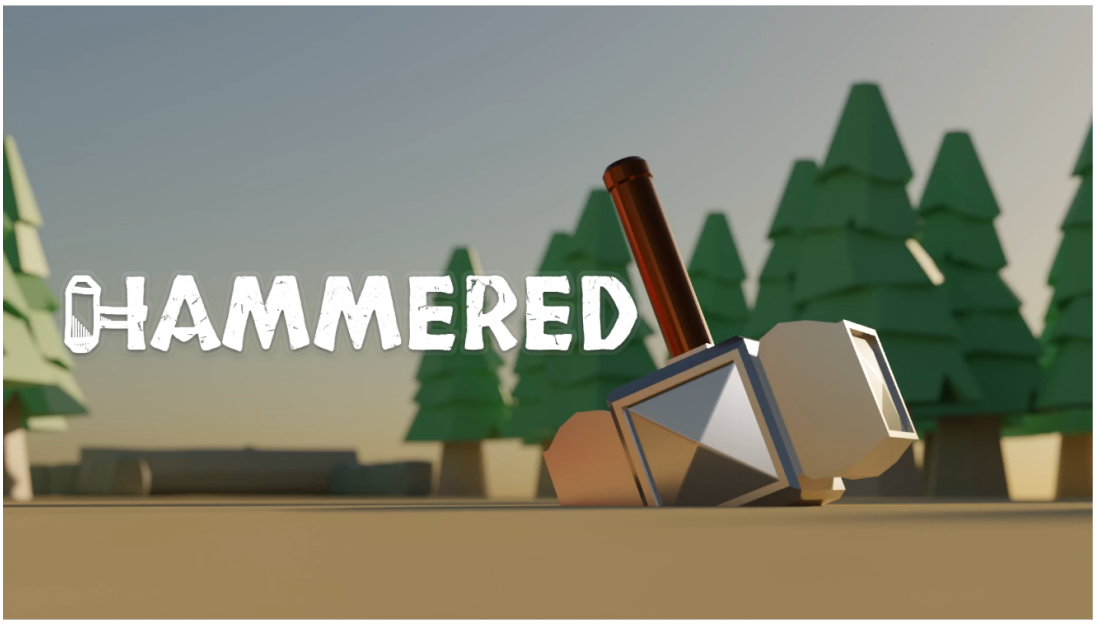
Game Programming Laboratory

HAMMERED

"Smash through puzzles with a flying hammer."



Yuto Takano - Producer, Programmer, Designer
Audrey Leong - Programmer, Sound Artist, Designer
Marie Jaillot - Programmer, Visual Artist, Designer
Andrew Dobis - Programmer, Visual Artist, Designer
Siddharth Menon - Programmer, QA Engineer, Sound Artist, Designer
Konstantinos Stavratis - Programmer, QA Engineer, Designer, Writer

Contents

1

Formal Project Proposal

1.1 Game Description

1.1.1 Overview

Our game is a single-player puzzle adventure game, centered around a lost hammer that belongs to Thor. The player will progress through island stages on a third-person 3D isometric map, in order to deliver the hammer back to him. Each stage contains puzzles that need to be solved using the hammer's magical ability: it flies back to the player when thrown away, destroying objects in the trajectory. There may be rocks blocking a path, there may be mysterious levers that can't be reached by hand, or there may be a maze unsolvable without destroying one wall somewhere.

We want the players to exhibit creativity in solving puzzles, akin to games such as "Portal"[1] or "Looking Grimm"[2]. There are no time limits nor explicit Game Overs, with the only thing that blocks progress being players stuck on a puzzle. Instead, players are rewarded upon completion of puzzles with visually and aurally satisfying effects, revealing of lore, and minor upgrades to the core mechanic.

The game's overarching backstory (about returning the hammer to Thor) will be simple so that we hope players will easily understand and attach themselves to emotionally.

[1] https://store.steampowered.com/app/400/Portal/
[2] https://goat-productions.itch.io/lookinggrimm

1.1.2 Background Story

Oh god. You wake up, head pounding as you squint at the piercing sun above you. It's gotta be at least midday, and you're probably late for work too. One look at the floor and a whiff of your clothes is all it takes to confirm last night's adventures. *Where's the nearest exit...* Oh wait. Island. Right. Instead, you trudge towards the coast, gauging the path of islands that lead back to the mainland before running straight into a giant hammer.

Figure 1.1: *Gee, thanks.*

Sighing, you stare at the hammer and give in. "Psst, here Mjolnir, good boy–" It flies straight at you, nearly knocking you over. So that's how it works... this will be fun :)

1.1.3 Design Decisions

Island Puzzles The game will be set on a collection of islands. Each island will have puzzles that need to be completed to reach where Thor is. The puzzles will require players to repeatedly use the core mechanic: a hammer that returns to the thrower. Using this, players will destroy certain environmental objects, such as trees or rocks, activate levers or cause other effects. Some of the puzzles will also require players to use the hammer in non-trivial ways, as illustrated in figure 1.2.

Completing each puzzle will reward players with story clues and satisfying visual effects (such as a filling progress bar). After having clearing all puzzles on an island, the game teaches the player a new way to use the hammer, which will be used in the subsequent islands.

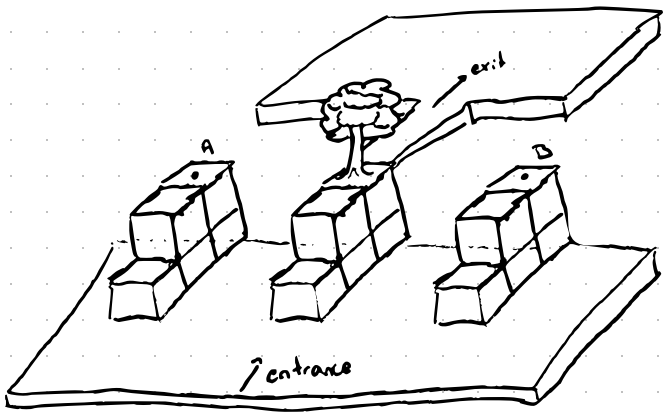


Figure 1.2: *Example of a very basic platforming puzzle that can only be solved by placing the hammer in point A and then calling from point B to destroy the tree blocking the exit.*

Hammer Mechanics The core mechanic of our game is using Thor's hammer "Mjölnir". In Norse mythology, Thor's hammer return to the user from anywhere. Our mechanic is inspired by this, and is illustrated in figure 1.4.

The hammer is used to destroy environmental objects which are blocking the player's progression through the puzzle. Player can drop the hammer, move to another location, and call the hammer back to them. The hammer will then follow a natural path to the player, destroying any environmental object it intersects on its way back to the player. There are 3 main ways that the hammer can be used to destroy the surrounding environment:

- **Default:** The player drops the hammer at a position A, moves to a position B and calls the hammer to said position. In this case the hammer will take the shortest path that avoids all indestructible objects and destroys all destructible objects.
- **Throw & Return:** The player throws the hammer from a position A_1 to a point A_2. The hammer lands at that location. The player then moves to a position B, and calls it back. This is an extension of the Default behaviour to allow hammers to be placed in places unreachable by foot.
- **Player Defined Direction:** The player drops the hammer at a position A_1, then places anchors at new positions A_i before finally calling the hammer back from a position B. The hammer will subsequently traverse the terrain following the shortest path from one point to another in the order they were placed in.

The hammer path possibilities are summarized and illustrated in figure 1.3.

