Babies & Basilisks: An Asymmetric VR Party Game

A Major Qualifying Project
Submitted to the Faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
Degree in Bachelor of Science
In
Interactive Media and Game Development
By

_____
Paige Cox

_____
Aaron Graham

_____
Hannah Jauris

_____
Vincent Miller

_____
Edward Shaddock

Date: 25 April 2019
Project Advisors:

_____
Professor Mark Claypool, Advisor

_____
Professor Brian Moriarty, Advisor

_____
Professor Ralph Sutter, Advisor

Abstract

Babies & Basilisks is a virtual reality (VR) party game featuring asymmetric, local multi-user gameplay which allows two categories of players to experience one game entirely differently. One player, wearing a VR headset and holding tracked hand controllers, builds a dungeon-like gameplay environment from prefabricated hallway sections and rooms, which can be assembled in many different configurations. Up to four additional players, viewing the game on a separate TV monitor in third-person perspective, use a standard gamepad to move their characters through the constructed environment, overcoming navigational and combat challenges by using abilities unique to each of the four characters. The goal of the gamepad players is to reach a specified endpoint, while the goal of the VR player is to thwart them by strategic deployment of enemies and traps, together with a "boss monster," which the VR player directly controls.

Babies & Basilisks was developed in the Unity game engine for Windows, and is primarily targeted for the HTC Vive VR headset. It features novel AI and control systems, custom art assets and sounds, and was playtested by 68 people as well as being exhibited at PAX East.

Acknowledgements

This project was made possible by the excellent guidance and advice of Professors Ralph Sutter, Mark Claypool, and Brian Moriarty. The characters were brought to life through the talented voice acting of Christine Flores and Justin Harris. Additionally, the staff at WPI's Global Lab in the Foisie Innovation Studio was essential in assisting the team in using the lab for development, especially Mikel Matticoli and Karen Royer. Furthermore, the team is very grateful to the WPI PAX IQP team, Jordan Cattelona, Grant Ferguson, and Kate Olguin, who enabled *Babies & Basilisks* to be demonstrated at PAX East. Finally, many thanks to all of the playtesters, whose invaluable feedback helped shape *Babies & Basilisks* into its final form.

Table of Contents

List of Figures

1. Introduction

Virtual reality (VR) is emerging as a readily available technology, and creating games specifically for VR has become commercially viable. According to findings by SuperData Research, revenue from VR hardware and software reached $3.6 billion in 2018, and is expected to grow to $16.3 billion by 2022 [27]. While the high cost of a VR system and computer capable of its high processing requirements is still a barrier to entry, people can experience VR games in social settings, which gives VR games a great capacity to bring groups of friends together into a single space. For this reason, there is a large opportunity for VR party games, or games that can be played by multiple people using a single VR system and computer. Games in this category include *Keep Talking and Nobody Explodes* [1], *Ruckus Ridge* [2], *Black Hat Cooperative* [3], and *Playroom VR* [4] [23].

Because only one player can wear the VR apparatus at a time, VR party games require alternative input methods for the other players. On a computer or console, a keyboard, mouse, or gamepad is a good option, since many people probably own these items already. However, the difference in input methods leads to an interesting design challenge: these games have inherent asymmetry between the one player using VR (referred to as the VR player) and the rest of the players (referred to as the controller players). While a designer could attempt to offer the same gameplay to both types of players, this fails to make use of the capabilities of modern VR systems, especially those with precisely tracked motion controllers.

An alternate approach is to incorporate asymmetry directly into the game by offering entirely different gameplay for each player type, but still including ample interaction between all players. This was the approach the team took when designing *Babies & Basilisks*, an asymmetric virtual reality party game for the HTC Vive. In *Babies & Basilisks*, a VR player builds an enemy- and trap-filled dungeon by selecting and placing prefabricated maze sections using the Vive's motion controllers. Up to four controller players each choose one of several available characters, and attempt to navigate and survive the VR player's maze as it is constructed. The controller characters, represented by toys, are dwarfed by the VR player, a baby over 20 times their size who serves as an imposing presence throughout the game.

The game's development took place over four seven-week terms. In addition to the work of designing the specifics of the theme and gameplay, the creation of the game required substantial technical, artistic, and audio development. The team created 13 character models for the VR player, controller players, and AI-driven enemies, as well as eight modular hallway sections deployed by the VR player, and 21 static objects comprising the surrounding environment. On the technical side, the team implemented third-person character controls and abilities for four distinct controller players, five enemies with different AI behaviors, a VR building system, a VR-controlled boss with three abilities, a per-room shared camera system for the controller players, and an auxiliary audio library. In addition, the team developed 61 sound

effects, 16 voice lines, and four music tracks, adhering to two distinct themes: a lighthearted and playful theme for the VR player and a dark and gritty one for the controller players.

The team tested the game on four occasions: formally at AlphaFest and again with two game design classes, and informally at PAX East. With the formal sessions, there were 68 total playtesters, and the team was able to refine the game in response to their feedback. In addition to minor tweaks and design changes, the team implemented several major design changes as a result of observed player behavior and suggestions.

This paper describes the process of designing *Babies & Basilisks*, the development of art, tech, and audio, and the testing and refinement of the game. Chapter 2 presents the background and overarching inspirations for the game; Chapter 3 describes the design decisions and process; Chapter 4 discusses artistic development; Chapter 5 details technical implementations; Chapter 6 describes the audio process; Chapter 7 examines playtesting results; and Chapter 8 reflects upon the project.

2. Background

This section describes background information on games, movies, and toys that inspired *Babies & Basilisks'* design and art style. The three games that most inspired the gameplay were *Panoptic* [5], *Garry's Mod* [6], and *Overwatch* [7], described in Section 2.1. The games that most inspired the art style were *Job Simulator* [8], *Donut County* [9], *Ni No Kuni* [10], and *Legend of Zelda: The Wind Waker* [11], detailed in Section 2.2.1. Additional artistic inspiration came from toys (Section 2.2.2) and 3D films such as *Toy Story* series [12], *The Incredibles I & II* [13], and *The Boss Baby* [14] (Section 2.2.3).

2.1 Game Inspirations

In preparation for designing *Babies & Basilisks*, the team researched other games in order to determine the best practices for asymmetric VR gameplay and style, as well as for inspiration for design as a whole.

2.1.1 Panoptic

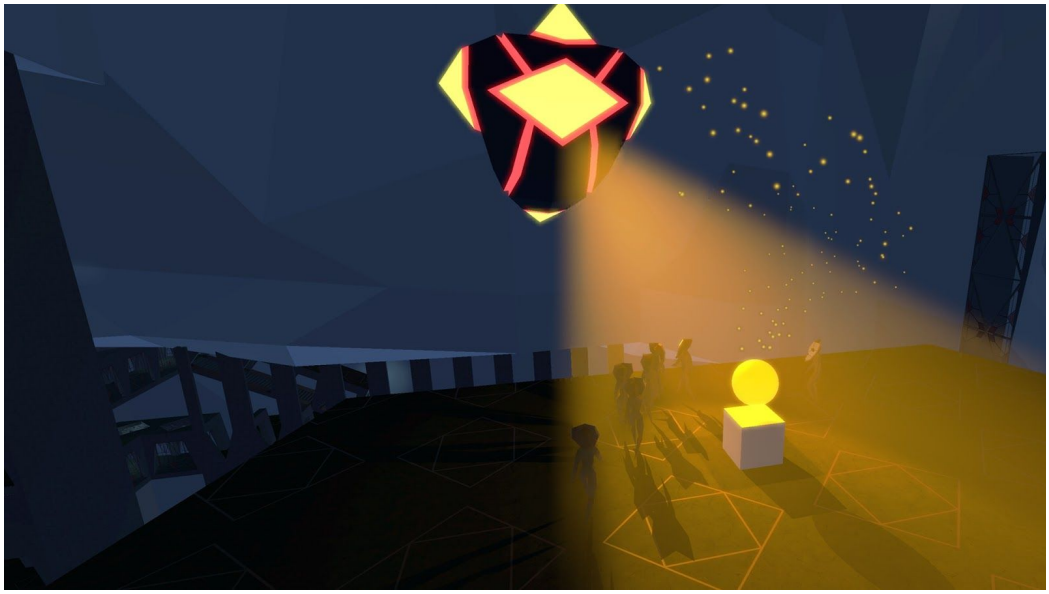


Figure 2.1.1a: In-Game Image from Panoptic (2017) [5]

Panoptic [5], shown in Figure 2.1.1a, is a 1v1 asymmetrical VR/desktop game that pits a colossal Overseer, armed with a slow but devastating eye laser, against a tiny dissident Being intent on destroying the Overseer's energy source by mimicking the movements of others to remain undetected. When creating *Babies & Basilisks*, the team wanted to achieve the same level of immersive, asymmetric gameplay as found in *Panoptic*. Further, the massive scale difference

between the VR player and the controller player was an interesting idea that the team wanted to explore in greater depth.

2.1.2 Garry's Mod

Figure 2.1.2a: In-Game Image of a Deathrun Trap in Garry's Mod (2004) [6]

Garry's Mod [6], Figure 2.1.2a, is a physics-based sandbox game released on PC with a prominent modding community and several game modes. One of these modes, deathrun, significantly influenced the design of *Babies & Basilisks*. In the deathrun game mode, one participant - the "death player" - activates traps in the game environment while the other players try to navigate the area and survive until they reach the end of the map.

2.1.3 Overwatch

Figure 2.1.3a: Promotional Image from Overwatch (2016) [7]

Overwatch [7], Figure 2.1.3a, is a team-based multiplayer first-person shooter. Players select from a roster of 30 characters, each with a unique style of play and multiple abilities. When designing the characters for *Babies & Basilisks*, the team adopted this style of play, creating a specific role and ability set for each character.

2.2 Art Inspirations

A diverse range of media inspired the aesthetic design of *Babies & Basilisks*. The initial goal was to create a lively, asymmetric game themed as a child's toybox. In the conceptualization process, the team decided to pursue a simple, stylized art style with bright colors. To inform the development of this look, the team referenced several video games, vintage toys, and 3D film characters.

2.2.1 Stylized Games

Video games with a visual style similar to what the team was aiming for included *Job Simulator* [8], *Donut County* [9], *Ni No Kuni* [10], and *Legend of Zelda: The Wind Waker* [11]. Major takeaways from looking at these games were the character designs (seen in Figure 2.2.1a), which had minimal surface defining features, simplified real-life details, often exaggerated facial features, and strong silhouettes, all of which helped give their respective games a fun, lively vibe without being overly complex.

Figure 2.2.1a: Characters from Donut County (2018) [9] and Ni No Kuni (2010) [10]

2.2.2 Vintage Toys

Other inspirations for the game's aesthetic came from various vintage toys, such as wind-up metal figures, older plush animals, and small squeaky toys. The general takeaway from these was strong silhouettes and clear, no-frill designs.

2.2.3 3D Film Characters

Figure 2.2.3a: Toy Characters from Toy Story (1995) [12]

Possibly the biggest source of character inspiration came from 3D animated films, the most-referenced being the *Toy Story* series [12], *The Incredibles I & II* [13], and *The Boss Baby*

[14]. The *Toy Story* films provided a vast range of characters with various toy elements, such as in Figure 2.2.3a, and showed how otherwise stationary objects could move in a natural way, all of which helped greatly in the design of *Babies & Basilisks'* toy characters, some of which can be seen in Figure 2.2.3b.

Figure 2.2.3b: A Few of the Toy Characters from Babies & Basilisks

The Incredibles and *The Boss Baby* did not have the toy elements, but they did provide a great deal of inspiration for the VR character model, particularly with the head shape and facial details. A side-by-side comparison of this character inspiration and resulting model is shown in Figure 2.2.3c.

Figure 2.2.3c: Inspiration from The Incredibles [13] (Left) & Resulting Model (Right)

3. Design

This section details the design process of *Babies & Basilisks*, starting with initial design constraints and progressing to gameplay specifics. Sections 3.1, 3.2, and 3.3 describe the experience goal, target audience, and original scope, respectively, all of which had to be determined before concrete game design could begin. Section 3.4 describes the character ability system for controller players, with a description of each character. Section 3.5 covers the modular building system used by the VR player, and all of the placeable components. Sections 3.6 and 3.7 describe the gameplay progression for a single round for the VR and controller players, respectively. Finally, section 3.8 describes the end conditions for both categories of player.

3.1 Experience Goal

Babies & Basilisks' experience goals are to provide its players with a fun, immersive party game with asymmetric gameplay where the VR and controller players compete against one another, but also have opportunities for cooperation if they desired. The game should be a relatively quick--but repeatable--experience, allowing players to experiment with different strategies or characters as desired.

3.2 Target Audience

Babies & Basilisks is designed to cater to both younger and mature players, with the players themselves being able to define how competitive they want the game to be. One restricting factor is access to a VR setup, which may not be available to younger players. Players who have some experience in action games and with VR may find the game easier to pick up and learn.

3.3 Original Scope

Over the first few months of production, the team focused heavily on the general design of *Babies & Basilisks*. The initial concept was an asymmetric VR game featuring *kaijus* (giant monsters). From this basic pitch, the team was given creative control over every element of the game, aside from the fact that it was meant to be an asymmetric VR game. Several ideas were considered for the direction of the game, including the initial monster theme, an internal computer world, and the final toybox theme. Ultimately, the team was charmed by the notion of a lighthearted game and decided on the toybox theme.

Once the team had settled on the theme, discussion over how the game would look and play began. The team went through several iterations for the base concept of the game, the

building mechanics, and the boss fight structure. During this process, many gameplay elements were conceived and redesigned. Part of the initial scope was to include a non-playable character which would act as a narrator throughout, as well as several base environments with themed enemies based on the environment in which the game was currently taking place. For instance, one environment might have been a bathroom, and within this space there would be water and bath-themed toys. However, having a strong overarching narrative did not ultimately fit the team's goals for the game, and was discarded to focus on making a game that would be fun to pick up and play. The multiple rooms and themes were ultimately beyond the scope of the project, and the design was cut back to only include one environment with a higher level of polish.

3.4 Character Designs & Abilities

In order to diversify gameplay, the game was designed around multiple distinct playable characters. By allowing controller players to choose a character and offering significantly different gameplay based on that choice, the game would be more replayable, meaning it would offer more opportunities for unique gameplay experiences in each session. The model the team chose to follow when designing these characters was *Overwatch* [7], which features a large cast of playable characters, each with a specific role. Each character comes with a set of approximately three abilities, which are limited only by cooldowns and cannot be changed in any way. Avoiding further customization of character strengths, such as in *Team Fortress 2* [15], allows the players to quickly jump into the game without having to go through a loadout screen. In order to facilitate players trying different characters, a uniform control scheme was decided upon. Each character would have three abilities: a main attack and two special powers. In designing the abilities, the aim was to create a different play style for each character that was balanced in relation to the other characters. The controller buttons used to activate these abilities are the same for every character.

Because each character would have a rigid, non-customizable role, it was important that a large range of roles were covered, without anything clearly lacking. Four key roles were identified, inspired by the roles in both *Overwatch* [7] and *Team Fortress 2* [15]. The first role is the healer, a support character who can provide assistance to teammates but is not very effective on their own. The second role is a tank, who is good at absorbing large amounts of damage, but not as strong in offensive capabilities. The third role is the melee character, good at quickly closing the distance to enemies, hitting them with close-range attacks, then retreating. The fourth and final role is ranged DPS (damage per second), a character with strong offensive capabilities but vulnerable to damage. Images of each of these characters can be seen in Figure 3.4a.

Figure 3.4a: Character Line-Up, featuring (left to right) Doctor Bear, Tess, Kangaroo, & Amos

3.4.1 Healer

In designing the healer character, Doctor Bear, the artists wanted something that seemed comforting and soft. Since the game has a toy theme, a stuffed bear was a good option. To signify his healing role, he was given a doctor's vest and head mirror. His main attack is tackling, a weak and slow melee strike that should only be used in dire circumstances. His most important ability is to create a healing blanket, an object that stays in one place and heals allies who stand on it. Finally, he can use a roar to push back nearby enemies, protecting his teammates and allowing them to heal in safety.

3.4.2 Tank

For the tank character, the team wanted to subvert the cliché of a large, muscular figure seen both in the Heavy class in *Team Fortress 2* [15] and in characters such Reinhardt and Zarya in *Overwatch* [7]. Instead, the artists decided on a small doll-like figure. From here, Tess the Tiny Terrors Doll was created, complete with oversized sweater, pigtails, demon horns, and an oversized rocket launcher. Her main attack is to fire a rocket, an ability which is somewhat slow but powerful. Her second ability is to taunt nearby enemies, causing them to focus all their attacks on her. She can pair this with her third ability, which significantly reduces the amount of damage she takes for a short amount of time in exchange for slower movement speed.

3.4.3 Melee

For the melee character, the initial design was a knight action figure who would attack with a sword. However, to avoid having mostly humanoid characters in the game, the design was

changed early on to a kangaroo wind-up toy. Boxing gloves and a helmet were added to convey his focus on melee combat. His primary attack is a quick punch that hits a single enemy. His second ability is a spinning tail attack, which can damage a group of enemies around him. Finally, his third ability is a quick dash, which can be used either to damage an enemy in his path, or to quickly escape a dangerous situation.

3.4.4 Ranged DPS

Finally, for the ranged DPS character, the artists decided on a cowboy action figure, who the team named Amos. With *Toy Story* [12] as a clear inspiration for the game's overall look, it was important to distinguish this character from *Toy Story'*s Woody. Therefore, he was made to look much older and given a gruff personality. His main attack is to shoot with his revolver. While this does not do as much damage as Tess's rockets, it can be fired more quickly, giving it a higher overall damage per second. His second ability is to throw an explosive stick of dynamite, which both deals significant damage and knocks back nearby enemies, making it excel at crowd control. Finally, he can throw a lasso, which can either constrain enemies to a small space, or create a small safe area for players to stand in.

3.5 Modular Dungeon Pieces

Overall, there are three types of hallways that the VR player can place, and four types of rooms. These pieces all fit together modularly into a grid, occupying either a 1x1 or 2x2 area. The VR player retrieves these sections from a shelf, where they spawn automatically.

3.5.1 Hallways

Two of the hallway types are a simple straight section and a 90-degree elbow. While these do not provide gameplay in the form of combat encounters or platforming challenges, they can provide the spacing necessary for the VR player to design the level in whatever shape they want, and serve as small breaks for the controller players between challenges. The third type of hallway is a longer, curved section. This type of hallway has no floor, but instead features a series of platforms and traps for the controller players to navigate. This hallway has a camera that moves at a fixed speed, and the characters must keep up with this camera in order to make it to the end without falling and suffering damage. The VR player also has the opportunity to interact with the traps, such as aiming flamethrowers and adjusting the heights of saw blades, if they want to make the hallways more difficult. Each of these hallway types are shown in Figure 3.5.1.a.

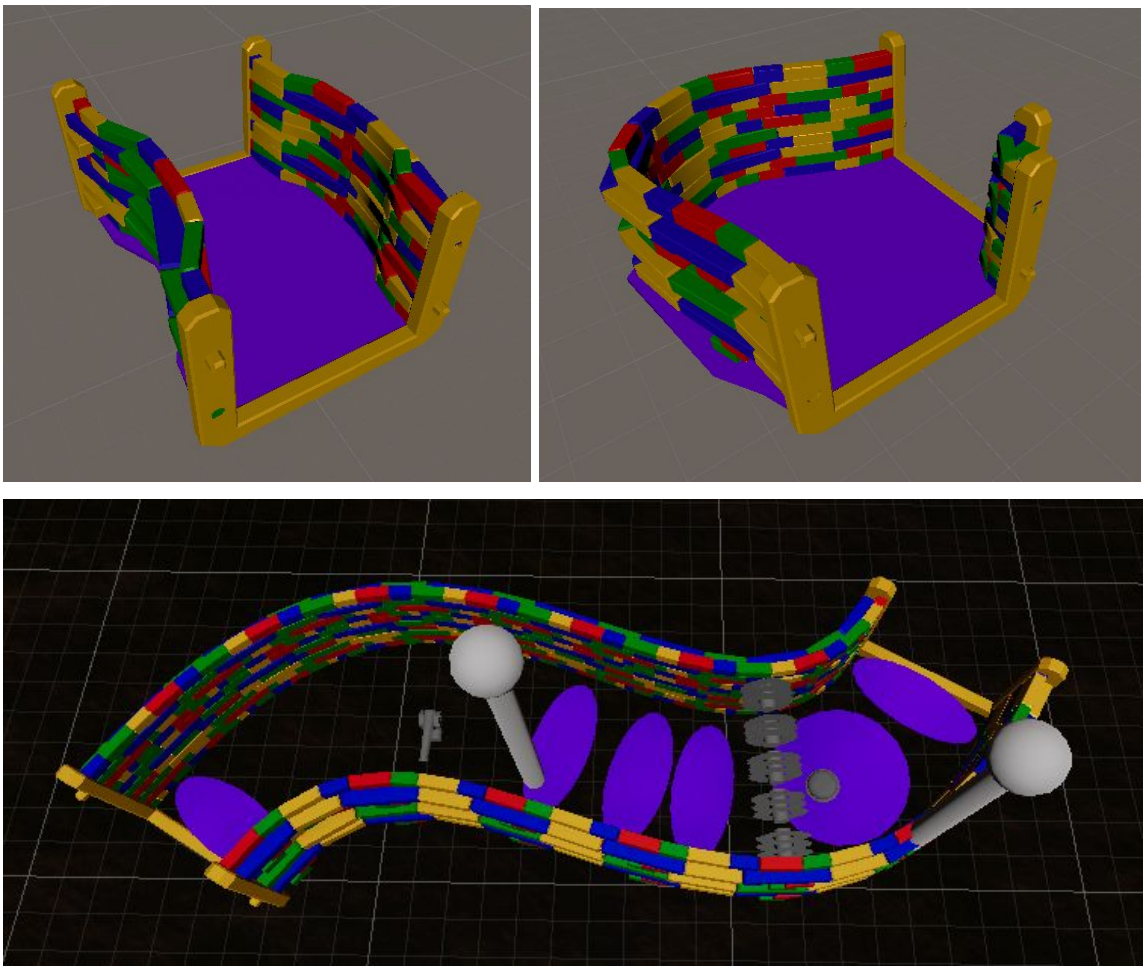


Figure 3.5.1a: The hallway sections

3.5.2 Rooms

The four room types the VR player has access to are a small combat room, a large combat arena, the end goal room, and a portal to a boss fight, depicted in Figure 3.5.2a. Both types of combat rooms function in the same way from the VR player's perspective. The VR player simply places the room, and when the controller players enter, they will be unable to leave until all enemies are defeated. From the controller players' perspectives, the small combat room immediately confronts the players with waves of enemies to fight. For the combat arena, the controller players are first presented with an elevator they all must enter in order to rise into the arena, but there are more waves of enemies they must fight before they can leave. The final goal room is straightforward as well, simply awarding the controller players with a victory screen once they enter.

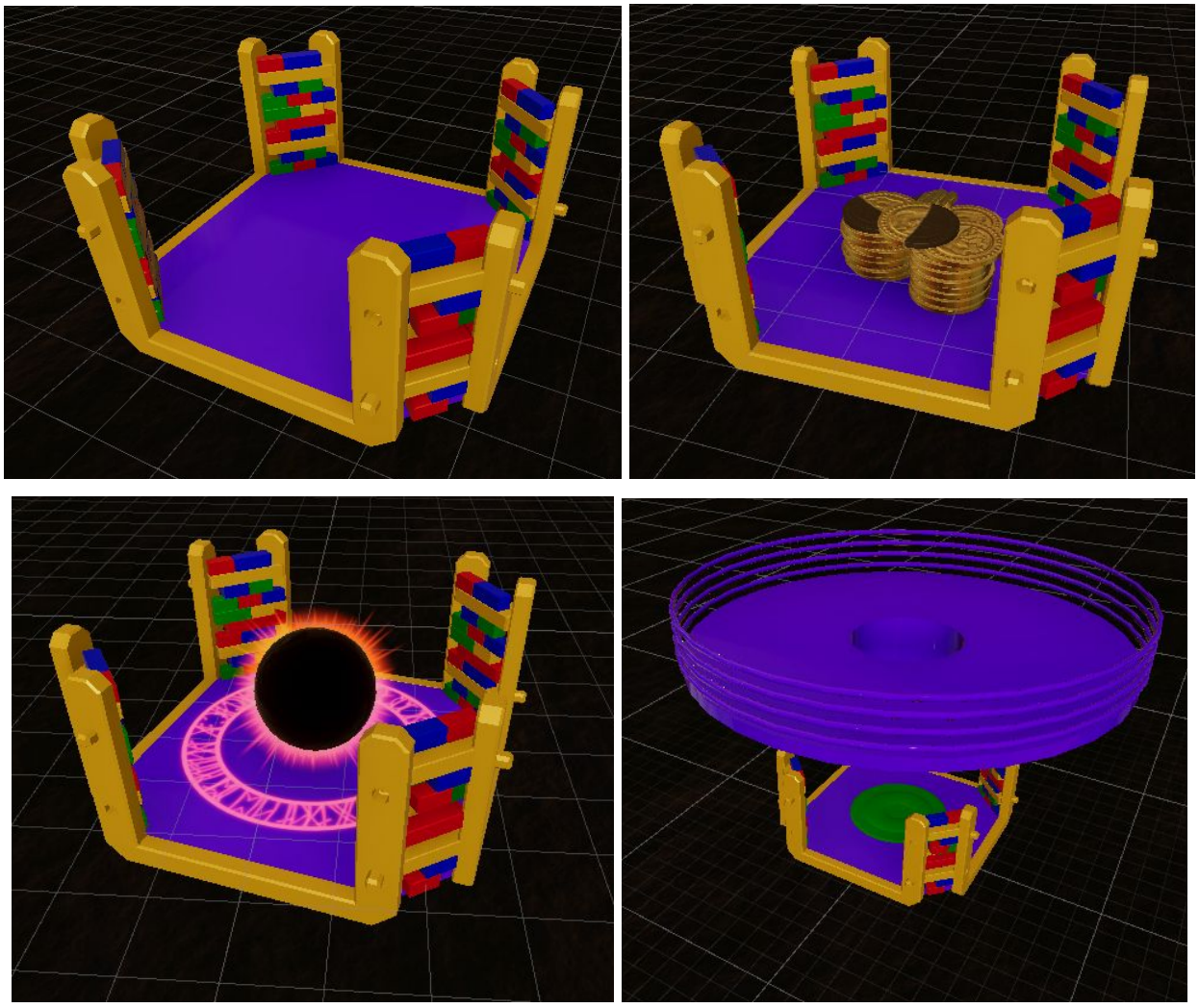


Figure 3.5.2a: The room types (from upper-left to lower-right: small combat room, goal room, boss room, arena room)

One special type of room that the VR player can place is a boss room. Once the controller players enter this room, all players are teleported to an open arena, which is actually a table located within the VR player's environment. Here, the VR player takes control of a dragon and is able to have a first-person battle with the other players. The boss fight is described in section 3.6.

3.6 Gameplay Loop: VR

Gameplay for the VR player begins once they put on the HTC Vive. While the controller players configure options and select characters, the VR player can immediately begin building the level. The level is built out of placeable hallways and rooms, which snap together at doorways. Hallways provide two openings that serve as attachment points, while rooms provide four. Any non-connected openings of placed sections are automatically blocked by a fog wall, to

prevent players from falling out of the map. These fog walls were inspired by the *Dark Souls* [16] games by FromSoftware. An example of a completed dungeon is shown in Figure 3.6a.

