The Outcast God

by

Tom Colebrooke

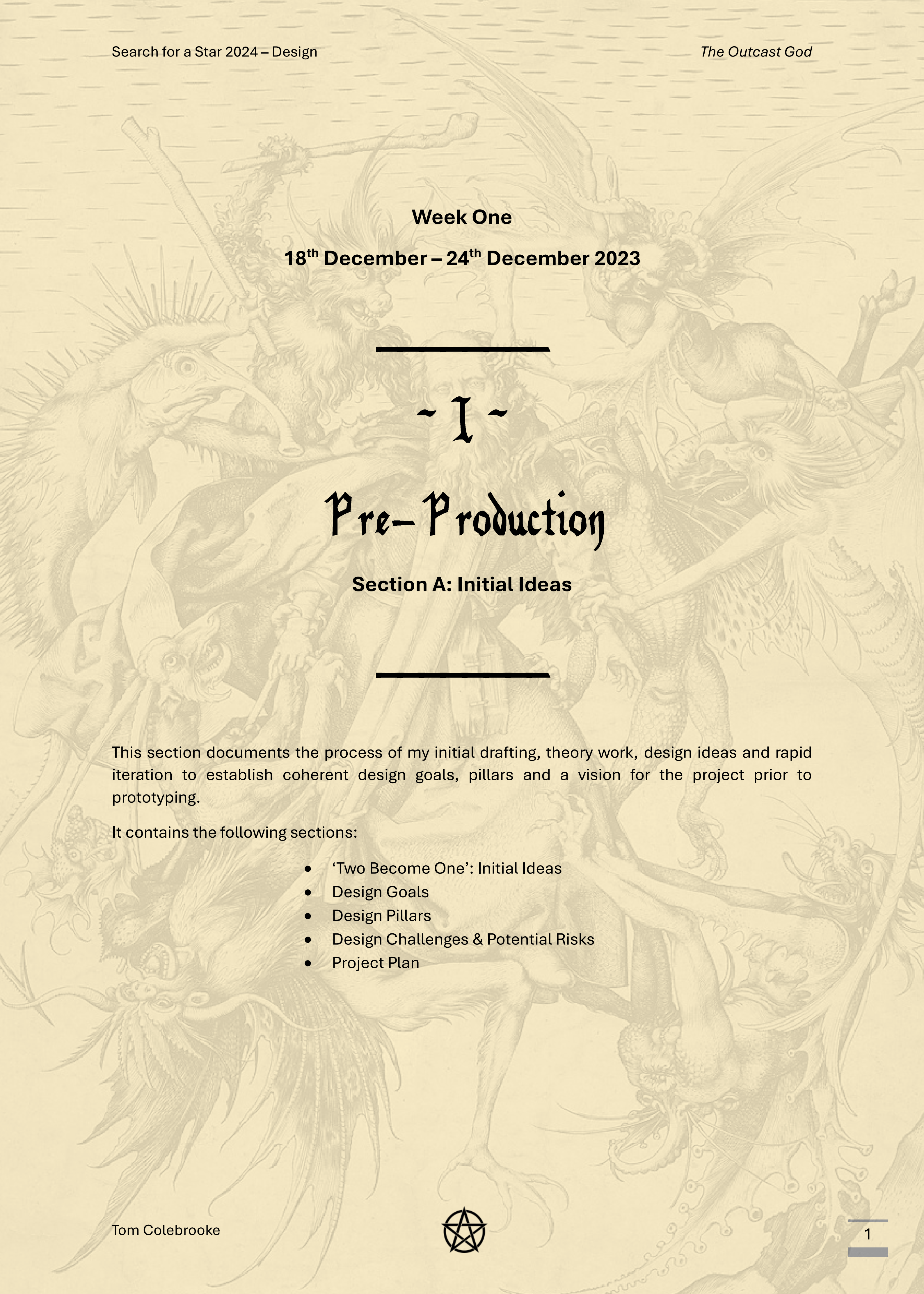
Search for a Star 2024

Game Design Category

Contents & Structure

Elevator Pitch

The Outcast God is a fast-paced, 3D, first-person supernatural action/puzzle game that borrows the best elements of FPS and Puzzle games to create an explosive, chaotic, unique gameplay experience. Shoot fire from your hands, slow time, use the environment, perform executions, and manage your powers carefully to solve puzzles and elude death at the hands of hordes of vicious demonic entities and stop them from summoning a great and terrible power.



Week One

18th December – 24th December 2023

~ I ~

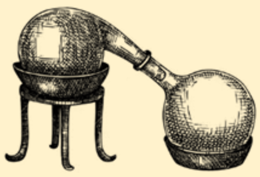
Pre-Production

Section A: Initial Ideas

This section documents the process of my initial drafting, theory work, design ideas and rapid iteration to establish coherent design goals, pillars and a vision for the project prior to prototyping.

It contains the following sections:

- ‘Two Become One’: Initial Ideas
- Design Goals
- Design Pillars
- Design Challenges & Potential Risks
- Project Plan



'Two Become One': Initial Ideas

What to combine?

This brief allows a great deal of room for interpretation: encompassing genres, mechanics, objects, and presumably those aspects of gameplay that fall somewhere in between those labels. The task is to combine two things that 'don't conventionally go together'.

I started by considering popular and widely understood genres, and aspects of their mechanics or primary goals or rules that I could swap out for something creative and unexpected. I explored a 'Battle Royale' game in which the goal is to die as quickly as possible, and a horror game in which tickling or laughter was the only weapon to fight off ghosts.

However, I realised that these ideas were **inversions** of formulae for games or mechanics that work well already, rather than new or interesting combinations. To me, this seemed to be slightly missing the point – or at least the potential – of the brief. I could see the design challenge becoming less about bringing together two incongruous things and more a way of thinking how to keep one aspect of a successful formula intact while dropping one of the core principles that makes it succeed. This just didn't seem alchemical enough. It wasn't **combining**, just **turning upside down**.

I wanted to push the envelope, and to see if I could think of two things that, far beyond not conventionally 'go[ing] together', operate in absolute antithesis to one another. In this way, the challenge was to combine two things while **keeping them intact** in a meaningful way. I wanted to find two genres and mechanical systems that have totally different goals, operate by different rules, and offer separate experiences to meet the wildly different expectations and motivations for their audiences.

Of all the genres that came to mind, two stood out as being diametrically opposed in their aims, assumptions, values, mechanics, presentation, and aesthetic. Those two genres are the exhilarating, acrobatic, frantic, fast-paced first-person shooter and the slow, cerebral, non-threatening, deliberate 3D puzzler.

- ***What if combat was combined with puzzling?***
- ***What if they depended on one another?***
- ***What if the puzzle itself was treated as another enemy in a level?***

This gave me my starting points:

First Person Shooters + 3D Puzzlers

e.g.

DOOM (2016)
Warhammer 40,000: Boltgun (2023)
BioShock (2007)

e.g.

The Talos Principle (2014)
The Vanishing of Ethan Carter (2014)
Portal (2007)

Why these two?

When speaking about level design for puzzles at GDC 2016, Jolie Menzel highlights the fundamental characteristics of a 'puzzle' as essentially a **non-competitive challenge** with **one solution**. Meanwhile, the core tenets of the FPS genre are the opposite of that. They are **highly competitive challenges** where the goal is to win **by any means necessary**. Many routes can be taken to that end. Some leap around avoiding gunfire, preferring to pick enemies off from a distance with accuracy, some prefer a more personal approach, using melee tools and mechanics to dash from enemy to enemy, healing and replenishing armour as they go. Some use rocket launchers, some use shotguns, some run out of ammo, some carefully manage all of their tools and resources. It doesn't matter; what matters is that all of these paths can and do lead to **successful outcomes** for the player. In a puzzle, meanwhile, only one specific and meticulously predetermined path will lead to a single successful outcome. All other outcomes are prohibited by the mechanics or environment. Menzel efficiently characterises this conflict between combat and puzzles when she says that 'games are played to *win*, puzzles are played to find a *solution*'.[1]

The task of combining the FPS and puzzle genres is therefore an irresistible challenge: these subtle distinctions and conflicts will be offered up for scrutiny, allowing me to question assumptions about genre conventions while resolving seemingly impossible contradictions in value that go to the core of design theory. I see choosing these two starting points and bringing them together in this way as an opportunity to learn and grow as a designer while generating ideas that have a real chance of yielding something unique, fun, and meaningful for players.

Finding Ingredients

At this point, the question is how to design a system in which a problem (combat) with various solutions must be used to solve a puzzle with one solution.

I have seen people describe combat in difficult games as a puzzle on the basis that deducing attack and dodge patterns, and learning to read telegraphs, is a puzzle, but I don't think this description is accurate. Combat is a **problem**: one with many answers or solutions. This distinction matters a great deal. It is the reason there are no head-scratching puzzles in *Call of*

[1] Jolie Menzel speaking at GDC 2016: 'Level Design Workshop: Solving Puzzle Design'. https://youtu.be/0xBJwrm9C8w?si=dB6Zl3auiqeuGgbO

Duty or *Wolfenstein*, and the reason that we don't have to glance at ammo or health bars in *Braid*, *Baba is You* or *The Room*. In order to avoid becoming bogged down in semantics too early on, I knew I had to draw up a comprehensive working list of conventional mechanics and characteristics for both of my starting genres. This list did not have to be exhaustive or authoritative, but instinctive and focussed on moment-to-moment gameplay.

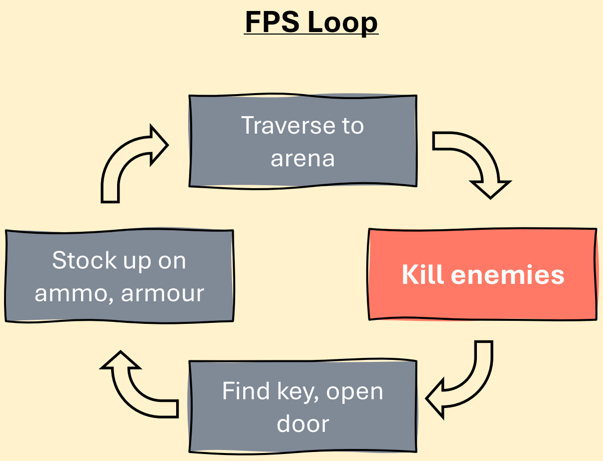
I revisited my primary inspirations: *DOOM*, *The Talos Principle*, *Boltgun*, *Portal*, *BioShock* and *The Vanishing of Ethan Carter*. I also briefly revisited some adjacent titles, including some that explicitly operate at genre boundaries, to cross-reference these characteristics. I compiled the essential core elements that all of these games share: the properties that tether them to their genre and the label of an **FPS** or **Puzzle** game. These were, in no particular order:

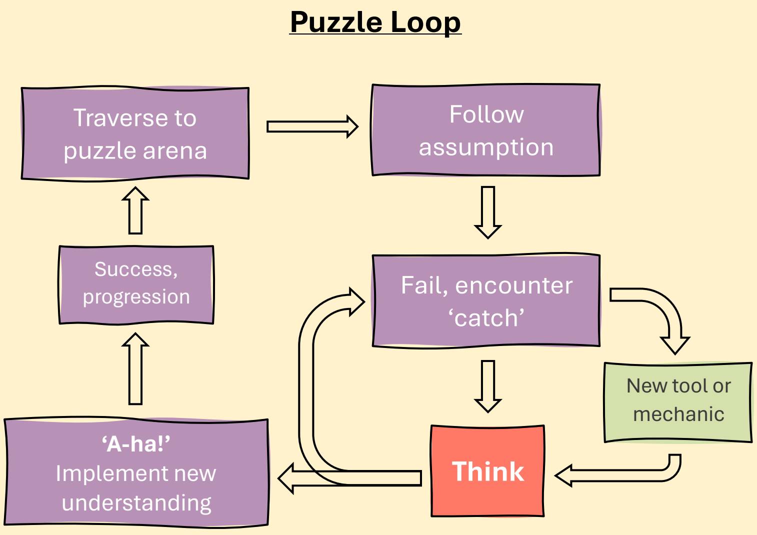
Combat FPS Gameplay Features	**Puzzle** Gameplay Features
• Shooting	• Time
• Movement: speed and agility	• Logical thinking
• Dodging	• Memory
• Targeting / Aiming	• Close observation of environment
• Reloading	• Understanding of mechanics
• Ammo management	• Sometimes no fail state, little or no consequences
• Weapon switching	• Reaching a clearly defined end goal
• Health management	
• Input precision	

These can be refined, sorted and classified as follows:

Feature Type	**Combat FPS**	**Puzzle**
Tools	Weapons, movement	Logical thinking, observation
Resources	Ammunition, health	Time
Mechanics	Plentiful, pliable	Limited, fixed
Playstyle	Many possible solutions and strategies (no dominant strategy)	A single unknown solution (dominant strategy)
Penalty	Death, progress removed	Stasis, frustration, lack of progress
Abilities	Optional and sometimes custom	Fixed and required
Goal	Win	Solve

Next, I had to consider the gameplay loops for both game types and start thinking about which steps I could effectively amalgamate into one experience.





'Think' is obviously a more nebulous and tricky-to-design-for quality than 'kill enemies', but the inputs to that stage in the loop are more important than the process itself. Assumptions, failure, and making sure the player has the mechanic necessary to overcome that assumption lead to the logical thinking that brings those elements together and results in a new level of understanding: that much sought-after 'a-ha' moment. In the puzzle loop, the 'features' I identified are spread more evenly between the loop elements 'Fail', 'New tool', 'Think' and 'A-ha', whereas in the FPS loop, almost all its mechanics and features exist within the 'Kill enemies' stage. This fact could be important in a moment. Before I take a crowbar to these long-established and successful formulae, however, it's important that I define my values for what a combination will look like.

Design Goals

In setting out to achieve a difficult goal, it is vital that I retain a solid sense of **scope** and what will be realistically achievable given the constraints of the competition, not least of which is a timeframe of only four or five weeks to take this project from its initial idea to completion and a playable demo. I also have to allow time for thorough documentation and presentation while bearing in mind that these will be the primary materials for assessment. Before asking myself what I *want* to achieve with the prototype, therefore, I first must establish what I *could* achieve with the prototype.

Considerations:

- A rigid deadline of 21st January, allowing approximately **four full working weeks** (accounting for time off over the festive period) to complete the project.
- A guideline project length of **30 hours**.
- I will be **learning Unreal 5.3 for the first time**, with a long hiatus from the engine after learning to use UE4 some years ago.
- The prototype should only contain a **maximum of ten minutes** of gameplay.

Given these considerations, time is against me, and I have the added pressure of re-learning the engine while in the process of prototyping. After weighing the implications of time and considering the brief and my vision for the final product, I set myself the following design goals. I will use these to aid decision-making throughout prototyping and further design to keep myself within scope and avoid becoming side-tracked by extraneous issues that are non-critical to these objectives.

1. **Design an engaging and elegant gameplay loop in which combat and puzzle solving rely on each other in a First-Person fast-paced setting.**

2. **Build a prototype in Unreal containing a tutorial to teach the core and secondary mechanics followed by a short use-case arena to show off the dynamics of the game.**

3. **Playtest and iterate the demo to produce a product that leaves a viable template for a complete game.**

In one sense, these goals reflect the ‘success criteria’ of the project and will be used as a benchmark against which to test the final gameplay experience. In another, they are almost mantras to consider when deciding if a feature, or a set of hours spent completing a task, is going to serve the overall vision.

Design Pillars

Design pillars are an important shorthand for the core principles that should guide decision-making in forming the concept and content of a system of mechanics and their use-cases. The rich variety of directions available to any project that attempts to combine two disparate genres or subjects gives all the more reason to establish these pillars firmly, and as early as possible in the design process. My initial vision for this project and the values I will refer back to when making decisions are as follows.

***The Outcast God* should be:**

#1 **Intuitive**	**#2** **Balanced**	**#3** **Cohesive**
• *Players should fight the environment, not the controls.* • *Encourage experimentation once core systems are learned.* • *Clear and satisfying audio-visual feedback.*	• *Combat and puzzling are symbiotic.* • *Encourage the use and mixing of all available tools.* • *Prevent dominant strategy.*	• *Appealing and challenging to players from both genres.* • *Mechanics-as-metaphor.* • *Straightforward lore should inform and justify systems and environment with internal logic.*

It would be easy to get carried away with complexity and bloat on a concept like this. I want to use the goals and the above pillars to remind myself not to over-complicate: try and let players jump in, understand the mechanics, feel powerful, and have fun. I need to stay laser-focussed on that vision in the most essential form possible, especially for a basic prototype.

I also want to keep the best parts of both genres intact throughout the grafting process. This will be difficult, but the goal is to explore new forms of gameplay. This will mean trying not to lean too heavily on one of the genres, demoting the experience to an FPS with 'puzzle elements' or a Puzzler with 'combat segments'. I want to strive for a fully integrated and **cohesive** FPS/Puzzle game. Combining two disparate genres successfully will also depend on the mechanics coming together seamlessly. This means making sure every aspect of gameplay is consistent through implementation, narrative, lore, level design, aesthetic and presentation. Inevitably, this comes with challenges and risks that need to be addressed before making a project plan.

Design Challenges and Potential Risks

Incorporating two directly conflicting genres that follow totally different design philosophies into a single game within a tight timescale and small-scale demo in an unfamiliar engine is not without its challenges. Perhaps they're more than challenges. This project risks abject failure, and I have to come to terms with that. In embarking on this challenge there is fear, but also freedom. This is an experiment, a design competition, and a rare opportunity to really challenge myself and my skillset to produce something unique and something I can be proud of. However, that margin for error will loom over me at all stages: during my initial designs and graphing, gnawing doubt during the prototyping before all features are implemented, and the dread when I distribute the demo for playtesting; let alone final submission.