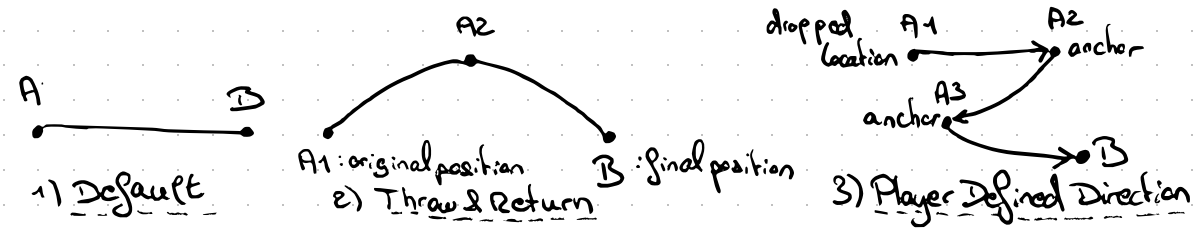


Figure 1.3: *Summary of the different hammer uses.*

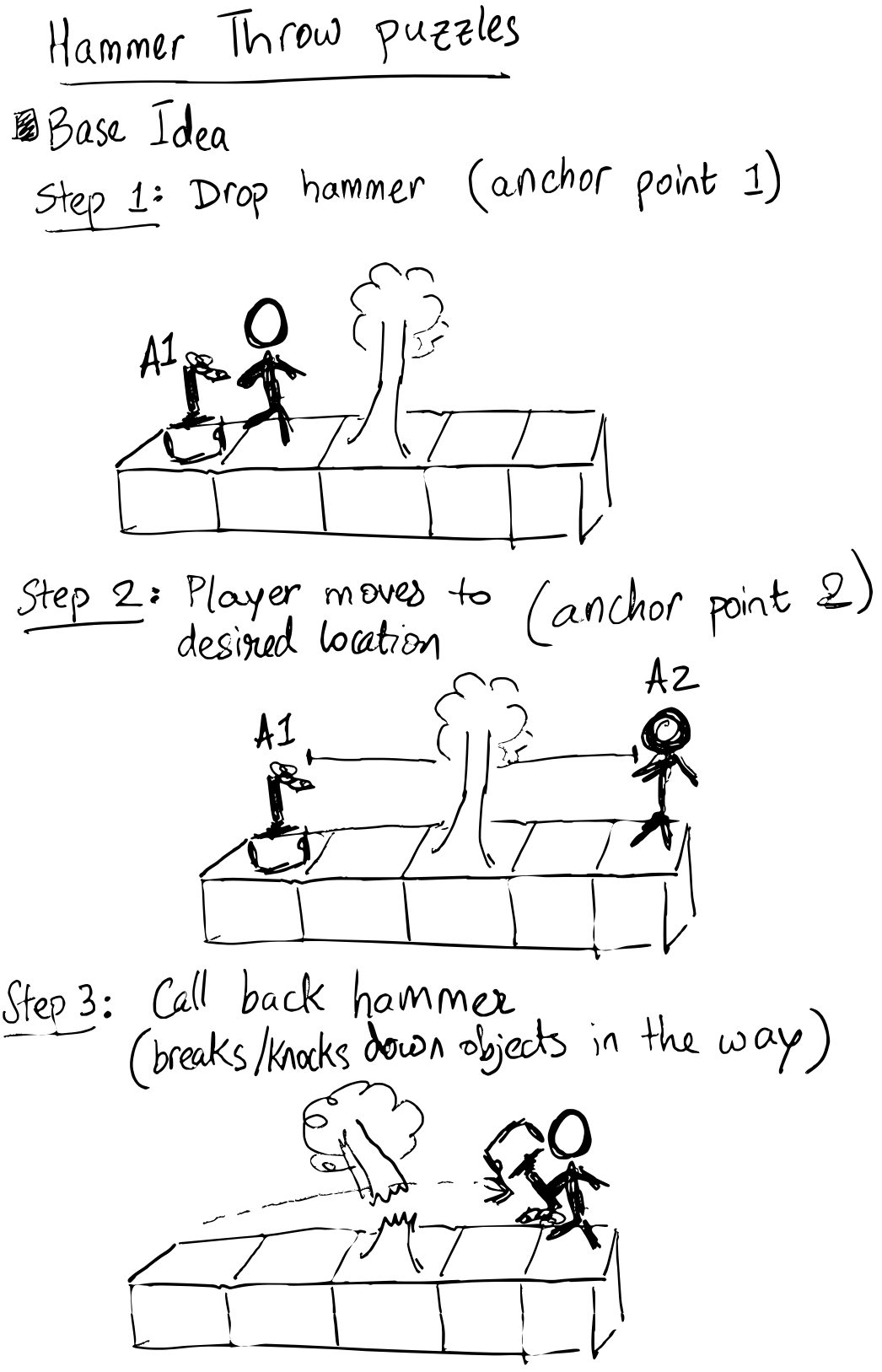


Figure 1.4: *Illustration of the core mechanic of our game.*

Visual Identity The game will take place as a cartoonish low-poly 3D environment with a distant isometric camera view, inspired by games like "TUNIC"[3] and "A Short Hike"[4].

We chose a low-poly look for the environment because it aids the players in understanding complex puzzles without driving them away with detailed graphics. The simple look also makes it easier to visually draw players' attention to the core hammer animations. Aiding this, we will use a saturated color palette and simple animations to further simplify the world.

We chose 3D instead of 2D, since it adds another dimension of depth to the puzzles, increasing the enjoyment. It also leads to puzzle designs that use perspective, such as secret objects hidden behind view.

We chose a distant third-person camera so that we can clearly show the hammer mechanic. First person cameras don't fit our theme, since it could be difficult for players to understand the hammer returning mechanic (due to perspective) or the mechanic could happen out-of-frame undesirably.

1.2 "Big Idea" Bullseye

The most important conceptual idea is the hammer mechanic, where you can drop or throw Thor's hammer and it will return to the player, destroying objects on the way.

The supporting impressive technical component is the hammer's path-finding trajectory, which takes a natural path in 3D space while avoiding all indestructible objects.

A visual for the bullseye is provided in figure 1.5.

1.3 Technical Achievement

The main technical achievement for our game is the algorithm to animate the hammer in a natural trajectory as it returns to the player. The hammer will follow the shortest path in 3D space that is natural and feels most intuitive to the player. The player may be able to dynamically change the trajectory mid-air by jumping or moving after calling for the hammer.

The trajectory calculation will be constrained to avoid any indestructible objects.

[3] https://store.steampowered.com/app/553420/TUNIC/
[4] https://store.steampowered.com/app/1055540/A_Short_Hike/

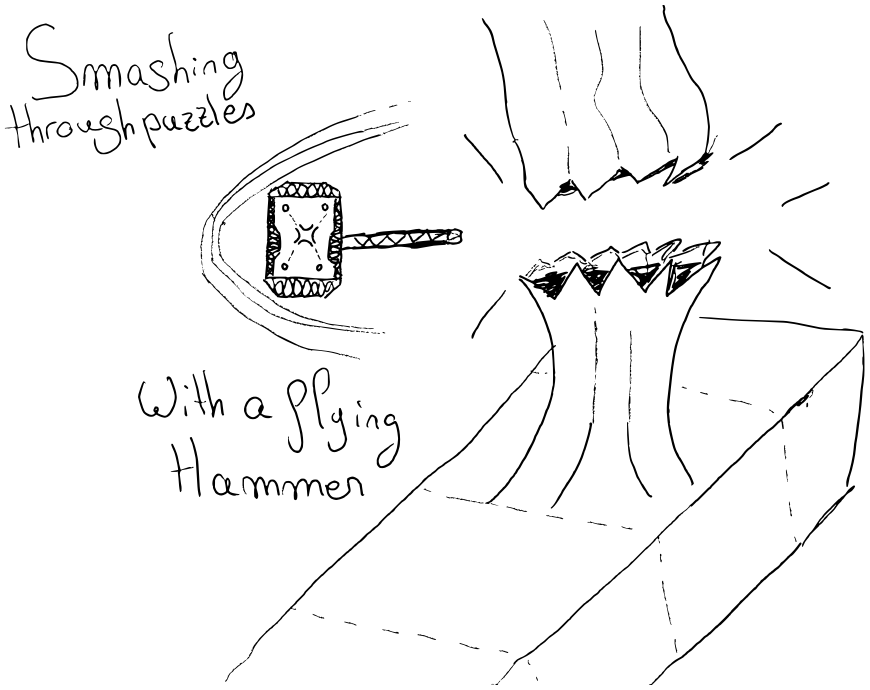


Figure 1.5: *The Bullseye*

1.4 Development Schedule

1.4.1 Layered Task Breakdown

Functional Minimum

- One island (blocks of ground and grass)
- Controllable character (not animated), camera follows character
- Hammer that can be picked up and dropped with no side effects

Low Target

- Island with dummy assets (trees, rocks, ...)
- Animated character
- Player can drop hammer and call it back in a straight line to break elements
- A few basic puzzles using the above mechanic
- Basic sound effects
- Basic story narration done as rewards for completing puzzles

Desired Target

- Player can throw the hammer and call it back (natural path finding that avoids indestructible obstacles)
- Background music
- Animations (such as object destruction, hammer throws, etc)
- More advanced puzzles
- Water around the island
- Tutorial stage

High Target

- Several islands with different landscape (snow/frozen island, fire island,...)
- Hammer can adopt properties from things it passes through in its trajectory, such as fire or ice – causing further effects on subsequent things it intersects with
- Sound effects for interactions and surroundings (wave sounds, ...)
- Puzzles with fire/ice mechanics
- Basic end game cinematic
- Game UI upon launch and save mechanics

Extras

- Fancy water graphics around the islands
- Beautiful shaders and complex dynamic lighting

1.4.2 Task List

See the timeline below for the task assignments and estimated hours each will take.