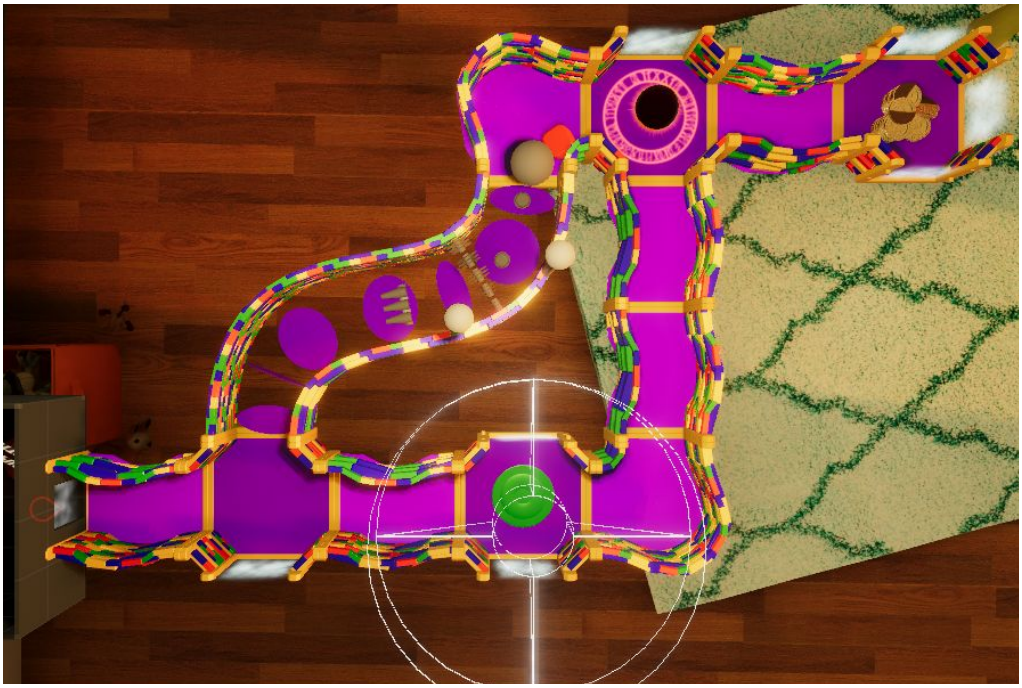


Figure 3.6a: Completed dungeon with two pathways through

Once the controller players join the game, the VR player might choose to observe them as they make their way through the level, or to keep building ahead of them. When the controller players navigate through platforming hallways, the VR player may also choose to interact with the traps in an attempt to hinder the controller players, using controls such as the lever shown in Figure 3.6b, which moves the saw blade trap seen in the hallway.

Figure 3.6b: VR player saw blade trap control (lever)

The VR player can also place a boss room, which, upon the controller players entering it, grants the VR player direct control over a boss enemy. As the boss, the VR player has three abilities at their disposal. They can perform slash attacks by simply swinging their arms, causing their in-game model's arms to mirror their actions. If the arms are swung fast enough, damage is dealt if the arm collides with a controller player. This ability is shown from the VR and controller player perspectives in Figure 3.6c.

Figure 3.6c: Boss slash attack (left: VR perspective, right: controller perspective)

Their second ability is to flap their arms, causing the in-game model to hover above the ground and fly forward, closing distance to other players, as depicted in Figure 3.6d from the controller player perspective.

Figure 3.6d: Boss flying ability

Finally, by pulling the triggers on both motion controllers, they can breathe a stream of flames, which deals damage to players on hit and can more easily reach players who try to keep their distance from the boss. This ability is illustrated in Figure 3.6e.

Figure 3.6e: Boss fire breath attack (left: VR perspective, right: controller perspective)

The VR player may choose to place the goal room at any point, in order to adjust the difficulty. However, after five rooms are placed, other room types will become unavailable, forcing them to place a goal rather than make the level go on endlessly. This constraint facilitates fairly short game length, allowing players to change roles frequently, and VR players to experiment with different level designs.

3.7 Gameplay Loop: Controller

Gameplay for the controller players begins with the character selection screen, where players can view the different characters and their abilities. Once all players have selected their characters and choose to start the game, they are dropped into a toybox starting area. If the tutorial option is activated, a fog wall will cover the starting area's exit, and each of the players must go over to a button on one end of the toybox and jump on the button a few times, with each button press briefly explaining a different aspect of the selected character. After each player has finished going through their character's tutorial, the fog wall lifts and the players are able to go into the maze. The tutorial, displaying a message for Dr. Bear, can be seen in Figure 3.7a.

Figure 3.7a: Controller player tutorial

Once in the maze, the players encounter a number of enemies, arena rooms, and trap-filled hallways. With the enemies, the players must use their unique abilities to fight back, and can often utilize the winding nature of the maze to backtrack and separate enemies to make the fight easier. In arena rooms, however, players ride an elevator to an enclosed space where they must defeat all of the enemies in order to return to the rest of the maze. In the trap hallways, players need to carefully navigate the path, utilizing platforming techniques. If a player falls or dies during one of these trap areas, they respawn at the end of the section. However, players that complete the hallway without falling are rewarded with healing. Throughout the maze, if any player reaches zero health, they are knocked down, and must be revived in order to continue playing. They may choose to revive themselves, although each player can only do this twice, or wait for an ally to revive them, which has no limit but is interrupted if the reviving player takes damage. If the downed player is not revived within a time limit, they die permanently, and have to sit out the rest of the game.

Another element within the maze is the boss room. Once the controller players enter this area, they are teleported to a different section of the map where they must face a particularly strong enemy in an open arena, as seen in Figure 3.7b. This enemy is controlled by the VR player, and its abilities are discussed in Section 3.4. Once the boss is defeated, players will be teleported back into the maze.

Figure 3.7b: Controller player perspective of boss fight

The last bit of gameplay for the controller players is finding the victory room. Once this room has been entered, “victory” will appear on-screen and the players will return to the character selection screen.

3.8 End Condition

Win conditions differ for the VR player and the controller players. The win condition for the VR player is intentionally vague, in order to best fit the experience that the VR player wants to create. If they are treating the game competitively, they may consider winning to be when the health for each of the controller players fall to zero, and losing to be when they reach the goal room. The VR player could do this through placing trap-filled hallways, boss battles, and enemy encounters. They could also simply want to make a fun, challenging experience in which the controller players ultimately succeed.

In order for the controller players to win the game, they have to battle their way through the VR player’s maze and make it to the victory room. If the health for all of the players falls to zero, they lose. When either end condition is reached, the controller players return to the player selection menu where they can choose to play the game again. In a party setting, this would be the chance for another player to take the role of the VR player, and begin building the level anew.

4. Artistic Implementation

This section describes the artistic techniques and tools used in developing *Babies & Basilisks*, including in-depth looks at asset pipelines for characters and environments. Section 4.1 describes the complete character production pipeline, beginning with conceptualization (Section 4.1.1), and moving through modeling (Section 4.1.2), texturing (Section 4.1.3), retopology (Section 4.1.4), rigging (Section 4.1.5) and finally animation (Section 4.1.6). Section 4.2 describes the steps used to create static (non-animated) objects including environments and props. This section also begins with conceptualization (Section 4.2.1), and moves on to modeling (Section 4.2.2), UV unwrapping and texturing (Section 4.2.3), and finally lighting (Section 4.2.4).

4.1 Character Production Pipeline

The art asset pipeline for characters and their animations, shown in Figure 4.1a, started with a quick conceptualization period before moving forward with the sculpting of the characters' models in ZBrush. Once a model was fully sculpted, it was textured and optimized using additional ZBrush features before being rigged in 3DS Max. After the rig was fully set up and skinned to the model, the character was animated in 3DS Max. These animations were then exported as FBX files and imported into Unity.

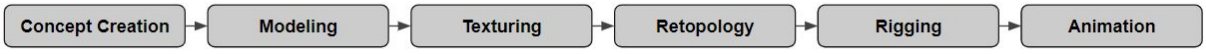


Figure 4.1a: Character Production Pipeline Diagram

4.1.1 Conceptualization

The character production pipeline began with gathering reference images of vintage toys and pre-existing stylized 3D models in order to better visualize the styles and forms of the characters. These images were used throughout the rest of the process. As an example, the toy designs shown in Figure 4.1.1a helped guide the design process for the kangaroo character.

Figure 4.1.1a: Wind-Up Kangaroo References [25][26]

4.1.2 Modeling

After conceptualization, the base meshes were created in Zbrush using sphere chains. With the rough base and shape created, the model could then be polished, and musculature and details could be sculpted. After the entire body for a character was finished, clothing and other organic pieces could be extracted from this base mesh, then edited slightly to look more accurate. Additional hard-surface pieces, such as metal keys and screws, were also created in Zbrush, using IMM brushes. Figure 4.1.2a shows the process from initial base mesh creation to polish and hard-surface details on Doctor Bear.

Figure 4.1.2a: Sculpting Process for Doctor Bear (Left is rough, Middle is polished, Right is polished with extracted clothing)

One other step for some of the characters was a process the team called “toyification”, which helped make the models appear more like physical toys. This was especially important for action figure-esque models, since they needed to have visible joints and connector pieces. This process was accomplished by cutting apart and softening split edges on these models, as seen in Figures 4.1.2b and 4.1.2c.

Figure 4.1.2b: Pre-Toyification Amos (Left) & Amos with Jointed Limbs (Right)

Figure 4.1.2c: Close-up of Amos’ Jointed Arm Sockets

Additional toyification on characters was completed by adding small plastic detailing, such as raised seams, higher level of surface polish, surface noise, and cleaner edges, as shown in Figure 4.1.2d below.

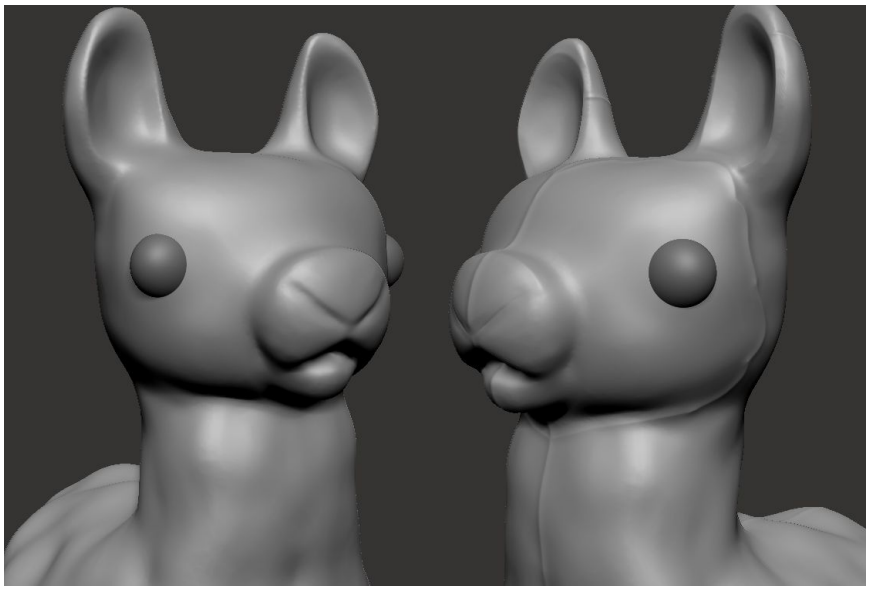


Figure 4.1.2d: Pre-Toyification Llama (Left) & Llama with Subtle Plastic Ridges (Right)

The only character model that did not follow this exact process was the VR player model (the baby). We used reference images of animated babies - such as Jack Jack from *The Incredibles I & II* [13] in Figure 4.1.2e and the various babies from *The Boss Baby* [14] in Figure 4.1.2f - to determine a stylized yet human look that would fit in with the toy-filled world.

Figure 4.1.2e: Jack Jack from Disney's The Incredibles II (2018) [13]

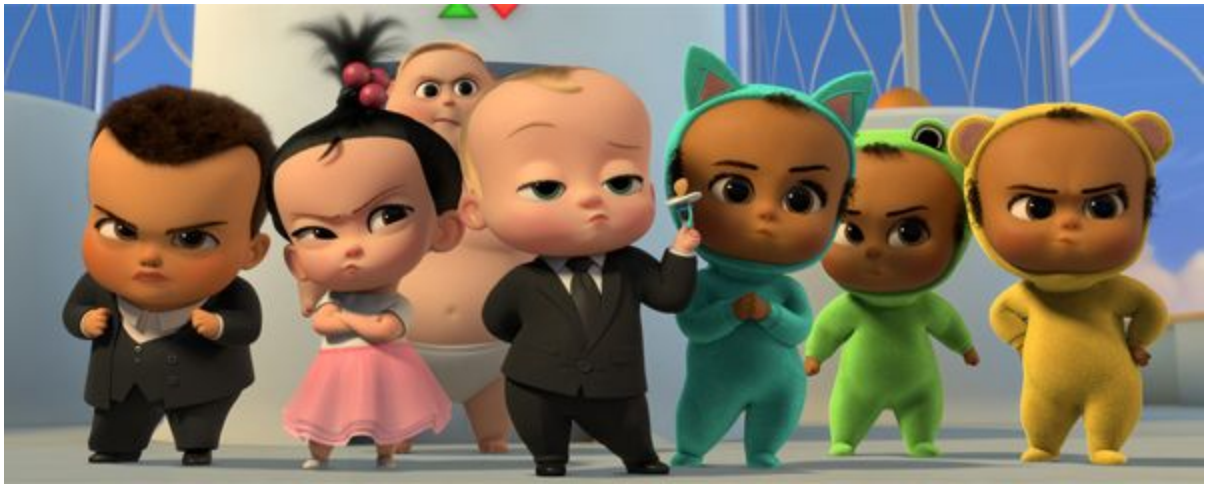


Figure 4.1.2f: Babies from Dreamworks' The Boss Baby (2017) [14]

Although the VR model did not need toyification, special considerations were required for its representation in VR. Because most players have vastly different proportions from a baby, the model would not be able to accurately track their movements. In order to avoid feeling disconnected, the team designed two versions of the baby model, one with the full character mesh, and another with just the hands. The full third-person mesh was used in the controller-character views, while the first-person hand models were used in the VR set-up; these meshes can be seen in Figure 4.1.2g.

Figure 4.1.2g: Third-Person Mesh (Left) & First-Person Mesh (Right)

4.1.3 Texturing

Once an entire character mesh and its pieces were generated, texturing in Zbrush could begin. This process began with basic color blocking to see how the various character designs would flow together. Once the basic look was satisfactory, secondary and tertiary color forms were added, as well as basic shading to give a slightly more realistic look. Depending on the model, the texturing process also involved generating surface noise on a few parts of a model. This gave the mesh a bumpier look, and was especially useful with characters that had cloth elements, such as Doctor Bear in Figure 4.1.3a.

Figure 4.1.3a: Surface Noise & Texturing on Doctor Bear

4.1.4 Retopology

After all of the assets were generated for an individual character, they were retopologized with Zbrush's Zremesher tool. Zremesher quickly reduced the polycount from a few million polygons to between five and ten thousand and produced clean edge loops, making the mesh friendly for rigging purposes. Following the Zremesher process, textures from the high poly version of the model were projected onto the low poly mesh, transferring surface details along with it. This projection step resulted in diffuse maps for the model, which could then be used to create normal, or bump, maps as well. Figure 4.1.4a below shows the difference between the high-poly sculpt and the optimized, game-ready version.

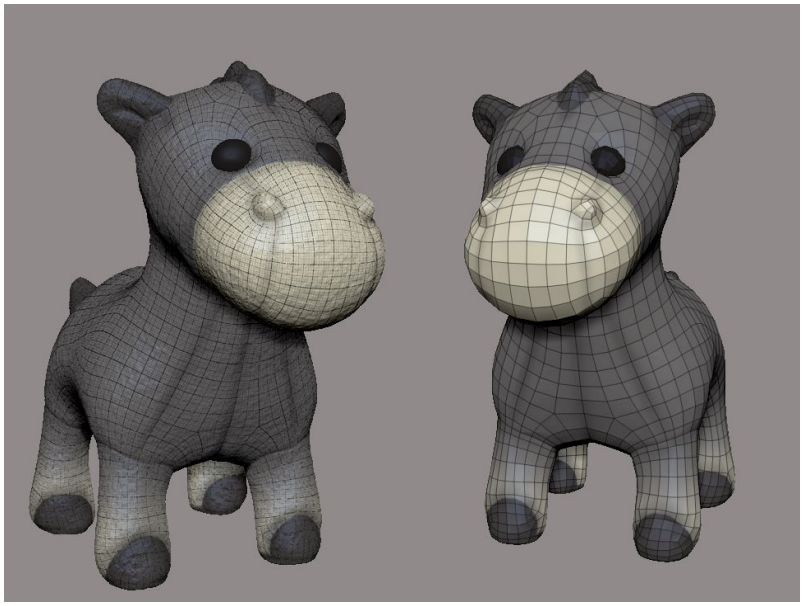


Figure 4.1.4a: High Poly of 2.8 Million (Left) versus Low Poly of 3000 (Right)

4.1.5 Rigging

Once characters were fully optimized, they were rigged using 3DS Max's CAT Rig system. This system allowed for relatively quick rig setup, and easy transfer of bones and other rig data into Unity. After setting up the base CAT rig, characters were skinned according to how many meshes they were comprised of. For instance, characters comprised of a single mesh, like Tess, could have all of the bones skinned to one model, whereas characters made of multiple, jointed sections, such as the Arm Sharks or Amos, needed to have certain bones skinned to certain sections of the mesh. Once the bones were skinned to the character models, their envelopes were tweaked in order to guarantee proper mesh deformations. Once this time-consuming process was completed, an animation layer was added to the system, and the model was ready for animating. An example of the rigged Arm Shark can be seen in Figure 4.1.5a.

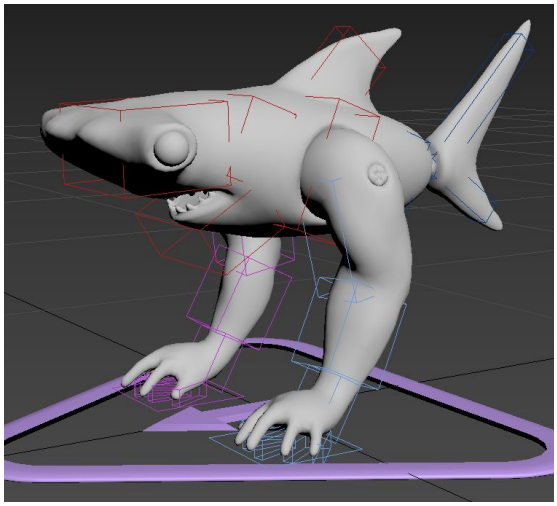


Figure 4.1.5a: Rigged Arm-Shark Using 3DS Max's CAT System

4.1.6 Animation

Once the rigs were completed, they were animated in 3DS Max, as seen in Figure 4.1.6a.

Figure 4.1.6a: Animated Amos in 3DS Max

The toy designs were incorporated into the animations, providing additional characterization for the enemies and player characters. For instance, since Doctor Bear is a plush character, his animations are a bit looser and floppier than the animations for a rigid character such as Amos.

Each character needed a specific set of animations, but some were common among all characters, such as idle and walk animations. An example moveset for Doctor Bear can be seen

in Figure 4.1.6b. These animations were exported as FBX files and imported into Unity ready for use in-game.

General Animations	Walk Cycle	Run Cycle	Idle	Jump

Attack/ Special Animations	Roar	Tackle	Healing Blanket

Figure 4.1.6b: Doctor Bear Character Moveset

4.2 Environment & Prop Production Pipeline

The pipeline for the environmental art, shown in Figure 4.2a, began with conceptualization and grey-boxing the environments in Autodesk Maya and 3DS Max. This grey-boxing process gave the team a visual representation of the room's basic geometry. Furthermore, it helped determine how large of a space was needed inside of Unity and where objects should be positioned for the most logical game flow. Further into the process, placeholder textures were applied to these environments to get a better idea for how the space would feel textured. After the grey-boxing stage, the placeholder geometry was replaced with more eloquent models. Once all the models for the environment were created, they were UV unwrapped and textured. Following this step, the environment was placed inside Unity for lighting setup.

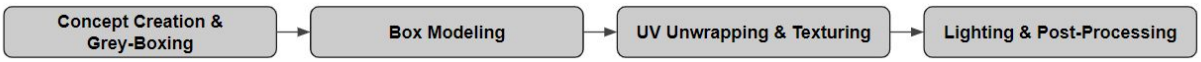


Figure 4.2a: Environment & Prop Production Pipeline Diagram

4.2.1 Conceptualization & Grey-Boxing

The environment and prop production pipeline began with gathering reference images for the type of space the team wanted to create, as shown in Figure 4.2.1a.

Figure 4.2.1a: Room Inspiration[24]

This was quickly followed by conceptualization of the space in Maya through a process called grey-boxing, which allowed for the team to get a general sense of the environment and props before refining. Once the initial grey-boxed environment shown in Figure 4.2.1b was created, it was transferred over to Unity for some basic testing, mostly to see how the layout felt in-game. The conceptualization process also involved looking up examples of playrooms for color inspiration, which the team then tested out by applying placeholder textures to the grey-boxed models.

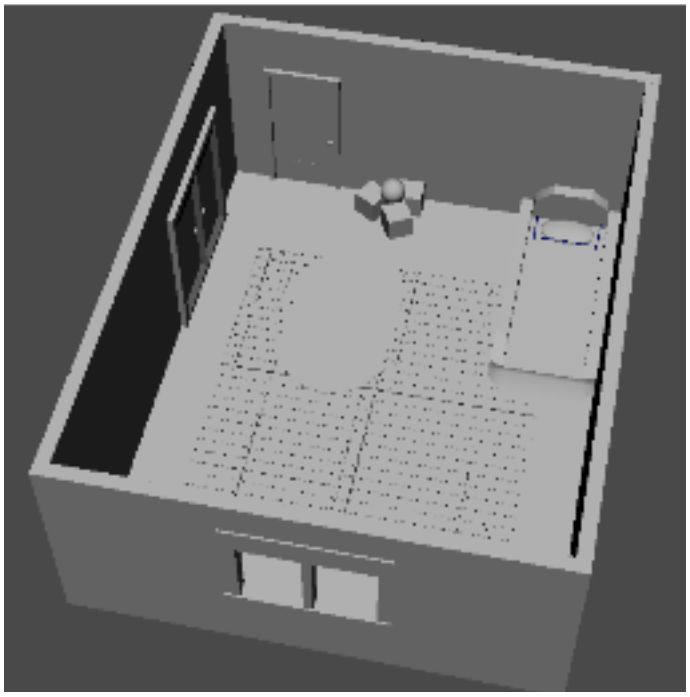


Figure 4.2.1b: Preliminary Grey-boxed Base Environment

This conceptualization and grey-boxing process went through several iterations over the course of the project in order to test different furniture layouts and architectural structures. The final room concept, shown in Figure 4.2.1c below, took substantial influence from the room layout shown previously in Figure 4.2.1a, and featured a playroom rather than a bedroom, with a prominent window nook area as a focal point for the environment.

Figure 4.2.1c: Final Grey-box Concept

4.2.2 Modeling

The next step of creating the game's environment was to replace the grey-box models with more detailed, visually appealing models, as seen in Figure 4.2.2a.

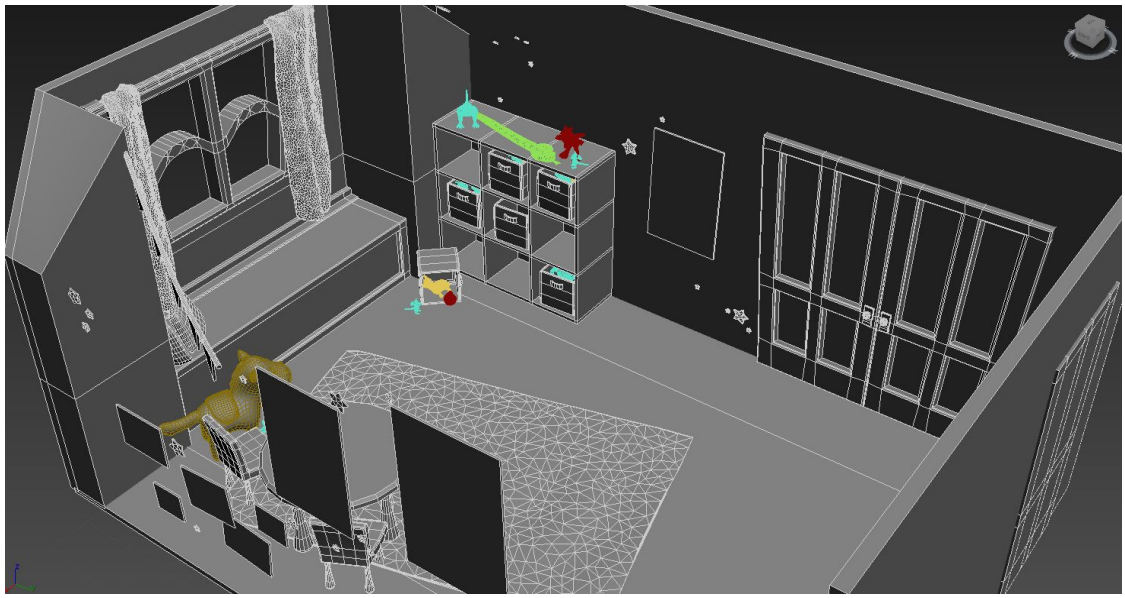


Figure 4.2.2a: Wireframe View of New Models

The team used box-modeling tools in Maya and 3DS Max to shape the various furniture and architectural pieces for the room, some of which are shown in Figures 4.2.2b and 4.2.2c. These models were then imported into Unity in a single FBX file, automatically creating a placeable prefab of the entire room.