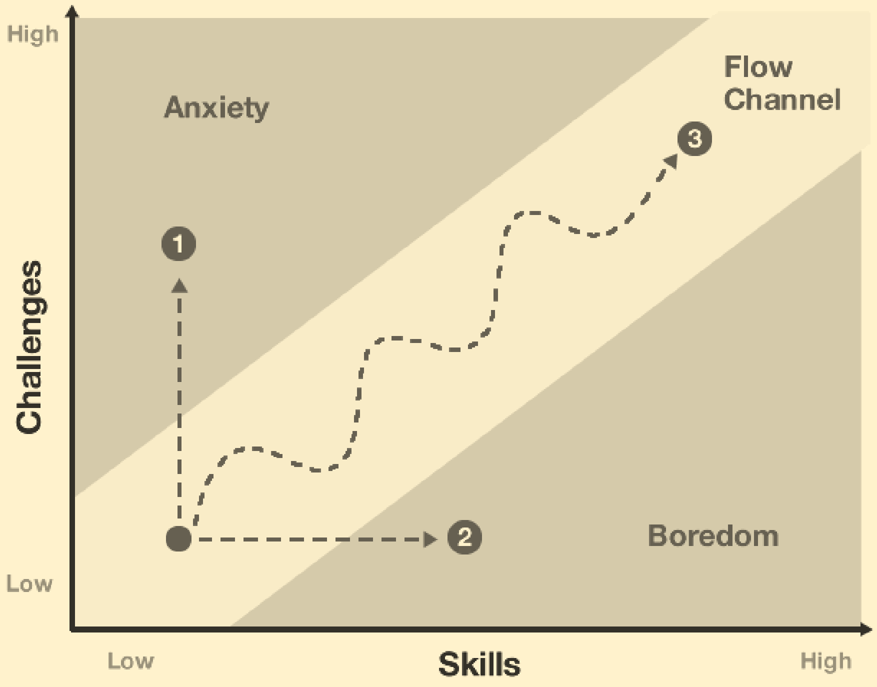
There are a number of ways that this project could fail, and at least if I acknowledge these outcomes, or at least the likely ways in which this project could go astray now, early on, I can plan for them and think through some best practices to either dig myself out of the hole, or note the loss, bookmark it for another day and move on before too much time is wasted. The following are some of the realistic issues that I can see facing both pre-production and production phases of this project:

Anticipated Issue	Possible Causes	Best Practice Response
Gameplay feels 'lopsided', privileges one mechanic or system over another and becomes too stale, static, spammy, or boring.	Either combat or puzzling yielding too many results too easily or too quickly.	Consider outcomes to each action in the gameplay loop and try introducing counters to each in turn, see if it resolves the issue.
Mechanics become too bloated, too many 'steps' to reach understanding.	Tying together too many elements from both FPS and puzzling genres, becoming afraid of the editor's pen when combining two fun and intriguing genres.	Consult design goals and pillars, be bold and slightly ruthless with trimming elements that don't serve the vision.
A mechanic I design is too complex to prototype in Unreal with my current skillset within the project timeline.	Lack of familiarity with more complex aspects of Blueprints or a function that requires custom C++ scripting.	Consult Unreal documentation, search for YouTube tutorials, or if all else fails, streamline the mechanic to something within my capabilities and make a note to follow up after the deadline to pick up the required skills.

A Note on Flow:

Another challenge, which is broader and more theoretical than any sort of logistical, skill or easily definable design problem, is a conceptual one concerning the elusive 'flow-state' that game designers hope to achieve when crafting experiences for players.

The concern I have is that the two elements of my game: the archetypal dynamics of the FPS and the Puzzle game will pull in opposite directions and prevent flow. Players will be anxious about the puzzle and bored or frustrated with the combat as it gets in the way of them instantly trying to implement a new perspective or piece of understanding they have about the problem they're trying to solve. The recently-established Second Wind recently talked about a similar issue with a puzzle in *Obduction*. Their primary gripe was the process(es) the designers put in ***between*** enacting changes on the puzzle pieces. These were seen as frustrating filler. This is easy to see in other puzzle games too: the successful ones generally allow the instantaneous and smooth implementation of different ideas and strategies through very basic controls, movement and mechanics. *Obduction*, it seems, took things too far – and, notably – lore and narrative was not enough to prop up the rationale.



Source: Amir Afianian on Medium.

FPS games generally have to implement tools to supress anxiety. Puzzle games generally have to carefully implement tools to prevent boredom. My work, if I'm not careful, may involve a highwire act of trying to juggle both. Striking that balance and widening that channel in which more than one type of player can feel successful, powerful, engaged, emotionally involved, and **having fun** seems to be the biggest challenge facing me when designing the core mechanic and gameplay loop and one I will have to devote serious time and energy to. Providing enough elegant interchange between the puzzling and combat elements, making consequences for straying too deeply into one system while neglecting another, will be key: **but how to get there?**

With those risks and challenges noted, the final step before beginning to make conceptual strides is to establish a rough workflow and project plan.

Project Plan

Beginning this project on Monday 18th December, with a deadline of Sunday 21st January, gives me precisely **five weeks**, or **34 days**. Allowing for a break over the festive period from Christmas Eve to Boxing Day, this includes **30 working days**. I have decided to break down the project into clearly defined weeks with tasks that centre around a particular stage in the process to help me maintain a sense of momentum within each stage, and to easily keep track of task deadlines.

Week 1 will be devoted to Pre-Production and Planning the project, including tasks like generating and documenting initial ideas and concepts, rapidly iterating designs and reaching a point where a solid workable template can be taken forward into the engine for prototyping. This phase will be tricky to track in terms of hours spent – it's always tough to account for 'thinking time', especially as the best ideas always seem to come when I'm away from my desk doing something entirely unrelated. However, the process of meticulous documentation and conceptualising in terms of visually representing these ideas can be accounted for.

Week 2 will see the beginning of initial prototyping in Unreal including setting up a working basis. I like to start with the base systems like movement, health, damage, death, etc. If I can make it fun to jump around an empty cube shooting projectiles at straightforward AI, it stands me in good stead for making it fun to do so in a polished environment with extra mechanics and tricky opponents. I'll try and get the **core mechanic** up and running in this week too so that I can get a sense, as soon as possible, of what the tone and feel of the gameplay loop is going to be, helping me adjust and fettle the finer points of balancing later on.

Week 3 will see the main in-engine production phase where **content**, **secondary mechanics** and **polish** are implemented including level design, use cases, tutorial, and sound and music. I have also allowed time in both weeks for bug fixing.

Week 4 is dedicated to further development, playtesting, and documentation and will see the most intense period of composing and editing this document to show my creative processes throughout the other weeks, as well as break down the final product from a design perspective.

Week 5 will see the final stages of presentation including the writing, recording, and editing of the showcase video, documenting my reflections and preparing the project for submission.

On the next page I have drawn up a comprehensive scheduling table breaking down each week's forecast of tasks and their allotted time. At the end of each week, I will compare the time taken to complete each task with the time budgeted and mark down time gains and losses to make any necessary adjustments to the next week's schedule. The aim with this system is to prevent a cascade of time building up and then reflecting on a 60 rather than a 30-hour project as intended. I will also be using a **Kanban board** hosted on **Jira** to visualise each week's tasks. At the end of the project, I can reflect on the time taken to complete all tasks and reflect on the success or failure of my planning methods.

Detailed week-by-week chronological task plan:

Week	Task	Time Budgeted (hrs)	Time Actual (hrs)
1 **w.c. 18.12.23** ***Pre-Production & Planning***	Draft a premise and story concept informed by the brief, goals and pillars established	0.5	?
	Draft Design Goals	0.2	?
	Draft Design Pillars	0.2	?
	Write up challenges and risks	0.3	?
	Put together a project plan	0.5	?
	Block out entire documentation structure	0.5	?
	Conceptualise Core Mechanic	2	?
	Conceptualise Secondary Mechanics	1	?
	Draft Core Gameplay Loop	1	?
	Reflect and plan production next steps	0.5	?
	Week 1 Total	**6.7**	?
2 **w.c. 25.12.23** ***Production***	Make Backlog from design documentation that will inform prototyping in Unreal	0.1	?
	Procure assets	0.5	?
	Prototype base systems: movement, health, damage, dying, implement basic enemies, get a basic AI working, controls	2	?
	Prototype and implement Core Mechanic	3	?
	Bug fixing 1	1	?
	Week 2 Total	**6.6**	?
3 **w.c. 01.01.24** ***Further Production***	Implement Secondary Mechanics	3	?
	Playtesting, Iteration	3	?
	Balancing	1	?
	Tutorial design	1	
	Level Design	3	?
	Sound & Music	1	?
	Bug fixing 2	1	?
	Week 3 Total	**13**	?
4 **w.c. 08.01.24** ***Further Development***	Further playtesting	0.5	?
	Implement and iterate more secondary mechanics	1	?
	Implement feedback from playtesting	1	?
	Scene dressing	1	?
	Secondary mechanics	0.5	?
	Bug fixing	0.5	?
	Write bulk of GDD	2	?
	Implement HUD / UI	1	?
	Week 4 Total	**7.5**	?
5 **w.c. 15.01.24** ***Presentation***	Write Elevator Pitch, Outline of Concept and USPs	0.5	?
	Write video script	1	?
	Cut together footage	0.5	?
	Record voiceover	0.5	?
	Write Reflections	0.5	?
	Write up ‘Presentation’ section	0.5	?
	Put together Appendices and References	0.2	?
	Final checks and submission	0.5	?
	Week 5 Total	**4.2**	?
		Budgeted Hours Total: 38	**Actual Hours Total: ?**

Week One (cont.)

18th December – 24th December 2023

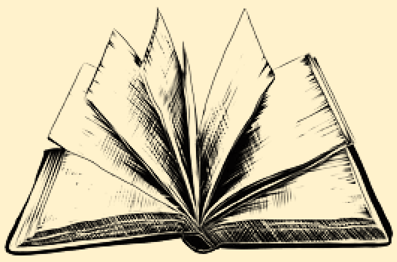
~ I ~

Pre-Production

Section B: Concept Development

This section details my thought process in converting my initial analysis, ideas, and reactions above into a workable design concept to use a basis for a prototype in Unreal. It reflects the output of a few productive hours of creative idea generation in the context of the brief and the objectives outlined previously. It contains the following sections:

- Premise & Story Concept
- Core Mechanic Design
- Secondary Mechanics
- Core Gameplay Loop: First Draft
- Week One: Recap and schedule update



Premise & Story Concept

Cast out of his home by the False God, Montekin, the ‘Halfborn’, the ‘Realmless’, has wandered adrift in undying fog for centuries; until a whisper came to him. His Mother’s voice, guiding him to the Everdoor and urging him to reclaim his place in the court of Pandaemonia: his home, and the realm of his Father, who carried out his banishment. With his spiteful kin standing before him, he faces a serious of perilous trials to get within striking distance of his Father and enact his revenge.

Mechanics-as-Metaphor

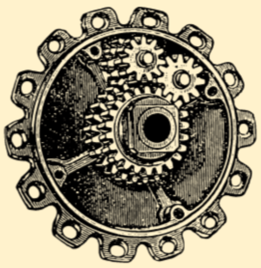
In reference to my **third design pillar**: cohesion, I wanted to make sure that the player character fit with the theme, ‘Two Become One’ every bit as much as the mechanics did. So, I crafted a simple lore premise that feeds into the mechanics and act as a basis that could be greatly expanded on in a larger game.

That is, Montekin’s nature as half-lightbringer half-daemon gives him Sight, that unique core mechanic ability that dominates much of the core gameplay loop. It’s in his ability to see what lightbringers see and wreak the destruction that daemons can wreak that make him able to string together both aspects of himself to overcome obstacles and challenges.

Making Montekin a half-angel half-demon also gives a rationale for the flight ability, and setting the game in a quasi-catholic gothic hellscape sets up lots of opportunity for creative and otherworldly level designs that give players a chance to express their skill in combat and their mastery over the puzzle aspects of the game. In fact, the world itself gives a solid artistic, stylistic and narrative basis for many of the mechanics you’re about to read about, namely:

- Demonic rituals summoning the False God, a round-ending presence
- Supernatural abilities
- Magical projectiles

It always adds that something special when a mechanic is backed up and justified in the lore of the universe that a developer has created. It adds a sense of internal logic that helps immerse players in the world and give a greater context to their interactions with it, generating positive experiences that leave a mark.



Core Mechanic Design

The core mechanic for *The Outcast God* has the intimidating task of combining the main elements of the **FPS** and **Puzzle** game loops into a single, elegant and intuitive form. It needs to be straightforward enough to prototype in the next two weeks, but complex enough to show a suite of use cases and potential for further growth. It needs to keep the elements of the gameplay loops I analysed earlier, namely 'Thinking' and 'Killing enemies' intact in a meaningful way. It needs to address the design pillars by being easy to control, balanced, and logically consistent with the supernatural premise and narrative that I've built.

To begin with, I need to go back to the **features list** I developed for both of my starting points, and decide which elements are **essential** to the union and which I can discard:

🟩 Green = **Keeping** Feature

🟨 Yellow = **Modifying** Feature

🟥 Red = **Discarding** Feature

Combat FPS Gameplay Features	**Puzzle** Gameplay Features
🟩 Shooting	🟨 Time
🟩 Movement: speed and agility	🟩 Logical thinking
🟥 Dodging	🟥 Memory
🟩 Targeting / Aiming	🟨 Close observation of environment
🟥 Reloading	🟩 Understanding of mechanics
🟥 Ammo management	🟥 Sometimes no fail state, little or no consequences
🟥 Weapon switching	🟨 Reaching a clearly defined end goal
🟩 Health management	
🟥 Input precision	

This gives me a coherent list of features that I am going to try and implement into the game and allows me to start deciding which elements to incorporate into my **core mechanic** and which to resolve with **secondary mechanics**.

Basic Features	**Modified Features**
• Shooting • Movement: Speed and Agility • Logical Thinking • Aiming • Understanding of Mechanics • Health Management	• Time • Close Observation of Environment • Clearly defined end goal

In order to generate a meaningful and consistent **symbiosis** between the two systems, I decided to take this list and further refine it by classifying each feature.

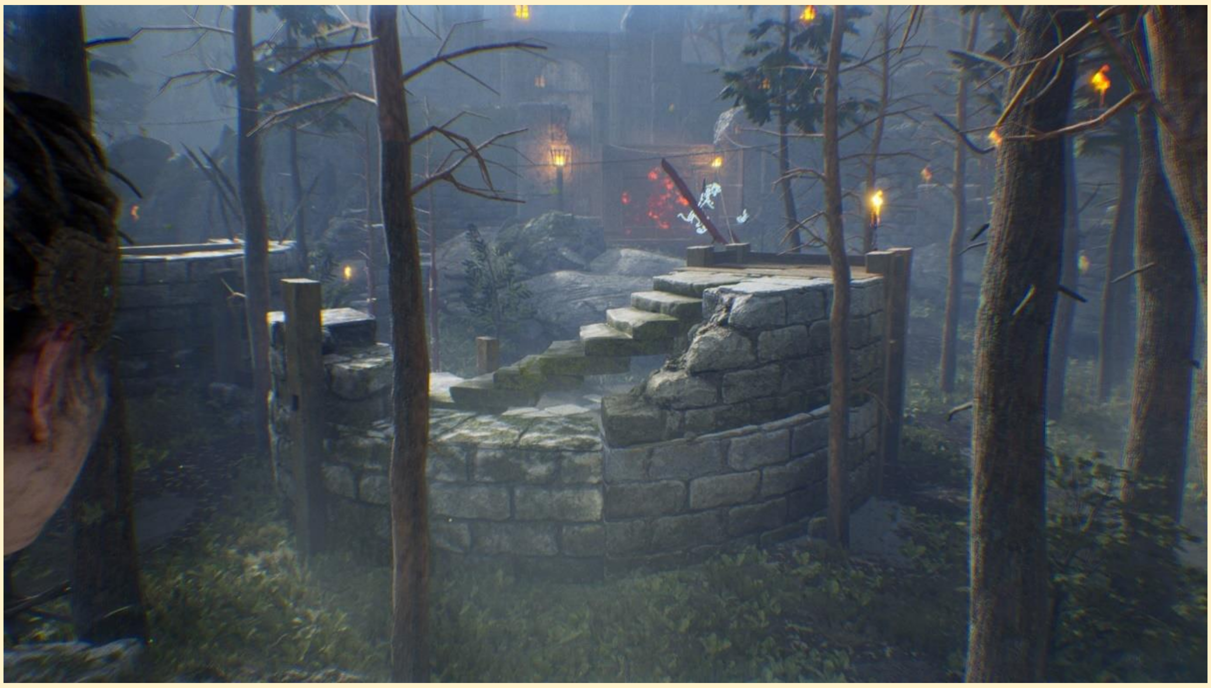
Feature Type	***The Outcast God* Mechanics**
Tools	• Weapons • Movement • Logical Thinking • Close Observation of Environment
Resources	• Health • Time
Mechanics	• Plentiful • Fixed
Playstyle	• Several uncertain opportunities of achieving one unknown solution
Penalty	• Death • Progress Removed
Abilities	• Fixed • Encouraged • Sometimes Required
Goal	• Win to Solve, Solve to Win

This list gives me the features I need to **combine** if I am to produce a design that acts as a meaningful blend of my two starting points. I can use it to test against any potential core mechanic ideas and as a touchstone for ensuring that both genres are being thoughtfully considered in the gameplay loop.

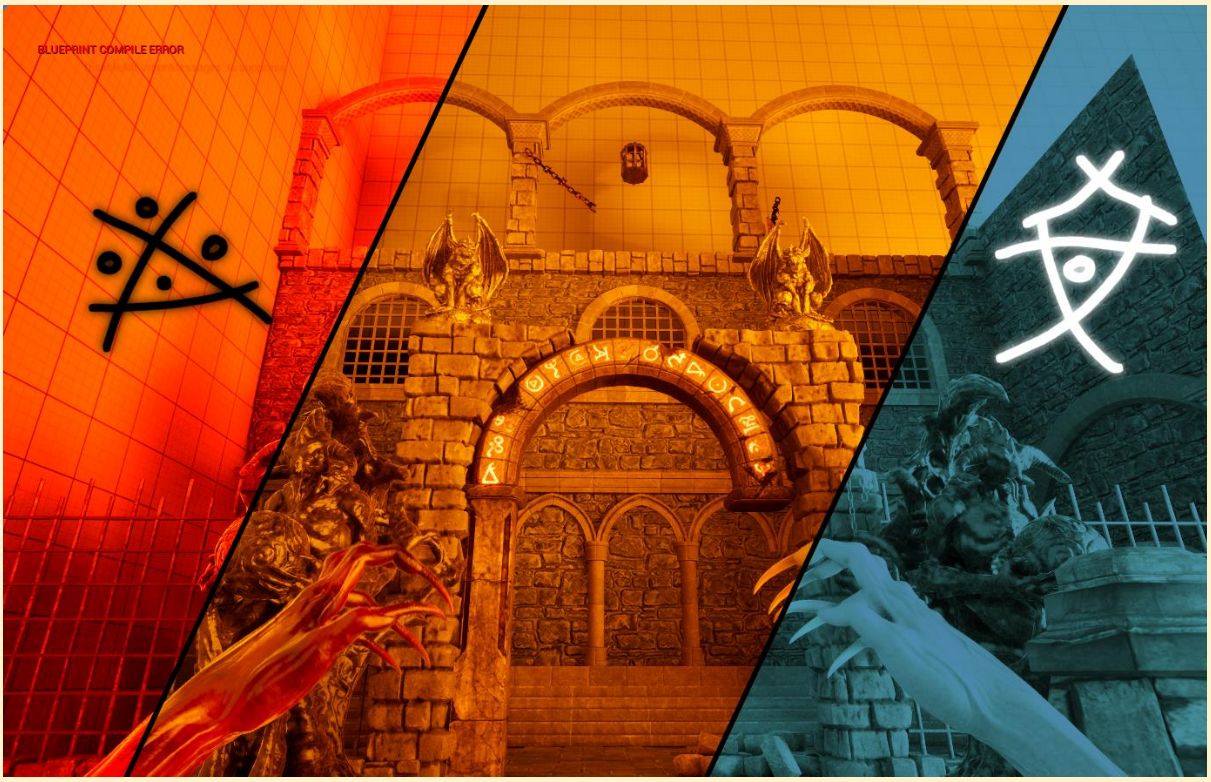
Initial Ideas

Having analysed and established the design theory needed to engage with my two starting points, I started conceptualising ideas that could serve as core mechanics.