1.4.3 Timeline

			Week	21.2	28.2	7.3	14.3	21.3	28.3	4.4	12.4	18.4	25.4	2.5	9.5	16.5	23.5	30.5
			No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
			Deadline		Project Proposal (Rough)		Final Proposal	Physical Prototype	Playable Demo	Interim Report & Demos			Alpha Release & Demo		Playtest & Gobo Final		Slides & videos for Final Presentation		Due!
Target Level	Task	Responsibility	Est. Hours	0	6	10	23	47	38	32	31	32	19	14	13	9	4	0
Functional Minimum	Formal game proposal	All	8		6h	2h												
	Physical prototype	All	5			3h	2h											
	3D model of one island with dummy assets (trees, rocks)	Andrew, Marie	10				4h	5h										
	Character model (no animation)	Marie	7				2h	5h										
	Controllable character (no animation)	Sid	6				6h											
	Hammer that can be picked up and dropped	Konstantinos, Yuto	9			5h	4h											
Low Target	Adding textures and basic lighting to island	Andrew, Yuto	14				3h	7h	4h									
	Character animation and between the islands	Marie	6					4h	2h									
	Drop hammer, call it back, and break elements (basic path finding)	Sid, Konstantinos	14					8h	6h									
	3 basic puzzles (on 1 island, using 1 mechanic)	All, Yuto, Audrey	9					5h	4h									
	Basic story narration (hints at different levels)	Konstantinos	5					5h										
	Basic sound effects (character movement, collision, throwing, etc)	Audrey, Sid	8				2h	1h	3h	2h								
Desired Target	Player can throw hammer and call it back (path finding)	Konstantinos, Sid	18						6h	12h								
	Animating (hammer movement, collision, etc)	Marie, Andrew	12						4h	6h	2h							
	More advanced puzzles	All	12						3h	4h	5h							
	Additional graphics (water around the island, etc)	Marie	14							2h	6h	6h						
	Background Music	Audrey, Sid	14					6h	6h	2h								
	More detailed narration (especially introduction and ending)	Audrey, Yuto	7							2h	5h							
	Game tutorial/trailer	Yuto, Andrew, Konstant.	19								6h	5h				4h	4h	
High Target	Multiple islands, each with different landscapes	Marie	18							2h	4h	6h	6h					
	Hammer with fire/ice properties when passing through elements	Sid, Yuto	18								3h	6h	4h	5h				
	Sound effects for interactions	Audrey, Sid	13									5h	5h	3h				
	Puzzles with fire/ice mechanics	All	17									4h	4h	5h	4h			
	Game saving mechanic	Yuto	4											1h	3h			
	Basic end game cinematic	Marie, Andrew	11												6h	5h		
Extras	Fancy water rendering around the islands	?	0															
	Dynamic lighting (or cast by objects such as fire)	?	0															
	Player can move while the hammer is returning (dynamic pathfind)	?	0															

Figure 1.6: *Task List + Timeline*

1.5 Assessment

The strength of our game will be its simple yet effective core mechanic. "A hammer that returns to the player" is a concept that's very easy for people to understand, which leaves the mind open to creatively solving puzzles and absorbing the lore.

We want to market our game towards a wide target audience, from casual to hardcore gamers alike. This will be done through the simple puzzles where the childlike creativity and the ability to think outside the box is important, and through more complex puzzles that might need specific timing or player control skills. Our game will have intuitive controls for movements, and expect to have minimal or no HUDs constantly on display – these efforts will hopefully lower the barrier of entry to our game.

Players will repeatedly perform the core hammer-throwing mechanic, so this must be designed with great care. The game will have a tutorial to teach players, as well as very simple levels in the beginning to train their muscle memory. We also aim to reduce repetitiveness of the mechanic by providing satisfactory visual candy, such as a swift animated movement of the hammer or effectful destruction of the environment.

Our game design will encourage players to think creatively to solve puzzles. We impose no strict time limits or social pressure through multiplayer – we want to give the player to think as much as they want. We will therefore consider our design to be a success if it manages to invoke some feelings of self-satisfaction when complex puzzles have been solved and the player unlocks rewards or lore.

2

Prototype

Since our game's core player experience is to do with solving puzzles, we decided to design our physical prototype around puzzles as well. We decided on a play style where multiple Game Masters design a puzzle on a rearrangable board, and challenge players to solve it with the least moves. This allows us to increase the replay-ability of the physical prototype, making it fun to play, but it also made the prototype a perfect real-life level editor for us to design various puzzles and check their difficulty.

2.1 Initial Prototype Setup

Puzzle Design We decided to build a flat game (with no y-axis movement) in the prototype to simplify asset creation, and to honor the idea that the board can be rearranged at will. Any physical elevation would need to be subdivided for it to be rearranged into any shape, which would lead to a messy board and unstable ground to place things on top. It was also physically impossible to create floating islands or any complex elevated mechanic anyway, so we stuck to what was possible.

We also stuck to the first hammer mechanic (drop at current position and later call back) as to reduce the amount of ambiguity in hammer motion. We expect the other hammer mechanics (throw and recall) to be learnt intuitively when in a 3D virtual space, but we omitted those from the board game since we couldn't be sure if it would be understood by players.

Our initial puzzle idea is shown in figure 2.1. This puzzle included many of our main elements, such as the tree that can be knocked down to be used as a bridge, the keys necessary to unlock doors, and both breakable and unbreakable walls.

Assets

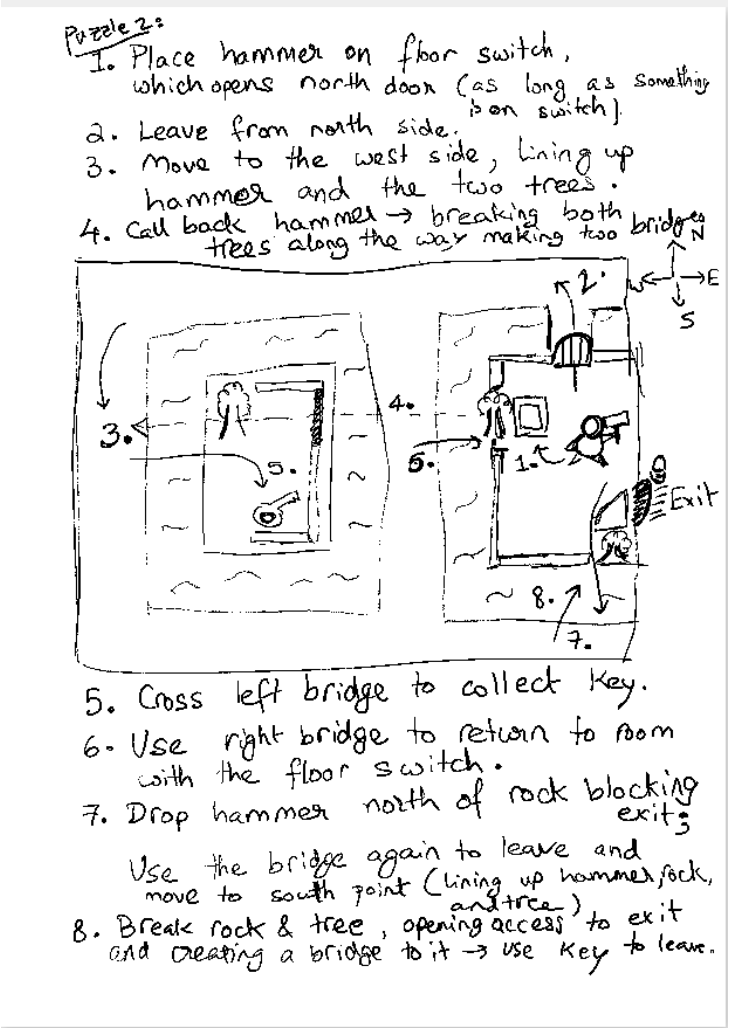


Figure 2.1: *Initial Prototype Puzzle Design*

Board

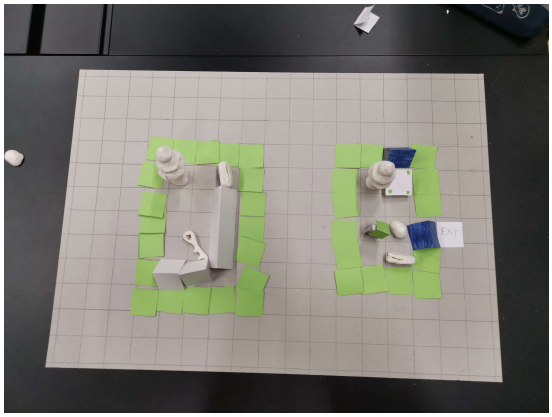
We created a grid on the board of our game. We introduced a grid instead of fluid movement for several reasons. First, it helped to clarify the paths the player and hammer can take; for example we could make the hammer only be allowed to fly in horizontal, vertical, or $45\,\mathrm{deg}$ diagonal paths. Second, it added a visual predictability to interactions (e.g. a tree will always fall in one of the 8 directions), easing the decisions the Game Master had to take to prevent unintended bugs. Finally, it was useful in finding the asset proportions required in the video game: e.g. if we wanted a tree to create a bridge over a chasm, it needed to be "at least 3 tiles long".