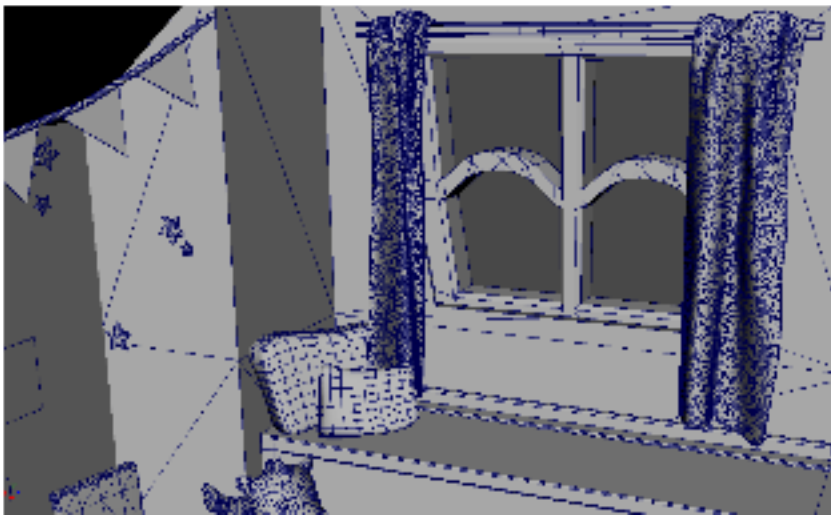


Figure 4.2.2b: Wireframe View of Window Nook Area

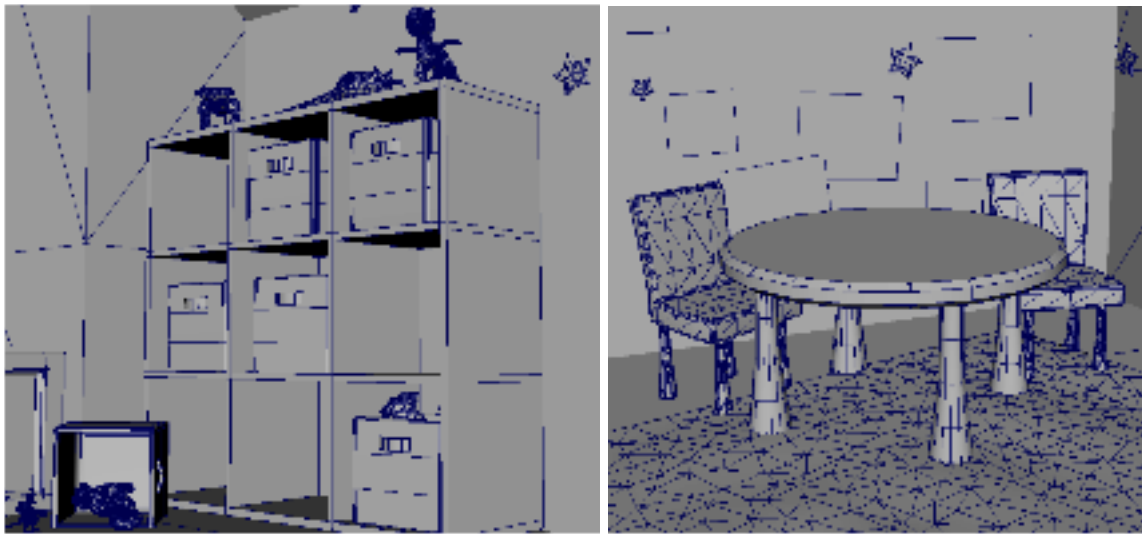


Figure 4.2.2c: Wireframe View of Toy Storage (Left) and Drawing Area (Right)

4.2.3 UV Unwrapping & Texturing

Before texturing could begin, the art team unwrapped the UVs for each model using Maya's automatic UV unwrap tool. The unwraps were then rotated and repositioned in the UV space in order to best utilize the texture space available. Following the UV process, the art team used a combination of Photoshop and GIMP to create the texture sets for the environment pieces. Much like in the character texturing process, environment texturing began with basic color blocking and planning in order to create the desired atmosphere. In order to create a playful, childlike feeling throughout, the team decided on a bright color palette with orange, teal, and white. Secondary colorful touches were added through textures of posters and children's drawings.

4.2.4 Lighting

The last step for creating the environment was adding lighting in Unity. Because *Babies & Basilisks* is a VR game requiring a steady 90 frames per second in order to work properly, the team had to be particular about the lighting, sticking only to baked lighting rather than real-time lighting which would be found in most games. The team used a spotlight for the primary "sun" effect in order to have more control than with a directional light; this light had a warm orange glow to it, and was angled to mimic a sunset. For additional lighting, ivory and light green point lights were placed around modeled light fixtures and small glow-in-the-dark stars. Once the lights were placed, post-processing effects were added, including color grading to improve the contrast range of lighting values, and ambient occlusion to create pooled shadows and further atmosphere. These lighting changes, as seen in Figures 4.2.4a and 4.2.4b, greatly improved the overall visual feel of the game.

Figure 4.2.4a: Before Lighting & Post-Processing

Figure 4.2.4b: After Lighting & Post-Processing

5. Technical Implementation

This chapter describes the technical systems that were implemented as part of *Babies & Basilisks'* development, and the evolution of each system. Section 5.1 provides a summary of these systems and the overall process of implementing them. Section 5.2 describes the player control systems, both for controller players (Section 5.2.1) and the VR player (Section 5.2.2). Section 5.3 details the VR building system and inner workings of modular components. Section 5.4 discusses the development of AI to suit each type of enemy. Section 5.5 describes the shared camera system used by controller players. Section 5.6 elucidates the tutorial systems that teach both the VR player (Section 5.6.1) and controller players (Section 5.6.2) how to play. Finally, Section 5.7 details the creation of an auxiliary audio system using a third-party library.

5.1 Overview

Several technical aspects were vital to the design and implementation of *Babies & Basilisks.* The player controls, for both the controller and VR players, provided challenges in creating simple and intuitive controls for the players. The VR player's building system was required for the mazes to be built effectively and in ways that made sense, such as preventing hallways from overlapping each other. The AI of the enemies the controller players had to face was also important, as it defined the combat experience of the controller players. Camera systems were carefully implemented to balance provide interesting views that were also effective in showing important parts of the scene. A tutorial system was created as well, in order to help the players understand what their controls were. In addition to this, a custom audio system was created to provide different audio output to separate sources.

For the process of implementing the technical aspect of the game, the technical team divided the responsibilities and implemented them in separate branches of source control, rebasing the branches into the master branch when implementation was complete. Before rebasing, each branch was reviewed and checked for bugs by the members of the technical team. This minimized bugs in each iteration of the game, as well as providing the technical team familiarity with all aspects of the code, even ones they had not written themselves.

5.2 Player Controls

5.2.1 Controller

The main challenge of designing the controls for the controller players was to make the characters intuitive to control, and the inputs similar enough to facilitate easily switching between characters across playthroughs.

5.2.1.1 Receiving and Interpreting Input

Each character's three unique abilities were implemented as methods that overrode an abstract attack script. Each of these abilities was assigned a cooldown timer which prevented the players from using that ability again until the cooldown time had passed. To receive the player input, input axes were entered into the Unity Input Manager for each button and control stick across the four controllers, as well as input axes that corresponded to any controller to allow for menu navigation. The Input Manager also included axes to receive input from the keyboard and mouse, for players who did not wish to use a controller. All of these axes were differentiated by adding a suffix with the controller number to the name. The input axes are read by a PlayerController script, which is assigned to one suffix. The controller then uses the values it reads to update a Channels object, which serves as intermediary storage between Controllers and Motors. Various Motor scripts interpret the values in the Channels, allowing them to move and perform actions. By using the same channels file but a different motor file specific to characters' abilities, characters could react to the same forms of input in different ways. Each character was also assigned a second motor script for movement, and this script was the same one across all characters so the movement would be calculated in similar ways.

With the initial design of the movement system, players could move using the left joystick of a controller or the WASD keys on the keyboard, and could jump with the A button or spacebar. Each of the characters' three abilities could be activated with the Q, E, and R keys and the left, right, and center mouse buttons for keyboard users, or the X, Y, and B buttons for the controller players. The appropriate buttons for the abilities are shown at the bottom of the screen for each player along with icon indications of each ability, as shown in Figure 5.2.1.1a.

Figure 5.2.1.1a: Controller Player Ability UI

It was found during the team's own testing of the game, and during formal playtesting, that the two characters with ranged attacks, the tank (Tess) and the ranged (Amos) characters, were difficult to aim with using this design, as the player had to make the character physically move in the direction they wanted before they could use their abilities in that direction. For these ranged characters, aiming independently of motion was added, with the use of the mouse or arrow keys for keyboard controls and the right joystick for controllers. As this was modified, the input for the abilities for all characters was updated to be mapped to not only the face buttons on

the controllers, but the back buttons as well, allowing players to aim and attack without removing their finger from the aiming stick on the controller.

5.2.1.2 Disabling Input

During some events in the game such as waiting for a hallway's moving camera to begin, being grabbed by an arm shark enemy, or while amid certain player abilities, player input must be disabled. Initially, the controller characters' inputs were manually enabled and re-enabled within the specific instances of the code. However, with this approach, multiple unrelated pieces of code could disable the movement controller at the same time, and re-enable it at times when it should be disabled. For instance, if the healer character, Dr. Bear, were in the middle of a tackle attack, thus disabling input for a moment, and was simultaneously grabbed by an arm shark, the arm shark's grab would also disable the input. This input should remain disabled until the arm shark drops Dr. Bear. However, when the tackle animation completed and he was still in the Arm Shark's grip, that Motor would re-activate, allowing Dr. Bear to move about even though the arm shark should still restrain him. To prevent this from occurring, a separate script to manage the player characters' input was added to all controller characters. With this script, any code attempting to enable or disable the player's input does so by calling functions in this script. With this modification, the script can keep track of when the input should and should not be disabled, and allows the player input to always be appropriately enabled or disabled, even with multiple influences on it.

5.2.1.3 Implementing Abilities

During development, character abilities were first roughly implemented using a variety of timers to track how long an attack should last, placing static colliders around the player objects, and using placeholder objects for abilities such as Dr. Bear's healing blanket. As the art team created animations for the characters, the abilities of each character were further refined, using Unity's animation events to trigger functions to enable and disable the attack colliders or player input during the use of their abilities, as well as parenting the colliders to the player characters, so enemies would be damaged in line with the character's animation. With these added, it was easier to test how balanced the abilities were, allowing for tweaks in the length and power of the attacks. Additionally, the code was updated to be more elegant, relying on animation frames calling the functions rather than timers which had to be checked every update frame to see if they had expired.

5.2.2 VR

In designing the controls for the VR player, there were additional considerations that had to be made. VR games typically accomplish movement through a combination of the room-scale tracking capabilities of the headset, using a trackpad or joystick to move the character as in a

traditional first-person game, or pressing a button to teleport. Of these options, room-scale tracking was found by the team's personal experience to be the least likely to induce discomfort. However, the movable area in the game exceeds the maximum tracking area of the Vive, which is four by three meters. Due to the size of the area in which the VR player must move, an additional movement option was necessary. Another challenge was to map the movements of the VR player to a 3D avatar in the game, which would be visible to the controller players. With only three available points of tracking (the headset and two controllers), some guesswork was used to achieve believable animation on this character. Finally, there was the question of how the VR player would interact with various elements in the game world, including placeable objects and a boss enemy that could be directly controlled.

5.2.2.1 Movement and Avatar Tracking

Initially, the VR player's only means of locomotion was physically moving around in their tracking area. However, their size relative to the in-game room is much smaller than the actual player's size relative to the maximum tracking area of the Vive. As a result, only a small part of the in-game room could be traversed using natural head-tracking movement. To remedy this, a teleportation system was added, allowing the VR player to point at the ground while holding the thumbpad on the controller, then release the button to teleport where they are pointing, depicted in Figure 5.2.2.1a. While this system breaks the immersion of a VR game somewhat, the team played multiple VR games and found that it caused less discomfort than joystick-based movement.

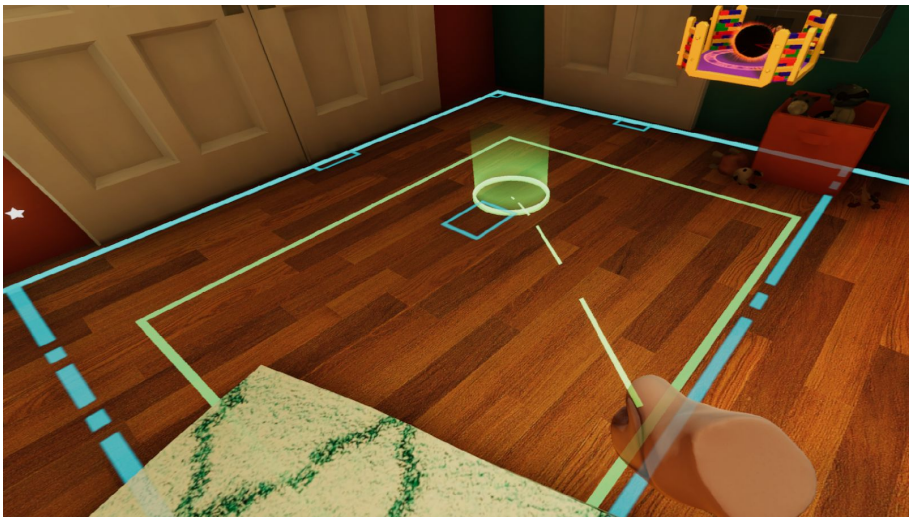


Figure 5.2.2.1a: VR Teleportation

For the VR player's avatar, a placeholder humanoid rig was used, which was set up to follow the horizontal position and rotation of the headset. The avatar would also blend between standing and crouching poses based on the height of the headset, attempting to keep its head at the same height as the player's head. Its hands were set up to track the location and rotation of

the Vive controllers using the inverse kinematics system provided by Unity for humanoid avatars. An example of the VR player in a crouching pose with hand tracking can be seen in Figure 5.2.2.1b. Details about the interaction system may be found in the “VR Building System” section.

Figure 5.2.2.1b: VR Avatar Tracking Hands and Height

5.2.2.2 Boss Controls

The initial design to control the boss enemy was a diegetic user interface resembling a remote control, and consisting of a joystick for movement and four buttons for performing attacks, as seen in Figure 5.2.2.2a. This input system was automatically created and attached to the boss as soon as the players reached it.

Figure 5.2.2.2a: Old Boss Controls

In initial user testing, however, it was found that this method of controlling the boss was unintuitive, and that more direct control might be better. For this reason, the existing avatar tracking system was adapted to enable the VR player to become the boss and control it directly. Instead of pressing buttons to perform different attacks, they could slash with their arms for basic attacks, or perform various gestures to trigger special attacks. While various attacks and gestures were considered, it was eventually decided that the boss would have one additional attack, the ability to breathe fire, as well as the ability to move by flapping their arms. The fire breath was initially triggered by pulling the triggers on both controllers and holding them behind one's head; however, this proved to be unintuitive to testers, so the gesture was reduced to just pressing the triggers. Each of these abilities are shown in use in the Design chapter in Figures 3.6c, 3.6d, and 3.6e.

5.3 VR Building System

Possibly the most crucial system from the perspective of the VR player was the building system. Since the VR player would be building the entire traversable area for the controller players, an intuitive and satisfying interface was needed. It was also essential to ensure the level the VR player creates is possible to complete, and free of places where controller players might be able to escape or notice inconsistent collision detection.

5.3.1 Hallway Snapping

It was decided that the map should be built out of rooms and hallways that snap together at specific points, rather than allowing the VR player to place individual walls and floors. Because of this, the collisions of each placeable section could be specifically designed and tested to line up well with adjacent sections. These map sections can be picked up by the VR player using the trigger on the Vive controllers. When they are held, they behave as a standard physics-affected object, allowing the player to drop or throw them. However, when held near another level section, they show a preview indicating that the two pieces can be snapped together. Releasing the trigger then causes the held piece to snap into place, as seen in Figure 5.3.1a.

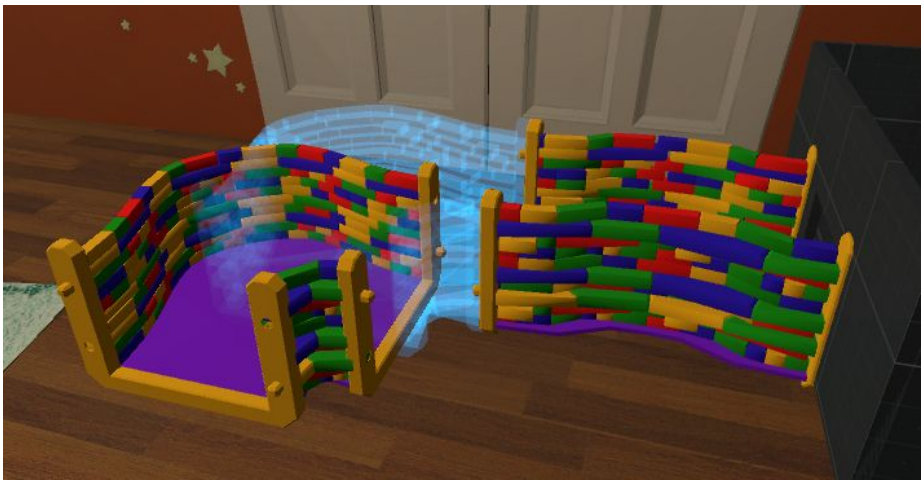


Figure 5.3.1a: A Hallway Section Snapping into Place

In addition to controlling which parts of these objects could snap together, it was also necessary to prevent two sections from being snapped into the same physical space. To accomplish this, a grid system was introduced, which divides the world into a uniform grid and keeps track of placed objects in each cell. Before a map section can be snapped into place, it must first ensure all necessary grid cells are empty and insert itself into these. This system is represented visually in Figure 5.3.1b.

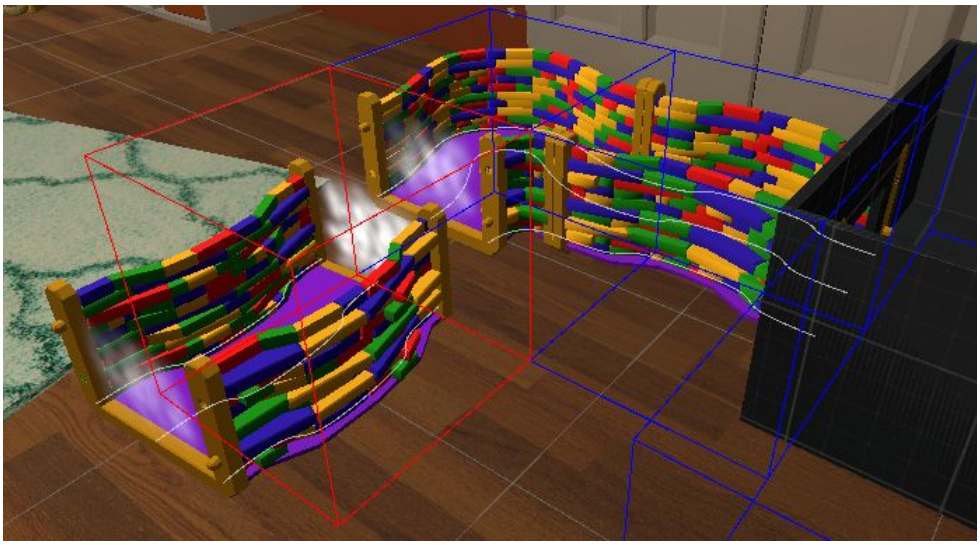


Figure 5.3.1b: Occupied Grid Cells (blue) and Required Cell for New Piece (red)

5.3.2 Shelf

The final piece of the building system was a mechanism from which the VR player receives these placeable sections. For this, a shelf was added to spawn each type of map section, and can be seen in Figure 5.3.2a.

Figure 5.3.2a: The Shelf

In the initial implementation, when the VR player picked up a placeable section and moved it a specified distance away, a new one would spawn. However, this led to a physics issue where new pieces could be spawned too soon, leading to a collision with the old piece. This sometimes happened without the VR player even touching the piece, causing new pieces to be repeatedly spawned as the old ones were pushed away. This could be remedied by increasing the distance that pieces needed to be moved in order to respawn, but that led to some pieces being placed too close to their spawn point, and a new one not spawning at all. Eventually, the shelf system was changed to only respawn sections after they were snapped in place, avoiding both previous issues.

5.3.3 Room Types

Since there were several room types planned (enemy encounter room, enemy arena, boss room, and victory room), spawning them individually would require the shelf to be quite large, and might confuse the VR player. To solve the problem, a piece of diegetic UI was introduced, similar to that used in the initial version of the boss controls, in the form of a small dial on the underside of a generic room template. The dial could be turned by the VR player to change what type of room they were holding. However, through testing, it quickly became apparent that turning the dial was unintuitive. Therefore, a more straightforward interaction was decided on, in which the VR player shakes the room once each time they want to change the type. The four types of rooms are shown in the Design chapter in Figure 3.5.2a.

5.3.4 Challenge Hallways

Finally, it was decided that the basic hallway sections would probably bore the controller players, as they failed to offer any challenge. To remedy this, a new type of hallway was

introduced. These new hallways contain a generated sequence of traps and platforms, requiring the controller players to carefully jump across as they avoid hazards. To create these hallways, the artists first modelled simple block-based components, as seen in Figure 5.3.4a.

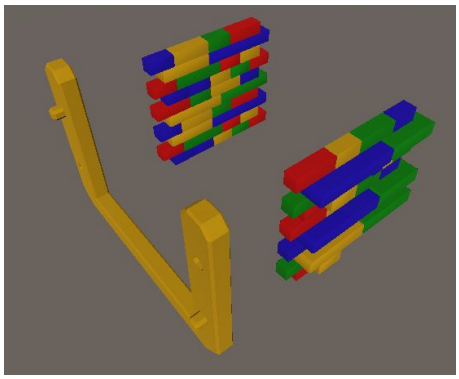


Figure 5.3.4a: Original Spline Hallway Components

In order to give the hallways a more interesting shape than a simple straight line, the components were duplicated and deformed along a spline, creating a visually-appealing curve, seen in Figure 5.3.4b.

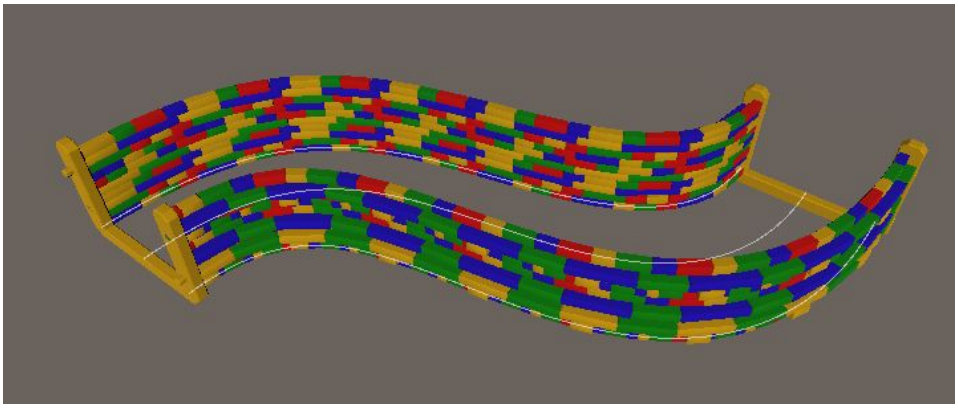


Figure 5.3.4b: Generated Spline Hallway

These shapes were generated dynamically using a Spline component, which was created by modifying code from the Cinemachine package. This script handled interpolation of points and rotations along the spline, and allowed for editing the control points and tangents using Unity's built-in transform tools. An additional component, the SplineMesh, was then created. This script would take as input a sequence of meshes and a reference to a spline. It would then copy each vertex of these original meshes, and deform the positions and normals based on the spline curvature. The output was a single render mesh and separate, simplified collision mesh.

The hallways' interiors were created by placing a generated sequence of traps and platforms along the spline. These include platforms that move or collapse when a weight limit is reached, as well as traps with sawblades, spikes, boulders, and flamethrowers. A fully-generated hallway is shown in Figure 5.2.4c.

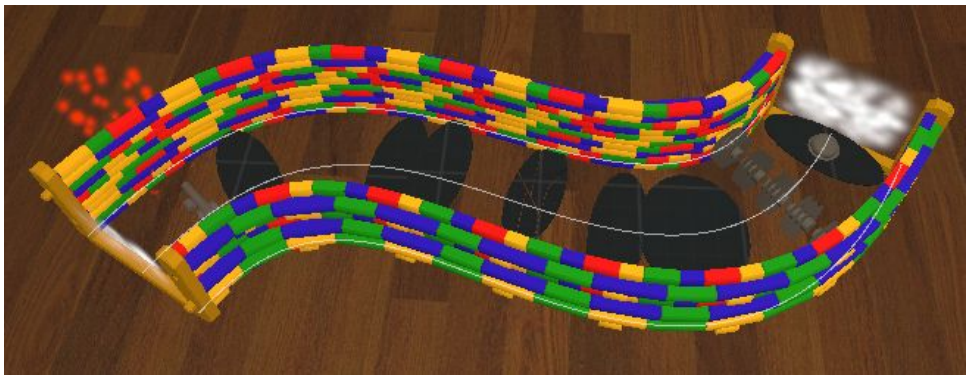


Figure 5.3.4c: A hallway with traps and platforms placed in it

Besides offering an additional challenge and a different type of gameplay from the rest of the maze, these hallways could also control the speed at which controller players progress through the level. This was accomplished using a camera that moves through the hallway at a fixed rate and prevents the players from leaving its view. A view of a controller player in the hallway is shown in Figure 5.3.4d.

Figure 5.3.4d: Controller player perspective in challenge hallway

In order to give the VR player more opportunities to interact with the controller players, a system was created so that they could control the traps in these hallways manually.
VR-intractable UI objects, including buttons, levers, and joysticks, had already been created for

the boss fight (see Section 5.2.2.2). These were repurposed to enable the VR player to activate the traps. These additional UI objects are shown in Figure 5.3.4d, and an example of the VR player interacting with them can be found in the Design chapter in Figure 3.6b.

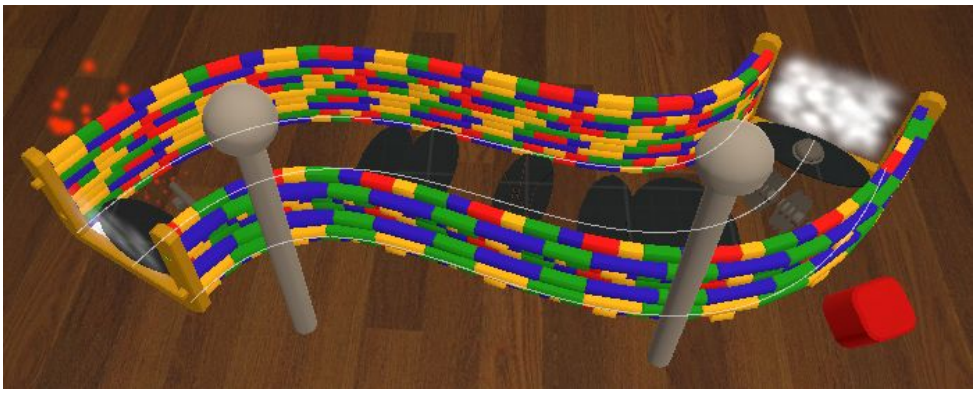


Figure 5.3.4d: The previous hallway with manual controls added

5.4 Enemy AI

In developing the behaviors of the enemies for the controller players to fight, the primary design considerations were to create a variety of interesting and noticeably different enemy behaviors which were balanced to operate in any combination against the players.

5.4.1 State Machines

For all AI enemies (as opposed to the VR player-controlled boss), behavior is defined in controller scripts which inherit from the EnemyController base class. This base class contains common methods used across different varieties of enemies, such as changing the targeted player or finding the rotation or distance to this targeted player. The behaviors specific to each enemy type were implemented using state machines. These state machines were created using a visual state machine editor developed by one of the team members, which automatically generated function templates to be called in each state, sub-state, transition, and transition check between states specified in the editor. These functions were then completed with the proper code for the desired behavior of the enemy at each state. Similarly to the player characters, each enemy also uses a channels file defining the inputs they can receive from the state machine controller script, with the values set in the controller being interpreted into movements and attacks with a motor script specialized to each enemy. One of the state machines, that of the llama enemy, can be seen in Figure 5.4.1a.