My first idea was to have a meter that goes more towards the 'Lightbringer' or 'Daemon' side depending on attacks used. If the player had a lethal attack in their left hand, and a non-lethal conversion or possession ability in their right hand, they would need to balance the use of each to stay in the middle of that meter. I toyed with the idea of seeing different versions of the environment based on which side you were using more of (seeing fire and demonic themes on one side that would fade to blue lighting and angelic elements on the opposite end). I thought this presented an interesting opportunity for players to express their preferences in approaching combat. I then thought this would be interesting to combine with perspective puzzles like those in *Hellblade: Senua's Sacrifice*. The player would need to use demonic attacks to reveal certain gates or portals, and lightbringer attacks to reveal corresponding parts of the puzzle.


Perspective puzzles from Hellblade

I even got as far as mocking up a basic visual in Unreal to see what this approach might look like and see if it helped me think of any secondary mechanics, or an overall gameplay loop that tied together both elements in a solid way.



The visual I mocked up in Unreal to help me conceptualise my first idea of perspective puzzles.

However, though it was interesting to generate ideas around this premise, I eventually abandoned this approach. For one thing, the scope just seemed too ambitious for the time I had available: implementing a mechanic where entire environments would shift in real time as the player performs different actions in the world sounds wonderful, but I can imagine prototyping that in two weeks in Unreal in a way that effectively communicates the mechanic would have had me pulling my hair out. Also, from a design perspective, I anticipated that players would end up just holding down one side of the mouse, then the other, as required, see-sawing between actions to get the desired perspective or view needed to 'solve' the puzzle.

Conceptualising Sight & Glyphs

I revisited my features list and knew that I wanted to introduce an element of time into the gameplay loop. Time is an essential element in puzzle solving: it's essentially the resource that players use to play the game. The penalty for failure is stasis and a lack of progress which generates frustration, rather than lost or reset progress as in FPS games. I also knew I wanted to avoid putting up a hard border or distinction between the FPS and Puzzle aspects of the gameplay loop as this would defeat the purpose of combining them, so having the loop basically be fight then solve, fight then solve, without mixing them up was out of the question. My idea was then to change just one word in that sequence. Change 'then' to 'to' and you've got fight to solve, and flip them around a bit and you've got solve to fight, fight to solve etc. This was the loop I wanted. I conceived of Sight and Glyphs as a way to work in tandem to tie together attacking, puzzling, healing, movement and time manipulation in an elegant, intuitive and easy-to-grasp core mechanic.

This tandem begins with the activation of **Sight**, which:

- Is a meter filled by killing enemies.
- Slows down time and gives the player time to solve the puzzle.
- Enables interaction with elements of the puzzle.
- Reveals a hint on the battle arena's central platform.

Glyphs are another resource that needs activating, and can be spent on:

- Interacting with puzzle pillars.
- Performing an execution move which kills a demon and restores a small amount of the player's health.

Both of these resources are gained by killing demons with the **base projectile brimstone attack** and with melee.

Crucially, this combines **all four** of the 'tools' from both the FPS and Puzzle genres: Weapons, Movement, Logical Thinking and 'Close Observation of the Environment'. Therefore, the **core mechanic** of Glyphs and Sight is an elegant way of combining my two starting points and is at the heart of my interpretation of the brief.

In coming up with an idea for the **puzzle template** for the game, I needed to keep things very straightforward for the purposes of prototyping. Eventually I settled on a sequencing puzzle that demands that the player link up a set of channels funnelling lava into the right sockets. Initially I wanted to implement the idea that if the lava is sent to the wrong pillar, there are consequences, pursuant to my third initial question of 'what would it be like if the puzzle itself was another enemy in a level?'. It would stop players from standing by a pillar, harvesting glyphs with ranged attacks, and spamming turns until the puzzle is solved. It also casts the environment as another enemy, encourages balanced use of the mechanics, and results in playstyle that forces players to go along with the premise of the mechanics. However, after some consideration, I decided not to take this forward as it would risk falling into the trap I acknowledged earlier of leaning too heavily on the combat side of things. After all, what would those consequences be? Likely to health, increasing the base level of threat and anxiety and therefore also risking a break in flow.

A caveat to the time manipulation feature of Sight is that when turning puzzles, the turning action itself is not affected by time dilation. I'm glad I caught this early as I might have overlooked it otherwise. By making the puzzle pillar transformations instantaneous, rather than smoothly animated in real-time, it enables players to enact several changes on one pillar without worrying about transition time or bobbing in and out of Sight to use multiple Glyphs one after the other.

Secondary Mechanics

Continuing with the theme of time as a resource in puzzle games, I wanted to conceive of an end condition that was not related to dying in combat, as this would push the player towards solving the puzzle and therefore immerse them fully in the gameplay loop I was designing.

I looked to tabletop games like *Forbidden Island* and *Betrayal at the House on the Hill* for inspiration. Both of these games have meters that steadily fill before some calamitous round-ending event takes place, meaning that players experience a real sense of urgency to get on with navigating and solving each round. As such, I conceived of a **round** or **wave** system that will pressurise the puzzle-solving aspect and get rid of the possibility of an endless (and eventually boring) loop of harvesting Glyphs and spending them on puzzle turns. I really liked the grim reaper from Vampire Survivors too – it meant that the roguelike aspect was crunched into a solid 30-minute timeframe rather than being infinite upward curve, which would take away a sense of urgency once your weapons were upgraded. It is a constant: you always know in each round you have 30 mins before the grim reaper comes. So, I implemented a similar mechanic into *The Outcast God* to make sure that players focus on the puzzle and beat the clock, using their abilities in combat to do so.

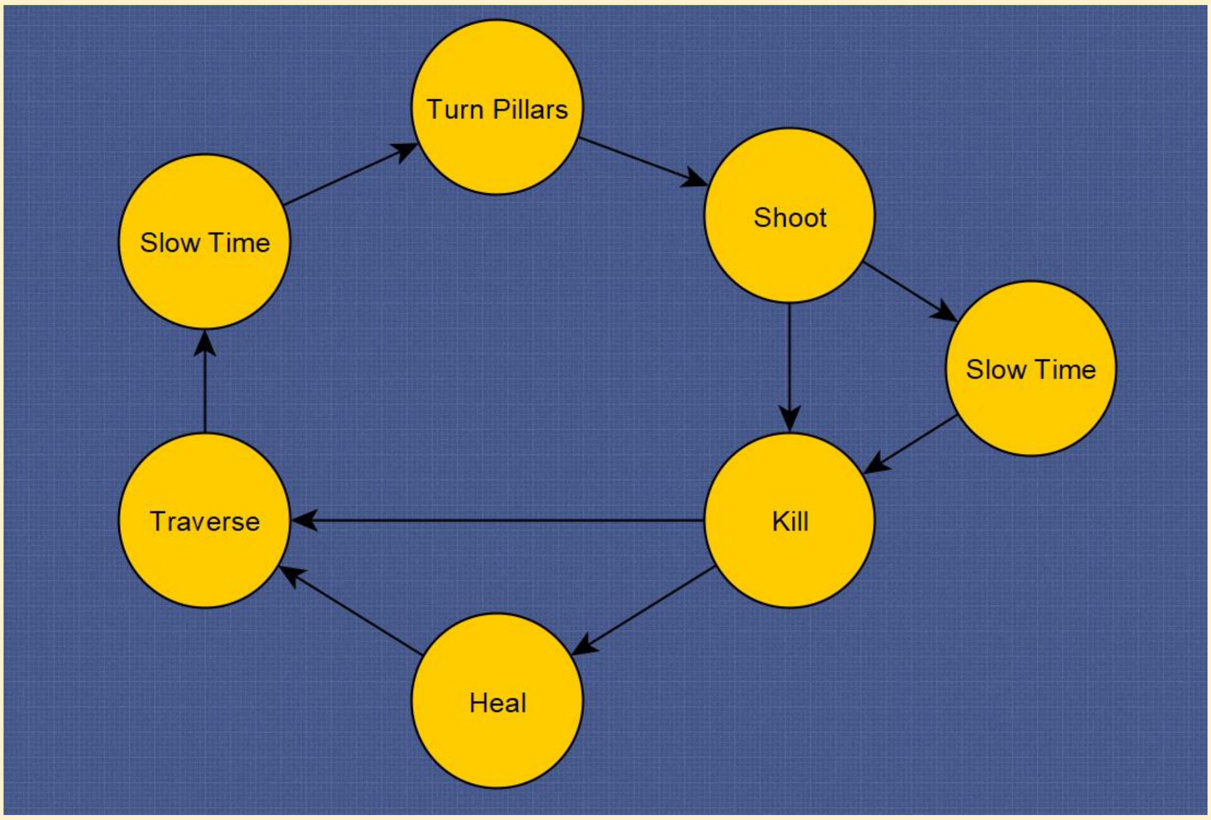
I also had to conceive of another possible way other than killing demons for players to earn Sight and Glyphs. It just seemed like it may become either too monotonous or too frustrating to have to kill an enemy before a solution could be reached, especially near the end of the five rounds. I initially thought that it might be exciting to have areas where players could stand to **passively** gain Sight or Glyphs, perhaps small ante chambers or chapels to the side of the arena. Standing still would be the drawback to these actions. But the infinite passive gain of these zones unnerved me and, within the context of the prototype, I thought would be too easy to misuse and over-optimise for players. So, I instead opted for consumables: statues that can be destroyed to redeem a fixed amount of Sight and Glyphs.

I also knew from the outset that I wanted to make traversal (a feature common to most games, but in this case, relevant to both the FPS and Puzzle genres) fun and engaging, as well as ‘useful’ in the sense that I have no plans at this stage to include a blocking or ‘formal’ dodge mechanic, so traversal should be tight, responsive and feature a flight ability.

Gameplay Loop

Having established the core and secondary mechanics, I can now organise them into a gameplay loop that considers the template loops of both the FPS and Puzzle genres in order to help decide the best sequence or order of repeating actions.

To integrate both the combat and puzzle aspects of Sight and Glyphs, I designed a **concentric** gameplay loop that contains two **key junctions** that represent player agency and decision making within each interaction. The nature of Sight and Glyphs as resources means that players should have a choice about where and how to use those tools to best reach their next goal in the moment.



The main gameplay loop for The Outcast God.

As illustrated by the loop, using Sight is **both** a choice **AND** a necessity, depending on the situation, and using Glyphs is likewise an essential tool for winning the level **AND** a costly method of healing if you're not careful and allow yourself to get damaged.

Now that I've managed to draft a full system of core and secondary mechanics to combine the FPS and Puzzle genres into a cohesive gameplay loop, let's revisit that list of requirements and see how the mechanics I've designed fit the brief, and integrate both game styles while keeping key parts of their successful formulae intact:

Feature Type	***The Outcast God* Mechanics**
Tools	• **Weapons** ○ A basic projectile attack – 'Brimstone' is activated by holding down the Left Mouse Button. • **Movement** ○ Quick running, jumping and flying with a double-jump style control that can be held for greater effect but not held indefinitely. • **Logical Thinking** ○ Pillar puzzles represent a basis for a puzzle template that can be built on and problematised later in the game with modular elements • **Close Observation of Environment** ○ In both combat and in puzzle-solving, close attention must be paid to the environment: ▪ To observe the 'hint' when using Sight. ▪ To not get caught out in a dead-end when trying to activate a Lightbringer or Daemon statue. ▪ To notice which pillars are connected or what the essential 'catch' of each arena puzzle is.
Resources	• **Health** ○ The only way to heal is to **spend sight and glyphs** on execution moves. This key aspect, inspired by the rush kill mechanic of *DOOM*, means that players must be aggressive and make use of **all three core mechanics** (basic attack, Sight and Glyphs) in order to maintain a pool of health. • **Time** ○ A key, indispensable element of almost all types of puzzles. Time is the endless resource pool that enables players to use their logical thinking and observational skills. In adding a time slowing effect to the Sight ability, it gives players an opportunity to process the puzzle in short bursts to integrate both combat and puzzling seamlessly into each passage of play.
Mechanics	• **Plentiful** ○ When in combat, there are enough tools (firing, dodging, flying, melee, executions) to give the

	player multiple ways to deal with each encounter. • **Fixed** o However, as the mechanics still have puzzling in their DNA, the end result needs to be one solution: turning the pillars in the correct order. This is done with one fixed action.
Playstyle	• **Several uncertain opportunities of achieving one unknown solution** o Combat gives players the tools necessary to engage enemies how they choose, and multiple branching opportunities to earn enough Sight and Glyphs to have a go at solving the puzzle.
Penalty	• **Death** • **Progress Removed** o Opting for two lose conditions: running out of health and running out of time combines the two pressures of fighting and puzzling into one.
Abilities	• **Fixed** o Players **must** use Sight and Glyphs in combination with their basic projectile attack in order to play and win the game. • **Encouraged** o Players are **incentivised** to use the execution (heal), flying, and statue attacks regularly to ensure an exciting game loop.
Goal	• **Win to Solve, Solve to Win** o The dynamic of each arena implements this mixture in a straightforward way: players combine 'winning' in combat with 'solving' the puzzle of each arena and cannot do one without doing the other. If they don't engage in the combat, they won't have the tools to solve the puzzle. If they don't solve the puzzle, they lose the fight, can't proceed and have to start again.

As outlined above, the core mechanics I've designed clearly address the fundamental elements of both the **FPS** and **Puzzle** genres, bringing them together as an intuitive, balanced, and cohesive whole. Heading into production week, I'm confident that I've given enough consideration to the nuances of this overall design structure to give myself a chance to produce a solid prototype.

However, I need to keep in mind that the initial design does not always reflect well in use cases or in practicality when put in front of players. I'm anticipating the need to refine and swap out elements of the core loop and adjust certain aspects of the design to make sure it's being used and expressed as intended to and by players.

Week One: Recap and Schedule Update

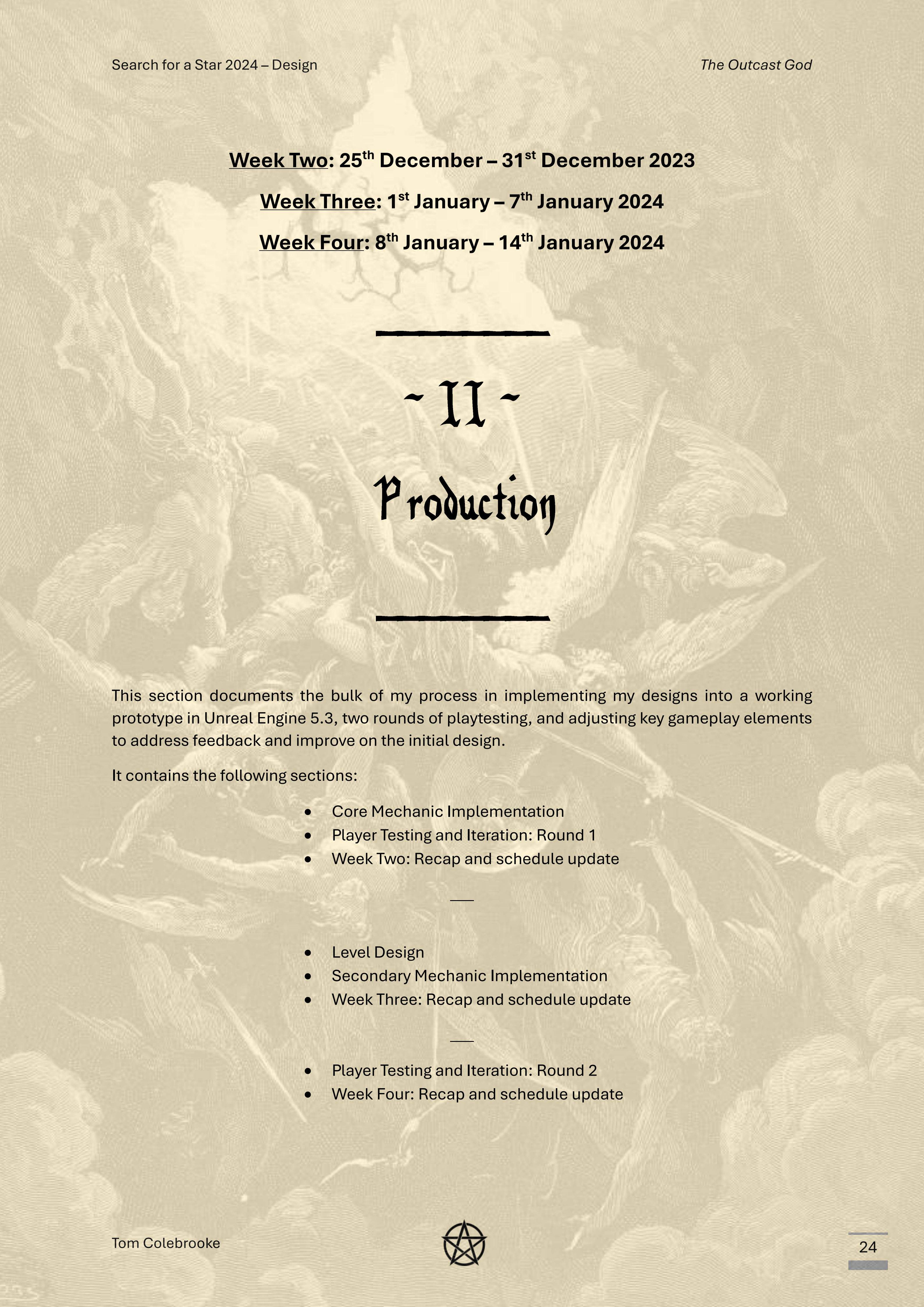
Each week, I will review the task list and time budget I assigned at the outset of the project against an updated timesheet that I've filled in as the week has progressed. This will allow me to review where things have gone to plan, where things have gone wrong, and make any necessary adjustments to the following week's schedule.