Props

We built objects that could be interacted with the hammer out of clay, such as trees, rocks, and breakable walls. Objects that were unbreakable were created out of paper. This provided a visual distinction between items that were "important" to gameplay and those that were not, removing the player's need to perform frustrating trial-and-error, instead allow them to focus on the puzzle.

Figure 2.2: *Assets Overview*

With the above initial idea and assets, the puzzle shown in figure 2.3(a) was built.



(a) The board containing the initial puzzle

(b) The process of designing the puzzle

Figure 2.3: *The initial puzzle idea, realised*

To introduce a small competitive aspect to the board game in order to compensate for the lack of multiple levels, we came up with the notion of "actions" that the players can take: move to any reachable location, drop the hammer, or pull back the hammer. The Game Master counted how many actions it took the player to solve the game, and encouraged players to solve them with the least amount of actions.

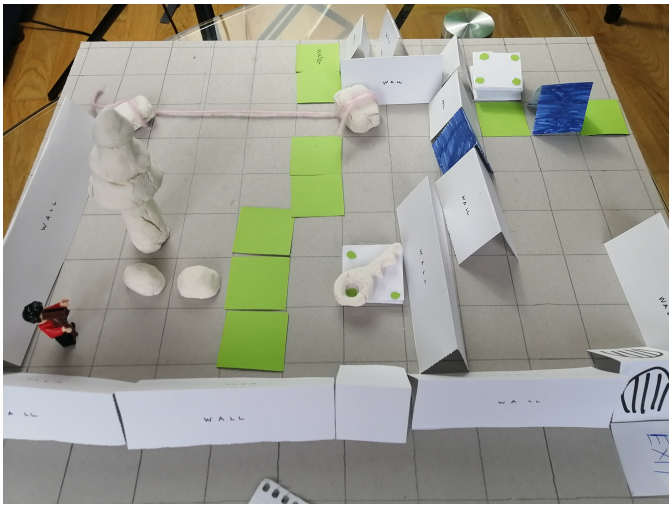
Playing Experience We were approached by an interested external person at the Student Project House when we were prototyping the game, so we asked them to play the game. He completed the puzzle in 11 actions, taking roughly 5 minutes. He said he liked the core mechanic of the hammer very much, and offered us lots of critical feedback, which can be summarised as follows:

- The puzzle was too linear: there was only one obvious action a player could take at any time, and at many times those actions were repetitive too.
- The breakable rocks and walls were useless, as they could just be ignored when mentally computing the trajectory of the hammer. In contrast, the tree falling to create a bridge was good because it changed the field in interesting ways, preventing the player from thinking ahead too much.

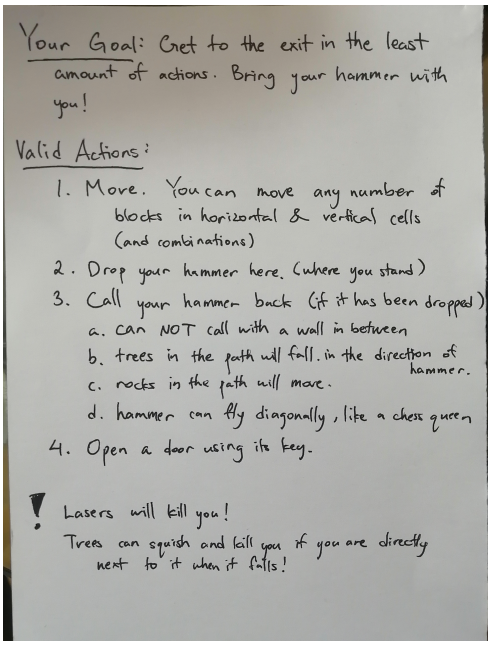
2.2 Final Prototype Setup

Iterating upon the feedback from the initial prototype playtest, we designed another puzzle. We removed all breakable rocks and walls. We introduced more objects that branched player decisions, such as a deadly laser that could be blocked by a fallen tree, a river in which a movable rock could be pushed into to form bridges, or pressure plates that activated doors in remote locations.

The puzzle we built is shown in figure 2.4(a).



(a) The board containing the final puzzle



(b) The rule sheet we handed players

Figure 2.4: *The complex puzzle we built upon feedback*

Playing Experience This puzzle was designed by three people in our team. We then let each of the other three in our team play, following the game concept that Game Masters design puzzles and challenge players to compete in the number of actions. Each person took around 45 minutes to complete this puzzle (with an average of 100 actions), and offered improvement points, such as assets to replace or clarifications that the Game Master should give players. When we had iterated this puzzle as much as we could, we also invited another external person to play through it.

The external person claimed that the core puzzle mechanic was fun and intuitive. He also praised the puzzle design, enjoying the difficulty of it and the creative thinking required to solve it. We received criticism that there were some visual cues lacking, and that there were repetitive tedious actions at some point in the puzzle. However, he offered several modification suggestions to the map to improve this, such as having a rock already in one river tile to indicate that it is possible to create bridges.

Based on the playing experience of four players who all enjoyed the puzzle, we became convinced that this was a good design, which we could port to the video game easily as well.

2.3 Findings and Conclusion

We learned that designing puzzles is more difficult than it seems. Our first attempts were too linear, with only one way to solve them. After many iterations, we removed elements that did not add anything to our game, like multiple breakable items on the same path. We came up with new interactive elements such as the lasers and movable blocks, to unlock many more puzzle variations while still using a limited set of elements.

It was very interesting to see that although we designers thought the initial puzzle was fine, an external person (who even saw the core mechanic for the first time) told us the puzzle was boring and too linear instead of challenging. To improve upon this, in the final puzzle design we split our team into designers and players, to give half of our team a chance to play the puzzle fresh without any knowledge of how to solve it. The positive feedback we got out of this led to a successful playthrough once we invited another external person. This experience has made us realise that play-testing will be extremely crucial to our video game, especially because our game is so focused around puzzles.

We spent 3 hours designing the final puzzle for the physical prototype. We don't expect to create many of this quality for our final game, but we modified the planned timeline of the project to increase the number of expected hours needed to design further puzzles.

The experience of creating and playing with the physical prototype helped us realise that it was perfect for a level editor for our game. Although drag-and-drop tilemap editors exist for 2D, we could not find intuitive level editors for 3D, let alone MonoGame. We would have to code our own, but it still would be hard to collaboratively build puzzles easily. Whereas with the physical prototype, just rearranging assets around allowed us to easily prototype new puzzle ideas, which we found extremely useful for a game where designs of good puzzles are crucial to the player experience.

3

Interim Report

3.1 Progress

We have completed all Layer 2 (Low Target) tasks and are roughly in the middle of Layer 3 (Desired Target) . Below is a screenshot of tasks which we have completed up to this point (from the initial Gantt chart).

Target Level	Done	Task	Responsibility
Functional Minimum	☑	Formal game proposal	All
	☑	Physical prototype	All
	☑	3D model of one island with dummy assets (trees, rocks)	Andrew, Marie
	☑	Character model (no animation)	Marie
	☑	Controllable character (no animation)	Sid
	☑	Hammer that can be picked up and dropped	Sid
Low Target	☑	1 tutorial puzzle (on 1 island, introduce hammer movement)	All
	☑	2 learning/study puzzles	All
	☑	Puzzle Editor	Yuto
	☑	Adding textures and basic lighting to island	Andrew, Sid
	☑	Character animation	Marie
	☑	Drop hammer, call it back, and break elements (basic path finding)	Sid, Konstantinos
	☑	Background Music	Audrey, Sid
	☑	Basic story narration (hint ideas at different levels)	Konstantinos
Desired Target	☐	Hammer callback trajectory	Konstantinos, Sid
	☐	Storyline fleshed out	onstantinos, Yuto, Audre
	☐	Additional animation (hammer movement, collision, etc)	Marie, Andrew
	☑	2 more learning/study puzzle, start 1 solid puzzle	All
	☐	Shaders and better rendering	Andrew
	☑	Additional 3D asset design	Marie, Andrew, Yuto
	☑	Animation loader (programming)	Marie, Sid
	☑	Basic sound effects (character movement, collision, throwing, etc)	Audrey, Sid
	☐	More detailed narration (especially introduction and ending)	Audrey, Yuto
	☑	Map restart UI and pause menu	Yuto
	☐	Game tutorial/trailer	Yuto, Andrew, Konstant.
High Target	☐	Multiple islands, each with different landscapes	Marie
	☐	Sound effects for interactions	Audrey, Sid
	☐	Game saving and restart mechanic	Yuto
	☐	Basic end game cinematic	Marie, Andrew

Figure 3.1: *Status of Schedules*

Specifically, the core foundations of the game has been completed, including sound playback, 3D animations, UI, and level editing. With these as tools, we are going to be focusing on working on the technical achievement until the Alpha.