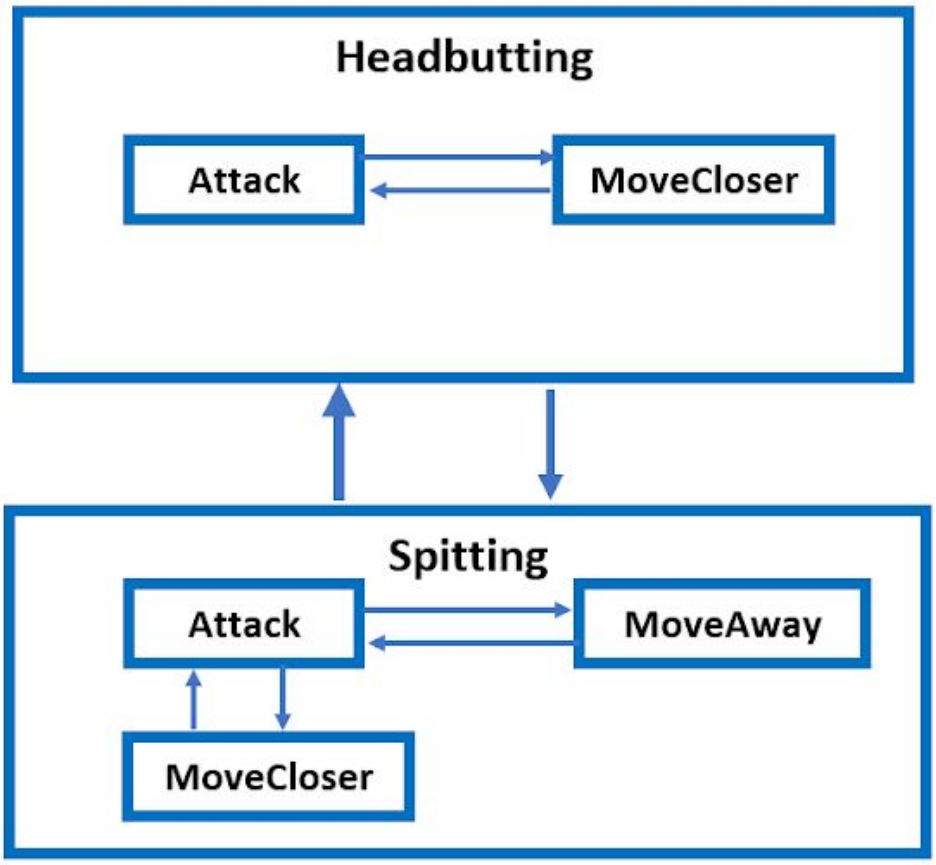


Figure 5.4.1a: Llama Enemy's State Machine

Each box represents a state, and boxes within boxes represent sub-states. Arrows show transitions between states. In this state machine, there are two main states: headbutting and spitting, representing what attack mode the enemy is in. Within these states, sub-states were defined, which the enemy would transition between to maintain a proper distance to the player it was targeting, and attack when possible. Since the headbutting state was an up-close, melee attack, the enemy was only concerned with getting closer to the player. However, with the spitting state, the enemy would attack at a range by shooting a projectile, and so would draw farther away if the player came closer, but also would follow the player so as not to get too far away. The llama would transition between the states of headbutting and spitting if the player became too close or far away. For instance, if the player continually runs away from the llama while in headbutting mode, eventually the llama falls behind and transitions to spitting mode. If it is in spitting mode and is backed into a corner, allowing the controller player to draw close, it switches to headbutting mode.

5.4.2 Implementing Attacks

To perform attacks, specific colliders are enabled and disabled at certain points in the animation, which deal damage to the players. As animations were added, some of these colliders were updated to be child objects of bones on the animation rig, allowing the collider to move with the animation and provide a more realistic path that damages the players. For ranged attacks, such as the llama's spit attack, projectiles were created and launched towards the player.

To transition between attack states, timers were initially used to check how long an attack had been occurring and to signal its end. This part was designed to be easily replaced by animations, which allowed for updated functionality using Unity's animation events. Instead of checking the time every frame, the events trigger functions at specific points in the animation. With this revision, the functions that ended the attacks could be called from the animation, and many timers could be removed from the code.

5.5 Cameras

One challenge present in many local multiplayer games is how to handle the camera. There are two broad approaches to this. The first approach uses a split-screen, where each player has their own camera, with a fraction of the screen dedicated to them. This approach is common in shooter games where the camera is tied to an individual character's aim, as seen in Figure 5.5a.

Figure 5.5a: Split-screen multiplayer in Call Of Duty: WWII (2017) [17]

The second possible approach is using a shared camera that automatically adjusts to keep all players on screen at all times, such as in the *Lego Star Wars* [18] games, as seen in Figure 5.5b.

Figure 5.5b: Shared camera in Lego Star Wars: The Complete Saga (2007) [18]

In deciding on which camera approach would be used, several factors were considered. The game would not have a focus on shooting, so precise aiming with the camera would not be needed. Furthermore, since VR imposes a substantial rendering cost, the additional rendering cost of the extra cameras would need to be kept down.

It was decided early on that a shared camera would be used. Since the game supports up to four players, using a shared camera as opposed to a split-screen reduced the rendering cost significantly. Additionally, using an automatic or fixed camera would reduce the amount of input required from the players, making the game easier to learn. To implement this shared camera, the "Cinemachine" Unity package was used. This package makes it easy to create a camera that tracks multiple targets, keeping them all on screen at the same time by moving or zooming.

5.5.1 Camera Types

Initially, a single top-down camera set to track all of the players was used, as seen in Figure 5.5.1a. During the boss fight, this top-down camera would add the boss to the list of objects to track. This initial camera effectively conveyed necessary visual information to the players and kept them all on screen, but failed to provide interesting angles.

Figure 5.5.1a: The top-down camera used in combat

In order to make the camera angles more cinematic and dynamic, it was decided that different types of camera would be used depending on the situation, using different angles and methods of tracking. For the hallways, a side-view camera provides players with a good view of platforms they can jump on, shown in Figure 5.5.1b.

Figure 5.5.1b: Side-view camera used in hallways

For combat scenarios, the top-down camera was kept, so that players could have a steady view of all threats. However, the tracking on these cameras was changed to be fixed with a constant view of the entire room, in order to avoid jarring camera zooms during quick movement. Finally, for the boss fight, an isometric view was decided on, in order to strike a good balance between providing enough information to the players and an exciting view of the action. These different cameras were specified on a per-room basis, with each activated when one player enters a trigger volume. The Cinemachine system created smooth transitions between these cameras.

5.5.2 Camera Issues

Some cameras required parts of the level to be hidden in order to avoid obscuring their view of the players. This problem was solved by separating the wall meshes from the rest of the rooms, and testing if a line from the camera to each player character intersects with each wall. If this test returns true, the wall is hidden. However, some issues of player visibility still arose with

the cameras. In particular, having any one player able to change the active camera by moving to another room was jarring, and caused the camera to lose focus on the majority of players. The issue was fixed by adjusting the camera to transition only when the majority of players enter a new room.

5.6 Tutorial

With a building system implemented for the VR player, it was necessary to teach them how to use it. Additionally, the controller players would need to be taught what challenges they might encounter in the level and how to deal with them. Due to the complex nature of the game, it would be ineffective to convey this information in a single screen. Furthermore, the nature of the game as a party game means that any number of players may be inexperienced, and that time spent teaching the mechanics should be minimized.

5.6.1 VR Tutorial

To accomplish this, an in-game tutorial was created that limits the VR player to building a specific sequence of map sections. This solution allows the VR player to observe what each section does, while also enabling the controller players to experience them one at a time in an order that makes sense. The tutorial works by creating one item at a time, and showing an on-screen text hint for the VR player to pick it up, as seen in Figure 5.6.1a.

Figure 5.6.1a: VR Player Text Hint

Once they do so, the hint moves to show where they can place the object. Some sections require additional input, such as shaking a room to change its type, or interacting with a lever to control a trap. Instructions for these actions are also shown as text hints.

After testing this system with users, some minor adjustments were made. It was found that the text hints were not noticeable enough, so they were made more evident by adding arrows pointing towards them when they were out of view. It was also hard for the VR player to judge how far away they were from the hint's location. This was improved by adding a line from the ground directly up to the text hint, so that the VR player could determine its position based on the floor. Additionally, a phone ringing sound was attached to the text hints, to aid the VR player in locating these hints when looking away from them.

5.6.2 Controller Player Tutorial

For the controller player tutorial, a button was added in the starting room, along with a text prompt for the first player to jump on the button. Each time it is jumped on, the button changes the displayed text to tell the player about one of their abilities, including a basic description and what button to press. This button is shown in the Design chapter in Figure 3.7a. More thorough descriptions of the abilities can be found on the character selection for players interested in learning more, but for players who pick a character at random and do not want to read about the abilities, the tutorial provides a basic overview of the use of each ability.

5.7 Audio System

While Unity makes it easy to show different viewpoints on the computer display and VR headset, it does not provide a way to create separate audio outputs. SteamVR has an option to mirror audio to a second device, but that was insufficient for the needs of this project. Instead, an entirely separate audio channel was needed to play sounds relevant to the controller players through speakers, while the VR player would hear sounds relevant to them through headphones.

5.7.1 Initial Implementation

To solve this, it was determined that a third-party audio library would be needed, since Unity's audio system would not provide the needed functionality. However, the built-in audio system does have the advantage of handling 3D spatialization and attenuation, features which are very important for VR immersion. Therefore, it was determined that the best solution would be to use the built-in audio system for the VR player and another audio library to play sounds for the controller players. This setup is shown in Figure 5.7.1a.

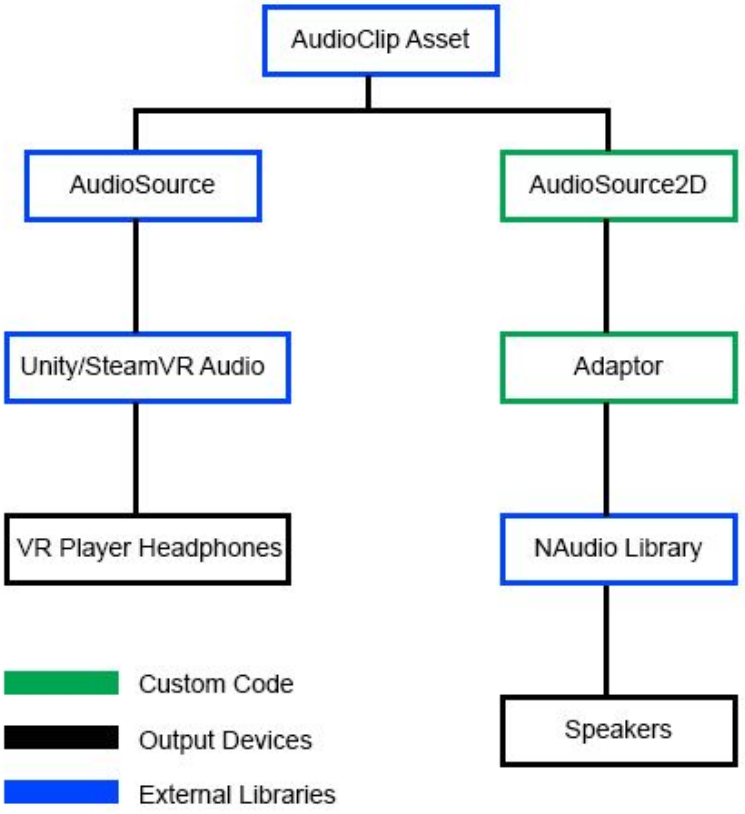


Figure 5.7.1a: Complete Multi-Output Audio System

Since the controller players use a shared third-person camera, spatialized audio was unnecessary in this system. For the external library, NAudio was chosen, as it allows the programmer to list available audio devices and select between them. These devices are presented to the controller players as a dropdown the first time they launch the game, after which their selection is saved. The major challenge in using this system was providing it with sound data in a format it could play. Since Unity stores audio files in a proprietary format, NAudio could not play these directly. One solution would be to distribute the sound files with the game in a standard format, but that would lead to unnecessary duplication of files played on both systems. Instead, the team created an adaptor that could read Unity's audio assets and convert them to the format expected by NAudio.

5.7.2 Refinements

After initial creation, there were several improvements made to this system. Firstly, while the initial version only offered an interface to play a sound given one of Unity's audio assets, a better interface was developed to mimic the one used by Unity. This interface involves

AudioSource2D components which automatically play a sound on the 2D audio system, and stop playing it when the level changes or the object is destroyed. A further refinement added attenuation based on distance from an AudioListener2D component. Later in development, it was found that rapidly playing many sounds using this system led to significant performance loss. The cause of this was the adaptor, which must copy the sound data from the Unity asset in order to use it in another thread. While there is no way to avoid this copying, the performance loss was improved by caching the copied data in a dictionary keyed by the Unity sound asset, to be reused every time that sound is played.

6. Audio

This section describes the procurement of all sounds necessary for the game. Section 6.1 provides an overview of the chapter, including overall design considerations and required sounds. Sections 6.2 and 6.3 describe the aesthetic and inspirations, respectively. Section 6.4 lists all tools used in audio production. All sounds created for the game fit into three categories: sound effects (Section 6.5), music (Section 6.6), and voiceovers (Section 6.7).

6.1 Overview

The job of the audio designer is to create or obtain any sounds that will help enhance the game currently being developed. In this game, sounds are used primarily for immersion and feedback. Looking closely at the artistic and technical sides of the game helped to decide on what sound would work best for each object, character, or environment, and changes were made based on player feedback or the needs of the game team.

Two audio pillars were used when constructing the sound library for this game: aesthetic and features and systems. Audio aesthetic simplifies the theme and features of the game to a handful of descriptors, allowing for more focus for creating thematically similar sounds. Audio features and systems consist of the tools and workflows used to craft the sounds and any third party software used during development.

The team required sound effects for the many character actions of the controller players like punching, shooting and jumping as well as any character interactions between the enemy and controller characters and between the VR player and the environment. Original music was also needed to craft a unique atmosphere for the game's environment as well as to take advantage of the asymmetrical audio system.

6.2 Audio Aesthetic

The basic description of *Babies & Basilisks* is "an asymmetric virtual reality party game for the HTC Vive that requires a VR player to build a maze and place traps and enemies that the controller players must then navigate and attempt to survive." From this description, several features can be extracted. The use of "party" indicates co-op gameplay which requires local functionality and mixing refinement for multiple sounds playing at once. The aesthetic of the game conveyed through words such as "babies," "basilisks," "maze," and "traps," is fantasy mixed with reality, with fantastical objects and scenarios being placed within a real-life setting. Sound design was constructed around this aesthetic, and by layering sounds together, everyday objects had a more fantasized tone. The game mechanics for the controller players are geared toward action-based gameplay, meaning sounds for character interactions, enemies, weapons,

and abilities are vital. Due to the asymmetric nature of the game, with separate mechanics for the VR and controller players, different sound environments had to be created for these two different experiences.

6.3 Audio Inspiration

6.3.1 Mario Party

Figure 6.3.1a: In-game graphics for Super Mario Party (2018) [21]

Mario Party [21], in-game graphics shown in Figure 6.3.1a, is local multiplayer party game featuring the main Nintendo mascot Mario. *Mario Party* takes the form of a traditional board game with players taking turns to roll dice to move across a number of spaces on the board corresponding to the number that they roll. The *Mario Party* series in conjunction with its bright and colorful visuals has a fun and excited tone to its sound effects. Simple actions like passing a ball as shown in Figure 6.3.1a would sound like a small whiff of air in real life, but in *Mario Party*, the sound effect is exaggerated and sounds like something that would be heard in a cartoon when a character slips on a banana. This sound design puts the player into a happy mindset and immerses them into the party theme of *Mario Party*. The team emulated this sound design style for the sound effects of *Babies & Basilisks* by making the audio cartoon-like and fun to fit the bright and colorful aesthetic of the environment and characters.

6.3.2 The Elder Scrolls V: Skyrim

Figure 6.3.2a: In-game graphics for Orcish dungeon in Skyrim (2011) [22]

The Elder Scrolls V: Skyrim [22], shown in Figure 6.3.2a, is an action role-playing game where the player spends most of their time traversing through dungeons and completing quests. Most of the music for the dungeon crawling sections of *Skyrim* are melancholy to immerse the player in the situation of traversing a perilous tomb with no way. The team wanted to design the controller players' audio around the dark and gritty backing tracks of *Skyrim* to fit the maze crawling aesthetic of the controller player characters in *Babies & Basilisks.* This choice was a major contrast to the sound design for *Mario Party* as *Babies & Basilisks* while also an enjoyable party game is also a dungeon crawler so using both sides of the spectrum emotional spectrum was necessary for the team's asymmetrical game.

6.4 Audio Tools

As referenced in the overview for the sound effects and music several programs were used to generate the audio library for the game. Most sounds were downloaded from *freesound.org* and edited in *Audacity* or *Adobe Audition*. Voice-lines were recorded and edited in *Adobe Audition*. Music was developed in *hookpad* and moved over to *Garageband* to attach digital instruments to midi files for higher-quality sound.

6.5 Sound Effects

The sounds are broken up into five categories: controller-player ability sounds, block connecting sounds, damage sounds, death sounds, and UI sounds. Most of the sounds in the game were initially sourced from *freesound.org,* a website where anyone can download free sound assets for any purpose. However, the voice lines required original recordings. The process of making sounds consisted of downloading the original files from freesound, editing them inside a DAW (Digital Audio Workstation), and layering and applying effects to them to make a unique sound and for mixing and mastering purposes. Some effects that were applied to these sounds were the multiband compressor, Fast Fourier Transform (FFT) filter, and pitch shift. The multiband compressor, as shown in Figure 6.5a, was used to separate tracks into different frequency ranges and compress them independently to eliminate any abnormalities like random booms or pops.

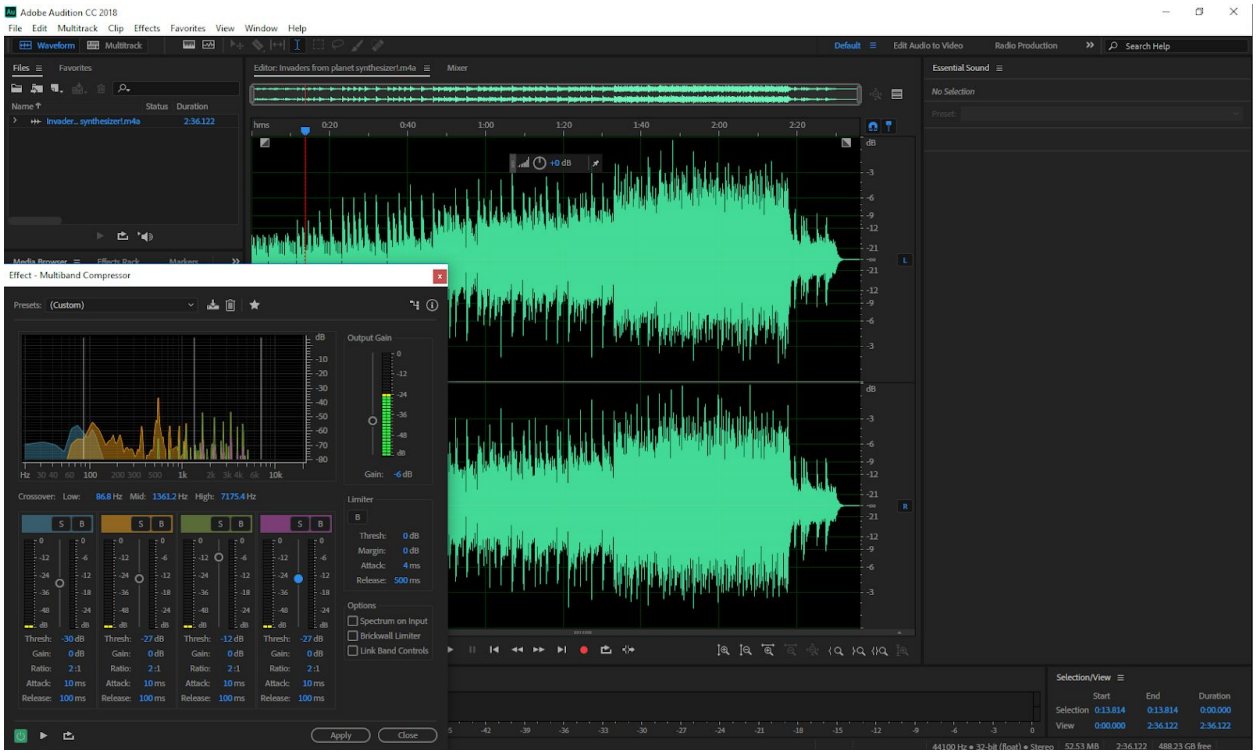


Figure 6.5a: Multiband Compressor

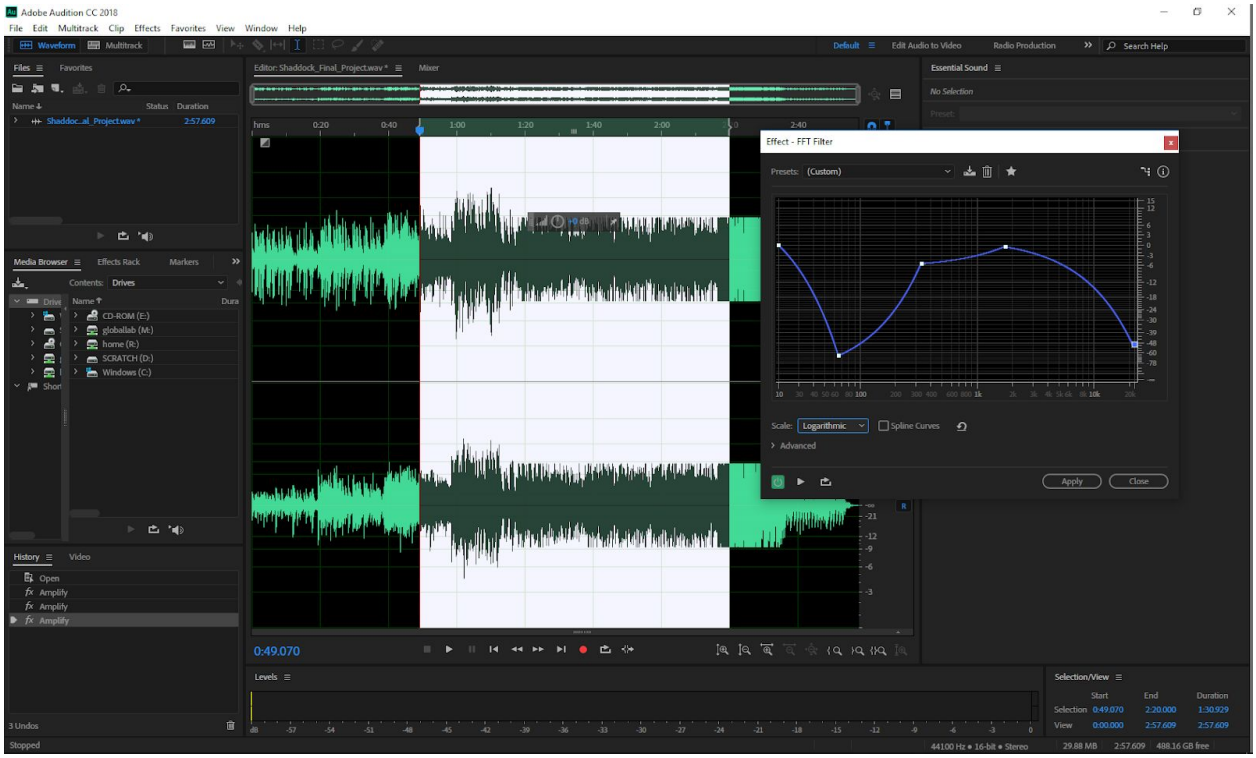


Figure 6.5b: FFT (Fast Fourier Transform) Filter

The FFT filter as shown in Figure 6.5b above was used for more precise equalization of the audio, as well as bringing out a deeper tone in voice lines by lowering the frequency.

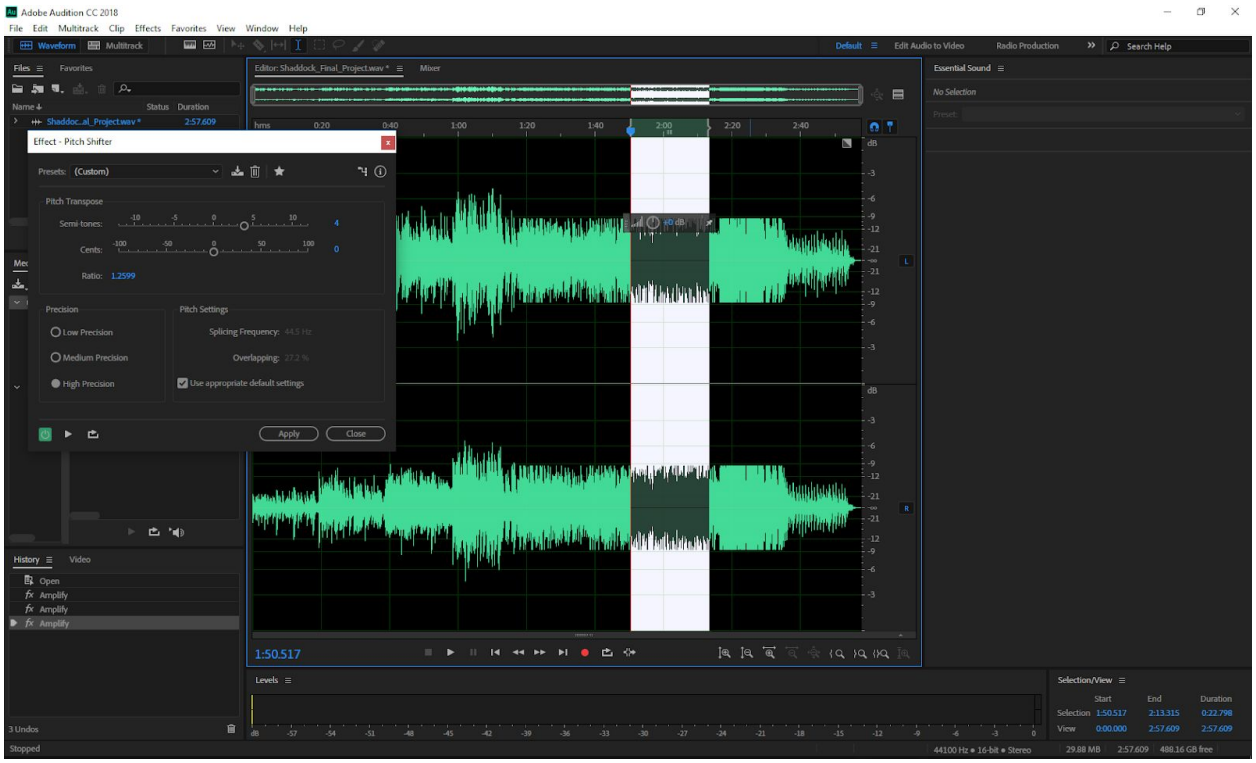


Figure 6.5c: Pitch Shifter

Along with the FFT filter, pitch shifting, as shown in Figure 6.5c, was used to bring out more precise frequencies for higher or lower tones for the voice lines. This effect was right to use since some of the recorded lines might not have been deep enough to fit the aesthetic of the character they were used for.

Once the sounds were created, they were implemented into the game as a sound cue for an Audio Source. The secondary audio system separates the sound channels for both the VR player and the controller player, meaning that what the VR player hears on their headphones will not be heard by the controller player's speakers (this system is described in more detail in the Technical Implementation: Audio System section, page 52). Leveraging this system, two different sound asset libraries were created, allowing for asymmetrical audio. Each sound was auditioned and approved by the team before implementation was finalized.

6.6 Music

The game has an original soundtrack which was developed mainly in hookpad and polished in garageband. The first step was to choose the instruments as shown in Figure 6.6a.

Figure 6.6a: Instruments

Once the instruments were chosen, the individual MIDI tracks were mapped to their instrument, and effects like reverb and panning were applied, as well as adding compression and

amplification parameters to raise and lower the volume at specific parts of the track. An example of these effects being applied in Garageband can be seen in Figure 6.6b.