The review for week one is as follows:

Week	**Task**	**Time Budgeted (hrs)**	**Time Actual (hrs)**
1 w.c. 18.12.23 ***Pre-Production & Planning***	Draft a premise and story concept informed by the brief, goals and pillars established	0.5	0.6
	Draft Design Goals	0.2	0.3
	Draft Design Pillars	0.2	0.3
	Write up challenges and risks	0.3	0.3
	Put together a project plan	0.5	0.5
	Block out entire documentation structure	0.5	0.5
	Conceptualise Core Mechanic	2	2.2
	Conceptualise Secondary Mechanics	1	0.9
	Draft Core Gameplay Loop	1	0.5
	Reflect and plan production next steps	0.5	0.5
	Week 1 Total	**6.7**	**6.6**

I am pleased that I have managed to remain both **within time budget** and **within the scope** I outline for the project in the first day or two. I knew that time was going to be a serious consideration in trying to tackle this ambitious challenge and so far, things seem to be going well.

I need to be careful not to get carried away with adding unnecessary features or functionality during prototyping, and to get a working model of the movement, mechanics, and environment up and running as soon as possible to get the game in the hands of some play testers.



<u>Week Two</u>: 25th December – 31st December 2023

<u>Week Three</u>: 1st January – 7th January 2024

<u>Week Four</u>: 8th January – 14th January 2024

~ II ~

Production

This section documents the bulk of my process in implementing my designs into a working prototype in Unreal Engine 5.3, two rounds of playtesting, and adjusting key gameplay elements to address feedback and improve on the initial design.

It contains the following sections:

- Core Mechanic Implementation
- Player Testing and Iteration: Round 1
- Week Two: Recap and schedule update

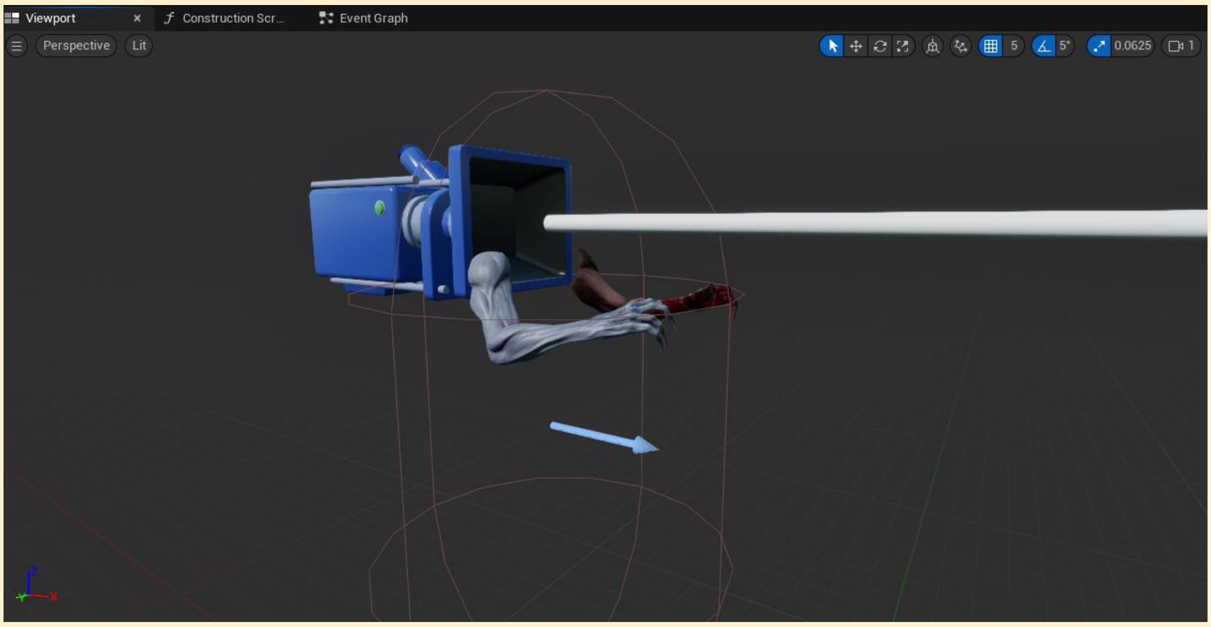
- Level Design
- Secondary Mechanic Implementation
- Week Three: Recap and schedule update

- Player Testing and Iteration: Round 2
- Week Four: Recap and schedule update

Core Mechanic Implementation

Movement and Flying

The first thing I had to do was set up the player character. The fact that Montekin is half-Lightbringer and half-Daemon meant that I knew I wanted some suitably creature-esque hands, and so I found some very cheap but high quality ones on the Epic Marketplace. Crucially, these hands had a modular texture that allowed me to make Montekin's right hand white (lightbringer) and his left hand red (Daemon). I implemented these into the engine with a basic animation blueprint and also gave the player a static mesh to overlap puzzle pieces and highlight them for interaction.



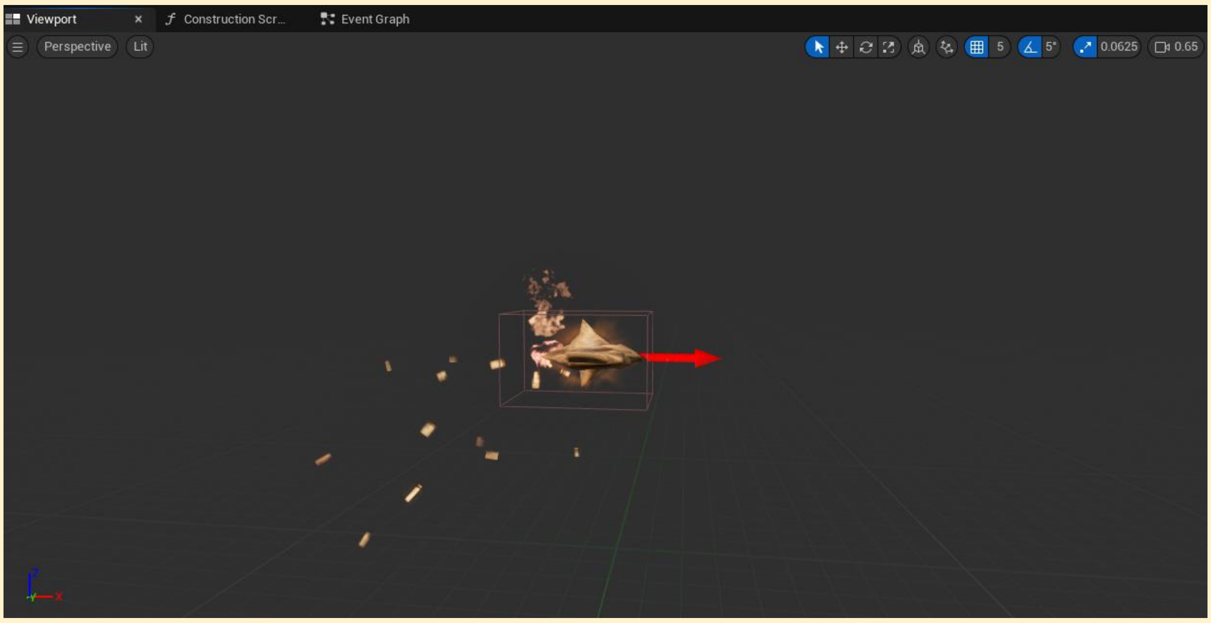
Implementing the player hands and a static mesh cylinder to generate overlap events in Blueprints without having to line trace.

After this was done, I mocked up a very quick test level to start tweaking movement values based on the default First Person Player Blueprint provided by Unreal. Conveniently, this template already has fully customisable movement options that include movement speed, jump height and control options, float amounts, friction values and more. This allowed me to get a basic movement system up and running very quickly and move on to more demanding tasks.

A simple test area to tweak movement settings, mainly the flight ability which I tested on the two tall rock obstacles.

Projectiles: Implementing the brimstone attack

To get the basic brimstone attack I first created a base projectile actor that had modular outputs for impact events, sounds, particle animations and more, to save time when implementing more than one type of projectile. I knew that I wanted each hand to shoot a different type of missile to reflect the two differently coloured hands (fire and ice) and also wanted to set up a template for including enemy AI projectiles later on.

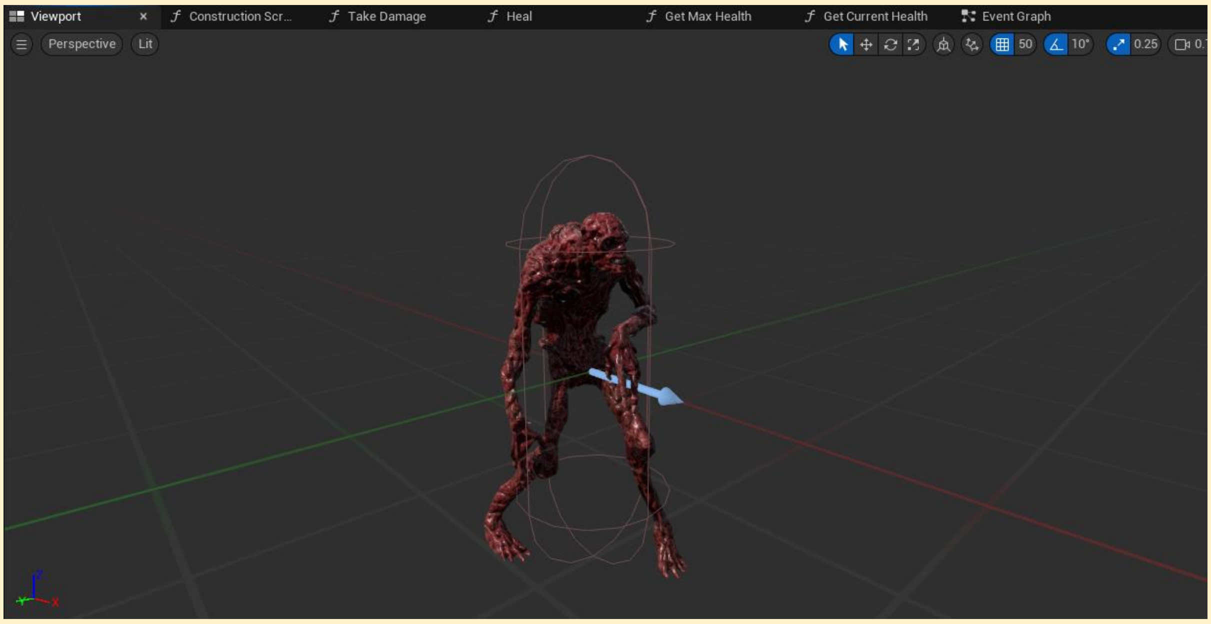


Making the base brimstone projectile

Adding Enemies

After this was done, I procured a basic demon enemy and repeated the process I'd used for the player mesh to get collision, navigation, and a basic AI working as intended.

Implementing basic enemies with animation blueprints and a simple follow and attack AI script.



Adding capsule collision, animation and AI to the enemy blueprint.

Once the demons were chasing and a damage system using a Class, Interface and Struct so that I could call on these functions across the player and all enemy actors without having to program each one individually.

Ironically, the core mechanics I'd designed, upon which the main combination of my starting points rested, were by far the easiest to prototype and get in the player's hands for playtesting.

Prototyping Sight

This was quite straightforward and involved defining Sight as a float value within the player blueprint. After attaching this to Unreal's built-in time dilation feature and creating some conditions around when it should and should not enable the player to interact with pillars, it worked as intended on first pass.

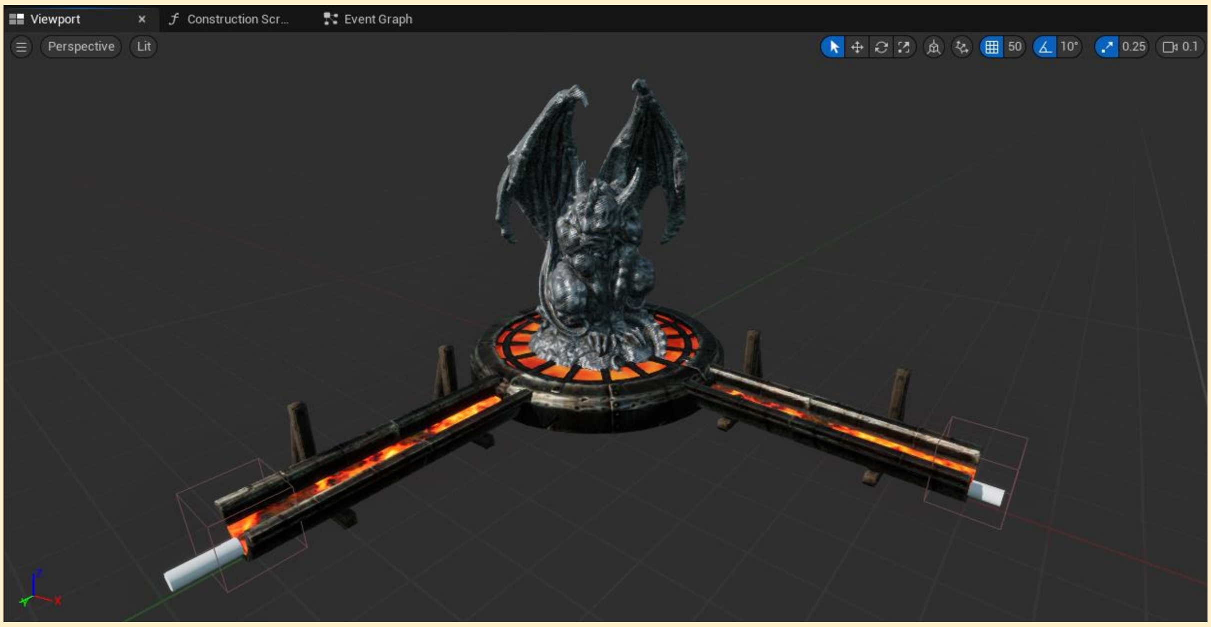
Prototyping Glyphs

Likewise, glyphs were easy and painless to set up. An integer value that is incremented up and down when interacting with the world.

As soon as these core mechanics were in place and working well enough to all be used together, I packaged a simple level and began playtesting and gathering feedback which was a hugely important step in iterating on the finer points of balancing the design.

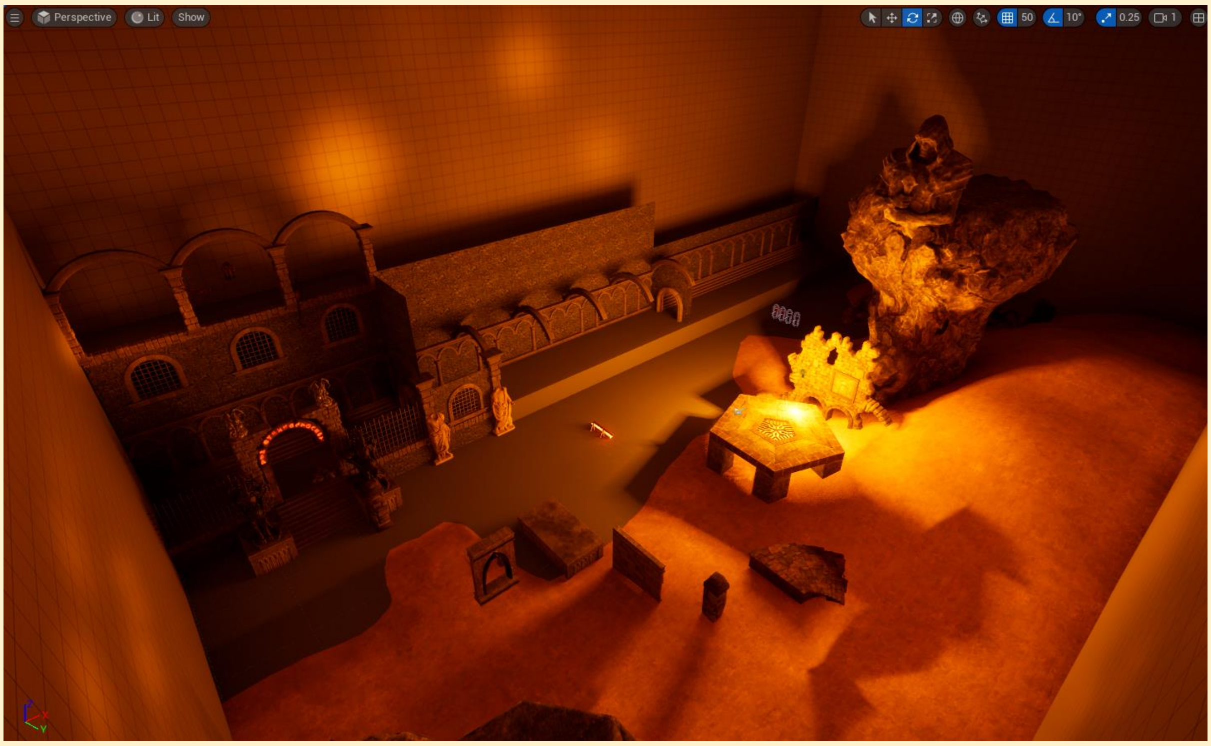
Implementing Basic Pillars

Making interactable pillars involved merging assets in a new blueprint, adding collision boxes to act as nodes through which the correct combination of overlaps can give a readout of a solved puzzle to the level blueprint.



The gargoyle pillar I created for prototyping purposes with basic collision detection boxes at the end of each lava channel.