During the Easter break, we separated tasks into 6 different main areas, and divided workloads between team members. These areas were: integrating a physics engine, working towards our technical achievement of intelligent path planning, creating and using animated meshes, adding a sound management system, improving shading/rendering, and other miscellaneous features such as UI or a scene editor.

3.1.1 Physics Engine Integration

We wanted to transition from the basic, naïve collision detection system we had implemented for our first playable demo to an implementation using an external physics library. With this in mind, we decided to integrate the BepuPhysics V1 library into our codebase and rebuild our mechanics around it. As of writing the interim report, the required elements of the library is integrated for the purposes of collision detection and the previously existing mechanics (player movement, hammer collision events, pressure plates, interactable trees, collectibles) have been reimplemented. Additionally, early implementations of a few newer mechanics, necessary for our puzzle design goals such as movable blocks and lasers, have also been added to the game.

3.1.2 Path Planning

In the Layer 2 (Low target) implementation of the project, a naïve straight-line routing scheme was adopted, so as to be able to test the functionality of all other basic components of the game. However, in order to stay faithful to the technical achievement of this project, a more complex path planning system needed to be devised. In particular, the path planning scheme should be able to avoid obstacles in the scene; a feat which the previous implementation could not achieve. As of time of writing, the path planning algorithm utilized for the routing is the A* (pronounced "A star") algorithm. Some reasons this particular algorithm was selected are:

1. It is well studied in the literature
2. The environment is fully known in advance; to be more exact, the environment may change due to the trajectory the hammer will follow (e.g. a tree falling), however those changes do not alter the trajectory of the hammer in any way.
3. It is easy to implement

The A* algorithm is studied in the context of graph traversal. Consequently, to complement the implementation of the A* algorithm (or any graph traversal algorithm which might be implemented), a data structure representing the actual 3D space of the environment needed to be implemented as well. The data structure selected was a simple uniform grid (i.e. an orthogonal parallelepiped subdivided into smaller cubes of fixed side length). The fact that the hammer (which is the only object which utilizes the path planning scheme) may also travel through the air further encouraged the use of the uniform grid.
Finally, there is a bijectional (1-1) relation between a cell of the uniform grid and a vertex re-

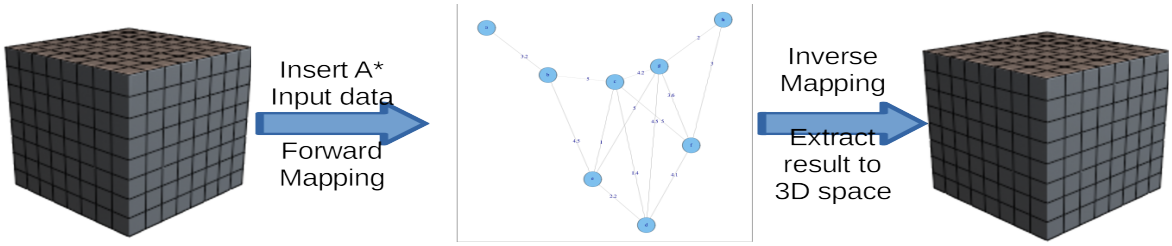


Figure 3.2: *Path planning pipeline.*
Two external image resources were used for the creation of this figure.
The cube (left and right image): UserTwoSix, CC BY-SA 4.0, via Wikimedia Commons
The graph (middle image): Sigbert, CC BY-SA 4.0, via Wikimedia Commons

quired in the underlying graph used during the A* evaluation: the A* is solved on the graph and the result is mapped to the corresponding 3D grid cells the hammer will travel through (figure 3.2).

Due to the incompleteness of the function, it has not been merged with the main branch (yet).

3.1.3 Assets and Animations

The character now has animations for running and idle standing. We used Mixamo to animate our character model and Aether.Extras to import animations from our fbx models. Now that we have learned how to integrate animations into our game, we will be able to add animations for the surroundings, such as water, trees knocked down and hammer throwing.

3.1.4 Sound Effects

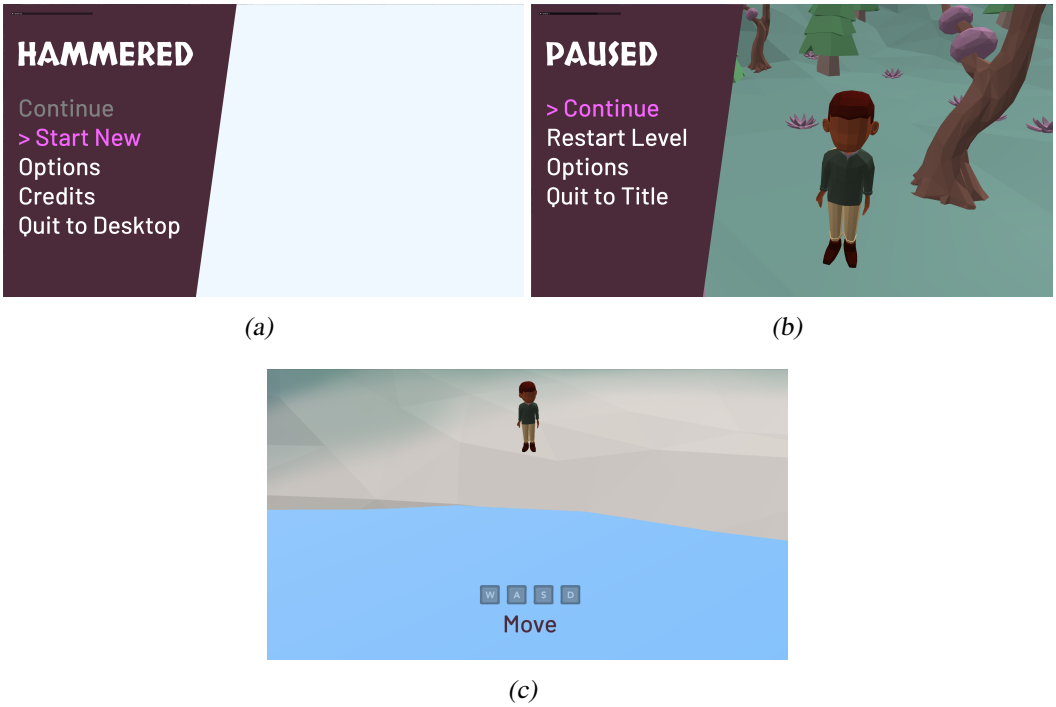
We have implemented an audio management subsystem that runs through the spine of our game. This allows us to call arbitrary sound effects and background music at will. Specifically, we have made it versatile enough that not only can fixed sounds be played from various game objects, but chosen sounds can be played on-demand through scripts that control cut scenes or level logic. We expect to be using this sound subsystem a lot in the upcoming two weeks to make immersive aural experiences.

3.1.5 UI

A functional UI for the menu has been completed as well, using Myra. This is in addition to a developer UI and active scene editor done with ImGui, which is enabled only in Debug releases. The UI supports mouse, keyboard, and controller input for item selection, and in this is where we have put the Level Restart functionality, taking inspiration from other games which employ a similar pattern.

During the process of implementing UI, we have added functionality to display layers of screen content on top of each other. This has enabled us to implement input prompts as well; see figure

3.3. The input prompts automatically change based on the input method, and currently supports keyboard prompts and Xbox controller prompts. We hope that this has a positive impact on the player experience for the Interim demo (to our classmates) and for the Alpha release in two weeks. In the future, we might want input prompts shown in world coordinates as well, such as on interactive props.



(a) *(b)*

(c)

Figure 3.3: *Menu and tutorial UI*

3.2 Challenges

There were several tasks that we underestimated the effort it would take to implement for. One is asset design. Not only did we need assets for small environmental props or key parts of our levels, we needed terrain and their textures as well. Inconsistencies between Blender and Monogame's color management and some frustrations with embedded textures in FBX formats required a lot of time to be spend on assets. Animation was also daunting, but we were rescued by Mixamo, which we have mentioned above.

The physics engine integration also proved to be a challenge at first, requiring nearly a full rewrite of our existing code. However, once the necessary changes were made to the base classes, modifying the specific mechanics/objects was pretty straightforward, with the main challenge being parameter tweaking to model the desired interactions between the player, hammer, and the environment.