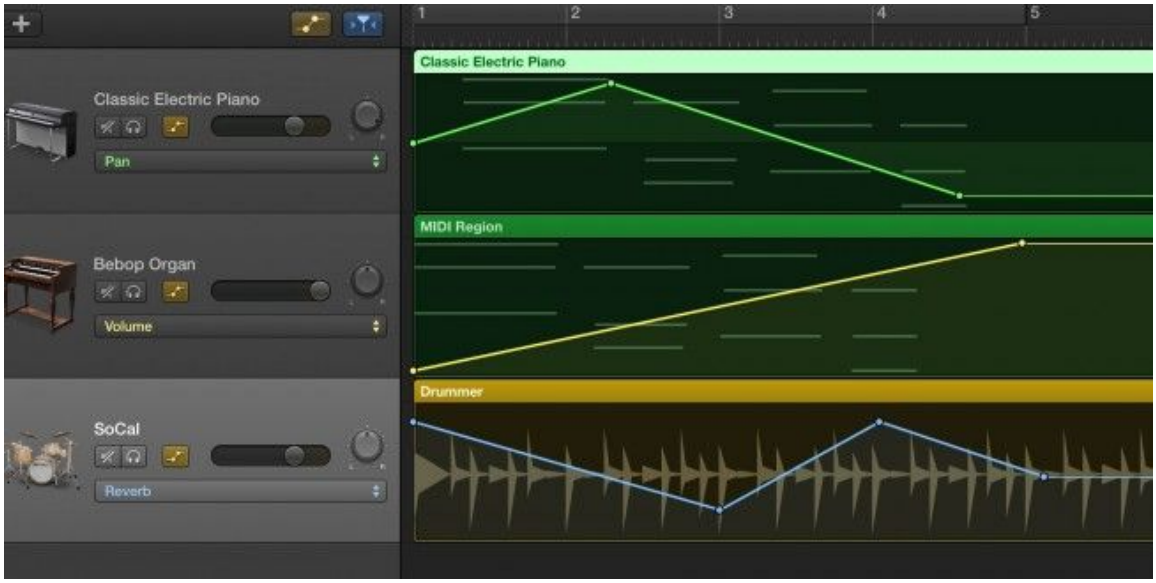


Figure 6.6b: Effects

Two main themes were created to take advantage of the asymmetrical audio system. The controller player's backing track is a dungeon crawling theme which gives off the atmosphere of a dark and melancholy experience as the players traverse the maze set up by the VR player. This theme was set in a minor key, which is a signature mostly used for sad tones as seen in Figure 6.6c.

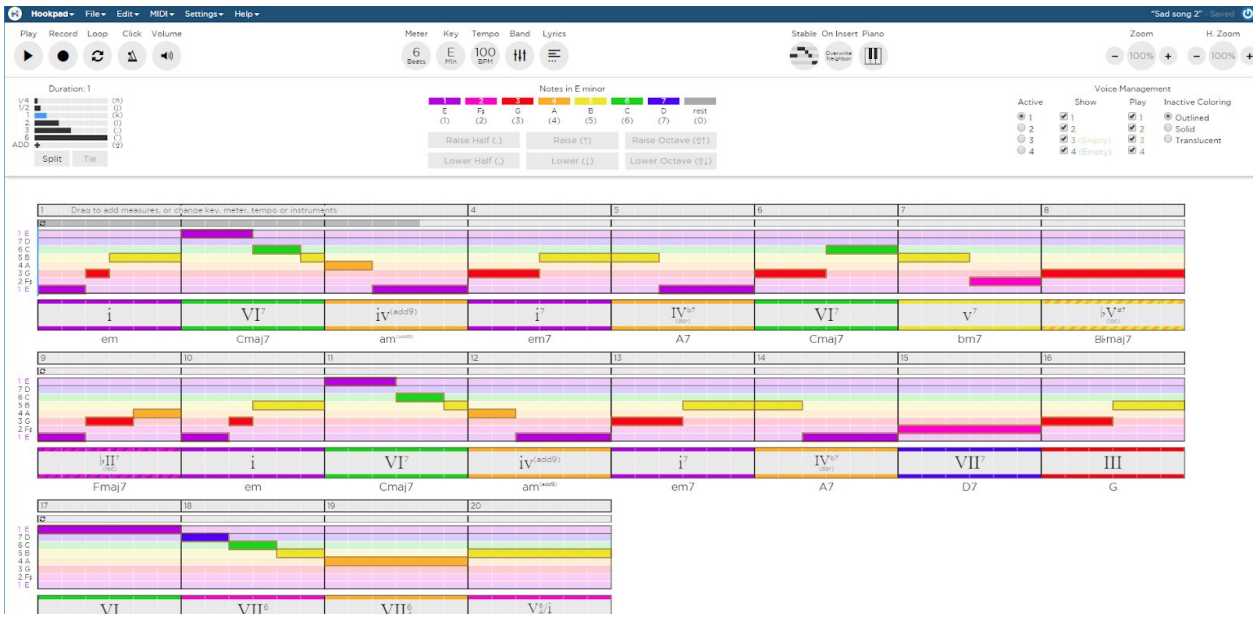


Figure 6.6c: Controller Player Theme

The VR player's backing track was more upbeat and fun to reflect the feeling of playing with toys in a child's room. The warm feeling of the theme was inspired by the Katrina and the Waves' song "Walking on Sunshine" [19]. The chord progression is the same as "Walking on Sunshine" but the melody is wholly original as seen in Figure 6.6d:

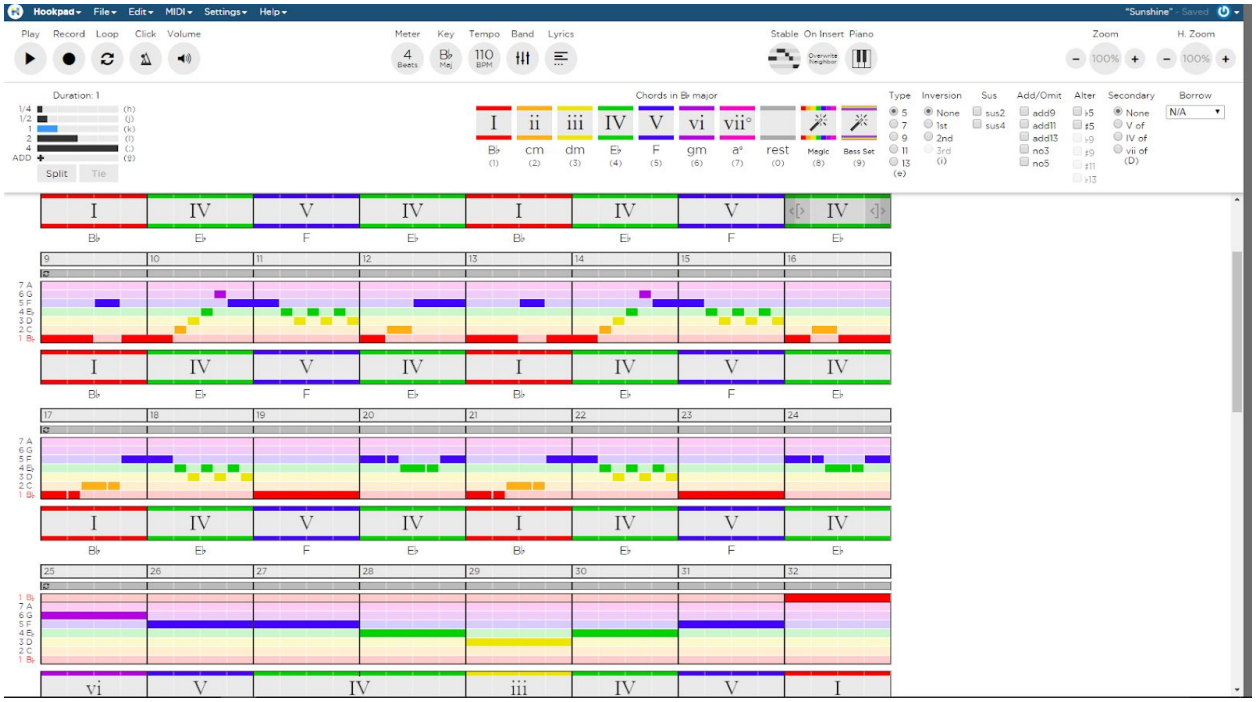


Figure 6.6d: VR Player Theme

The boss theme was made to be more bright and rhythmic so that the players would keep up their high energy while fighting. Synthesizers were the main instrument for the music while bass drums developed a deep sound for the beat.

6.7 Voice Acting

Voice lines were instrumental in adding more originality to the game's characters. The voice acting was performed by Christine Flores, who voiced Tess, and Justin Harris, who voiced Amos and Doctor Bear. Before recording could begin, a dialog script was written and approved by the team. Once the script was approved, the voice actors were scheduled to meet and record their voice lines together. It was easier to record on a tight timeline if the voice actors were together rather than recording separately. Once the voice actors read over the script, the team's audio designer, taking the role of voice-over director, gave direction on how the lines should be read, so the actors knew what emotion should be evoked and how to articulate it. Initially, the voice actor for Doctor Bear was asked to speak with a German accent, which he was unable to

do. In order to work around the strengths and weaknesses of the voice actor, the voice was changed to have an Australian accent, which he could more effectively mimic.

7. Testing

Throughout the development process, the team performed playtesting at several occasions to determine if the overall experience was satisfying, and to find which elements of the game needed the most improvement. Playtesting included the IMGD-sponsored AlphaFest event (section 7.1), as well as several sessions conducted in IMGD design classes (sections 7.2 and 7.3). All playtest sessions concluded with a survey, enabling players to provide anonymous feedback. The game was also demonstrated at WPI's PAX East booth, gathering informal qualitative data (section 7.4).

7.1 AlphaFest

Figure 7.1a: AlphaFest Playtesting

AlphaFest is an event hosted by the WPI IMGD department in which students can show their games-in-progress to other game developers, or anyone who is interested. As seen in Figure 7.1a, AlphaFest provided a great time to playtest our project with new faces. The team participated in the AlphaFest event held on Friday, November 16, 2018. At this time, the game was in an alpha stage, with basic, empty hallways the VR player could place, and preliminary versions of all rooms and enemies working. The enemies and controller players did have models, but only one controller player was rigged and animated. All other objects were placeholders. Since the game was still early in development and had not been tested by anyone outside of the team, Alphafest was a valuable opportunity to get feedback from players who did not know

anything about the game. The team administered two surveys: one for the VR player and one for the controller players, to help collect feedback after they had played the game. The full surveys are located in Appendices D and E. Testers who played as both the VR player and a controller player filled out both surveys.

AlphaFest testing took place in the Foisie 221 Global Lab room, in which the team had previously developed the game. This meant that the team was able to make sure the game would work as expected on the machines prior to the event. However, while this location provided a large screen for the controller players to look at and plenty of room for the VR player to maneuver, it was across the floor from the other AlphaFest games and away from foot traffic, reducing the number of potential testers. To combat this, the team created posters advertising the game and where to find it, and placed them around the floor when AlphaFest began. When this proved ineffective, members of the team went to the main gathering of AlphaFest to find and bring interested parties to the game. In total, the team received 12 survey responses for the controller players and 6 for the VR player.

7.1.1 Questions and Expectations

The surveys were designed to gather feedback on specific parts of the game, in order to provide insight into whether these elements needed to be changed. The controller player survey was ten questions, with nine Likert scale answers requiring a rating of 1-4 and one optional short answer. The VR player survey was also ten questions, with three optional short answer questions and seven Likert scales. For the VR player, the primary goal of the survey was to determine if the controls for building the maze and controlling the boss were intuitive, and if there was enough time to build the maze with the controller players actively moving through it. The main goals for the controller player survey were to see if the characters were simple to control, gauge how difficult the enemy encounters were, and gather feedback on the current camera implementation.

It was expected that the VR player would not have enough time to build the maze, as the team already planned to give the VR player a head start. This had not been implemented for AlphaFest, however, in order to determine if it was truly necessary. Given the team members' own experiences with the controls, it was expected that players would find the controls to be intuitive overall, but they might have difficulty figuring out how to turn the dial to change the room type. For the controller players, the team expected that they would not find the maze difficult to navigate, partially because of the expectation that the VR player would not have enough time to build the maze before the controller players could explore it. It was anticipated that the controller players would find the combat fairly simple, as the enemy behaviors were not polished and the health amounts of the characters and enemies were not fully balanced; however, it was unclear if the players would feel they had enough space to fight in. Finally, the team

expected that some players may take issue with the frequent movement of the camera, but that overall they would find it acceptable.

7.1.2 VR Response Results

The VR player surveys, although few, provided a clear idea of which controls were and were not working for the game in its current implementation.

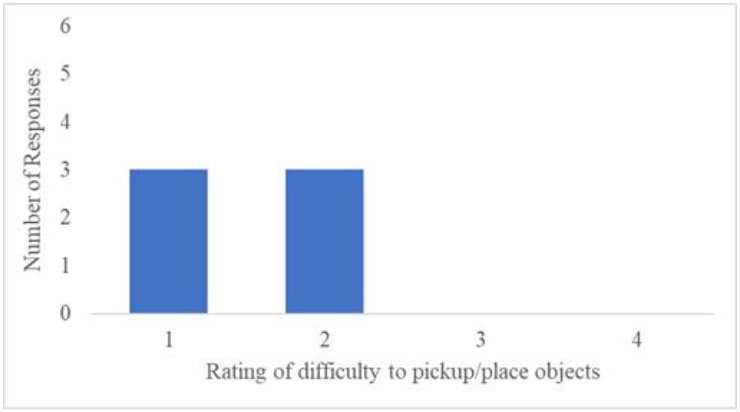


Figure 7.1.2a: How intuitive was it to pick up and place objects, from 1=Easy to 4=Difficult?

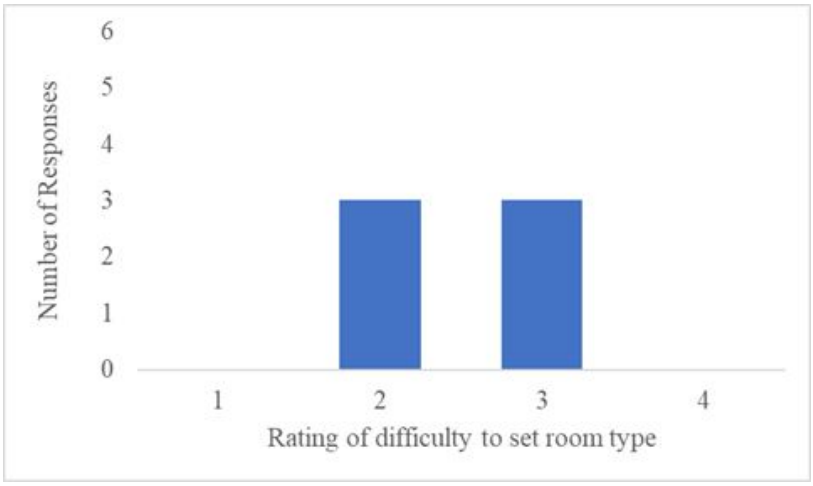


Figure 7.1.2b: How Intuitive was it to set the room type, from 1=Easy to 4=Difficult?

As seen in Figure 7.1.2a and 7.1.2b, most players found it easy to move and place objects in general, but it was more difficult for them to set the room type. Although Figure 7.1.2b does not indicate that any VR players had significant difficulty setting the room type, during AlphaFest most people had difficulty understanding that the room types could be set and how to turn the dial, even after having it explained. Ideally, these two figures would have results of mostly ones (easy), with a few twos. From this feedback, it was clear that a tutorial at the beginning was necessary to teach players how to interact with the game.

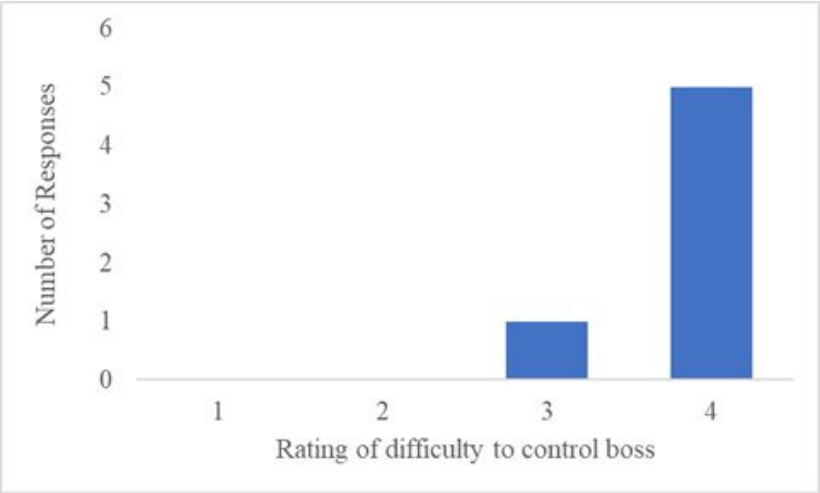


Figure 7.1.2c: How intuitive was it to control the boss, from 1=Easy to 4=Difficult?

While VR players did not think the game's maze-building controls were very difficult, the boss controls were exceedingly difficult for them. Figure 7.1.2c shows how players rated the ease of control for the boss, with four being difficult to control and one being easy to control. Every player found the boss difficult to control, with only one person not rating it at a four. Part of this issue could have arisen from the boss having to rotate before moving, which made it slow and clunky, as well as the lack of visual indicators on the buttons showing what the attacks were. Further, the controller players were quick and could easily stay out of range of attacks, and characters with ranged attacks were almost impossible for the boss to attack. Potential solutions to this included changing the existing UI to have these icons and designing a boss that would not have to rotate before moving. Ultimately, a completely different version of the boss fight was developed, as described in the Design chapter in section 3.5.2. Related to the difficulty of the boss, the team observed during AlphaFest that many VR players were unaware of when the boss fight started, and some only reached the boss controls as the boss was being destroyed by the controller players. The final design of the boss fight resolved this issue by putting the player directly into the location of the boss, leaving no ambiguity that the fight had begun.

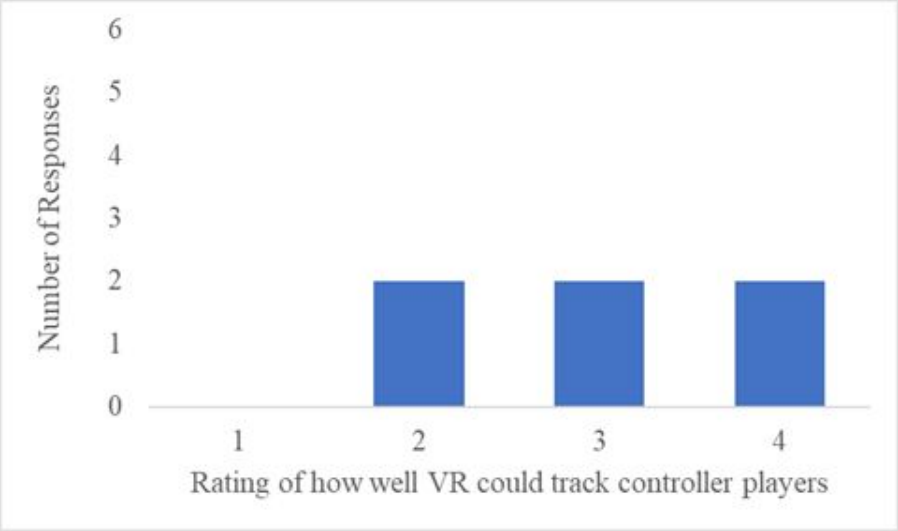


Figure 7.1.2d: How well could you keep track of the controller players, from 1=Could not find them to 4=Could always find them?

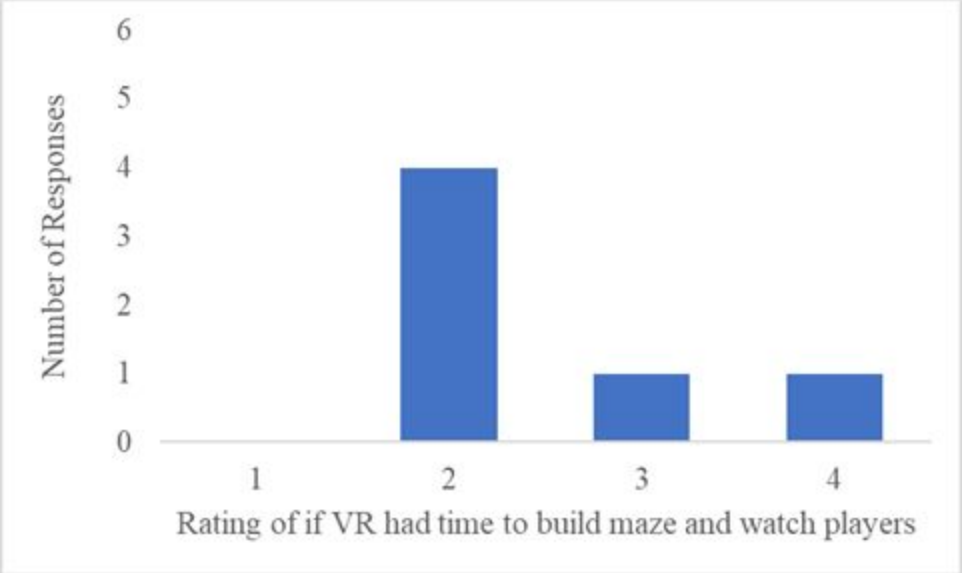


Figure 7.1.2e: Did you have enough time to both build the maze and monitor the players, from 1=No time to 4=More than enough time?

Figures 7.1.2d and 7.1.2e show results for how well the VR player could keep track of controller players in the maze and if they felt they had enough time to do so, respectively. The results of Figure 7.1.2d varied, but no players reported they could not find the controller players when they wanted to. This was perhaps due to a lack of incentive for the VR player to watch the controller players, indicating that the game should include more ways for the VR player to directly interact with the controller players. This was later improved by adding traps that the VR player can control when the players move through certain portions of the maze. Figure 7.1.2e

shows, as expected, that most players felt there was not enough time to watch the players. Ideally, this question would have received mostly threes and fours, indicating that the player was not only able to create the maze but also observe the players. This lack of time likely contributed to the varied responses in Figure 7.1.2d, as players could have been more focused on keeping the maze growing rather than watching the controller players. As anticipated, the VR player did need more time to begin building the maze before the controller players reached it. One change to give the player more time to build the maze was the addition of challenge hallways with a fixed-rate moving camera, so the controller players would not be able to rush through the hall before the VR player had the next room ready.

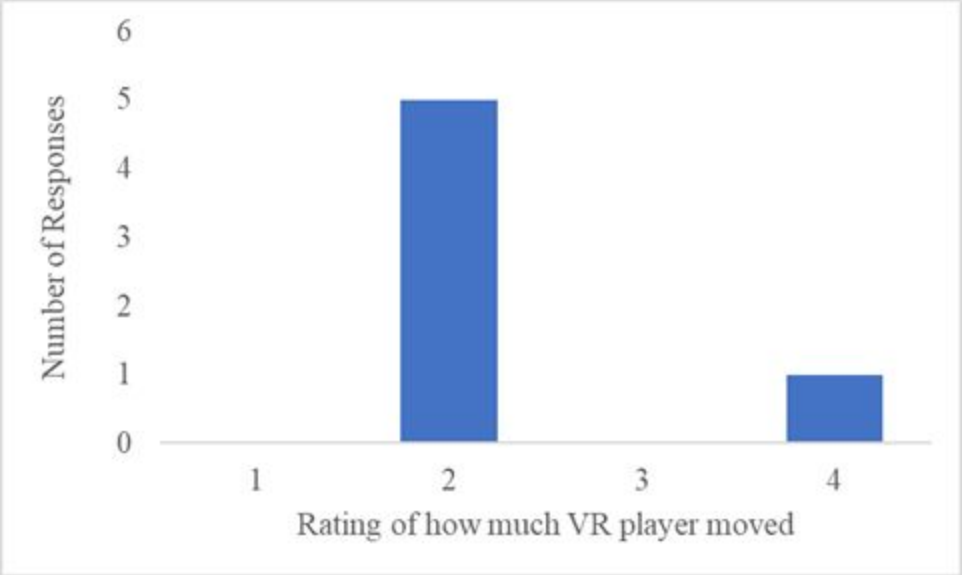


Figure 7.1.2f: How much did you have to move while playing, from 1=Stationary to 4=Mobile?

Figure 7.1.2f shows the responses to how often the players have to move, with one being stationary and four being constantly mobile. Most people rated their motion at a 2, and some specified in the general comments that they did not have to make use of the teleporting mechanism. While this result was ideal, it was likely influenced by the large space afforded by the demonstration area. Regardless, this is the answer the team was hoping to receive, indicating that the spacing of elements for the VR player was adequate for users to interact well with.

The VR player survey included three short-response questions, asking about their most frequent activities (such as watching VR players or building the maze), what goal they had when building the maze (such as making a fun maze or trying to destroy the controller players), and general comments (discussed in the Shared Questions Responses section). These questions were optional, but all of the VR players provided responses to the two VR-specific questions.

For the question of the VR player's primary activity, all players answered with one of the same three options: building the maze, placing enemies, or both. The results are summarized in Figure 7.1.2g below.

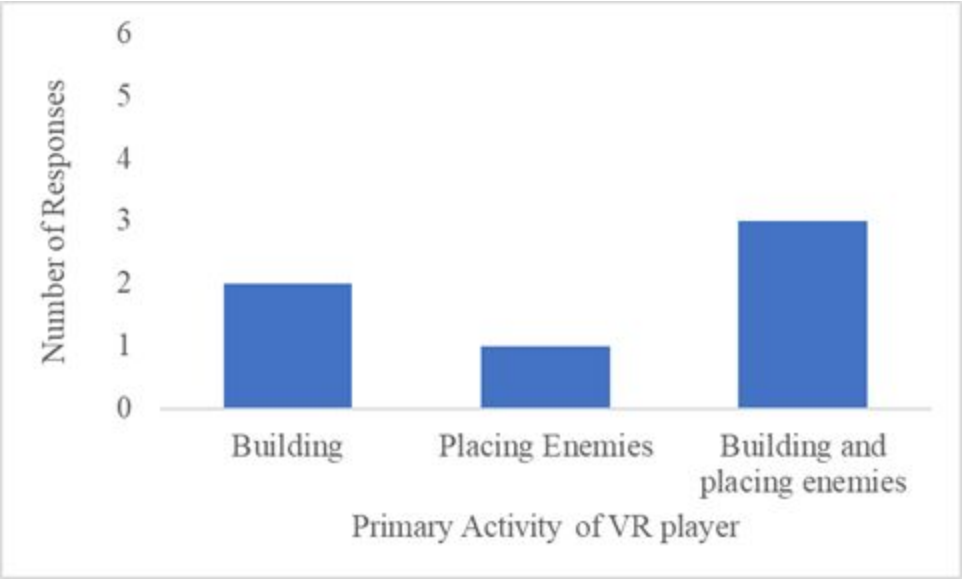


Figure 7.1.2g: What were the VR players' primary activities?

The responses indicated in Figure 7.1.2g were similar to what the group expected. While half of respondents were both building and placing enemies, some concentrated mainly on building, while one was mainly focused on placing enemies. Combined with the feedback about VR players not having enough time to build the maze, it was not surprising that no other types of activities, such as watching the controller players, appeared in the feedback. While this feedback reflected what the team wanted VR players to be doing, it was also desired that the VR players should interact with the controller players more. Because of this, it was determined that there should be more ways for the player to interact with the controller players, such as manually operated traps.

The question of the VR player's goal, and if they felt there was enough of a goal provided from the game, obtained responses too varied to represent in a graph. Players generally indicated that they were confused by the lack of goal, and one commented that it was unclear "because there are so many things happening" for the VR player to keep track of. Some said their goal was to "be the game master and set up the dungeon" or "make fun for players", and others only wanted to make the maze overly challenging. From this feedback, the team found that while players indicated confusion, most of them did mention the intended goals of presenting challenges and creating a dungeon. However, since there was blatant confusion, the team decided that it would need to make the goal much clearer, utilizing the idea of a tutorial decided upon earlier to explain what the VR player should be doing.