Player Testing and Iteration: Round 1



The first prototype level I shared with playtesters

Now that the core mechanics were set up in a basic sense, I shared the above very basic test level with my housemates to start getting feedback as soon as possible. This included rudimentary functionality of killing enemies, receiving sight and glyphs, and using these to interact with a test puzzle pillar and heal when the player takes damage. The feedback I received was immensely valuable. I have detailed the most important comments, and my responses to them from a design point of view, below.

Player #1

"I think I got the basic principle. Movement was fun, killing demons is always a good time. The puzzle element I honestly found kind of annoying though, mostly because every time I made some move, I'd mess it up when a demon was attacking me."

In a very early iteration of the project, before even this basic prototype level, Sight was activated by pressing the E key, and players could interact with puzzle pieces (and originally obstacles to enable and disable them like Tears in *BioShock Infinite*) via a one-time press of the Right Mouse Button. I realised this wasn't going to work as this player pointed out that when demons swarmed the puzzle platform, the controls were getting their wires crossed and muddling the dynamic of using them both together. This was something I quickly rectified by mapping the Sight to the RMB instead, and changing the puzzle interaction to make it so that players could only interact with pillars while using Sight, bringing the design more in to line with what I outlined above.

Player #2

“The slow-mo thing kind of felt underwhelming”

Once Sight reached this point where it made sense from a *mechanical* point of view, I still had to overcome the hurdle of making it make sense to players. This player didn’t really ‘get’ what Sight was for because it didn’t **appear** to change anything other than time dilation. So, **I decided to use an effect** similar to Wraith Mode from *Shadow of Mordor*, a very cool smoky effect that emphasises that *this is a different playspace* layered on top of another, while being easy to dip in and out of. This involved making a post-process material and mapping it to an infinite volume in the level. The player’s reaction to activating Sight went from ‘Oh right’ to ‘Oh sweet!’, which is an excellent return on quite a small adjustment to the mechanic. This really highlights the importance of audio-visual feedback, and this goes double for helping drive home the function and effect of core mechanics.

Player #3

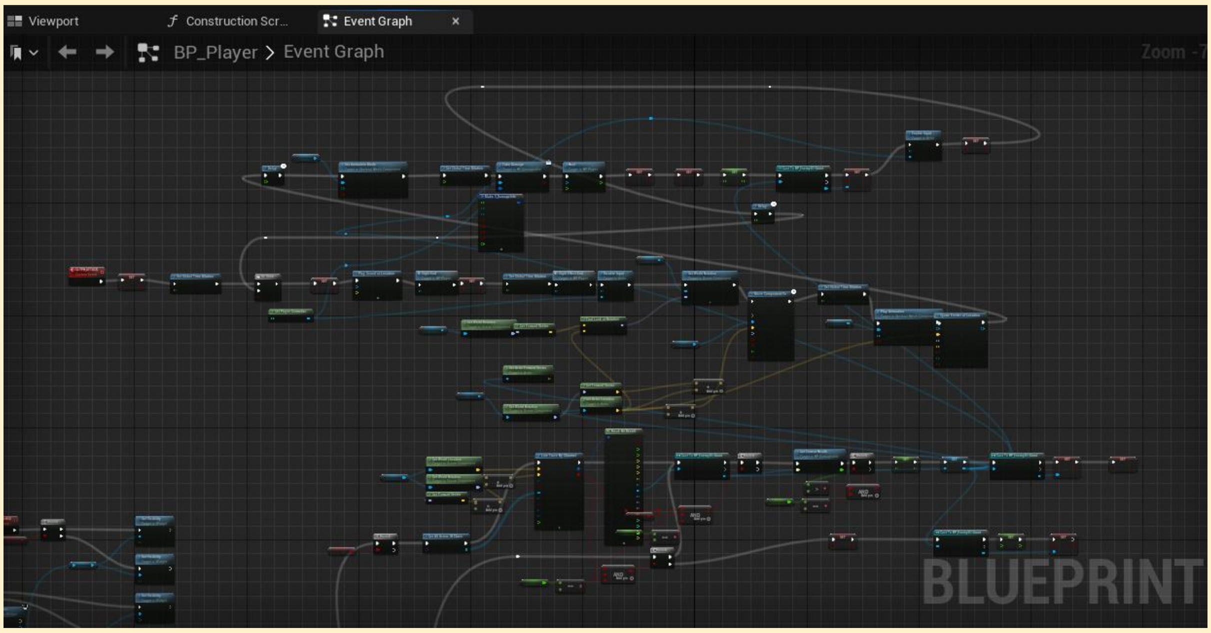
“The healing made me chuckle. Like I was running away from this pack of monsters just . But the thing is it almost didn’t matter if they hit me, because I could hit them back, kill them and immediately heal. So you’d be best off standing there, letting the demons surround you, entering a loop of just spamming the mouse buttons to heal as they do damage to you and do damage them back, wait for them all to die, see how many glyphs you have left and just use the time between waves to solve the puzzle. I don’t think that’s what you’re going for.”

Prototyping and Implementing the Execution Mechanic

This was a fair point. At this stage, healing was a simple button press to spend a glyph and receive health. There was no real process or cost to it other than that. After considering the need for a more *involved* way to receive health, I recalled my starting points: *DOOM* and *The Talos Principle*. In *DOOM*, players are encouraged to be aggressive and forward in their strategy because they leave themselves vulnerable without performing executions on downed enemies. Performing these executions rewards them with health. So, I decided to adapt this mechanic for my own purposes by combining it with Sight: players need to weigh the use of Sight and Glyphs to heal against using them to solve the puzzle.

The Execution move was conceived as a way to massively incentivise players to make use of Sight and Glyphs to heal: it’s a satisfying move to pull off as it comes with a sense of power and a positive feedback loop from having successfully targeted and completed the move. Now it needed to be implemented. There is no doubt that this was by far the most complex scripting task that I undertook in the course of this project.

After some experimentation and reading up on some techniques I had not used before, I ended up scripting a line trace to occur whenever the player was using Sight. If the trace hit an enemy, a small widget would be displayed above the targeted enemy’s head. If the Left Mouse Button was then pressed, the player would rapidly close in on the target and perform the execution. Getting the world location of the enemy and making the player close in to a relative point in front of wherever that enemy is, as well as showing and hiding the relevant widget for only one actor per class, was very challenging, but I got there in the end.



Blueprint for the Execution mechanic.

Week Two: Recap and Schedule Update

2 **w.c. 25.12.23** ***Production***	Make Backlog from design documentation that will inform prototyping in Unreal	0.1	0.2
	Procure assets	0.5	0.2
	Prototype base systems: movement, health, damage, dying, implement basic enemies, get a basic AI working, controls	2	4
	Prototype and implement Core Mechanic	3	4
	Bug fixing 1	1	0.5
	Week 2 Total	**6.6**	**8.9**

Prototyping the main systems in Unreal is quick and painless. Using the first-person game mode template, I was able to take the base blueprints and tweak their values. However, getting the blueprints working for the enemy AI, a damage system and bound events to death and damage functions took longer than expected.

On reflection, the reason this took longer than expected is because I rolled all of these different tasks into one, which gave the feeling of bloat and made me lose sight of the progress I needed to make on each one. It made the end goal for the week too diffuse and abstract, so in response to this, I decided to **set up a kanban board on Jira** for myself to follow for the next production week.

Level Design

When designing the levels for the final prototype, I had two main objectives:

- Construct a main battle arena to contain use cases of all my primary and secondary mechanics.
- Make a tutorial sequence that introduces each mechanic to the player in turn with a bite-sized use case.

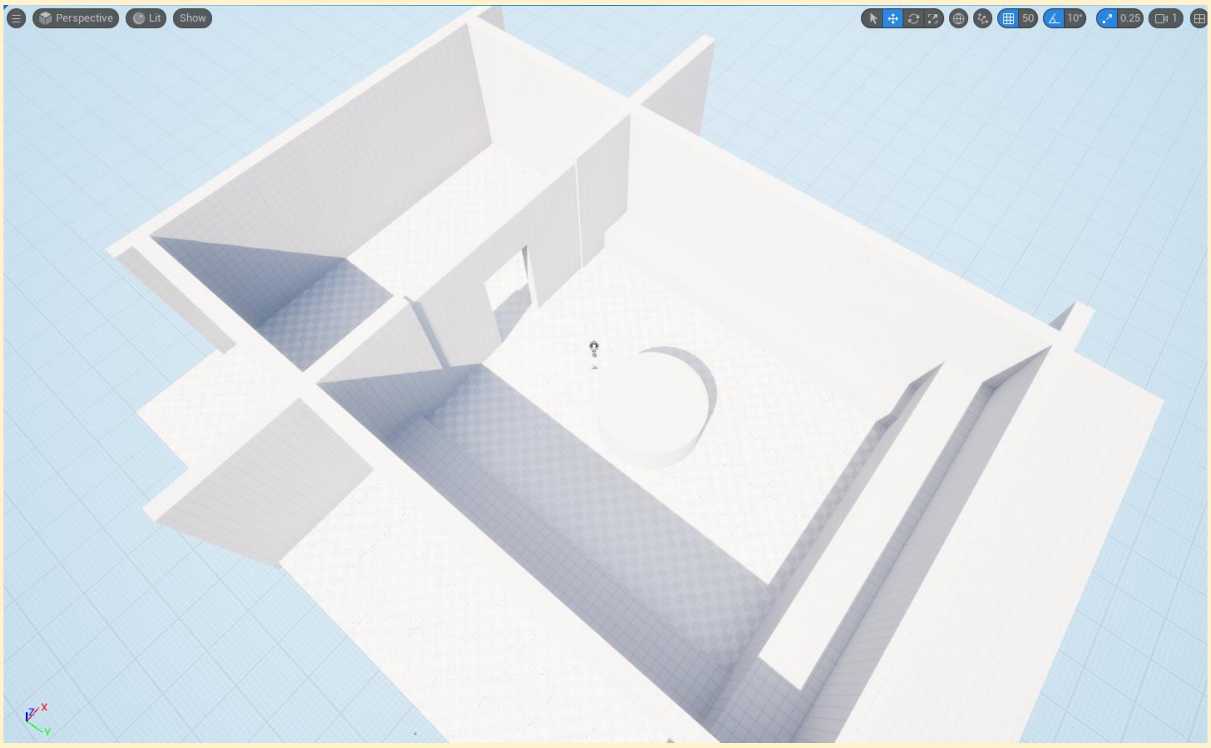
Given the project's nature as a small-scale prototype I knew it was vital to stay within scope and focus on the mechanics: it's all about the gameplay. So, I drew up a list of requirements for the main battle arena.

- Steep sides and lots of room for upward mobility to encourage flight and projectile combat. The execution move is much cooler when done from a high position, so this important aspect of the level design encourages players to engage in choices that will result in cooler, more emotive experiences in the game.
- A central platform to serve as a summoning zone, spawn area and indicator for which wave the round is currently on.
- Colonnades and nooks for the player to get trapped in when they go looking for refill statues.

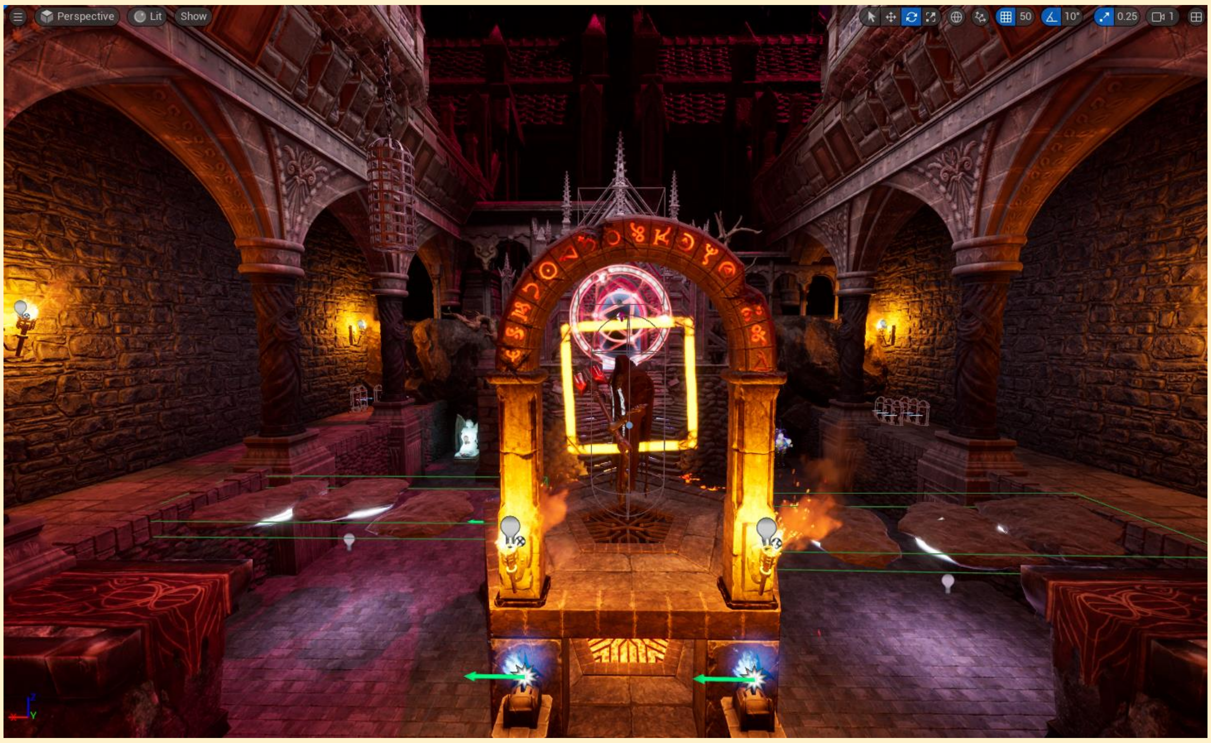
After sketching out some very rough paper designs marking out the key sightlines, I got to greyboxing. Having spent much more time on the core and secondary mechanic designs, I wanted to keep the level designs basic in order to emphasise the mechanic use-cases without relying too much on room-to-room traversal.

Keeping in mind the time limit and scope that I was keeping myself to, I made sure that the greybox clearly defined the arena boundaries and did not contain any open-ended doorways inviting un-planned expansions.

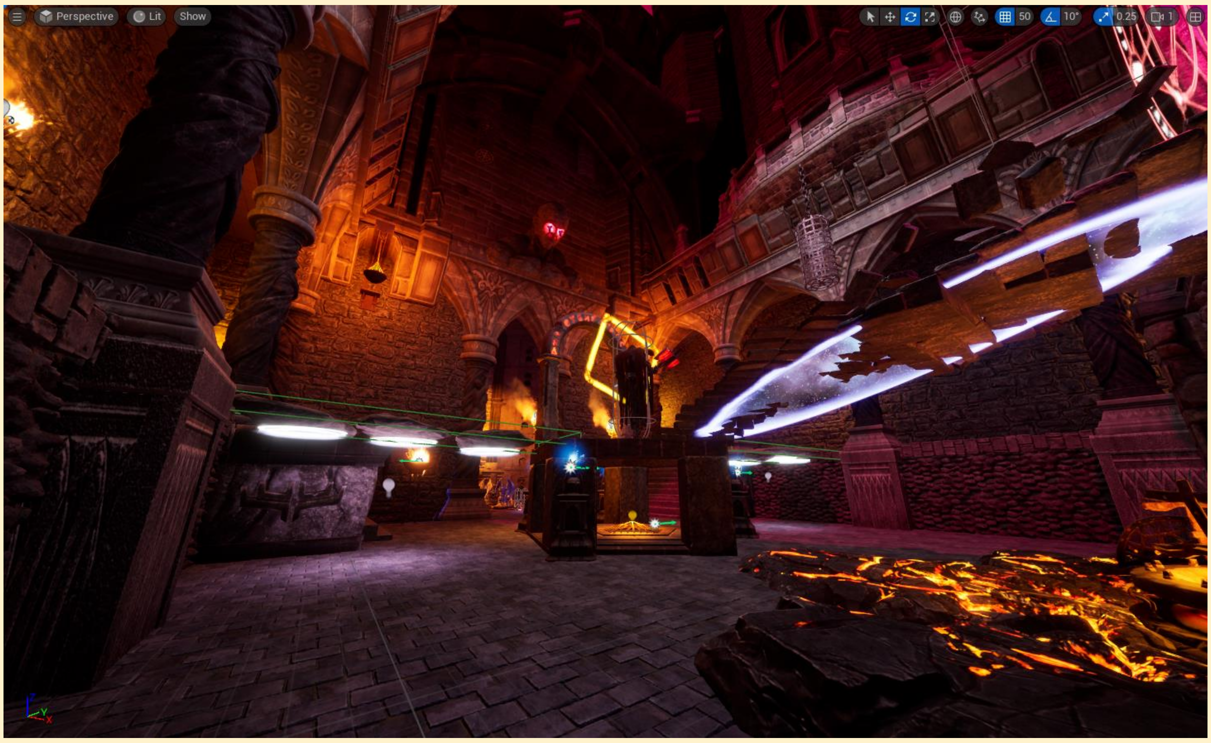
After this, I used a variety of free assets from the Epic Marketplace to set the tone of the game and make it visually appealing, consistent with the story and lore premise I had in mind, and added lighting to further compound the 'ghost-world' effect of the Sight mechanic as the post-process material interacts much better with fully lit environments.