3.2.1 Path Planning

1. Initially, an attempt to implement the A* algorithm without any external library was made. However, after careful and extended search and experiments, it was deemed counter-productive to attempt implementing the priority queue required for the A* algorithm. For this reason, an external fast implementation of the required priority queue was integrated instead.
2. The implementation suffers from an inherent problem of both the uniform grid data structure and the A* algorithm: the requirement of large quantities of memory. The former has a complexity of $\mathcal{O}(xyz)$, where x, y, z are the number of cells in each of the three dimensions of the parallelepiped, while the latter may reach exponential memory usage (w.r.t. the path length).
3. As of the time of writing, there is no connection between the physics engine and the uniform grid. As such, no efficient way to reduce the number of connections in the underlying graph has been implemented. This has three (3) drawbacks:
    - The efficient dimensions and subpartitioning of the parallelepiped are left to the XML document writer, instead of being automated.
    - The computational system which runs the game is encumbered with unnecessary memory.
    - The current implementation has as a convention that the grid can be dynamic (i.e. every grid can become available or unavailable at any time). As such, the whole graph is recreated (millions of objects are recreated), which slows down the system.
4. As of the time of writing, the current implementation cannot be run in real time (some cumbersome operations were mentioned above).
5. Currently, no smoothing techniques of the trajectory have been applied on the path. As such, the movement of the hammer looks "mechanical", being rigid and following vertices in an invisible uniform grid.

3.3 Future Work

Until the Alpha Release in 2 weeks, our planned tasks are to complete all points in Layer 3 (Desired Target) and get started on Layer 4 (High Target). Concretely, this means implementing our path planning, working on a better rendering pipeline, designing more props and levels, replacing temporary assets, and adding storyline elements such as monologues. Some of the original High Targets, such as saving and loading, are expected to be very easy to implement due to how we have implemented 3D scenes. We will therefore dedicate more time to our technical achievement, and to polishing the player experience.

4

Alpha Release

4.1 Progress

We have completed the majority of layer 3 (desired target) and half of layer 4 (high target). The initial Gantt chart and completed tasks can be seen below.

4.1.1 Path Planning

In the interim report, the scheme to integrate a full path planner was described. A visualization of the existing pipeline is entailed in figure 3.2

For the alpha demo, focus was put on a) making the path planning as real-time as possible, as well as b) smoothing the resulting path.

Desired Target	☑	Hammer callback trajectory	Konstantinos, Sid	18
	☐	Storyline fleshed out	onstantinos, Yuto, Audre	2
	☑	Additional animation (hammer movement, collision, etc)	Marie, Andrew	12
	☑	2 more learning/study puzzle, start 1 solid puzzle	All	12
	☑	Shaders and better rendering	Andrew	18
	☑	Additional 3D asset design	Marie, Andrew, Yuto	21
	☑	Animation loader (programming)	Marie, Sid	19
	☑	Basic sound effects (character movement, collision, throwing, etc)	Audrey, Sid	8
	☐	More detailed narration (especially introduction and ending)	Audrey, Yuto	7
	☑	Map restart UI and pause menu	Yuto	11
	☐	Game tutorial/trailer	Yuto, Andrew, Konstant.	19
High Target	☐	Multiple islands, each with different landscapes	Marie	18
	☑	Sound effects for interactions	Audrey, Sid	13
	☐	Game saving and restart mechanic	Yuto	4
	☑	3D audio and positioning and distance falloff	Audrey, Sid	
	☐	Basic end game cinematic	Marie, Andrew	11

Figure 4.1: *Status of Schedules*

Real-time responsiveness

Goal a) was deemed of utmost importance, as the feeling of responsiveness in a video game is an integral part of the user experience (UX). To combat the path computation overhead in the previous part, the principle of early path computation was adopted.

First of all, we took advantage of the assumption that the game environment is a Euclidean space.
The above assumption allowed us to exploit the (rephrased) observation which is ascribed to Archimedes of Syracuse:
"(In Euclidean space) the shortest-length curve between two points is a linear segment".

The agent (hammer) travels the distance between its starting position (where the hammer was dropped) and its destination (the position of the character when the "call back" action is invoked) in a shortest-path manner. Therefore, in cases where the linear path is unobstructed by objects of the environment, the linear path *is* the shortest path. Since we are aware of the path model (a line) the agent will take, we may save up computational time by directly evaluating said model, instead of executing the computationally demanding A* algorithm. It should be emphasized that both methods will return the same result.

The reason for taking this approach is the difference in algorithmic complexity between the two methods; a faster computation will extract the path faster, which in turn enhances the responsiveness of the game, with the agent reacting instantly to the command of the player.

How are cases in which the linear path is obstructed handled?

1. the linear path until the first obstacle is reached.
2. the shortest path from the position of the first obstacle position to the destination position is computed.

The above method "buys" time for the main game to compute path 2. (full solution) *while* the hammer is traveling the linear path (responsiveness).

Path smoothing

After having extracted the correct path, the next step would be to make its traversal seem more natural to the human eye. To achieve this, two (2) simple, yet effective techniques were incorporated.

The first one is reducing the amount of linear segments included in the path. This is achieved with the use of simple visibility checks between three consecutive points of the path; if the first point in the triplet can travel unobstructed to the third point of the triplet, the second one is deemed unnecessary and dropped.

Note: The fact that this still remains a shortest path is guaranteed by the triangle inequality.

The use of less anchor points results in significantly less artefacts which could arise from transitioning from one anchor point to the other, namely looking slightly like a stop-and-go motion.

The second technique is by utilizing quadratic Bézier curves. In order to account for curving paths, piece-wise quadratic Bézier curves are constructed in every triplet of points along the path.
The choice of a second-degree polynomial was lead by the assumption that the agent will turn around a corner of a single object in a hemicircular motion. This hemicircular motion is approximated by "concatenating" parabolas.
After the turn is made, the agent will follow a linear path until it reaches another obstacle, in which case the process is repeated, or it reaches its destination, in which case the path planning is terminated.

4.1.2 Physics and Map Design

In order to fix certain problems we had from our previous version (the character sliding after coming to a stop, ability to climb certain terrains due to unspecified bounds), we have changed our island meshes to be voxel based instead, in order to more clearly define what can and can't be traversed. This has allowed us to increase the friction coefficients to disallow players from climbing the mountains, and better see the various obstacles across the map. This has also helped with the collision interactions, such as pushing movable blocks on land versus into water.

4.1.3 Assets and Animations

We have continued adding animations and textures to our objects to make their purpose in the game more clear. An example is the tree falling animation, which transforms the tree into a log to make it more visually obvious that the log can now be traversed. We have also made new textures for the hammer, walls, and player to add to the visual immersion of the game and make the main mechanic with the hammer more obvious.

4.1.4 Audio

We have fully switched over to the audio management subsystem that was designed in time for the interim report, and have now added positional audio. Each interactable game object has been included as an audio emitter in order to provide the player with a better idea of where each object lies within the map. Sound effects for all of these have either been personally designed, or spliced and edited based off of available sounds from Zapsplat.

4.1.5 Shaders

In order to make the game environment more immersive, we implemented basic 3D shaders. Specifically, we implemented real time shadows using PCF, and added HDR and Bloom for glowing light sources. This immediately made the game feel more authentic compared to what we achieved with MonoGame's BasicEffect, but at a price – the computational workload on

both the CPU and GPU was much heavier. What remains in this field until the final release is optimization, implementing possible post-processing filters like LUTs, and making sure everything matches the look we want.

4.2 Challenges

Our main difficulty these last two weeks has been managing our time and workload across so many different tasks and branches. Though our core assets and mechanics had been completed prior to this, decisions such as redoing the entirety of the island meshes to better suit the game mechanics required a whole new set of assets, a dependency on which improvements on the movable block interactions, shading, and level design would have to wait on. Additionally, our particular task assignments across team members made it so that each member would have very specific knowledge about certain aspects of the code functions and varying timelines in which they would be done. This required not only frequent merges of the various pipelines, but much more intentional communication and collaboration from everyone in order to piece our various code and tasks together into a final product.

4.2.1 Path Planning

- The memory print of the uniform grid is something for which no solution was found. Considering the fact, however, that its impact is a few hundred MBs of memory, it was decided to let it be. Moreover, the visualization of the grid (in debug mode) allows for fine-tuning the grid for minimal memory consumption.
- There are cases where responsiveness is not fully achieved. This is in cases where the linear path traversal does not last long enough for the A* path to be computed in the background. Two main cases where observed during testing:

  1. The linear path is exhausted before the A* computation is complete; the hammer hovers mid-air before proceeding to follow the A* path.
  2. There was no linear path, since the hammer was in proximity of an obstacle. In this case, the player has to wait (1-2 seconds) for the program to fully compute the A* before seeing any response from the game.

  An idea as to how to counter this issue is to follow the early path computation principle mentioned above: partially compute the A* solution and have the agent execute this partial path, while in the background it computes the A* from the end of this partial solution until the destination. Although it might not necessarily need to truly optimal solutions, there are few edge cases which would result in the partial path being significantly different from the truly optimal one.
- Computing or incrementing the parameter value $t \in [0, 1]$ of the Bézier curves.

  1. *Estimating the t parameter.* An analytic solution –which would allow for the instant computation of the velocity of the agent at any point of the curve– would require an

(infinite) integral to compute the length of the curve. Moreover, the discrete nature of a game engine could result in the agent going slightly off the curve in cases where the position is not hard coded. This can result in the t estimation scheme becoming unstable and the motion suffering for it.