7.1.3 Controller Response Results

For the controller players, the responses received by the surveys gave the team a clear understanding about what was and was not working well in this portion of the game.

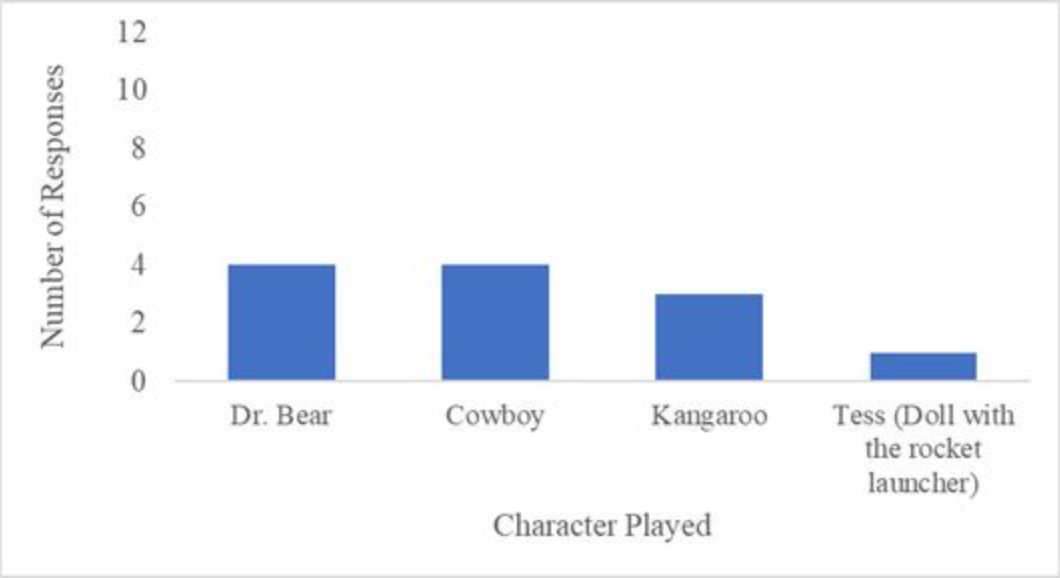


Figure 7.1.3a: Which character did you play as?

Figure 7.1.3a shows which characters players picked. Most players played multiple times, and were instructed to answer based on the character they enjoyed most of the ones they played as. Overall, most players seemed to choose Amos the Cowboy (simply referred to as “Cowboy” at the time of AlphaFest) or Dr. Bear. Ideally, there would be an even spread of responses here, and the team determined that providing information in the character selection menu about how many characters there were to choose from, as well as what each character did, would help avoid players choosing whichever character “looked coolest” to them, or simply picking the earliest character in the list. This data was also used to gain more nuanced feedback in the questions asking about character controls, discussed next.

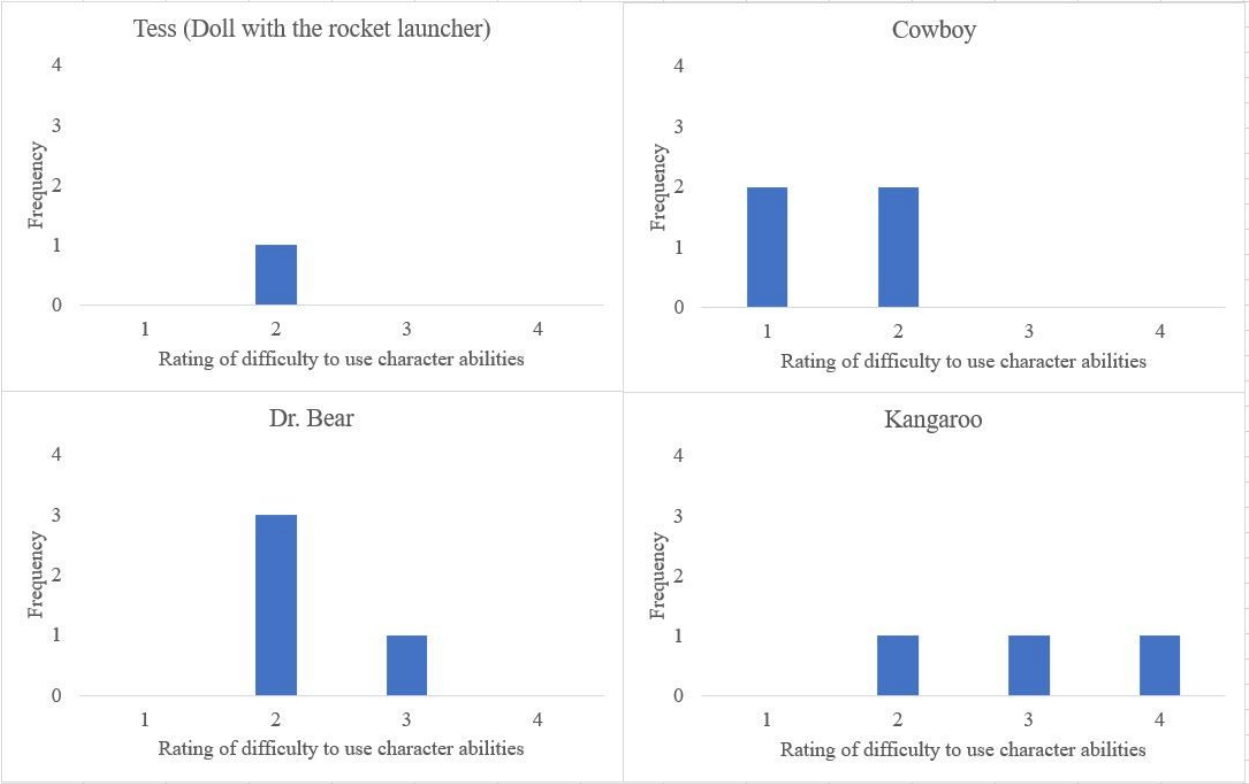


Figure 7.1.3b: How difficult was it to use abilities effectively for the character you played as, with 1=Easy to use and 4=Difficult to use?

Figure 7.1.3b rates the difficulty of using characters' abilities. Lower ratings indicate the abilities were easier to use, and higher ratings indicate the abilities were more difficult to use. Overall, most players found it somewhat easy to use the abilities, although several players asked for clarification on what abilities' effects were during AlphaFest. The results also indicate that the Cowboy's abilities were the easiest to use, with the four Cowboy players evenly divided between selecting 1 and 2 for the difficulty, and the only ratings of 1 belonging to the Cowboy. Dr. Bear also appeared to be somewhat easy to use, with three of the four responses rating the difficulty at 2 and only one person rating it at 3. Since only one player reported using Tess, it was difficult to draw conclusions on her abilities without more feedback. The Kangaroo, however, seemed to be the most difficult to use, with one of the Kangaroo players rating the difficulty at 4, which none of the other characters received, and another at 3. The team decided that while the other characters' abilities could remain mostly the same, the Kangaroo would need more work polishing its abilities so they would be easier for players to use and understand. Another conclusion drawn from this question was that animations and particle effects would help greatly in players' understanding of the abilities they were using. Of the characters, only the Cowboy had visible effects for all of his attacks, which could be linked to why he was considered the easiest to use. In order to reduce this difficulty, the team planned to provide brief descriptions of character abilities from the character selection menu.

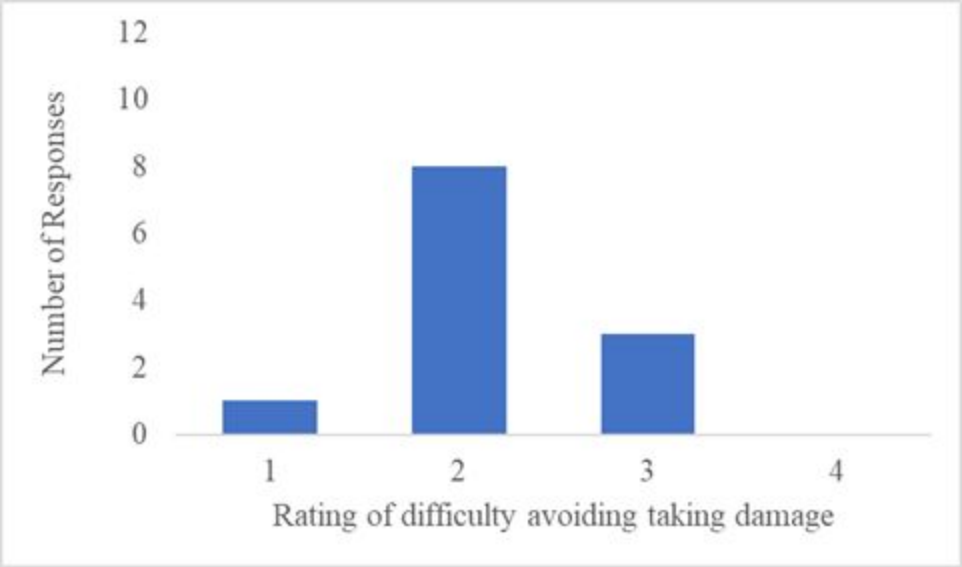


Figure 7.1.3c: How difficult was it to avoid getting damaged, from 1=Easy to 4=Difficult?

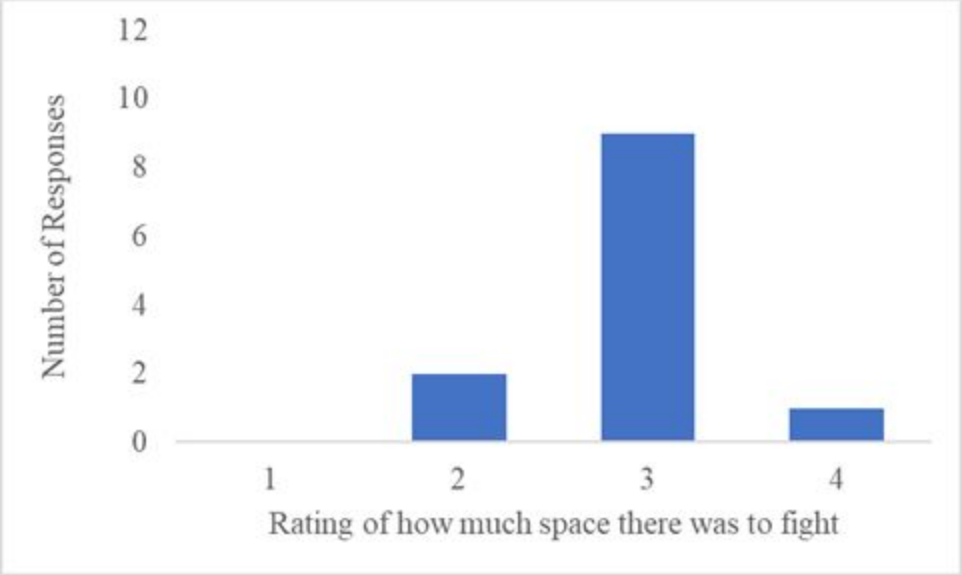


Figure 7.1.3d: Did you have enough space to move around and fight, from 1=Not enough space to 4=Lots of space?

Figures 7.1.3c and 7.1.3d show how well players were able to maneuver in fights. Figure 7.1.3c displays how difficult it was for players to avoid getting damaged by enemies, with 1 being easy to avoid and 4 being difficult to avoid. The ideal response for this question would have been mostly 3s, so players would get hit frequently but be able to dodge attacks some of the time. Instead the responses received mostly 2s. From this, it was decided that the enemy AI should be tweaked to be more challenging, but also that, similar to player abilities, the enemy

animations would be necessary to help the players understand what each enemy could do and where it was attacking. Figure 7.1.3d indicates whether players felt there was enough space to move around and fight. While players were expected to not have enough room, the overwhelming rating was that there was quite a bit of space for them to fight, with most ratings being a 3. This was the ideal answer, so the team decided that the space-to-character scale was adequate and did not need to be changed.

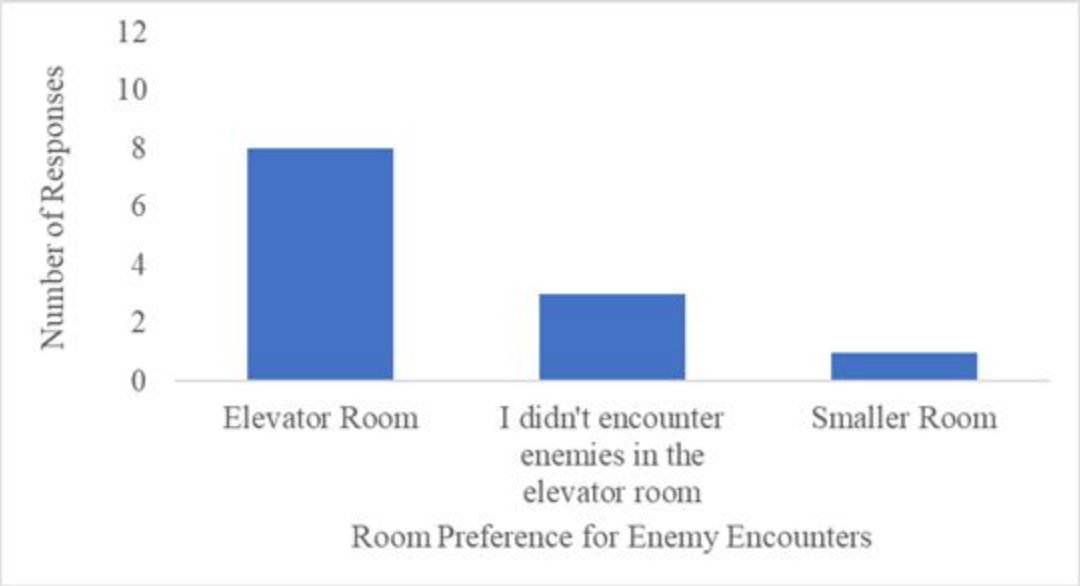


Figure 7.1.3e: Room preference for enemy encounters: elevator rooms or smaller rooms?

The survey also asked players if they preferred their enemy encounters in the larger "elevator" rooms (named such because of the elevator bringing players into the room from a smaller room) or the generic smaller room. The team hoped that most would prefer the elevator rooms, since they were already implemented and provided the players with more space to move, but the question was asked to avoid the team assuming the elevator element was better simply because it required more programming to implement. The players, in general, did prefer the elevator rooms, as seen in Figure 7.1.3e, with three times as many players preferring the elevator rooms to smaller rooms. A few players did not go into an elevator room, although that was due to the VR player not placing any elevator rooms with enemies in them, most likely because the VR controls for room selection were unclear. From these results, the team was able to conclude that the elevator rooms were worth including in the game and added to the enjoyment of the game rather than subtracting from it.

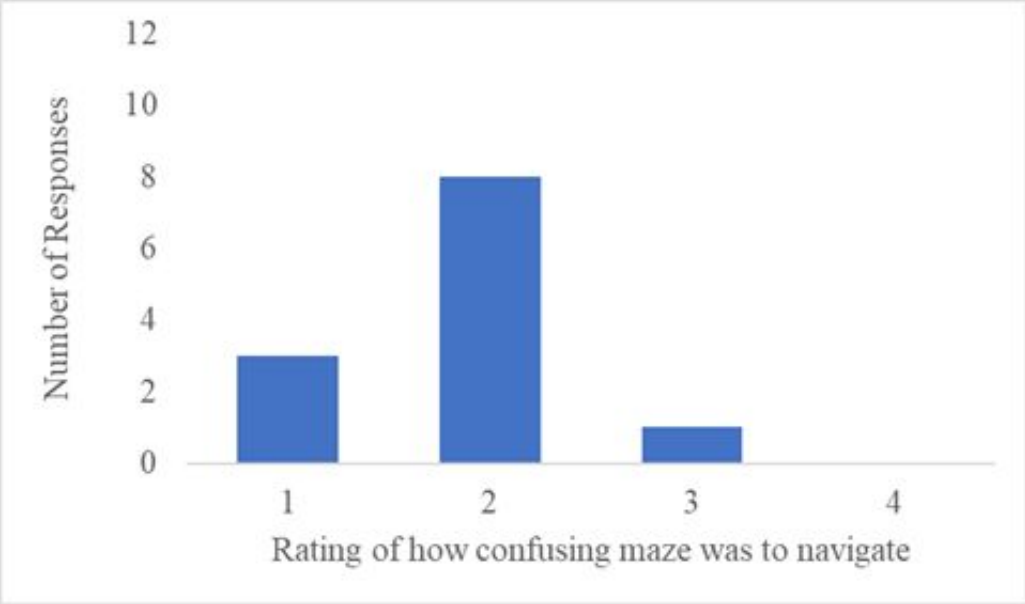


Figure 7.1.3f: How confusing was it to navigate the maze, from 1=Easy to 4=Very confusing?

Figure 7.1.3f shows the responses for how difficult it was to navigate the maze, with one being easy to navigate and four being quite confusing. While the ideal response would have been 3s and 4s, only one player found it that difficult. This was probably due in part to the VR player not having time to begin building the maze before the controller players entered it. Although adding a delay before the players entered would have possibly solved this issue in part, the team decided that the significant challenge should come from completing individual maze sections, rather than a confusing layout. This was accomplished in part by adding hallways with traps and platforming challenges, as opposed to only having connectors without any challenging content. This shift in focus enabled the game to be simpler for the VR player, as they could create a linear, challenging "maze," or have actual differing paths, without feeling pressure to create a complex layout which players may not fully explore.

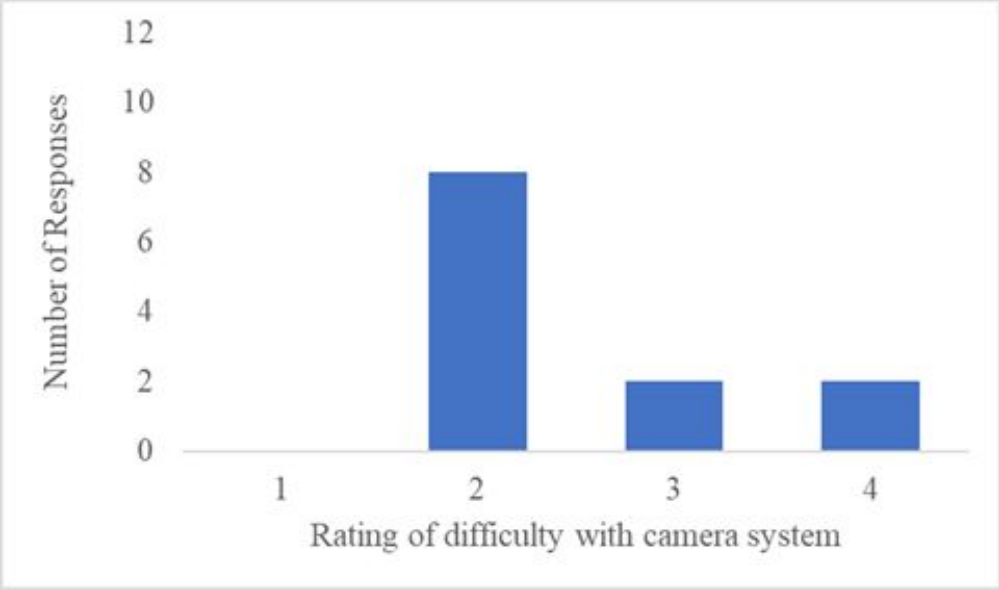


Figure 7.1.3g: Were the cameras effective, from 1=Could not see well to 4=Could see everything?

One major concern was how players would feel about the cameras, which moved and rotated based on which piece of the maze most of the characters were occupying. This could result in some sharp movements when players rounded right-angle corners. During AlphaFest, the team noticed several camera issues, such as hiding players behind multiple walls. Figure 7.1.3g summarizes the results of player opinion on the camera, with a rating of 1 meaning the camera was ineffective, and a rating of 4 meaning they could see everything they wanted to. The results confirmed that significant improvement was needed, with eight of the twelve respondents rating the system at a 2. Because of this, the team reconsidered how the camera could operate. A strictly top-down camera was not desired, as this would provide a far less interesting view. Eventually, it was decided to keep the camera's side-view position and instead change the hallways themselves to be composed of curves, which would allow the camera to rotate more smoothly and avoid harsh rotations from right-angle corners.

The survey also asked for opinions on how well the current sound effects were working for the game. This question was not asked to the VR player since there were not many VR sound effects implemented yet.

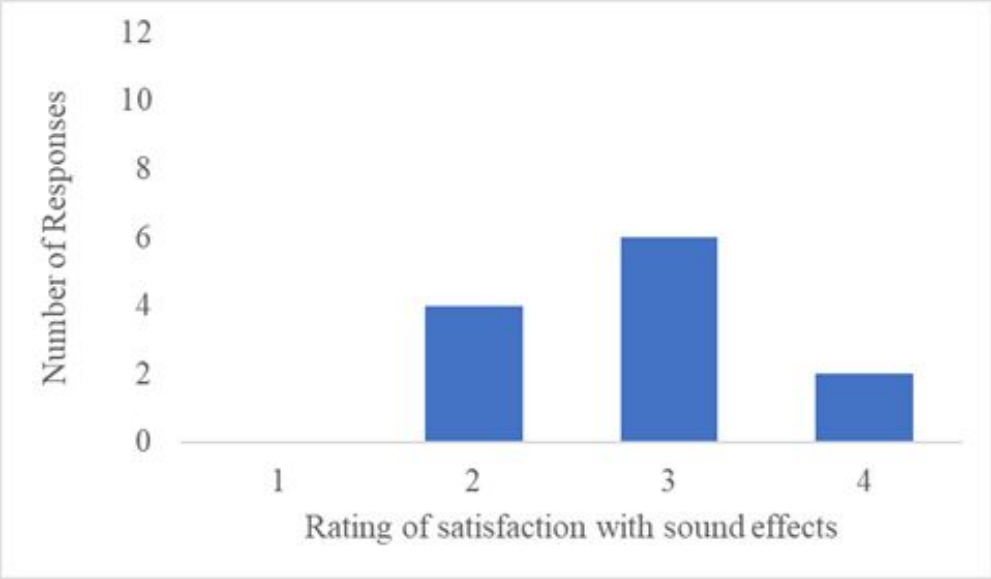


Figure 7.1.3h: Were the sound effects satisfying, from 1=Hate the sounds to 4=Love the sounds?

As indicated by Figure 7.1.3h, the responses were split, with a third of respondents giving a rating of 2, although a majority rated the sounds positively, with a 3 or 4. From this, the team determined that the sound design was effective, but that some sounds would perhaps need more polish.

7.1.4 Shared Question Responses

Both the controller and VR players were asked about the music of the game, and for general comments. While the music question was mandatory, for an accurate rating, the players had to hear the music in order to respond well. However, due to a bug in the audio system code, the output device had to be set with every game, and in some instances this was forgotten. While most players stayed for multiple rounds and were able to play at least one game with sound, two players indicated in their comments that they had not heard any sound. Those two responses were omitted from the results, for a total of sixteen responses for the question.

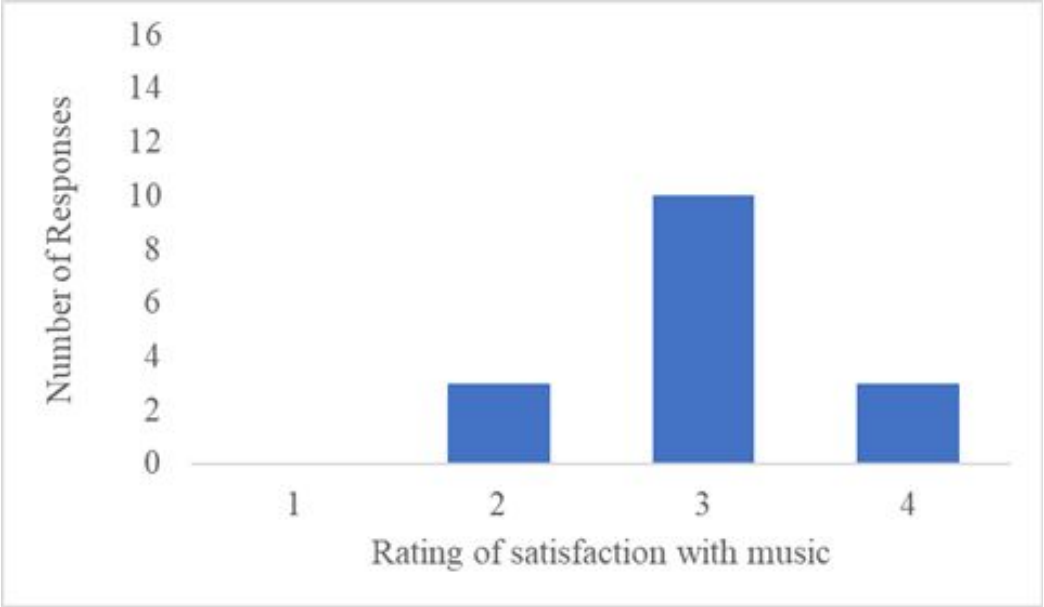


Figure 7.1.4a: Was the music satisfying, from 1=Hate it to 4=Love it?

Figure 7.1.4a shows the results of the responses. Most players found the music acceptable, with three players loving the music and three others feeling dissatisfied with it. Since music is subjective and players have different individual opinions, the team considered the thirteen positive responses indicative that the music was working well with the game.

The short-answer comments for both surveys were optional, and twelve out of the total eighteen responses included comments, with four of the responses from VR players and eight from controller players. These comments generally reinforced conclusions drawn from the rest of the information in the responses. Some comments reiterated conclusions drawn from the multiple-choice questions, such as mentioning jarring camera transitions, a desire for character ability descriptions, and confusion around boss controls. Recurring comments also mentioned vague goals and lack of direction for the VR player, and a need for extra time for the VR player to set up the maze. Despite these issues, comments were positive overall and players seemed to enjoy playing the game, with one player even commenting they would “love to see more of it in the future.”

7.2 IMGD 2900

The next set of playtesting was conducted with the students of Professor Moriarty’s Digital Game Design I class in C-Term. 25-minute long playtesting sessions were held, with four to five students present for each session. These sessions took place on February 20 and 25 , 2019, in the same location as AlphaFest’s playtesting. While intermittent loss of the HTC Vive’s tracking hindered the VR player’s ability to interact with the game, the team was still able to

collect feedback from select portions of the game. Overall, there were 19 controller player responses and 7 VR player responses.

