regular reassessment and flexibility in response to evolving circumstances are crucial components of effective risk management.

Appendices

Project Phases and Milestones

Phase I

Type	Date	Description
Document	10/19/2023	Problem Statement
Document	10/26/2023	Requirements Document
Document	11/09/2023	Design Document (Draft)

Release	11/16/2023	v0.0.1
Document	11/30/2023	Design Document (Final)
Release	12/06/2023	v0.0.2

Phase 2

TBD.

Phase 3

TBD.

References

[1] F. Font, G. Roma, and X. Serra, “Freesound technical demo,” in *Proceedings of the 21st ACM International Conference on Multimedia*, ser. MM ’13, Barcelona, Spain: Association for Computing Machinery, 2013, pp. 411–412, isbn: 9781450324045. doi: 10.1145/2502081. 2502245. [Online]. Available: https://doi.org/10.1145/2502081.2502245 (visited on 11/08/2023).

[2] A. Agostinelli, T. I. Denk, Z. Borsos, et al., *MusicLM: Generating music from text*, 2023. arXiv: 2301.11325 [cs.SD]. Available: https://arxiv.org/abs/2301.11325 (visited on 11/08/2023).

[3] J. Copet, F. Kreuk, I. Gat, et al., *Simple and controllable music generation,* 2023. arXiv: 2306.05284 [cs.SD]. Available: https://arxiv.org/abs/2306.05284 (visited on 11/08/2023).

[4] Epic Games, *Audio in Unreal Engine 5,* 2022. [Online]. Available: https://docs.unrealengine. com/5.0/en-US/audio-in-unreal-engine-5/ (visited on 11/08/2023).

[5] Epic Games, *Spatialization overview in Unreal Engine,* 2022. [Online]. Available: https://docs. unrealengine.com/5.0/en-US/spatialization-overview-in-unreal-engine/ (visited on 11/08/2023).

[6] Boomstick Gaming, "The Last Of Us Part 2 - Crack Every Safe Using Only Sound | NO CODES EVER NEEDED," YouTube. Jun. 20, 2020. [Online]. Available: https://www.youtube.com/watch?v=ayoest6OJZ8 (visited on 11/08/2023).

[7] LOCHO53, "I'm of the opinion that there are more games that would benefit without a HUD, than with." Reddit. Jul. 17, 2019. [Online]. Available: https://www.reddit.com/r/truegaming/comments/cegpf6/im_of_the_opinion_that_there_re_more_games_that/ (visited on 10/28/2023).

[8] "Global Escape Room Market to Reach $31.00 Billion, by 2032 at 14.8% CAGR: Allied Market Research," *Yahoo Finance*, Jul. 11, 2023. Available: https://finance.yahoo.com/news/global-escape-room-market-reach-085200922.html (visited on 11/04/2023).

[9] Epic Games, *Understanding the Slate UI Architecture in Unreal Engine,* 2022. [Online]. Available: https://docs.unrealengine.com/5.1/en-US/understanding-the-slate-ui-architecture-in-unreal-engine/ (visited on 11/08/2023).

[10] Epic Games, *Online Subsystems and Services in Unreal Engine,* 2022. [Online]. Available: https://docs.unrealengine.com/5.1/en-US/online-subsystems-and-services-in-unreal-engine/ (visited on 11/08/2023).