2. *Incrementing the t parameter*. The t parameter expresses the percentage of the curve covered ($0 \rightarrow$ start , $1 \rightarrow$ destination). However, what we wish to achieve is constant speed of the agent along *any* curve. Therefore, the amount that t needs to be increased is curve-dependent. Currently, no reliable method to do this has been devised.

The inconsistency of the t manipulation results in the hammer accelerating and/or decelerating in paths which include obstacles.

- *A* is an offline algorithm*. This results in the agent not following the character when they move after invoking the "call back" function. However, it was a team decision that this would pose no problem, as the end product will have the character freeze in place; this makes sense for a puzzle game, where the player wants to see the results of the action they took, without requiring maneuvers.

4.2.2 Audio

Another difficulty we faced was implementing 3D audio and understanding how to balance out the soundspace, especially with looped sound effects. In the case of both the footsteps and the laser sound effects, the challenge arose in timing the delay of the triggering action enough that the footsteps were distinct rather than layered, without making the delay such that the sound didn't feel responsive. That being said, looped sounds take up a lot of the sound space and end up quite distracting in terms of the game play (footsteps feel repetitive instead of natural, and the laser buzzing is constant), which needs to be improved upon for the next deadline.

4.3 Future Work

Upcoming next is user play-testing and project submission for the jury awards. As we do play-testing, we'd like to make sure that our game is fully complete in terms of puzzles and storyline. We also plan to complete all remaining tasks in the Desired and High Targets. What we have achieved in the last two weeks was basically an asset overhaul and an advancement in path planning, so we'd like to build around this focus and polish the game further. Specifically, more puzzle design, detailed tutorials, narration within each level to show progress, as well as an end-game cinematic are necessary. Our "do one thing, do it well" is using path planning to solve puzzles, so our biggest focus will be on puzzle and map design. The bland-looking islands will receive more foliage and textures as well. For audio, a distance based audio falloff will be implemented to denote the distance of the player relative to the sound emitting obstacle.

5

Playtest

5.1 Playtesting Session

5.1.1

We scheduled all of our playtesting schedules on Friday 5th May, with one session during at lunchtime and one in the afternoon, each led by different halves of our team. In total, we had 6 participants, 4 of which who had never seen or heard of the game before. This included Yevhen, Sam, Steven, Roxi, Saikiran, and Boyko.

(a) *(b)*

Figure 5.1: *(a) Lunchtime playtesting session with Steven and Sam (b) Afternoon session with Saikiran*

5.2 Questions and Comments

Below we list the questions we asked our participants.

How quickly did you understand the core Hammer mechanic?

All players claimed that they understood the "summon / drop" hammer mechanic within the first scene, and some of them said that the cut scene on the first island was helpful as a tutorial.

How quickly did you understand the extensions to the core mechanic (falling trees, moving rocks)?

Many testers said it took a bit of time to understand, but that it was intuitive once you got it. There were two conflicting kinds of feedback here: one claimed that this was fine for a puzzle game as it introduced a bit of thinking rather than just linear and monotonous progression, while the other feedback suggested more visual tutorials (like a tree that's already toppled over) to aid the less familiar.

Although this is drawing on assumptions slightly, there seemed to be a trend that the more avid gamers were quicker to test out the hammer with the environment (and realising the behaviour of trees/rocks), whereas the less gaming-inclined participants thought it was not obvious.

Was there anything unclear?

The overall trend of answers here were similar to the previous question – that the tree/rock behavior was unclear at first. Some players also said that they remained confused after having cleared the levels, asking us if what they did was the intended solution or not (it was).

Was there anything you thought would improve the game?

The participants said that adding more levels and more story elements would improve our game. Some also said it would help to have more tutorial levels to gradually understand the hammer mechanics and then move on to harder puzzles once the player got the hang of it.

What did you think about the play time?

Again we got two kinds of feedback: some said it was alright and others said it would be nice to have more levels, or have longer puzzles, in which case it might be good to have some checkpoints throughout the level.

Keyboard or Controller? Were the controls intuitive?

We told participants prior to each session that our game supported both controller and keyboard input, but was designed primarily with controllers in mind. Despite this, 50% of our players played with the keyboard while the other half played with an XBox Controller. This could be due to surrounding circumstances though, as some of the playtesting sessions were done at the Mensa and participants did not want to take out controllers.

We received mixed feedback on the control mappings. Testers liked most of the movement and interaction controls, but expressed their confusion for an inverted camera control (we had gotten used to a control mapping opposite to the industry standard), and using Z on the keyboard for confirmation instead of the space or enter keys.

5.3 Other Feedback

The questions we posed above were not enough for participants to express their feedback, and so we have consolidated the ones that did not fit below, categorized into the various aspects of our game. The specific revisions that we are doing as a response to these feedback are outlined in Section 5.4.

5.3.1 Game Mechanics

- Players expressed confusion at the hammer floating mid-air after summoning it and moving away. This is because of the behaviour of the hammer's path-planning to reach the location that the player was at when they called it, but not chase the player.
- When the hammer was outside the viewport, players often expressed confusion on where it was and how it would behave if they summoned it.
- Players found the movement slow in comparison to the map size. In levels where you had to go back and forth across the map, players said they wanted a dash button. Similarly, a jump button was requested by some players, although they said it wasn't a large problem.
- One player exploited the physics engine interactions, and spammed keys to cross blocked parts such as the sea, and reaching unintended areas.
- The trees could be knocked over in unintended angles at unintended locations. Specifically, some players exploited the trees on the perimeter of islands by placing a hammer on the inner side and standing on the thin ledge between the tree and sea, causing the tree to fall on them and create a bridge. One player specifically used this trick to clear all levels without relying on any of the other mechanics.
- With regards to the physics engine, there were several occasions of the players getting stuck on a slope between the land and water, resulting in them having to restart the level.
- When dropping the hammer next to a tree, players often got stuck between the tree and hammer bounding boxes and had to walk around it, which was not intuitive or pleasant.

- Restarting a level in our game through the menu restarted it from the very beginning of the map. This was fine for the first few small maps, but some players were exasperated at having to re-do the last large map when they got soft-locked near the exit.
- Some participants were not sure which doors where supposed to be unlocked with a key or with a pressure-plate. They did not understand when a key was used, due to the lack of feedback.
- One player wanted more indication on when he was soft-locked and needs to restart. He said he couldn't tell the difference between when he simply couldn't think of the solution, and when he had performed wrong interactions and was actually in a soft-lock situation.
- All players found that the player-water interaction was not intuitive. Most of them thought it would make more sense to be unable to fall in the water by putting boundaries on water areas, instead of falling in it and reappearing on the ground next to it.

5.3.2 Storyline

- The first 4 participants played a version of our game without any end-game cinematics. We had to tell players that the game was finished. These testers requested a more explicit completion of the game.
  The last 2 participants played a modified version that had a basic end-game cinematic. Although this was a quick temporary solution, participants said they found it satisfying, and that it was clear that the story was over.
- Players expressed confusion towards the short bits of monologue that we had included by the playtest build. These mentioned intriguing phrases like "polaroid photos with Thor" or downright confusing topics like "You found some traffic cones". Since we didn't have any other story elements built yet, the participants told us that they felt the storyline was confusing, slightly lacking, and not attachable.

5.3.3 Graphics

- We were told that the game freezing when loading levels was not good at all. The participants said that it reminded them of frozen software and losing work due to bugs, and that it did not come across as intended. Some players offered ideas such as a fading loading screen and loading content asynchronously.

5.4 Design Revisions

5.4.1 Maps that allow creative puzzle solving

Our initial game pitch was that we would have a puzzle game that challenges creativity, and would allow many alternative solutions to the same puzzle. In the physical prototype phase for

example, we specifically designed our puzzle so that there would be multiple intended solutions.

However, as we started playtesting, we saw many players solve puzzles in *unintended* ways, often exploiting bugs in the map layout, such as knocking down trees in particular angles to create bridges to bypass locked areas. The players told us that they felt like they had cheated and found an unintended solution, which was true, but it made us question ourselves: How much creativity were we willing to allow players as they solve our puzzles? And how should we confine that freedom to expected bounds?

Going forward, we are considering two approaches to this. One is to keep the player's freedom of being able to knock down every tree, but to remove trees from island perimeters and other areas that could be used to bypass another desired solution. The other is to limit the player's freedom itself, by keeping the map the same but marking certain trees as unable to fall. There are benefits and drawbacks to both, and we need to experiment more by the final release.

5.4.2 Hammer Behaviour

The hammer's path-planning behaviour has been modified so that it attempts to continuously search a path towards the current player position, performing A* when necessary or otherwise using a straight-line path. This results in a very smooth experience as a player, since you can keep moving while calling the hammer back and it will do almost exactly what you intend for it to do – avoid obstacles while following you.