7.2.1 Controller Player Responses

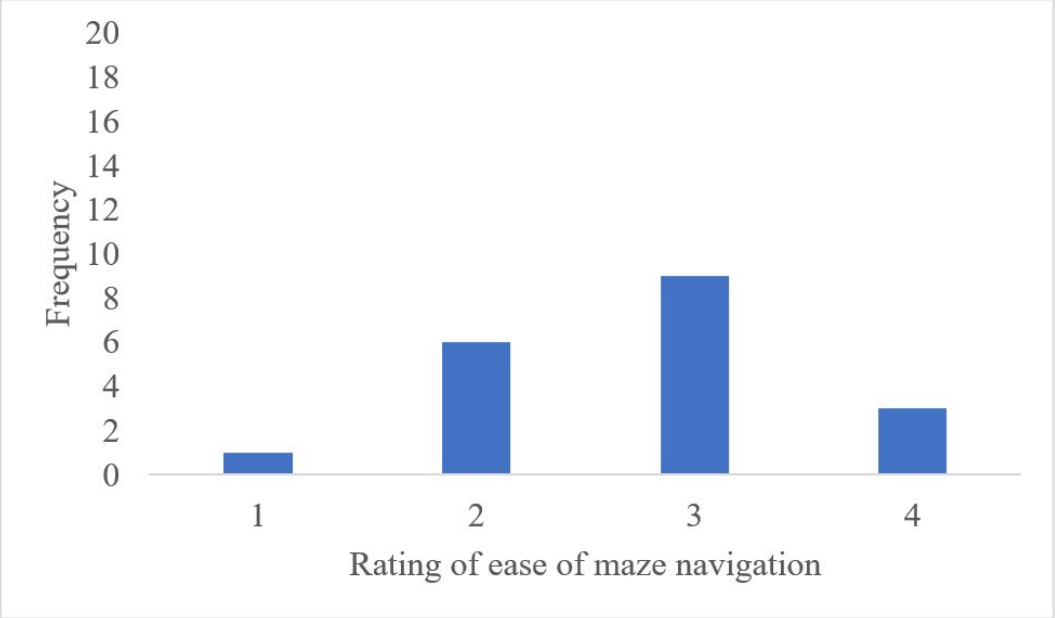


Figure 7.2.1a: How easy it was for players to navigate the maze, with 1 = difficult and 4 = easy.

Figure 7.2.1a shows the difficulty of navigating the maze, with a rating of 1 being difficult and 4 being easy. Most players gave the maze a medium to hard rating. This data suggested that the new platforming hallways added localized challenge without increasing overall difficulty. These values fit with the team's expected values for the new design of the maze, indicating that the platforming hallways effectively provided more challenge for the players.

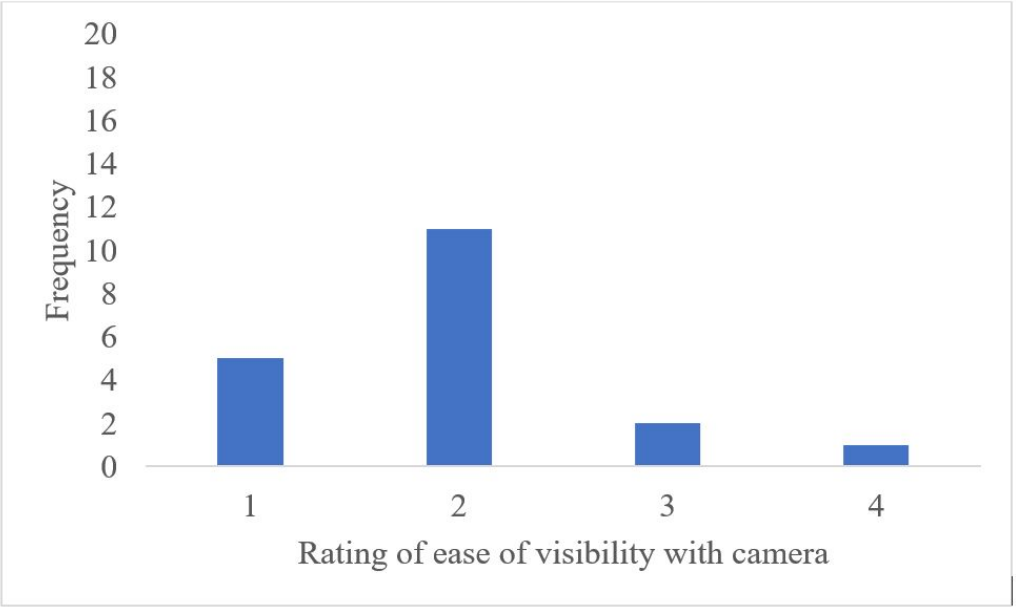


Figure 7.2.1b: How easy it was to see with the camera, with 1 = difficult, 4 = easy

As shown in Figure 7.2.1b, players also rated the effectiveness of the camera, with a rating of 1 being difficult to see and a rating of 4 being easy to see what they wanted. Players still found the camera system difficult, and reported specific problems including the camera following the first person to move into a new room, as well as continuing to track players even after they had fallen or died. These issues were taken into consideration and remedied by the team.

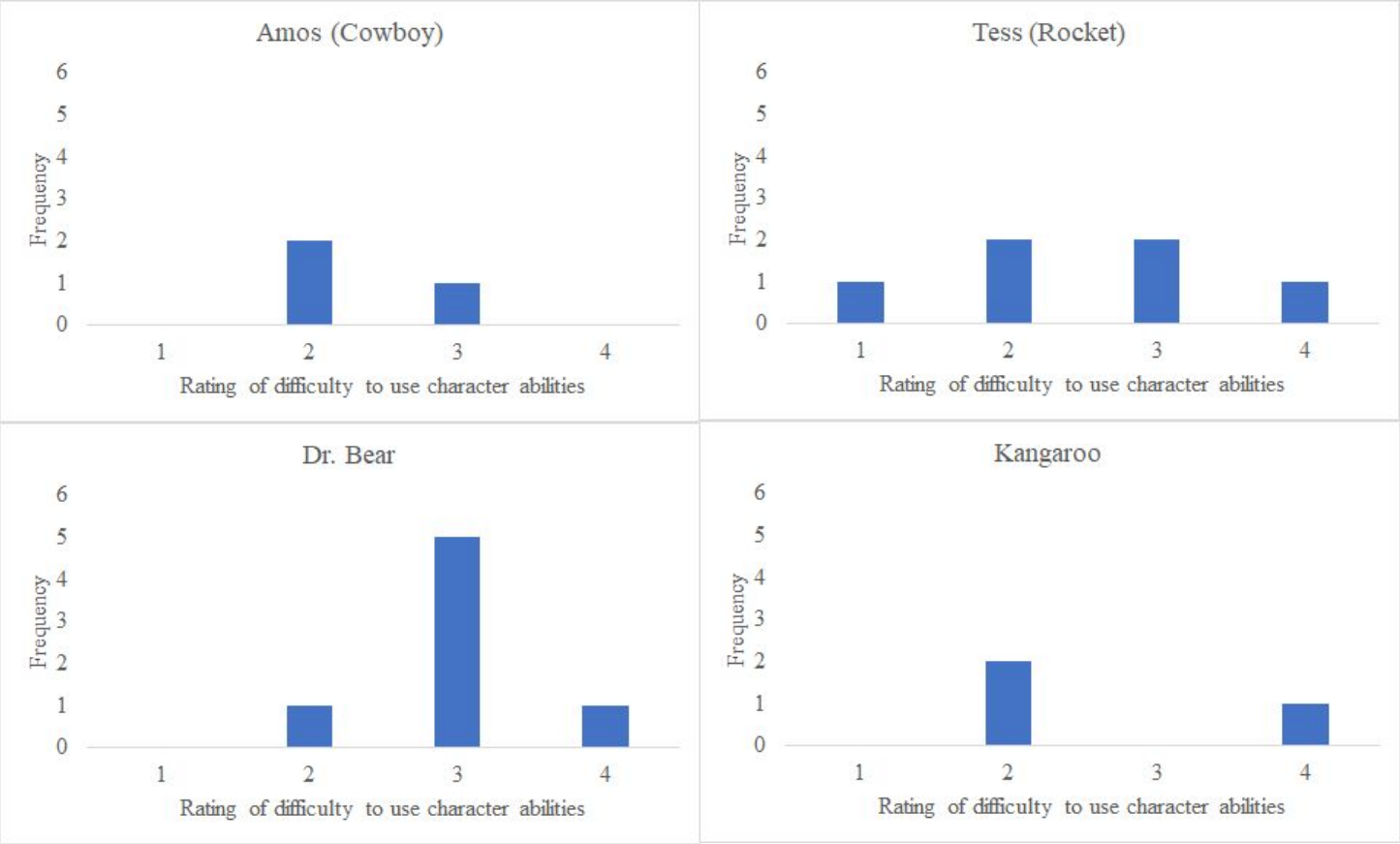


Figure 7.2.1c: How easy it was to use the characters' abilities, 1 = difficult, 4 = easy

This round of playtesting also saw a change in the ease of use for character abilities, as seen in Figure 7.2.1c. Ratings of 1 represented difficulty in controlling the players' abilities, while 4 indicated it was easy. While Dr. Bear's rating stayed largely similar in rating, leaning towards easy, players using Amos seemed to find him harder to use than at AlphaFest. In the general comments, players explained that they felt the aiming for this character seemed slightly off compared to the others. The difficulty of using Tess seemed to be evenly split, while the Kangaroo's ratings showed a small improvement. Overall, difficulty ratings leaned slightly towards difficult.

Answers to open-response questions were consistent with notes gathered from verbal feedback during playtesting. The most common response was that players desired more visual and auditory feedback for damage dealt and taken. A few players commented on the camera system, suggesting a split-screen camera, or that the camera followed the majority of players rather than the first player to enter a new room. This latter recommendation was ultimately implemented into the game.

7.2.2 VR Player Responses

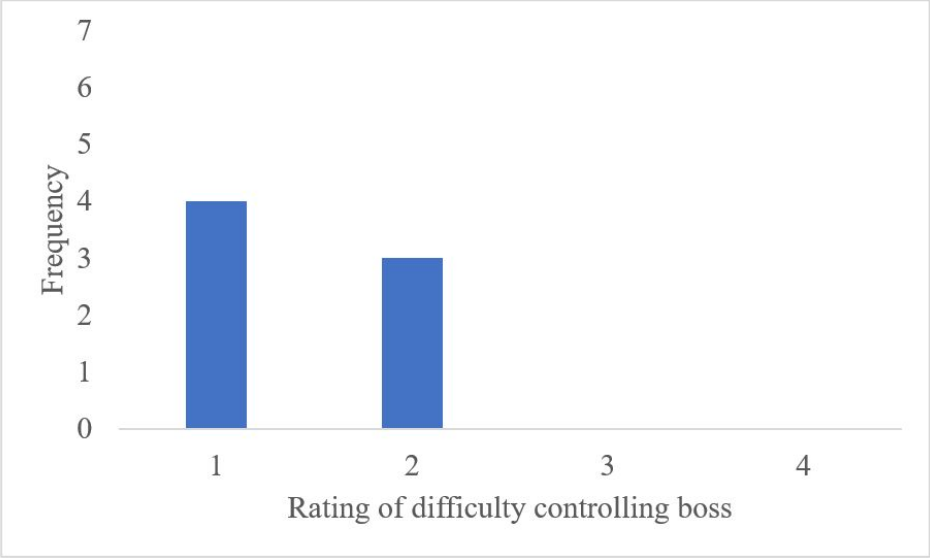


Figure 7.2.2a: How difficult players found controlling the boss, 4 = easy, 1 = difficult

Figure 7.2.2a shows players' ratings of how easy the new boss implementation was to control, with a rating of 1 being difficult and a rating of 4 being easy. At the time, there was no tutorial to teach the controls of the boss to the VR player. While still difficult, results were more evenly split between a 1 and a 2 than AlphaFest's overwhelming consensus of maximum difficulty.

There were very few comments for the open-response questions. One player stated that they were "waiting for levels to appear", indicating that either they did not notice when the next piece of the tutorial had appeared, or that the controller players could not traverse the tutorial maze as quickly as the VR player was building it. Of the players who reached the boss stage (some were unable due to VR tracking losses, requiring a restart of the game), some mentioned the desire for mobility options as the boss. The team decided this feature would be worthwhile to implement into the game, as it would provide more versatility when fighting the controller players.

7.3 IMGD 3900

The final formal playtesting session was conducted with Professor Moriarty's Digital Game Design II class during D-Term. Unlike previous sessions, this was performed in an empty room on a VR-capable laptop. Overall, there were 10 controller player responses and 4 VR player responses.

7.3.1 Controller Player Responses

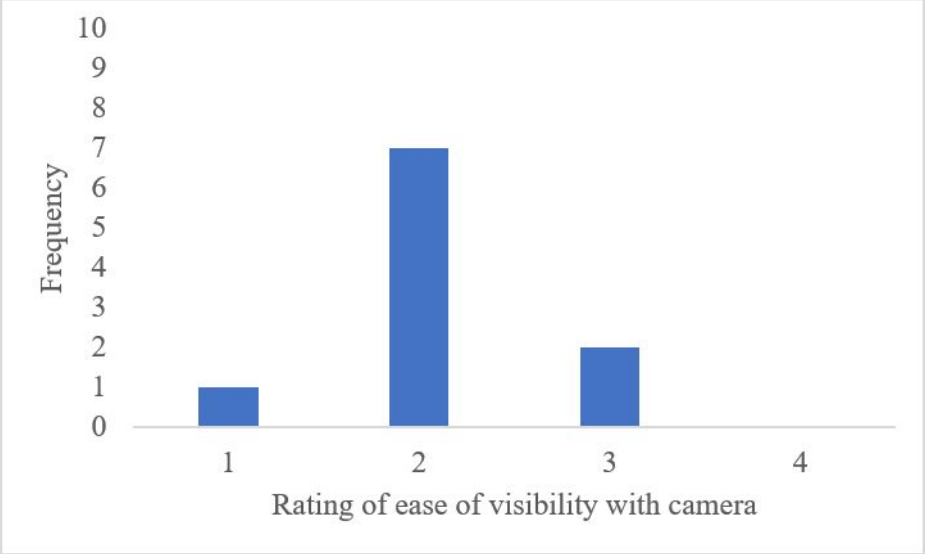


Figure 7.3.1a: How easy it was to see with the camera, with 1 = difficult, 4 = easy

Despite previous improvements, players still found the cameras difficult to use, as seen in Figure 7.3.1a. During playtesting, several bugs were discovered, such as incorrect camera transitions and characters respawning outside the camera's view. These issues could have contributed to the poor ratings of the camera's effectiveness.

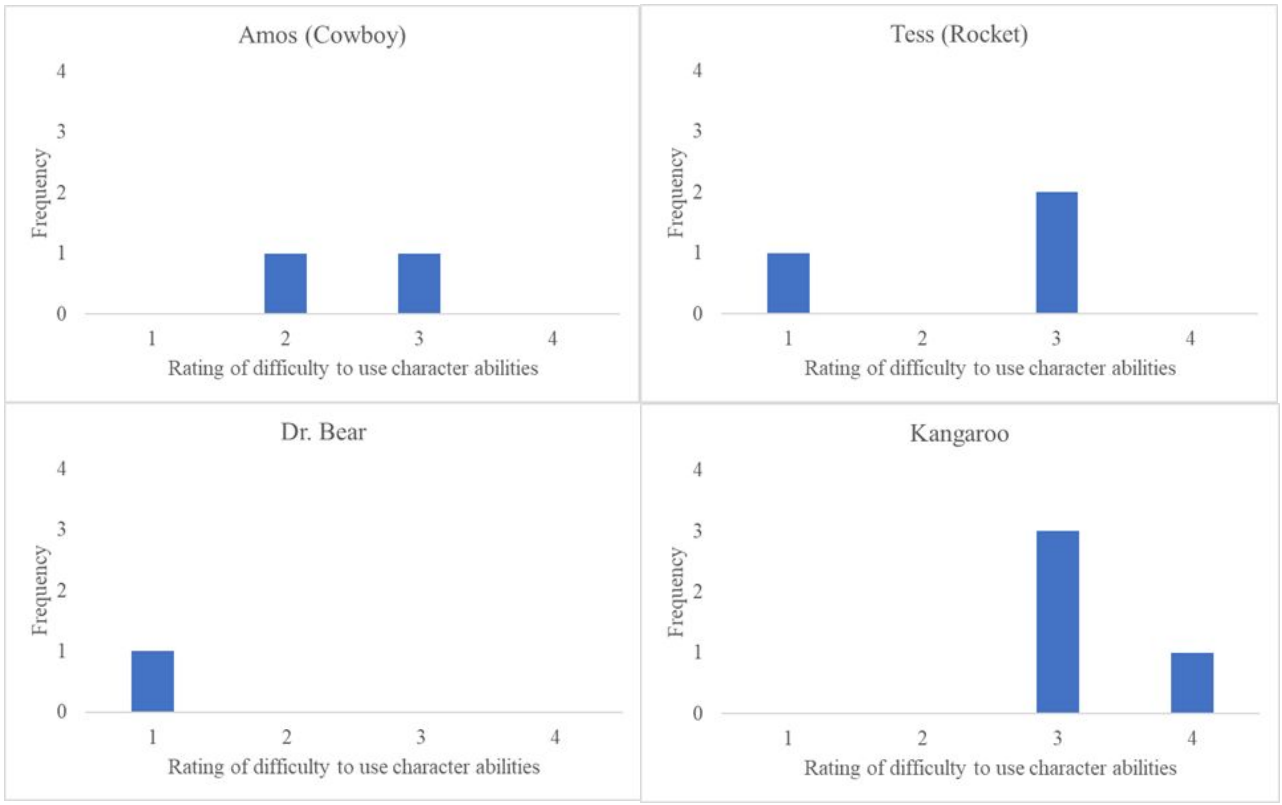


Figure 7.3.1b: How difficult the players found the character abilities, 1 = difficult, 4 = easy

In this playtesting session, players found the controller players generally easier to use, as seen in Figure 7.3.1b. Interestingly, most players reported using the Kangaroo, and that it was easy to use. With few changes having been made to character abilities, this seemingly contradictory result was perhaps due to the audience finding the Kangaroo's fighting style easier to use than previous players had.

In the open response sections, testers commented that the effects of abilities were still unclear, despite the addition of icons to represent the attacks. Players also included positive feedback for the new self-revive mechanic, as well as indicating that waiting for the VR player to build the next section of the maze was rather boring.

7.3.2 VR Player Responses

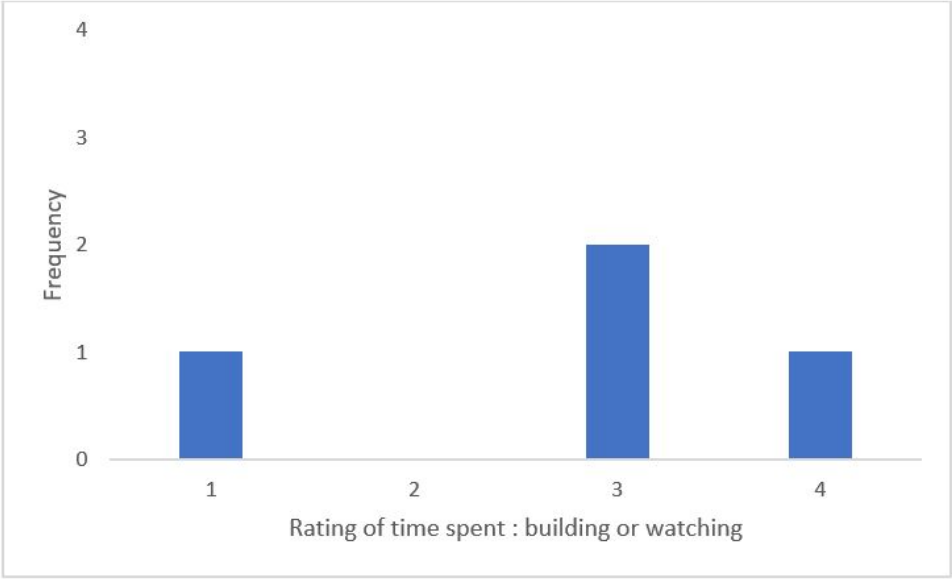


Figure 7.3.2a: Rating of VR player's time spent, 1 = building, 4 = watching controller players

As seen in Figure 7.3.2a, VR players indicated whether they spent more time building the maze or watching the controller players move through it. Since all players used the tutorial, which contained one point at which the VR player was given no more pieces to place until the controller players reached the boss room. This resulted in several long waits while controller players fought through enemy combat rooms to catch up to where the VR player had placed the boss room. Overall, players spent more time watching the players than they did building the maze, exactly the opposite of earlier feedback. This was likely due to enemies now spawning automatically in rooms when players entered, rather than having to be individually placed by the VR player. In a non-tutorial setting, the VR player would have been able to continue building the maze if they desired, suggesting that the automatically spawning enemies were effective in giving the VR player additional time to build the maze ahead of the controller players.

Due to the small number of VR player survey responses, not many open-ended comments were received. Most prominently, players expressed confusion over learning that enemies spawned automatically in rooms, as well as explaining that they were confused as to what their attacks were during the boss fight. A tutorial for the attacks existed at this point, but the on-screen hints were set to follow the position of the controller players, making them unreadable when these players moved quickly. The team remedied this issue by having the instructions track the VR player's head rotation instead.

7.3.3 Critique Comments and Observations

One major issue that arose during this playtest session was the ability to create impossible levels. This occurred when the VR player placed the goal room before the start of a challenge hallway. Since the platforms of challenge hallways collapse as players move through them, it was then impossible for players to travel back and reach the goal room. This problem was fixed by adding a floor to challenge hallways, which only appears after they have been completed. This change allows the controller players to backtrack to any part of the level. Furthermore, it addresses a separate issue where controller players did not have enough room to stand upon completing a hallway.

It was also noticed that the VR player was able to push controller players and enemies around using their head. This issue was caused by a head collider included in the SteamVR interaction system, which was changed to be non-solid to fix the issue. However, it was also commented that something like this could actually improve the game, as it would allow the VR player to interact more with the controller players. While no such system was implemented, discussion of what it might involve can be found in the Future Work section.

Finally, it was occasionally difficult for both controller and VR players to keep track of where the controller player characters were. This was especially problematic in combat rooms, where some enemy characters are visually similar to controller players (such as the Kangaroo character and Wallaby enemy). To resolve this issue, and help the VR player to quickly find the controller players, a semi-transparent arrow was attached to the feet of all controller players. This arrow is always drawn on top of all other geometry, so that it can be seen through walls.

7.4 PAX East WPI Booth

One final opportunity the team had for playtesting was at the PAX East event in Boston as seen in Figure 7.4a. The event lasted for four days (March 28 - 31, 2019), and team members took shifts demonstrating *Babies & Basilisks* at WPI's booth. Due to the fast-paced nature of the event and infeasibility of collecting informed consent forms, no survey was administered, and therefore no quantitative data could be collected. However, the booth had a near-constant stream of visitors, and many observations and comments were recorded in order to make final improvements to the game.

Figure 7.4a: Babies & Basilisks at the PAX East WPI booth

7.4.1 Bugs

The most severe bug encountered at PAX was the boss fight starting prematurely. This bug usually occurred when controller players were fighting enemies in the large combat arena. When the boss room spawned on the shelf, but before the VR player could even place it, the players would be teleported to the boss arena, and the boss fight would begin. However, because the controller players would not have completed the combat arena, the camera would stay fixed on their previous location, preventing them from actually fighting the boss.

Another bug that occurred frequently at PAX was players falling out of a hallway, and instead of respawning at the end, they would continuously fall to the floor and take damage. If a solo player fell out of a hallway, the camera would additionally follow their fall instead of remaining in the hallway as it should have. Enemies also had a falling-related bug, where they would fall from a room, either by glitching through the falls and floor, or by being unintentionally pushed by the VR player. Since enemy encounters could not be completed until all of the enemies were defeated, once these enemies had fallen, the controller players could not progress until the VR player found the enemy on the floor and picked it up, returning it to the room.

7.4.2 Observations and Comments

Overall, the game was received well by players at PAX. Many people asked if the game would be put on Steam, and some stayed for more than one round. The game seemed especially popular among younger visitors, suggesting the playful theme is effective. Some players were put off by the lack of refined lighting and placeholder UI visuals, and improvements to these areas were completed shortly after the event concluded.

The team also discovered an unexpected trend in the VR player's behavior which affected the subsequent design of the game. The PAX build was created shortly after the design decision to remove the VR player's ability to individually place enemies into the level and replace it with automatically spawning enemies was implemented. While the team had made this change, the ability of the VR player to be able to pick up enemies was overlooked, and accidentally included in the PAX build. However, VR players used this mechanic to aid the controller players during fights by holding the enemies in place so they could not attack but the controller players could defeat them. Because of this unexpected but frequent cooperation between the VR player and controller players, the team decided to keep this functionality in the game and, in keeping with this spirit, updated the game so that if a VR player flung an enemy from the room, the controller players no longer had to defeat it to proceed.

8. Conclusion

The team initially set out to create a VR party game, as this fits well within the current state of VR technology and availability. While several game concepts were considered, a level-construction-based game was decided on, with one VR player building levels and up to four controller players navigating them. To create a lighthearted and fun feel to the game, a toy theme was decided on.

The overall gameplay of *Babies & Basilisks* features two distinct sides. The VR player constructs a dungeon out of prefabricated sections, while the controller players navigate through this dungeon as it is built, fighting enemies and avoiding obstacles. This simultaneous, dual-sided gameplay creates a unique, dynamic experience. The significant scale difference between the two kinds of players, as well as the different audio environments, leads to very different experiences despite being in the same environment.

The artists were able to create a large quantity of assets, which allowed the game to have four distinct controller player characters and five enemies, all of which had entirely unique attacks and abilities that were fully animated. These characters all fit well into the environments, giving the game a uniform art style despite the varied look. The programmers had to develop many systems for the game to work properly, integrating multiple third-party libraries and developing some novel solutions for which no third-party library existed. The game's code is

clean and performs well, and there are no known major bugs in any of the game's systems. The audio designer had the challenging task of creating two separate and distinct sound environments. These two environments work well together to convey the different experiences of the VR and controller players. Additionally, the inclusion of original voice lines into the game greatly improves the originality and personality of the characters.

8.1 What Went Wrong

The most significant issues occurred during playtesting. Invariably, the team tested the build prior to the playtesting session (on the machine itself in the sessions held in the Global Lab), but when the playtesting sessions were about to start, the game would usually exhibit bugs that required restarting the computer to fix. Furthermore, several times the usefulness of the playtesting was detracted from by bugs that occurred during playtesting. While discovering these bugs was helpful, the team had hoped to get more playtesting feedback that was not encumbered by buggy encounters. This perhaps could have been mitigated by more testing of the game by the team with many players. Despite the team consisting of five players, testing was usually only done with one or both of the technical team, which led to some bugs that were only encountered when multiple people were playing the game, which did not come up in testing.

Another problem that occurred during development was when the technical team updated the version of Unity they were using from 2018.2 to 2018.3. Although the team knew the potential risks of changing the development platform halfway through development, the new version of Unity included support for prefabs, pre-made objects that could be placed or spawned into the level, to be nested within each other. At the point in development when Unity 2018.3 was released, the team was working with the challenge hallways, which required traps and platforms, which were all prefabs, to be placed within preset sequences to load the hallway with, which were also prefabs. With Unity 2018.2, changing one trap prefab resulted in many more prefabs needing to be updated before they could be effectively tested, and so the team decided it would save a great amount of time to update to Unity 2018.3. While this aspect of 2018.3 did improve development time significantly, it also caused unexpected difficulties. For instance, the particle system in 2018.3 had a bug that created a memory leak specifically when using VR. This resulted in the team being forced to disable the particle systems so the computer running the game would not crash, and begin considering alternatives to using particle effects. Fortunately, 2018.3 was eventually updated several weeks after the upgrade, fixing this issue and allowing the particle systems and nested prefabs to remain.

There were also some issues implementing the art into the game for the VR player model, most notably for the boss. Because of the constraints given by Unity's VR skeleton, namely that it needed adult humanoid proportions, the bones for these two models needed to be set up differently. For the VR player model (the baby), the team initially created a model which did not follow these constraints, causing the model to work incorrectly with the skeleton's arms. To

mitigate this, the artists made the baby model's arms longer. However, this attempted fix did not fully help the situation, and it was found that the VR camera would clip through the model's head. The end solution for this model was to create an entirely different player model of just the baby's hands for the first-person view, while keeping the full model for the third-person view from the controller players. For the boss rig (dragon/ basilisk), the model did not match up to the humanoid proportions which Unity's skeleton needed, causing the limbs to break repeatedly. In order to get around this issue, the team made some tweaks to the bones once the rig was in-engine, and made it so the limbs worked more smoothly.