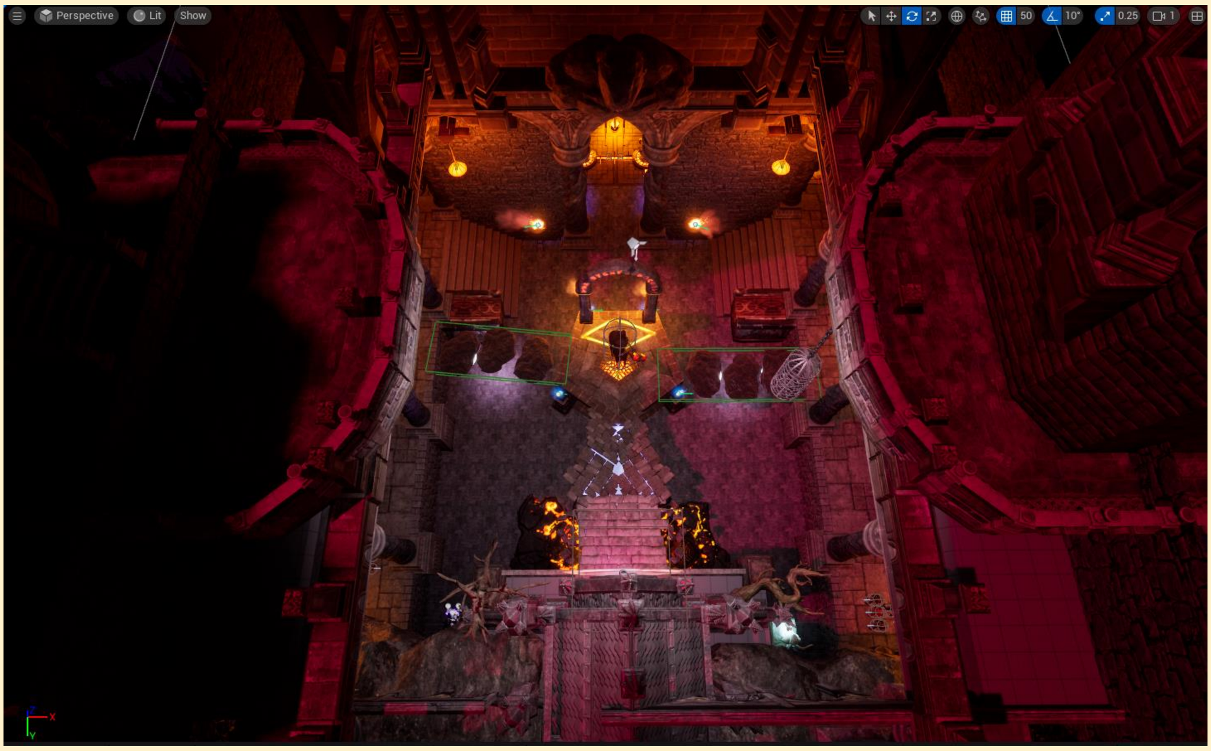
The early greybox of the main prototype level, featuring a puzzle chamber, a main hall, colonnades and a central platform.



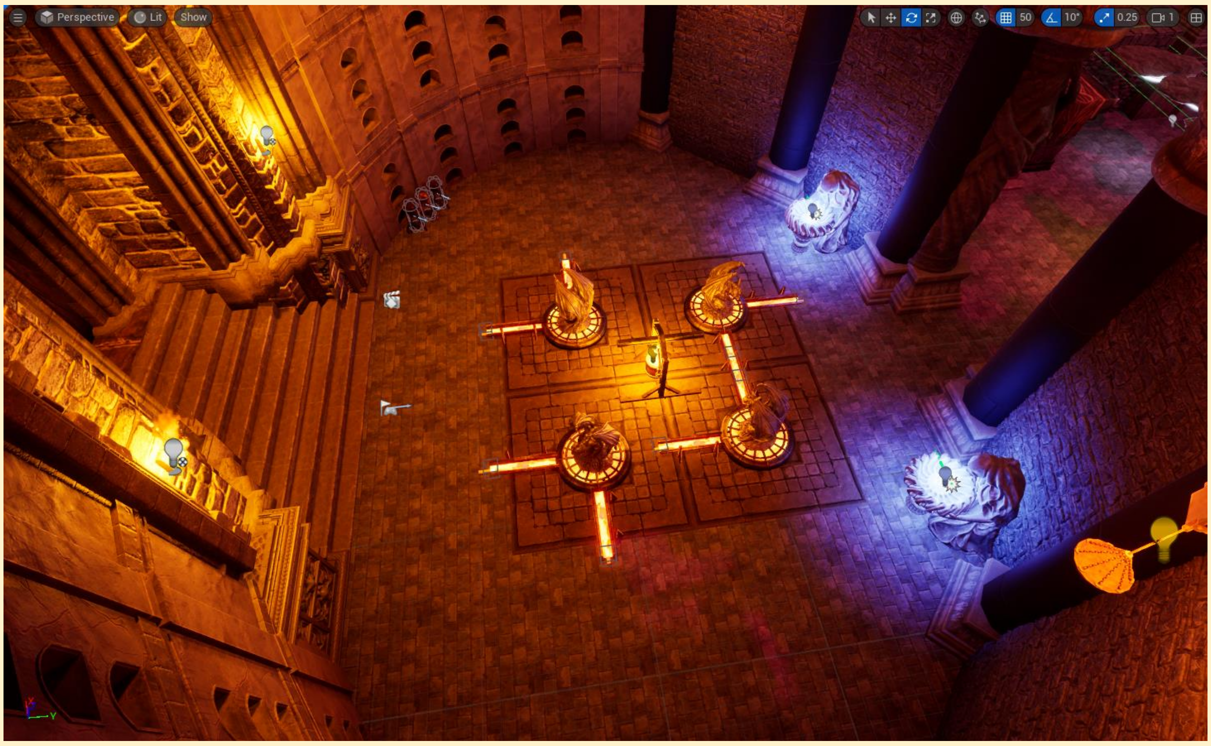
The finished prototype level's main summoning platform. The torches at the base of the platform show which wave the level is currently on, indicating how much time you have left to solve the puzzle before the False God spawns and ends the round.



The player's perspective from one of the statue nooks in the corner of the arena.

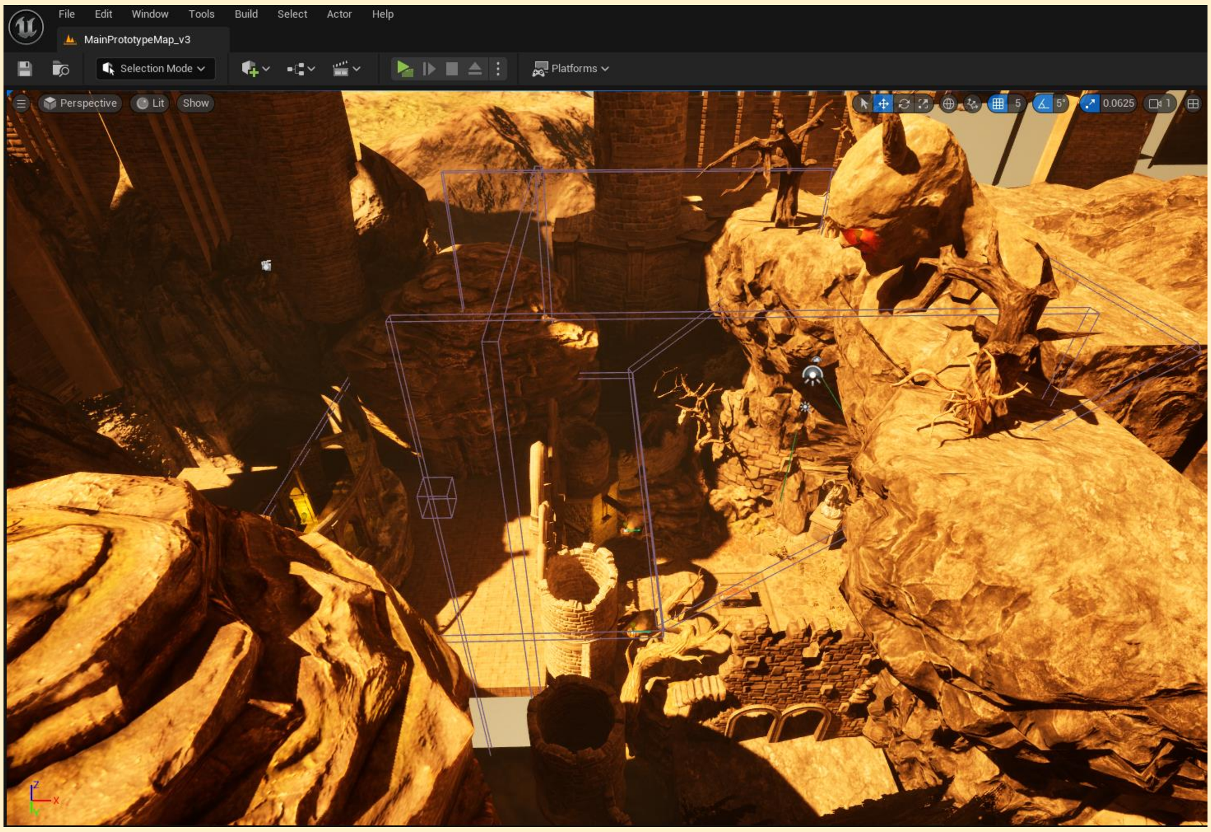


A view of the central chamber from above.

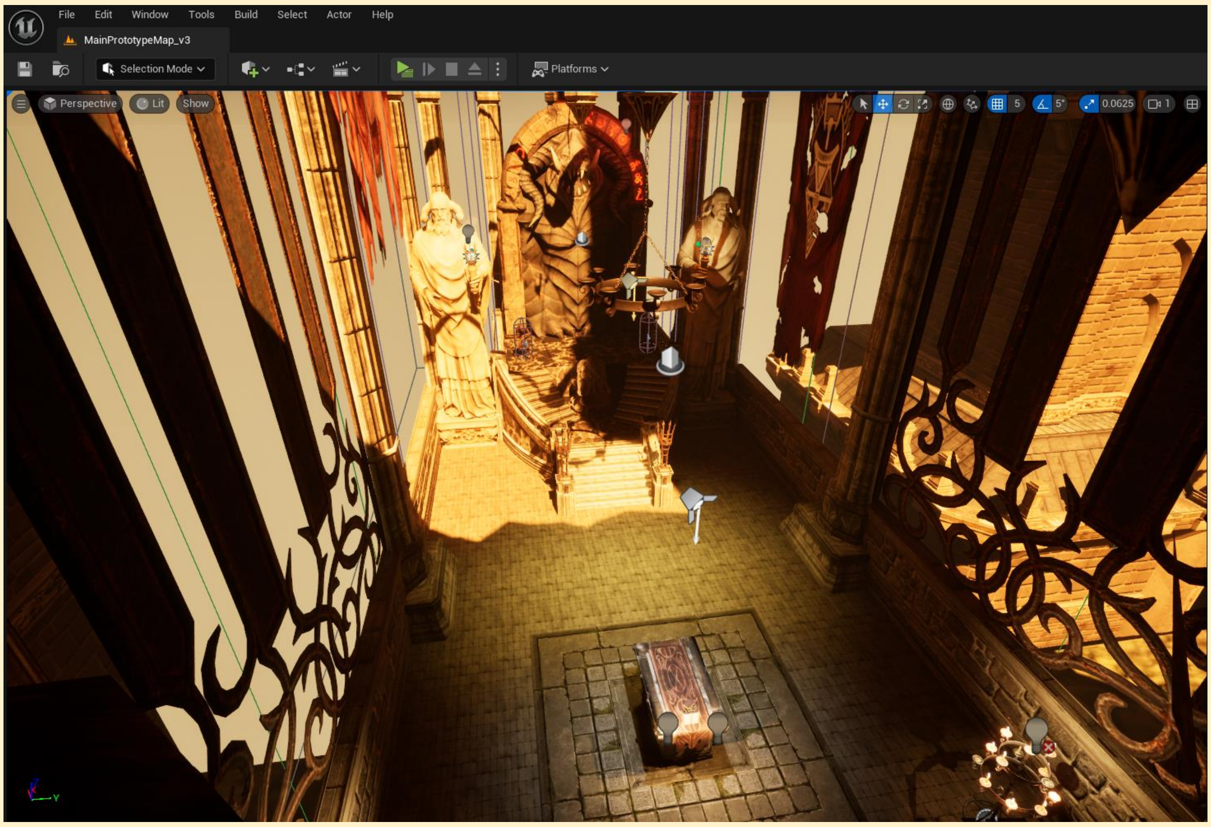


The basic statue puzzle grid in the prototype level.

After the main battle arena was designed, I set about designing a pathway-style tutorial section to lead the player into the main arena while introducing the main mechanics in turn.



Completed scenery for the tutorial area with collision boxes and invisible walls to prevent clipping or escaping the tutorial zone.



A further tutorial zone where players learn to shoot brimstone and use Sight.

Week Three: Recap and Schedule Update

This was an intense week that saw most of the prototype's content being implemented in-engine. I was able to save time on getting the secondary mechanics in place and working as intended, as I blew out the cobwebs on my blueprint abilities and started to become much more efficient at mocking up ideas and functionality quickly. However, I lost time by becoming slightly lost in the fun of scene dressing each level to make it look as good is it can.

I adjusted my initial plans for sound and music as I realised that they were not important to the functioning of the prototype and would not enrich the deliverables for this competition.

3 w.c. 01.01.24 ***Further Production***	Implement Secondary Mechanics	3	2
	Playtesting, Iteration	3	2.5
	Balancing	1	1.1
	Tutorial design	1	1
	Level Design	3	4
	Sound & Music	1	1.5
	Bug fixing 2	1	2
	Week 3 Total	**13**	**14.1**

Some stubborn bugs with the tutorial widget script also cost me an hour over my budget in ironing out casting between level and actor blueprints. In the end, I had to put together a complex system of variables to check the player's state and get their progress in the tutorial process.

Player Testing and Iteration: Round 2

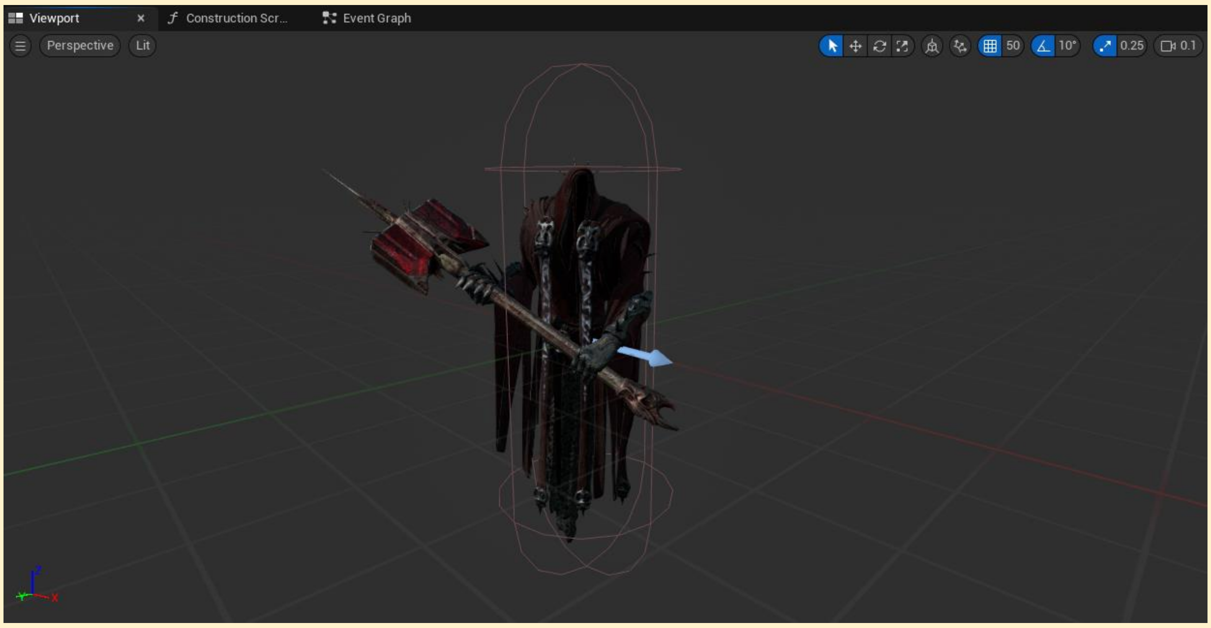
The second round of playtesting gave me the opportunity to further refine the core mechanics now that there were built levels and an established gameplay loop to test. From this round of playtesting, I gleaned two incredibly valuable points of feedback.

Player #1

“The combat is really fun! I kind of wanted to just... do that... like, I get the mix of puzzle and FPS but I am more drawn to the shooty bits.”

Implementing the False God (Round End) mechanic

This is the feedback that led to the implementation of the ‘grim reaper’ round end mechanic. I’d already drafted the idea in my head but had not yet put it into the prototype. It was listening to this feedback that reminded me why this was such an essential mechanic to include. It was simple to do. I got the free ‘Sevarog’ asset from Epic and programmed a straightforward blueprint to spawn at the end of the fifth wave, find the player and do a lethal amount of damage with a brutal hammer animation.



Implementing the False God, the entity that is summoned by the demons, finds the player and executes them if they've failed to solve the puzzle by the end of the fifth round.

Player #2

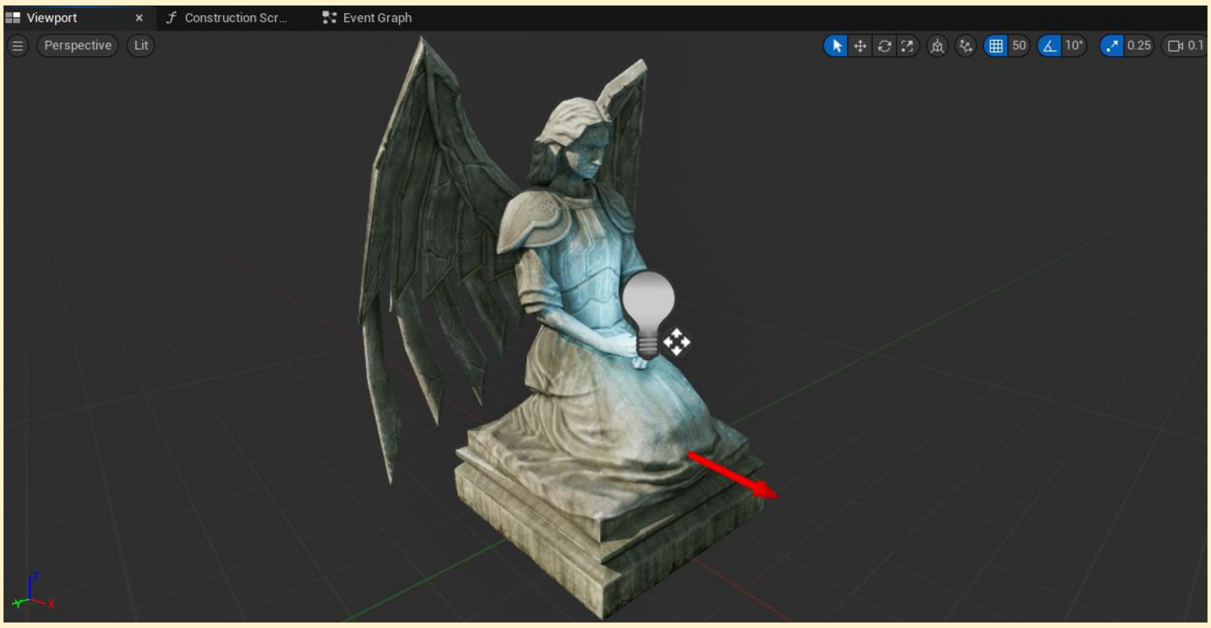
"There isn't much strategy to it, I'll be honest. I just constantly had the LMB pressed down and went for the good old spray and pray strategy. As it turns out I didn't need much praying. Mowed them down."