Since we had occasions where the player could outrun the hammer, we have revised the player's movement to introduce a slowdown while the hammer is being called. This also works as an indication that the hammer is mid-trajectory, which wasn't very obvious before.

5.4.3 Other small changes and planned revisions

We have additional planned revisions that either have already been implemented or will be implemented next week. These are planned to be completed by next Tuesday, for the jury award submission deadline.

- An addition of a complete end-game cinematic, improving upon the makeshift one created during the playtest sessions.
- An improved storyline with more dialogues and images.
- Improved interactions with the doors: visual and audio feedback when using a key.
- More decorative assets on our maps to reduce visual boredom as players move rather slowly and this cannot be changed due to puzzle limitations.
- A loading screen to asynchronously load content for the next map while keeping the UI thread at a steady FPS.
- Changing the key configurations and adding an options screen

6

Conclusion

6.1 Final Results

We list below the main changes made since our Alpha release.

Design:

- Redesign of the laser tutorial map
- Two complete new islands and a map for end-game
- Complete story with beginning and end cutscene cinematics
- Many new 3D assets (bridge, boat, laser base, vegetation, bottles, ...)
- More character animations
- Victory animation and sound for level completion
- New puzzle mechanics (e.g. moving lasers, pressure plates that need to be pressed in specific orders, pressure plates toggling lasers)

UI:

- End credits
- Options menu
- Loading screens
- Improved dialogue interface
- Sound feedback in menu

Audio:

- Sound effects improvements
- Ambient sounds
- Different music identity for each island

Technical:

- Shader improvements
- Better physics and interactions with environment
- Improved and more efficient path finding
- Improved level editor
- Checkpoints throughout levels to save progression
- Hardware instancing
- Player can now step on short obstacles
- Many bug fixes

Juiciness:

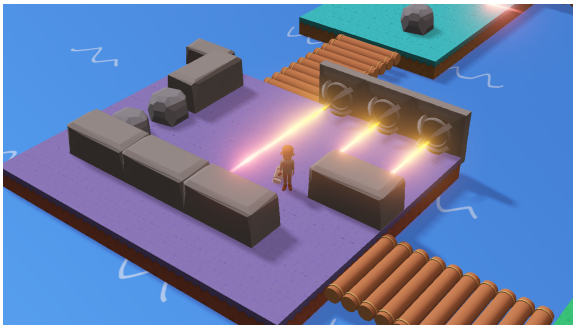
- Dust particles for hammer and trees
- Accelerating flying hammer
- Screen and controller shakes

6.2 Experience

Our experience during Games Programming Lab (GPL) can be summarized in one word: fun! Frequently, members of the team treated working on the game as leisure activities and a form of procrastination from other assignments – a result thanks to the highly enjoyable and motivating course.

Looking back at the original game proposal, the majority of its design ideas have materialized into the final game. We were able to build a third-person 3D game about delivering Thor's hammer, and have created many well-received puzzles that use the core mechanic of a flying hammer. We've stuck loyally to the original goals of not having any stressful time limits or Game Overs, as well as to the goals of having a low-poly saturated asset design. Our Bullseye Idea hasn't changed over the course based on user feedback nor based on development requirements, and we think this is a testament to the strong and stable idea all of us members had from the beginning.

We were able to follow the development schedule well, respecting all deadlines imposed on us by the course. The Gantt chart we developed initially was not as useful as we had intended, since it failed to enumerate several large tasks such as the integration with the physics engine (and the accompanying "can of worms", like collisions, stepping, etc) or UI, while it overestimated the efforts of some small tasks (e.g. "Player can move while the hammer is being summoned" in



(a) Lasers from the second island

(b) Cutscene after waking up

(c) The map editor and vegetation

Figure 6.1: *A showcase of interesting screenshots from our game*

the Extras section was achieved very simply by turning a boolean flag on). Instead, we kept up-to-date on the schedule through holding weekly meetings on Friday to re-enumerate immediate and long-term tasks, re-prioritize them, and re-assign them based on individual availabilities that week. When tasks were delayed due to technical or personal reasons, it was efficiently re-assigned to someone else or we modified the date it should be merged into the main code branch. This agile strategy proved to be very successful in making sure everybody was aware of their main responsibilities at any point.

There are two things we wish we could have implemented from our initial design had there been more time. The first is the creation of puzzles and mechanics that utilise the height dimension, which was originally a driving motivation for choosing 3D over 2D. The second is the additional hammer mechanics such as being able to throw it like a boomerang. These two would increase the possibilities of puzzle design for our game.

The milestone framework provided to us by the course (such as prototype, playtesting, etc) played an important part in keeping a steady pace for progress. Although we weren't very strict about the deadlines internally (and shifted the merge into the main branch if features couldn't be completed on time), they served as indicators for the general progress we should be making, and were useful in deciding how to balance time commitments with other courses throughout the weeks. Additionally, the course presentations following each milestone was a large source of motivation since we could learn from the progress of other teams. The physical prototype

milestone specifically holds an important place in our hearts, since it was something we would have only done as part of a course, but proved to be very useful in designing puzzles that translated well into the game.

6.3 Personal Impressions

6.3.1 Influences from the Norse Mythology theme

Not many of us knew about Norse mythology outside of what's seen in the Marvel universe. In the first week, we researched various potential topics like a resource management game during Ragnarok, or a multi-God theme survival game. In the end though, we decided we'd like to create a laid back game, so we limited the influence of the theme to just the loose concept of Thor's hammer, which Thor in Norse myths can summon no matter how far away he is from it. We avoided more complex story-level influences in aims to achieve a lighthearted story-line. We think that the theme was nonetheless essential to the core game idea we settled upon, and that it has guided us nicely when we were lost on mechanics or story.

6.3.2 Technical Difficulties

The largest technical difficulty was with working in 3D and not 2D, which we chose out of puzzle design goals and artist capabilities. Although we are proud of our result and believe that our 3D solution looks and performs adequately, the process of achieving this was difficult. Specific difficulties arose when trying to integrate model animations, when attempting to design levels and place props, and when we wanted shadows and had to re-implement the entirety of the default BasicEffect shader that MonoGame provides.

Architecting the project was also a difficulty. ***Hammered*** has many components internally: an audio/sfx manager, shaders and rendering pipelines, various game objects, an internal grid representation of the map for path planning and its accompanying algorithms, a checkpoint system, a settings manager, a screen system for UI overlays and transitions, map managers and in-game scene editors, a particle system built from scratch, and a scripting engine built from scratch for cutscenes. These all interact with each other even for an action as simple as summoning the hammer (sfx, particles, path-planning, specific shaders, controllable settings). We settled on using many manager classes.

6.3.3 Thoughts on MonoGame

MonoGame was a great introduction to game development since it was so barebones. We believe we have grown highly transferable skills like software and GPU optimization or software architecture design that we wouldn't have acquired had we opted for a high level game engine, such as Unity.

There were a fair share of issues with MonoGame though, notably the lack of recent and in-

depth documentation. We frequently had to translate XNA tutorials from 2009, and the library ecosystem surrounding MonoGame was limited at best, especially around 3D games. Additionally, its internal audio engine was quite frugal and did not sound rich. On top of this, MonoGame lacked support for compute and tessellation shaders, which would have helped us achieve the visual and performance goals we had in mind.

6.3.4 Final Thoughts on the Project

We feel that the time commitment and deadlines made this project very much like a project in industry. Every one of us had the opportunity to develop project management, time management, and collaboration skills. Some of us have also acquired skills they've long wanted to learn, such as asset modeling in Blender, music composition, or GLSL shader programming.

We consider our project to be a success, and are happy with the results. Our original visions for the game have been realised to a satisfactory degree, as we've completed all plans up to the High Target except for some that were removed for design choices, and honestly any further expansion of the game would consist of bug fixes, QoL improvements, or adding several new puzzles.

For the GPL course overall, we think that it would be nice to get more active feedback from the TAs with regards to the game's progress and its features or lack thereof. Some lectures felt irrelevant given the timing, and I think it would be beneficial for all future teams if there were sessions dedicated to TAs going around and giving feedback based on both the reports and the playtests.

6.4 Acknowledgments

We'd like to thank the many friends who playtested our game and discovered the many hilarious cheats and physics bugs that we initially had: Boyko, Oana, Roxana, Saikiran, Samuel and Steven. Thank you also to Morten for casting Thor in our final game trailer, it was awesome. We'd also like to offer Studio Gobo our deepest gratitude for playtesting our game and giving us valuable feedback.

All of our sounds are royalty free or composed from scratch. We have sourced the former from ZapSplat, samples by Avery Berman, Kenney's UI Audio pack, and Ellr's Universal UI/Menu Soundpack.

For many of our technical difficulties, the MonoGame User Forum and the MonoGame Discord Server have been immensely helpful in finding the path to a solution. Thank you.

Finally, we'd like to thank the Game Technology Center for providing us with this wonderful opportunity.