Implementing the brimstone cooldown

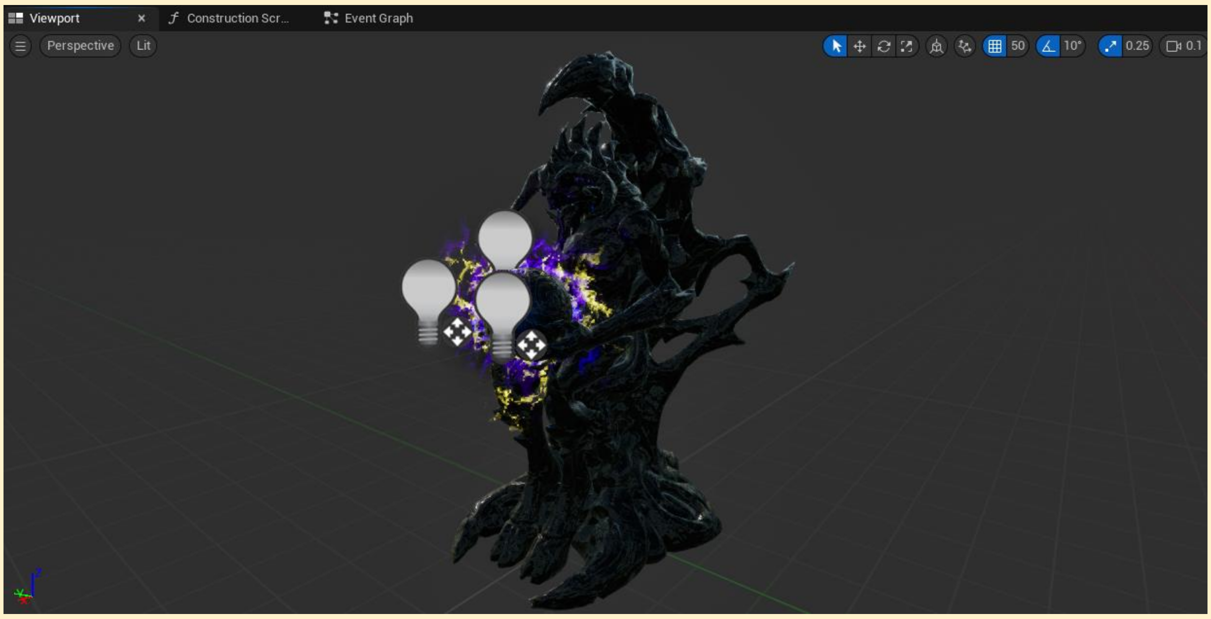
This was an indispensable piece of feedback. When I was playtesting the game myself in the editor I could feel that the brimstone mechanic was somehow incomplete, but couldn't put my finger on why. This brought the issue into sharp relief. It was too easy to just endlessly shoot fireballs, harvest a ridiculous amount of Glyphs, then spend the entire time in Sight mode solving the puzzle. So, I went into the player blueprint and defined a Boolean variable 'isShooting' and 'CanShoot' with a float value 'ShootCooldown' and set up the necessary increments and branch checks. I then quickly designed the UI elements below and seated the cooldown icon just above the crosshair. It all implemented very nicely, and is one of the aspects of the final game I think feels the most polished and like it would remain unchanged in further development.

Implementing further secondary mechanics

After playtesting and addressing the feedback, I decided to use the remaining time in the week to put together the blueprint for the consumable Sight and Glyph refill statues. These were static meshes with a light to signpost their location to players at a glance, and when activated, executed a blueprint function that called to a Sight/Glyph add function in the player blueprint, vanished and spawned a particle effect explosion to add a bit of polish. I was pleased with how these turned out considering they didn't take long.



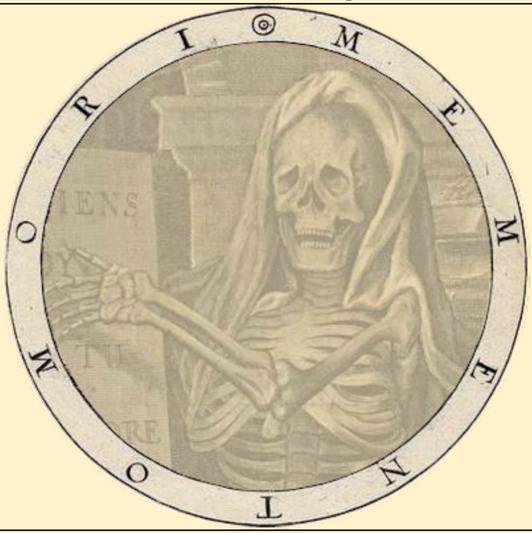
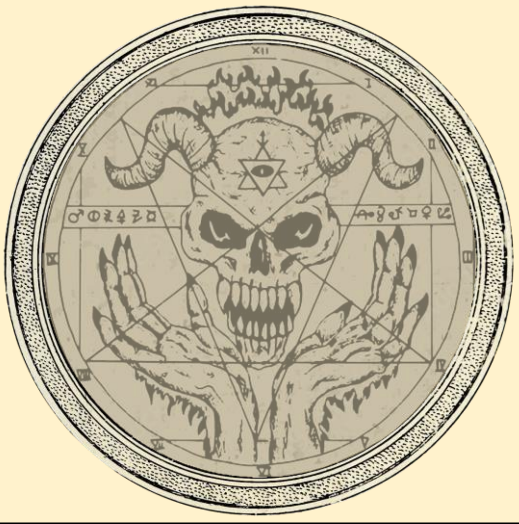
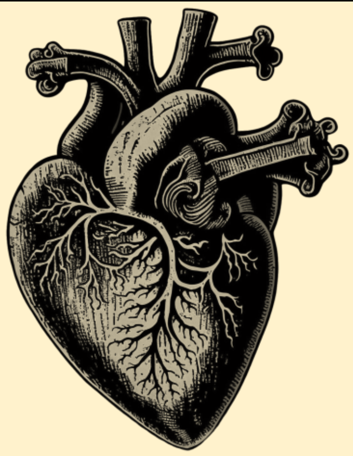
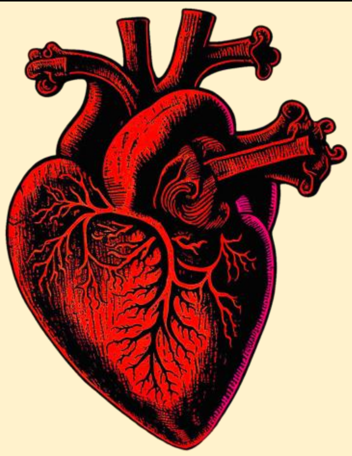
The Sight refill statue.

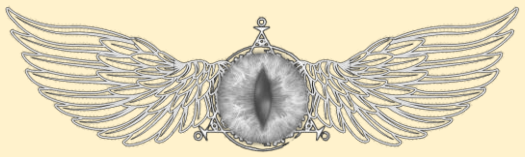
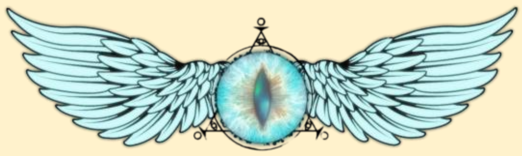
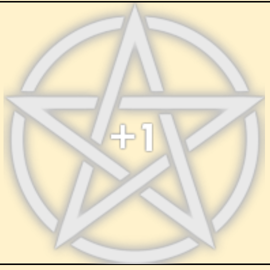
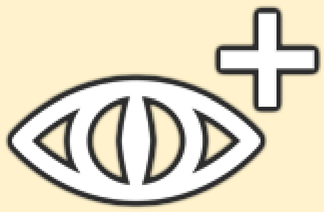
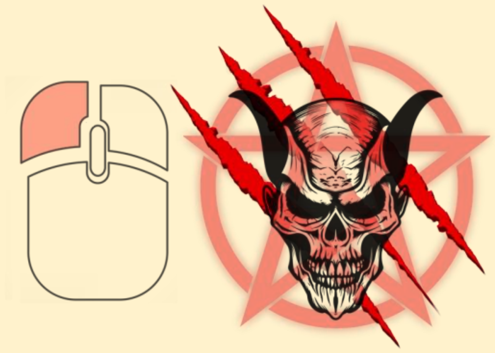
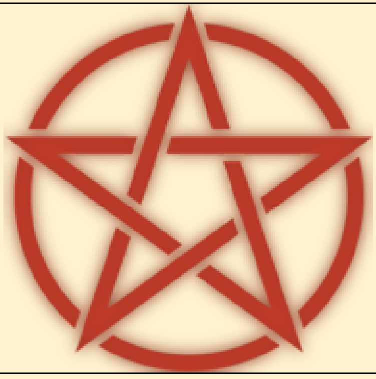
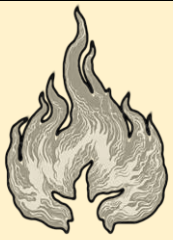
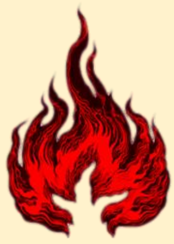
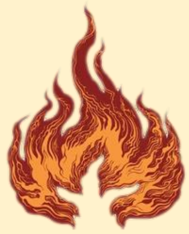
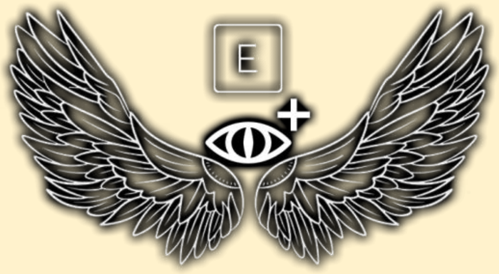
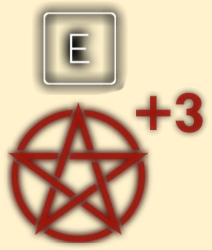


The Daemon statue that offers players a chance to get 3 Glyphs to help solve the puzzle.

UI Design

Making the HUD for *The Outcast God* was a fun opportunity to emphasise the grim and demonic aesthetic of the game and create graphics that are informative and nice to look at while also not taking too much attention away from the centre of the screen. I used Paint.NET to make PNG files for each element and added them all to a viewport widget which I implemented in the player blueprint inside Unreal. All graphics were exported as PNGs.

Health Meter Background	**Glyph Icon Background**
	
Health Meter Empty	**Health Meter Full**
	

Sight Meter Empty		**Sight Meter Full**
		
Glyph Add		**Sight Add**
		
Execution Prompt		**Glyph Spend**
		
Brimstone Empty	**Brimstone Full**	**Brimstone Overheat**
		
Sight Statue Refill Prompt		**Glyph Statue Refill Prompt**
		

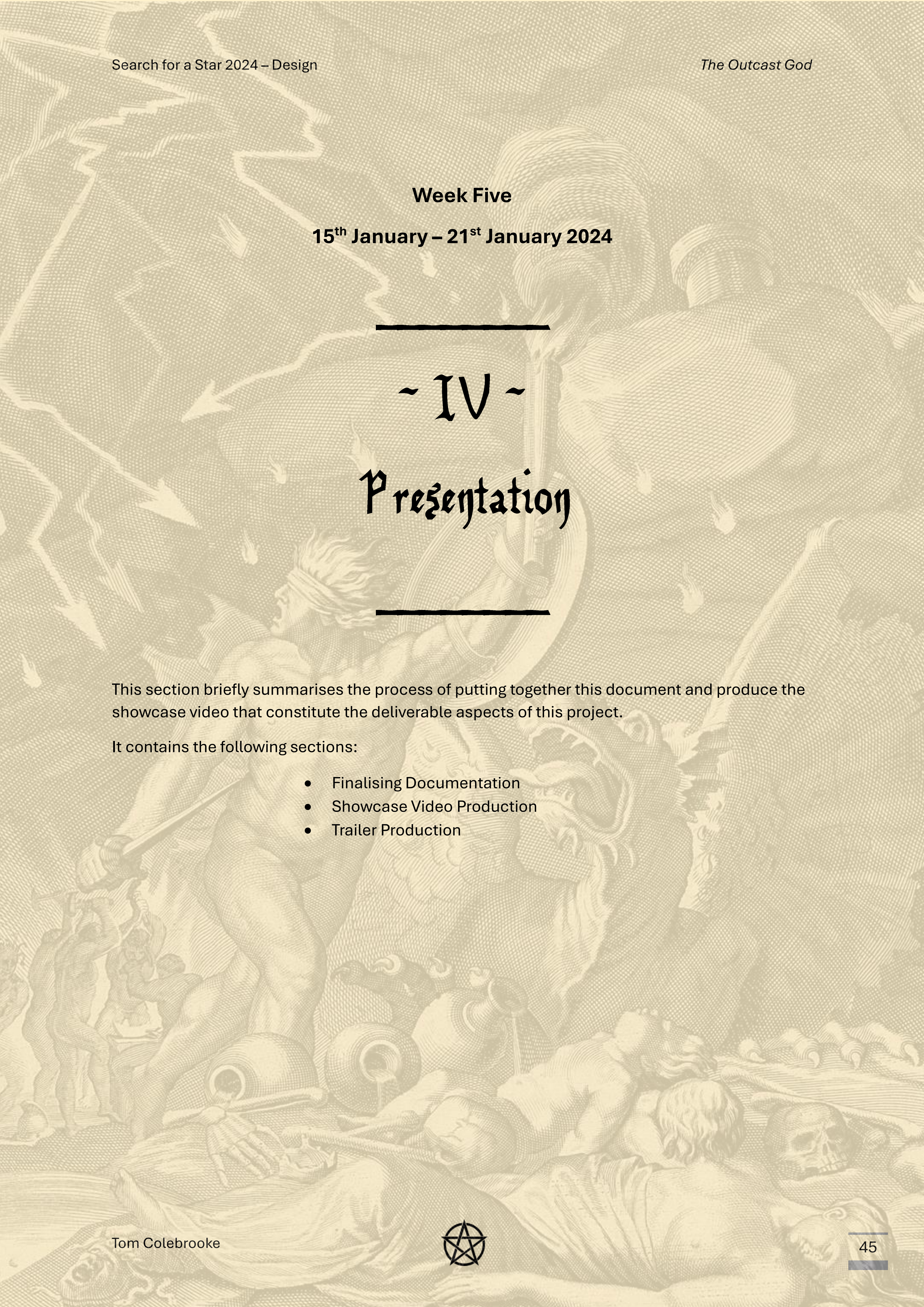
Once these graphics were displayed in the player's viewport, replacing the placeholder ones I had been using in playtesting up until this point, it totally transformed the gameplay experience. I was incredibly happy with how they enriched the gameplay experience by communicating information clearly and effectively with cool, clean animations and stylish visuals that fit perfectly with the game's aesthetic. This was well worth the hour spent on it, and massively improved the prototype, and also communicates the mechanics excellently which helps with the deliverable aspects of the competition – notably the video that contains gameplay footage.

Week Four: Recap and Schedule Update

This week I was able to devote time to polishing the core and secondary mechanics in response to playtesting feedback and ready the final prototype level for capturing gameplay footage.

4 **w.c. 08.01.24** ***Further Development***	Further playtesting	0.5	1
	Implement and iterate more secondary mechanics	1	1
	Implement feedback from playtesting	1	1
	Scene dressing	1	0
	Secondary mechanics (statues)	0.5	0.5
	Bug fixing	0.5	0.5
	Write bulk of GDD	2	2.5
	Implement HUD / UI	1	1
	Week 4 Total	**7.5**	**7.5**

I had ended up doing all of my scene dressing the previous week, so was able to reclaim that hour to come in on-budget this week, which I was very pleased with.



Week Five

15th January – 21st January 2024

~ IV ~

Presentation

This section briefly summarises the process of putting together this document and produce the showcase video that constitute the deliverable aspects of this project.

It contains the following sections:

- Finalising Documentation
- Showcase Video Production
- Trailer Production

Finalising Documentation & Producing Showcase Video

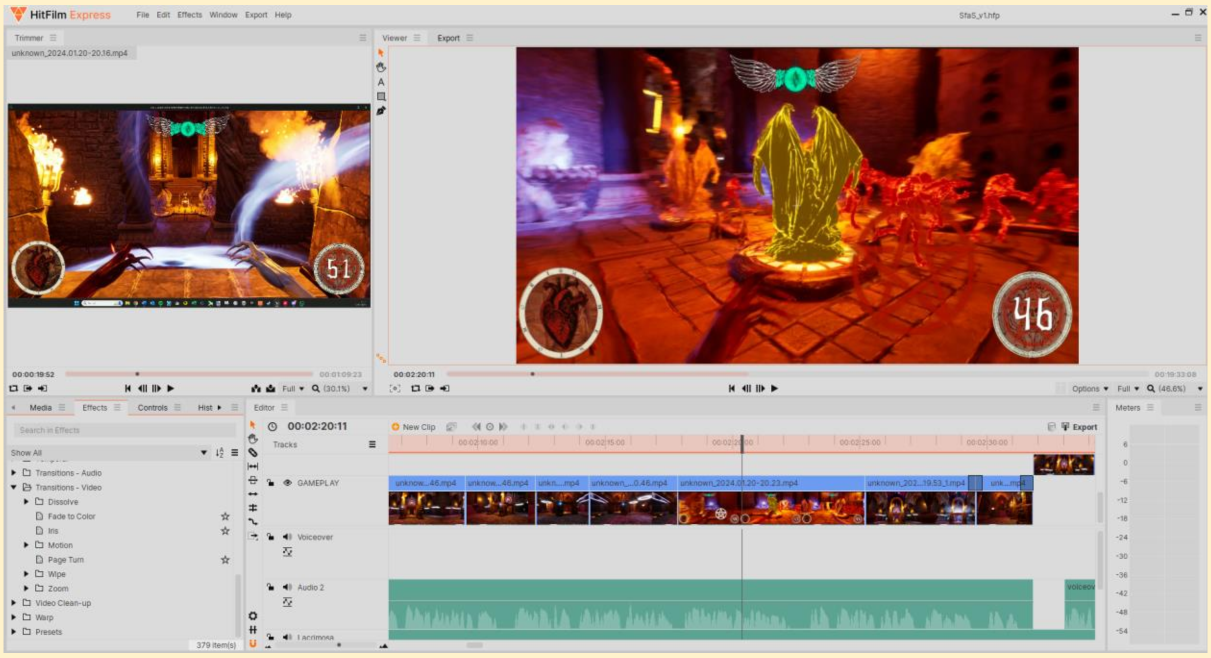
During the final week, I have put together this document and tried to show the creative development of my ideas, from initial conception to their function and use-cases in a fully built prototype in Unreal. Keeping a meticulous record of my working is something I've found challenging in the past, so I have tried to over-correct this time. I'm still slightly frustrated that I don't have more in-progress shots to draw from that show the precise development of every decision and considered filming time-lapses of putting together mechanics and levels, but in the end could not justify the extra time it would have taken to facilitate that process. It would also have been difficult to present that footage in the showcase video.

I have structured this document to address each of the requirements, suggestions, and assessment criteria present in the competition brief document, and have thoroughly recorded the step-by-step journey from a Word document to an exciting vertical slice of gameplay that I am immensely proud of. As such, I wanted to make sure that my passion for the project came through with the showcase video.

Producing the Video

Producing the showcase video involved taking each of my core and secondary mechanics in turn and summarising them in a direct and jargon-free way. I wanted the video to be swiftly paced, accessible, and clear in its direction with no extraneous or indulgent sequences. I ended up cutting an entire chapter on the theory behind the FPS and Puzzle mechanics that I felt was covered by this document and did not serve to clarify the function of those mechanics in the game. Rolling the short one-minute gameplay and story trailer into the same video was a decision I made in order to set the tone of the showcase video for assessors, introduce the concept and aesthetic of the world I'd created to give context to the mechanics, and to show off some of the music I had made for the game without it having to be muted or covered by my walkthrough voiceover.

The process of making the showcase video was technically straightforward. I used Radeon ReLive to capture gameplay footage, recorded the voiceover and layered it over a massively time stretched rendition of *Lacrimosa* to inject a bit of atmosphere. After layering on some clear and straightforward graphics to supplement my demonstration of key gameplay features, it was ready to render out.

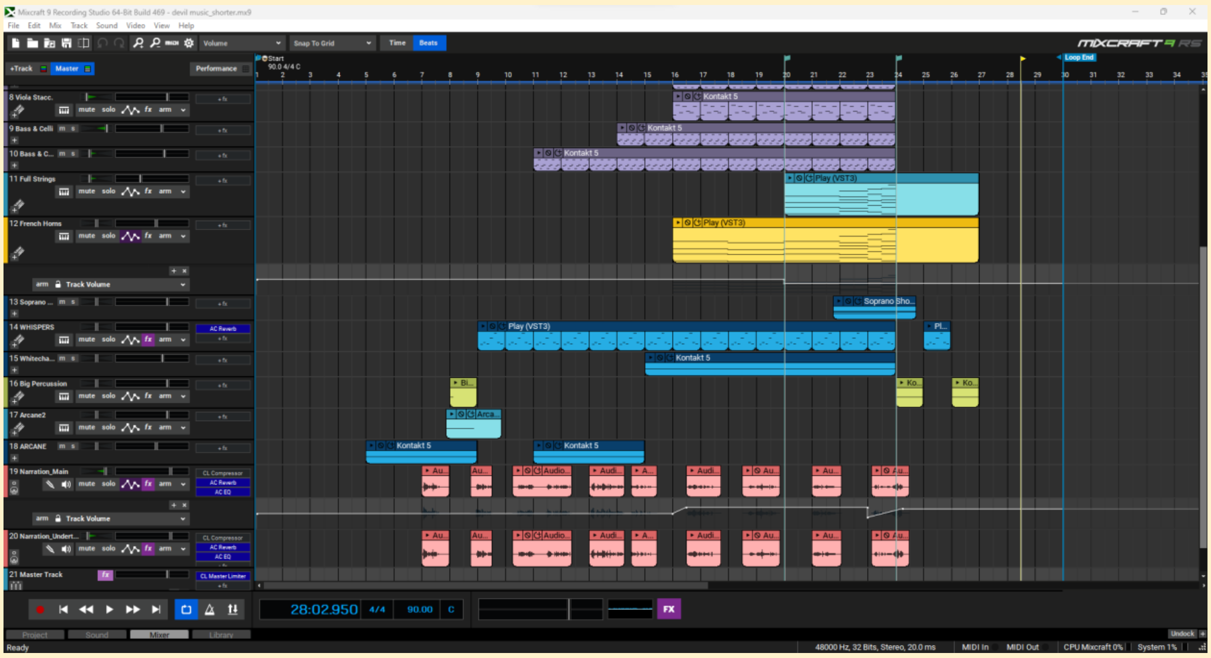


The timeline of the showcase video in HitFilm.

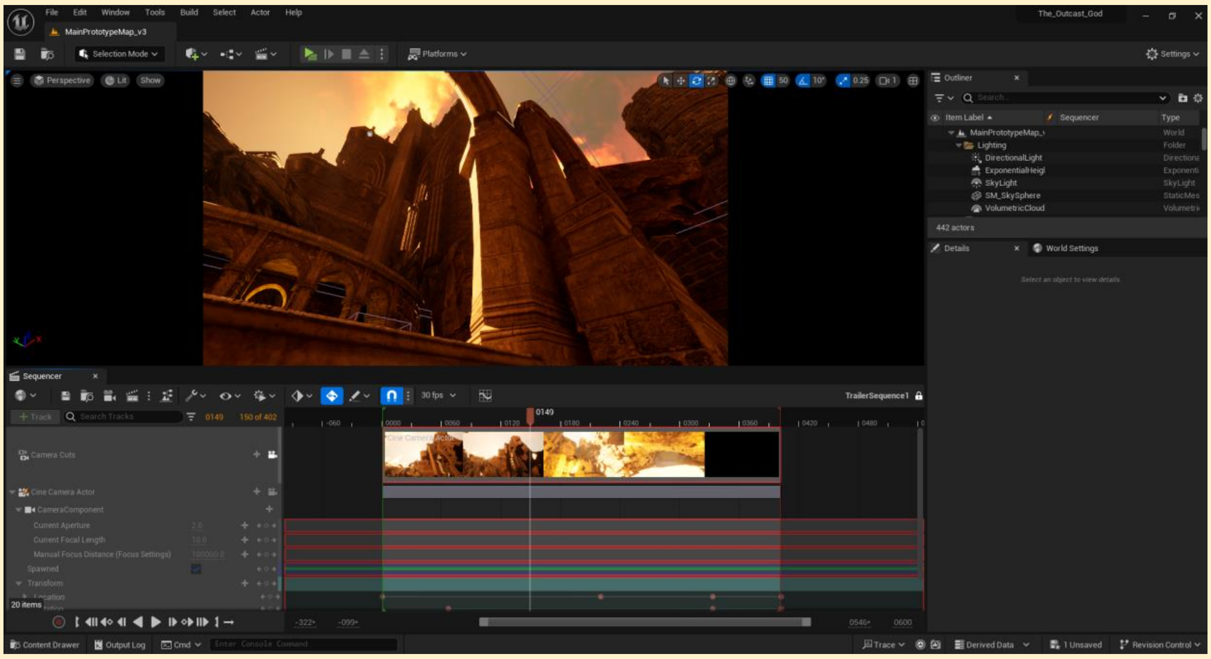
Producing the Trailer

To make the trailer, besides capturing gameplay footage using the methods described above, I also made some basic **Sequences** in Unreal, and exported these to video. For the sake of a very quick 10-20 minute process, these short panning sequences with a smooth camera added a huge amount of polish to the final trailer and gave a great visual overview of both the starting tutorial (the entrance to the cave) and the main battle arena.

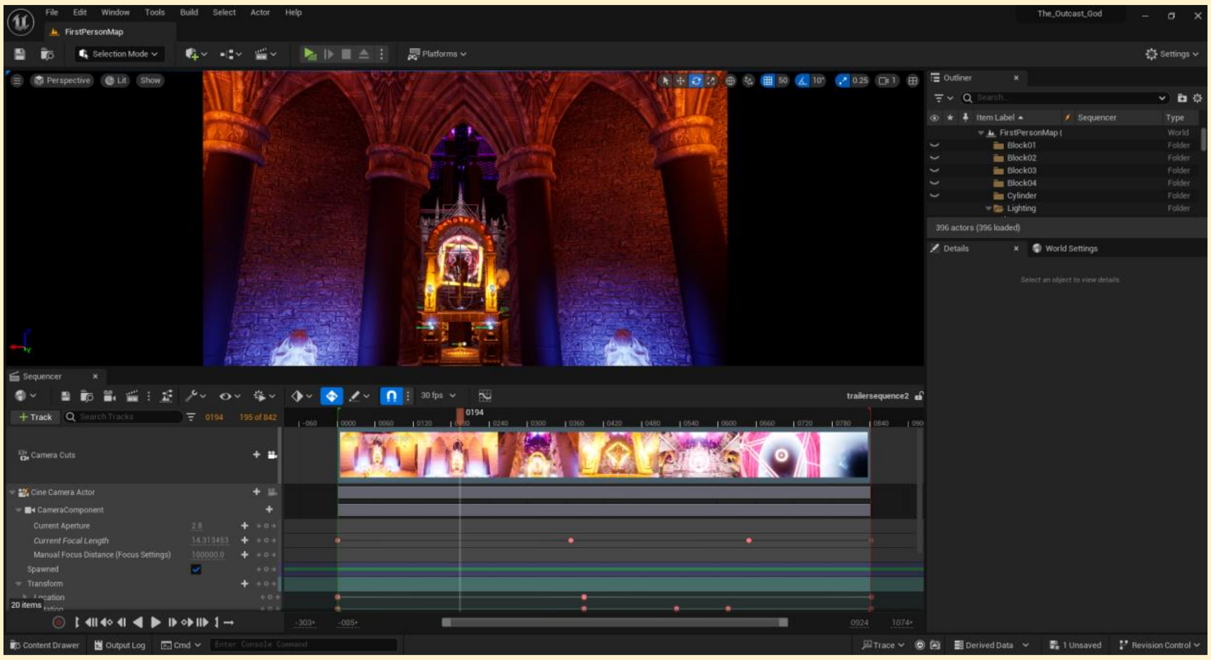
Including a short story trailer at the start of the showcase video also relates back to my design pillar of cohesion: just like the prototype I wanted to find ways to create flow, and to roll all of the various aspects of the project into as seamless a process as possible. So, I budgeted one minute at the start of the mechanic showcase video to introduce the premise and story concept of the game that contextualises the mechanics by expressing it artistically rather than just explaining the story. Underpinning every great trailer is a great score. I got to work on the music, with a booming gothic choir, staccato bass and celli and had lots of fun doing the demonic voiceover.



Producing the trailer music and voiceover in Mixcraft 9, a Digital Audio Workstation (DAW)



Making the camera sequence in the tutorial zone.



Making the camera sequence in the main battle arena for the story trailer.

Week Five (cont.)

15th January – 21st January 2024

~ V ~

Reflection

This section contains my evaluation and reflections on the process of participating in this competition, putting together my prototype and designs, and tallies my final schedule.

It contains the following sections:

- Did my design achieve its goals?
- Issues with the final product
- How I will continue to develop the prototype
- What did I learn?
- Final schedule review and evaluation

Reflections

I am incredibly pleased with the final prototype I have produced. It has been challenging: I spent most of the Christmas break away from my desk frustratedly trying to put together design solutions in my head to what seemed insurmountable problems. There were times when I considered throwing in the towel, but in the end, I stuck with it and have put together a solid, enjoyable, visually impressive prototype that I am proud to include in my portfolio, show off to friends and send off to studios.

Did I achieve my Design Goals?

1. Design an engaging and elegant gameplay loop in which combat and puzzle solving rely on each other in a First-Person fast-paced setting.

I believe I have achieved this goal. Sight and Glyphs, as two symbiotic mechanics, tie together combat and puzzle solving in a seamless gameplay loop that my playtesters enjoyed. Those moments, particularly those generated when all the different primary and secondary mechanics come together, like in the Execution move from above, generated a few 'wicked!' moments from playtesters that are priceless as a designer. A critique I would raise to my own design is that I think I lean *slightly* too much on the combat aspect of the fusion between genres. It's definitely a fast, hectic, FPS at its heart, but implements enough of the puzzle mechanics and features I outlined in the opening theory section to be considered a successful response to 'Two Become One', and that, I think, merits a tick in this box.

2. Build a prototype in Unreal containing a tutorial to teach the core and secondary mechanics followed by a short use-case arena to show off the dynamics of the game.

My prototype contains a complete tutorial introducing the mechanics to the player one-by-one in a one-way path to the main arena. The arena itself contains enough opportunity for players to experience a complete gameplay loop using the primary and secondary mechanics with their various benefits and drawbacks, and risk/

3. Playtest and iterate the demo to produce a product that leaves a viable template for a complete game.

This is a goal I believe was achieved solidly. There is a lot of interesting potential for further growth and development here in terms of iterating ideas for more deeply nuanced and complex use-cases in later levels that build on the mechanics I've introduced in the prototype. Overall, reflecting closely on all three of these goals, I find myself being very proud and satisfied that I **stayed within scope** for this project, producing working prototypes of all my initial designs, having refined them based on persistent feedback and iteration rounds.

Problems with the final product

There are some blueprint bugs that I need to iron out before I publish a publicly -available playable build, mostly relating to the execution mechanic (finding the world location of the target), and the enemy AI when firing projectiles. I would also like to implement accessibility support for colour blindness with the highlight colours – currently Red for enemies and Yellow for puzzle pillars when using Sight – this would have to be adjustable.

What have you learned?

I was able to hugely improve on my abilities using Unreal, using things like sequences, multi-gates, interfaces, widgets and more for the first time while greatly improving my literacy in Blueprints. If nothing else, this project has equipped me with the tools to produce much better prototypes in Unreal than before, which is an invaluable skillset as I build my portfolio.

It was also my first time actively blending genres in this way. I massively enjoyed the symbiotic aspects in games like *Brothers: A Tale of Two Sons* and *It Takes Two*, for example, and now I feel I've gained a deeper understanding of concepts like concentric gameplay loops that I can take forward to new projects and help me develop this one into something more fully-fledged.

Week Five: Planning Review and Evaluation

Below is the final tally of time spent on each task I planned out at the beginning of the project.

Week	**Task**	**Time Budgeted (hrs)**	**Time Actual (hrs)**
1 **w.c. 18.12.23** ***Pre-Production & Planning***	Draft a premise and story concept informed by the brief, goals and pillars established	0.5	0.6
	Draft Design Goals	0.2	0.3
	Draft Design Pillars	0.2	0.3
	Write up challenges and risks	0.3	0.3
	Put together a project plan	0.5	0.5
	Block out entire documentation structure	0.5	0.5
	Conceptualise Core Mechanic	2	2.2
	Conceptualise Secondary Mechanics	1	0.9
	Draft Core Gameplay Loop	1	0.5
	Reflect and plan production next steps	0.5	0.5
	Week 1 Total	**6.7**	**6.6**
2 **w.c. 25.12.23** ***Production***	Make Backlog from design documentation that will inform prototyping in Unreal	0.1	0.2
	Procure assets	0.5	0.2
	Prototype base systems: movement, health, damage, dying, implement basic enemies, get a basic AI working, controls	2	4
	Prototype and implement Core Mechanic	3	4
	HUD / UI	1	1
	Bug fixing 1	1	0.5
	Week 2 Total	**7.6**	**9.9**
3 **w.c. 01.01.24** ***Further Production***	Implement Secondary Mechanics	3	2
	Playtesting, Iteration	3	2.5
	Balancing	1	1.1
	Tutorial design	1	1
	Level Design	3	4
	Sound & Music	1	1.5
	Bug fixing 2	1	2
	Week 3 Total	**13**	**14.1**
4 **w.c. 08.01.24** ***Further Development***	Further playtesting	0.5	1
	Implement and iterate more secondary mechanics	1	1
	Implement feedback from playtesting	1	1
	Scene dressing	1	0
	Secondary mechanics (statues)	0.5	0.5
	Bug fixing	0.5	0.5
	Write bulk of GDD	2	2.5
	Implement HUD / UI	1	1

	Week 4 Total	**7.5**	**7.5**
5 **w.c. 15.01.24** ***Presentation***	Write Elevator Pitch, Outline of Concept and USPs	0.5	0.5
	Write video script	1	1
	Cut together footage	0.5	1
	Record voiceover	0.5	0.3
	Write Reflections	0.5	0.5
	Write up 'Presentation' section	0.5	0.5
	Put together Appendices and References	0.2	0.2
	Final checks and submission	0.5	0.5
	Week 5 Total	**4.2**	**4.5**
		Budgeted Hours Total: **38**	**Actual Hours Total:** **42.6**

Overall, I am pleased that I managed to produce a piece that looks slick, high-quality and that I am proud of within a reasonable excess of time budget, and within the scope I established for myself at the start of the project. I managed to avoid getting drawn into blocking out multiple levels, writing a grand narrative or spending too much time on things like animations or particles – the starter content and freely available assets did the heavy lifting there, allowing me to effectively communicate my mechanics with minimal customisation or extra effort.

Looking at the final time sheet, I realise I **overbudgeted** time for small administrative tasks and **underbudgeted** time for key development processes, and this is a process I would change in future projects.

A Final Word

Thank you to Grads in Games, to the assessors, for facilitating this competition and taking time out of their schedules to make it happen. It's massively appreciated, and I've learnt a lot from taking part. I'm proud of what I've produced and hope to engage much more with the community this summer as we all continue to develop our prototypes!

Thank you for reading and engaging with my project.

Software Used:

- Unreal Engine 5.3
- Paint.NET
- yEd Graph Editor
- Mixcraft 9
- HitFilm Express
- Jira
- Radeon ReCapture

Assets Used:

All assets procured from Epic Marketplace

Free assets:

- ‘Dungeon Playable Demo’
- Soul: Cave
- Infinity Blade Fire Land
- Infinity Blade Grass Land
- Ithris Cemetery
- Medieval Dungeon
- Paragon: Gideon
- Paragon: Sevarog

Paid assets:

- Monster Hands
- Mummy-zombie withered