While not disastrous, one potential misstep was using the SteamVR interaction system, included with the SteamVR Unity plugin. This system was developed by Valve when creating their early VR demo *The Lab* [20]. While it does contain many useful pre-made features, it also comes with serious limitations, such as difficulty in showing certain elements only on the VR player's camera, and temporarily disabling the intractability of objects. Furthermore, it appears that the system is not commonly used, as there are very few questions about it on the Unity answers and forums sites. It would have been better to start the VR development process by considering other alternatives, such as just using the base SteamVR code, or VRTK (another VR Unity library).

8.2 What Went Right

Babies & Basilisks is a well-developed prototype of an asymmetric VR party game involving two separate branches of gameplay, each requiring their own systems and presenting their own issues. As is common in game development, many parts of the game required close collaboration between two or more departments, providing additional experience to all team members outside their area of expertise.

Several factors contributed to the overall success of the project. While the team did not prepare a formal design document prior to beginning work, we did spend the term prior determining specifics of the design. This enabled all team members to immediately begin development at the start of the year, without waiting for the design to be ready.

The composition of the team provided a highly effective distribution of skills. With two artists and two programmers, no one person was forced to take on one of these major portions alone. While there was only a single audio developer, the programmers were able to help with the technical aspects of audio implementation. Using tools that team members were already familiar with was essential as well. For example, due to previous experience in Unity, the programmers were able to focus on developing specific features instead of learning how to use the engine.

Overall, the entire team felt that this MQP was a success, as well as a valuable learning experience. From an art perspective, the toy theme provided a challenge to create a wide variety of models with a fun, lively color scheme and visual style inspired by stylized media. From a

technical perspective, functionality for both VR and controller players had to be implemented, providing challenges in implementing two entirely different types of gameplay and polishing both experiences. Based on the results and feedback that the team received both in person and through written comments, the team felt as though the target experience goal was met and that this MQP was a project to be proud of.

8.3 Future Work

There are several ways that the game could be expanded and improved from its current state. From a usability perspective, it might be beneficial to make the audio device selection more clear, so that players know exactly how to set up both the default Windows audio output and the in-game output selection. Furthermore, since the game should appeal to both young and mature audiences, adjustable VR player height would be very helpful. The height of certain elements, such as hallways, the shelf, and the boss should adjust so they are accessible to players of any height.

In terms of expanding gameplay, it would be beneficial to add more variety for both the controller and VR players. On the level building side, hallways with more varied looks could increase the visual appeal of levels. Combat rooms of different shapes would also be helpful, both in offering choice to the VR player and variety to the controller players. These rooms could incorporate verticality, and might have entrances at different levels so that the VR player could build a multi-level maze. Finally, a larger variety of enemies and bosses would greatly enhance the replayability of the game, especially if bosses had different abilities for the VR player to try.

Another element that could be expanded upon is interaction between the VR player and controller players. Currently, the VR player can interact with the controller players directly though the boss fight, operating traps in the hallways, and holding enemies during combat. Some additional opportunities for interaction might be the ability for the VR player to pick up or push the controller players in a limited manner, enabling the controller players to attack the VR player if they get too close, and creating a way for the VR player to give commands to the AI enemies.

In addition to these design changes, some more minor improvements could be beneficial as well. The certain player abilities could be improved, such as the Kangaroo. This character was reported by playtesters to be more difficult to use, suggesting its abilities might need refinement. The abilities of the boss were also difficult for some players to understand, suggesting that the boss tutorial needs to be improved. These improvements would help to enhance replayability and decrease frustration.

9. References

[1] Steel Crate Games. Keep Talking and Nobody Explodes. Game [PC]. Steel Crate Games (8 October 2015).

[2] Foreignvr. Ruckus Ridge VR Party. Game [PC]. Foreignvr (5 April 2016).

[3] Team Future LLC. Black Hat Cooperative. Game [PC]. Team Future LLC (29 July 2016).

[4] Japan Studio. Playroom VR. Game [PS4]. SIEA (13 October 2016).

[5] Team Panoptes. Panoptic Demo. Game [PC]. Team Panoptes (2016). Retrieved from https://store.steampowered.com/app/541930/Panoptic/. Accessed 19 April 2019.

[6] Facepunch Studios. Garry's Mod. Game [PC]. Valve (24 December 2004). Retrieved from https://www.viveos.net/video/iZpRDkuqmgw/gmod-death-run-fu.html. Accessed 19 April 2019.

[7] Blizzard Entertainment. Overwatch. Game [PC, Playstation 4, Xbox One]. Blizzard Entertainment (24 May 2016). Retrieved from https://www.businesswire.com/news/home/20160602006554/en/Overwatch®-Global-Hit-7-Million-Players—and-Counting. Accessed 19 April 2019.

[8] Owlchemy Labs. Job Simulator. Game [PC, PS4]. Owlchemy Labs (5 April 2016).

[9] Ben Esposito. Donut County. Game [iOS, PC, Playstation 4, Nintendo Switch, Xbox One]. Annapurna Interactive (28 August 2018). Retrieved from https://www.epicgames.com/store/en-US/product/donut-county/home. Accessed 19 April 2019.

[10] Level-5. Ni No Kuni. Game [Playstation 3, Playstation 4, PC, Nintendo DS]. Bandai Namco Entertainment (9 December 2010). Retrived from https://www.playstationlifestyle.net/2019/03/05/ni-no-kuni-familiars-should-have-stayed-for-sequel/. Accessed 19 April 2019.

[11] Nintendo. The Legend of Zelda: The Wind Waker. Game [GameCube]. Nintendo (13 December 2002).

[12] Walt Disney Pictures, Pixar Animation Studios. Toy Story. Film. Buena Vista Pictures Distribution (22 November 1995). Retrieved from http://www.thedisneymoviereview.com/five-characters-that-should-find-love-in-toy-story-4/. Accessed 19 April 2019.

[13] Walt Disney Pictures, Pixar Animation Studios. The Incredibles. Film. Buena Vista Pictures Distribution (5 November 2004). Retrieved from https://d13ezvd6yrslxm.cloudfront.net/wp/wp-content/images/incredibles2-jackjack-bear-floor-700x326.jpg and https://pixar.fandom.com/wiki/Jack-Jack_Parr. Accessed 5 April 2019 and 24 April 2019.

[14] Dreamworks Animation. The Boss Baby. Film. 20th Century Fox (31 March 2017). Retrieved from https://statici.behindthevoiceactors.com/behindthevoiceactors/_img/shows/banner_4007.jpg. Accessed 5 April 2019.

[15] Valve Corporation. Team Fortress 2. Game [PC, Xbox 360, Playstation 3]. Valve Corporation (10 October 2007).

[16] From Software. Dark Souls. Game [PC, Xbox 360, Playstation 3]. FromSoftware, Namco Bandai Games (22 September 2011).

[17] Sledgehammer Games. Call Of Duty: WWII. Game [PC, Playstation 4, Xbox One]. Activision (3 November 2017). Retrieved from: https://i.redd.it/j443t47yrdwz.jpg. Accessed 20 April 2019.

[18] Traveller's Tales. Lego Star Wars: The Complete Saga. Game [PC, Playstation 3, Wii, Xbox 360]. LucasArts (6 November 2007). Retrieved from https://static.gamespot.com/uploads/original/gamespot/images/2007/309/reviews/836499-939576_20071106_002.jpg. Accessed 20 April 2019.

[19] Katrina and the Waves. Walking on Sunshine. Katrina and the Waves. Song. (26 April 1985).

[20] Valve Corporation. The Lab. Game [PC]. Valve Corporation (5 April 2016).

[21] Nintendo. Mario Party Series. Game [Nintendo 64, Gamecube, Nintendo DS, Nintendo Wii, Nintendo Switch]. Nintendo (December 1999). Retrieved from: https://cdn.gamerant.com/wp-content/uploads/super-mario-party-online-play-luigi-bowser.jpg.optimal.jpg. Accessed 24 April 2019.

[22] Bethesda Game Studios. The Elder Scrolls V: Skyrim. Game [PC, Playstation 3, Xbox 360]. Bethesda Softworks (11 November 2011). Retrieved from: http://i.imgur.com/tqCqeZP.jpg. Accessed 24 April 2019.

[23] Team VRGC. Great VR Party Games That'll Make Any Game Night. Web. vrgamecritic (30 May 2017). Retrieved from: https://vrgamecritic.com/article/great-vr-party-games-that-will-make-any-game-night. Accessed 23 April 2019.

[24] How to Turn an Attic Into a Playroom. Web. Embrace Family (21 July 2016). Retrieved from: https://www.embracefamily.com/how-to-turn-the-attic-into-a-playroom.html. Accessed 10 April 2019.

[25] Kangaroo Hopping Jumping Windup Tin Toy. Web. Tiny Toy Arcade. Retrieved from https://www.tintoyarcade.com/kangaroo-tin-toy.html. Accessed 19 April 2019.

[26] TOMY Wind-Up Kangaroo Toy Flips Works. Web. PicClick. Retrieved from https://picclick.com/TOMY-Wind-Up-Kangaroo-Toy-Flips-Works-172605466955.html. Accessed 24 April 2019.

[27] Oculus expected to sell 1.3M Quest units in 2019; XR revenue reached $6.6B in 2018 and is projected to increase 442% by 2022. Web. SuperData Research Holdings (24 January 2019). Retrieved from https://www.superdataresearch.com/xrupdate/. Accessed 25 April 2019.

10. Appendices

Appendix A. Tool Choices

Game Engine

Choosing a game engine was perhaps the most important technical decision in the pre-production stage of the project. Making the wrong choice here could have significantly hampered progress later on, or caused lots of wasted time. While for some games, especially 2D, there are many choices for the engine, VR games are almost exclusively made in either the Unity or Unreal engine, or a proprietary engine not publicly available. Both Unity and Unreal have been used to create many popular VR games, and have comparatively mature support for VR. Therefore, the team narrowed the game engine choices to these two engines.

In choosing between Unity and Unreal, there were several considerations to be made. Since both engines have been extensively used in creating VR titles, it was clear that both engines would suit VR game development in general. The first thing that had to be considered outside of that was the general ease of development, and how quickly new features could be prototyped and iterated upon. Performance and graphics would also need to be considered, since the rendering requirements of the Vive are greater than those of a typical desktop or console game. General flexibility of the engine was important, since some engines are best suited to a certain genre, and game mechanics seen as unconventional may be difficult to implement. Outside of engine features, the experience of the programmers using each engine was an important factor, which could significantly speed up or slow down development. Finally, outside of the technical space, licensing had to be considered. As a small team working on an educational project, a free engine would be preferable.

The first engine considered was Unity. Unity is a highly flexible game engine which as been used to create many diverse experiences in nearly every genre, from card games to VR experiences. The only programming language used in Unity development is C#, which can be a barrier to non-programmers, but makes it easy for those familiar in object-oriented programming to write clean-looking and robust code. The design of Unity makes it easy to quickly test and iterate on features, due in part to the fast compile times and lack of editor crashes that are a result of using C#. Performance, however, is the main drawback of Unity. While it can support high-end, near photorealistic graphics, the engine will struggle greatly to deliver this level of fidelity with stable performance. This is reflected in popular titles made in Unity, as nearly all of them use stylized, non-photorealistic art. Additionally, some considerations must be taken when using C# in game code. The use of a garbage collector means that dynamic memory allocations must be used sparingly, and lack of statically-allocated arrays further complicates this. For non-commercial projects or teams raising under $100,000 per year, all features of Unity can be

used for free. The only issue is the mandatory splash screen in the packaged game, which can be removed with the “Plus” version.

Unreal Engine was considered next. Unlike Unity, games made with Unreal tend to be less diverse. The engine, originally developed by Epic Games for their first-person shooter series of the same name, is still primarily used to create first- and third-person shooters. However, it has been successfully used in other genres, including VR. Unreal currently offers two means of programming: through C++ coding or a visual scripting system called Blueprint. Blueprint, while great for creating simple features and one-time events, quickly becomes messy and unmanageable when implementing more complex systems. C++, on the other hand, has other problems. The compile times for C++ in Unreal tend to be long, increasing iteration times. Furthermore, due to the unsafe nature of C++, it is easy for simple programmer mistakes to crash the editor, which runs in the same process as game code. However, where Unity and C# fail in performance, Unreal does well. Unlike Unity, Unreal is well optimized for high-fidelity photorealistic graphics. Additionally, a skilled programmer can write more performant code using C++ than would be possible in C#, due primarily to manual memory management. Similar to Unity, Unreal is free for limited commercial use. However, it has the additional benefit of not having a mandatory splash screen.

After carefully reviewing both game engines, Unity was selected for this project. The ease and speed of developing and iterating would make it invaluable for meeting the AlphaFest milestone. If the game was meant to be photorealistic, using Unity might lead to performance issues or worse-than-desired visuals. However, due to the simple art style of the game, this would not be an issue. In the event that the game was to be published, a Unity Plus license could be purchased cheaply in order to remove the splash screen. Finally, Unity makes it easy to use external libraries in code, which would be essential in creation of the secondary audio system, discussed later on.

Source Control

After the engine was decided, another important tech decision was the version control software. Three options were considered for this: Git, Unity Collaborate, and Perforce.

Git has the advantage of being hosted for free via services such as Github and Bitbucket. It is used extensively in software development, including with Unity. It offers important features such as creating branches to work concurrently on different features, and pull requests to easily review code. The main drawback of git is its clumsiness when working with unknown file types. With text files such as source code, Git and other source control systems can easily merge changes. With binary files, however, Git is unable to merge changes and must discard one set of changes, which can lead to lost work. Some files created by Unity, such as scenes and prefabs, can be treated as text files, but merging them with Git is not always successful.

Unity Collaborate, a built in feature of Unity, is much simpler and easier to use than Git. It is able to more effectively merge Unity's prefab and scene files (although it will still sometimes require discarding one), and does not require an external client. However, it lacks the branching and code review features provided by sites such as Github.

Finally, Perforce is a less commonly used option that works well with Unity. Unlike Git and Unity Collaborate, it has an exclusive checkout feature where only one person can work on certain files at a time. This prevents the case where changes have to be discarded, but doesn't actually allow merging of these files. Furthermore, unlike the other two options, Perforce would require the team to set up and maintain a Perforce server. This would provide hosting of the project, but not other services such as code review.

Of these three options, Git was determined to be the most suitable to the project. While it is more complicated to use, the decided that the functionality provided by a hosting platform such as Github would be highly beneficial. Since the programmers were already familiar with Git, the complexity was less of an issue. Furthermore, it would require neither additional setup nor payment by the team. The binary merge issue could be largely avoided by working in separate scenes files, and making sure to only stage these files for commit if significant changes were made.

Artistic Software

The final tool decisions to be made were by the art team. Given the multi-step process for setting up fully optimized character models, multiple programs and tools would be needed at different stages of the process.

For the character modeling process, Zbrush was chosen over box-modeling programs like Maya and 3DS Max. This was due to its focus on sculpting organic shapes, and because the artists had a general preference for Zbrush. For texturing, the primary consideration was how complex the textures should be. Initially, the artists planned to use Substance painter, due to its ability to create a great range of visuals and ability to realistically mimic the look of rusted metal, plastic, cloth, and even skin. However, as production went forward, the team decided on Zbrush's polypaint tool as it would help create a more uniform look throughout the various art elements. Rigging for each of the characters was done in 3DS Max, utilizing the CAT Rig system. 3DS Max was chosen over programs like Maya and Mixamo due to the art team's prior experience with Max, as well as the necessity for a variety of non-standard rig types - including quadrupedal characters, and models with need for secondary animation layering. Given this choice, animation for all of the models was done in 3DS Max as well in order to eliminate cross-program issues with skeleton transfer.

For the environmental modeling process, two programs were used: Maya and 3DS Max. Maya was used for the grey-boxing stage, due to its simplicity and ease of use. For the finalized versions of the environmental models, both Maya and 3DS Max were used. As with the

grey-boxing process, Maya was generally easier to use for the overall shapes and starting models, which were then moved into 3DS Max for finishing details, largely due to 3DS Max's extensive modifier library.

Project Management

The team used a variety of project management software in order to best facilitate clarity and productivity. Git was the chosen source control platform. The project's main repository was hosted on Github, which provides private repositories to students. Sourcetree was used as the client by both programmers, due to its helpful visual interface.

Most of the team's written documents and progress files were hosted on Google Drive. Google Drive was chosen over similar platforms such as Dropbox due to ease of access and useful collaboration features.

Almost all of the team's online communication was done through Slack. This allowed the team to communicate quickly and effectively, as well as sharing documents when necessary. The communication was also divided into multiple separate channels, including Tech, Art, and General, so communications could be organized and multiple conversations would not clutter each other.

Appendix B. Dialogue Scripts

Dr. Bear:

1. “Let’s play Dr.Bear!”
2. “Give me a hug!”
3. “I’m the doctor and you’re my patient! Let’s hope I cut into the right part this time...”
4. “I got my doctorate at Hugs & Kisses Medical School. My degree came with candy and malpractice waivers!”
5. “My supervisor said if I bring my mortality rates down, I get a cookie!”
6. “Oooh… my tummy hurts.”

Cowboy:

1. “A plastic rifle in one hand and a plastic whiskey in the other, it’s a toy cowboy’s dream.”
2. “I’d tip my hat to ya, but it’s permanently glued to my head.”
3. “I’m wanted in 12 toy boxes for skipping out on playtime. My bounty; 2 juice-boxes and nappies”
4. “What in tarnation!”
5. “Well, I’ll be an arm shark’s uncle!”
6. “Want to know how I get my moustache this shape? Grit and lots of industrial adhesive!”

Tess:

1. “I got evicted from Darbie’s Dreamhouse for playing too, rough (laugh)”
2. “Warning, explosions may be imminent (laugh)”
3. “Pew pew, pow, pshbuh, explosions!”
4. “I made a bet for a weapon with an army guy. Told him he wouldn’t survive being melted with a magnifying glass. Did I win? Well, let’s just say I haven’t been without this rocket launcher since that day. ”

Kangaroo:

1. “Hello I am a Kangaroo”

Appendix C. Informed Consent Form

Informed Consent Agreement for Participation in a WPI Research Study

Investigator: Brian Moriarty, IMGD Professor of Practice
Contact Information:
Brian Moriarty, bmoriarty@wpi.edu, 508 831-5638
Title of Research Study: Asymmetrical VR Game
Sponsor: WPI

Introduction: You are being asked to participate in a research study. Before you agree, however, you must be fully informed about the purpose of the study, the procedures to be followed, and any benefits, risks or discomfort that you may experience as a result of your participation. This form presents information about the study so that you may make a fully informed decision regarding your participation.

Purpose of the study: The purpose of this study is to obtain playtest feedback in order to locate/address operational bugs, to identify opportunities for design improvement, and to gather data to conduct statistical analyses on to measure games effectiveness towards the experience goal.

Procedures to be followed: You will be asked to play a 5-player game lasting less than fifteen minutes. After completing the game, you will be asked to complete a brief, anonymous survey describing your subjective experience and opinions. At no point during your play session, or in the survey, will any sort of personal and/or identifying information about you be recorded.

Risks to study participants: One player in each play session will operate in a virtual reality (VR) simulation, located within a limited physical space in the testing laboratory. Virtual “walls” (clearly discernible within the simulation) will indicate the boundary of the corresponding physical space.

Inadvertently crossing this boundary poses a risk of collision with real-world walls or objects, with a resulting potential for injury or property damage. The testing team will remove as many physical objects from the testing vicinity as possible, and provide verbal warnings to prevent the VR player from moving beyond the safe limits of the physical test space.

The VR device worn during the test is connected to its associated hardware with a cable harness. The cable poses a potential risk if it becomes entangled as the VR player moves around the test area. The testing team will monitor the position of the cable, adjust it continuously to keep it safely positioned, and provide a verbal warning in the event of entanglement risk.

Virtual reality hardware is known to cause dizziness or nausea in some users after prolonged use. The VR player is strongly encouraged to pause or end the test session at any time if they begin to experience any sensation of disorientation or discomfort.

Benefits to research participants and others: You will have an opportunity to enjoy and comment on a new game under active development. Your feedback will help improve the game experience for future players.

Record keeping and confidentiality: Records of your participation in this study will be held confidential so far as permitted by law. However, the study investigators and, under certain circumstances, the Worcester Polytechnic Institute Institutional Review Board (WPI IRB) will be able to inspect and have access to this confidential data. Any publication or presentation of the data will not identify you.

Compensation or treatment in the event of injury: There is minimal foreseeable risk of injury associated with this research study. Nevertheless, you do not give up any of your legal rights by signing this statement.

For more information about this research or about the rights of research participants, or in case of research-related injury, contact:

Professor Brian Moriarty, Project Advisor: 508-831-5638, bmoriarty@wpi.edu
Professor Kent Rissmiller, IRB Chair: 508-831-5019, kjr@wpi.edu
Gabriel Johnson, Human Protection Administrator: 508-831-4989, gjohnson@wpi.edu

Your participation in this research is voluntary. Your refusal to participate will not result in any penalty to you or any loss of benefits to which you may otherwise be entitled. You may decide to stop participating in the research at any time without penalty or loss of other benefits. The project investigators retain the right to cancel or postpone the experimental procedures at any time they see fit.

By signing below, you acknowledge that you have been informed about and consent to be a participant in the study described above. Make sure that your questions are answered to your satisfaction before signing. You are entitled to retain a copy of this consent agreement.

_____ Date: _____
Study participant signature

_____
Study participant name (please print)

_____ Date: _____
Signature of Person who explained this study

Appendix D. AlphaFest Survey

VR player

1. How intuitive was it to pick up and place objects? 1 (hard) - 4 (easy)
2. How intuitive was it to set the room type? 1 (hard) - 4 (easy)
3. What goal did you have while playing the game? Did you feel there was enough of a goal presented by the game?
4. How well could you keep track of the controller players? 1 (hard) - 4 (easy)
5. How much did you have to move while playing? 1 (not at all) - 4 (lots)
6. What activity were you primarily doing? (building, watching, placing enemies, controlling boss, etc.)
7. Did you have enough time to both build the maze and monitor the players? 1 (no) - 4 (yes)
8. How intuitive was it to control the boss? 1 (hard) - 4 (easy)
9. Was the music satisfying to listen to? 1 (no) - 4 (yes)
10. Do you have any general comments or recommendations?

Controller players

1. Which character did you play as? (if you played more than one, pick your favorite)
    - Doctor Bear
    - Tess
    - Amos
    - Kangaroo
2. Which did you prefer: elevator room or encountering enemies in the smaller rooms?
    - Elevator rooms
    - Smaller rooms
    - Didn't encounter enemies in elevator rooms)
3. How confusing was it to explore the maze? 1 (easy) - 4 (confusing)
4. How difficult was it to avoid getting damaged? 1 (easy) - 4 (hard)
5. How difficult was it to use abilities/attacks effectively for the character you played as? 1 (easy) - 4 (hard)
6. Were the cameras effective in displaying enough of the scene? 1 (no) - 4 (yes)
7. Did you have enough space to move around and fight? 1 (not enough) - 4 (plenty)
8. Were the sound effects satisfying to listen to? 1 (yes) - 4 (no)
9. Was the music satisfying to listen to? 1 (no) - 4 (yes)
10. Do you have any general comments or recommendations?

Appendix E. IMGD2900 and IMGD3900 Survey

Controller Player Survey

1. Which character did you play? If you played more than one, indicate your favorite.
    - Dr. Bear
    - Amos (Cowboy)
    - Tess (Rocket)
    - Kangaroo
2. Which type of play area did you prefer?
    - Large Elevator Rooms
    - Small Combat Rooms
    - No Preference
3. How easy was it to navigate the maze? 1(hard) - 4(easy)
4. How easy was it to avoid getting damaged? 1(hard) - 4(easy)
5. How easy was it to use the abilities of the character(s) you played? 1(hard) - 4(easy)
6. How easy was it to see what you wanted to look at? 1(hard) - 4(easy)
7. How adequate was the space available for moving around? 1(hard) - 4(easy)
8. How did you feel about the background music? 1(disliked) - 4(liked)
9. How did you feel about the sound effects? 1(disliked) - 4(liked)
10. Is there anything you would like to see added to the game, or that you felt was missing? (open response)
11. Is there anything you felt should be removed from the game? (open response)
12. Please enter any general comments or suggestions below: (open response)

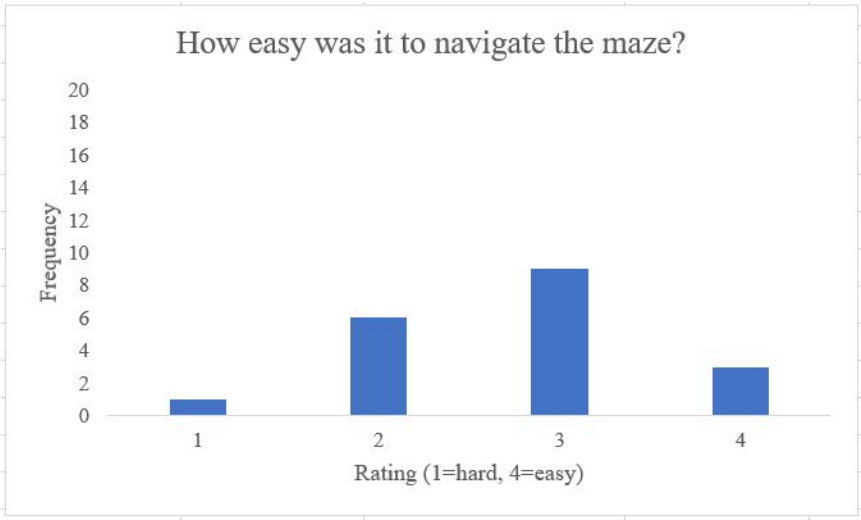
VR Player Survey

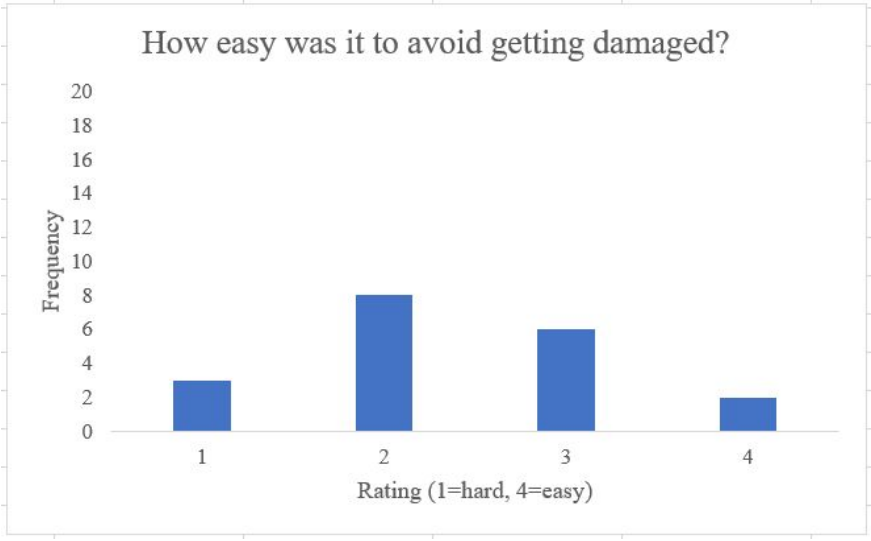
1. How easy was it to pick up and place objects? 1(hard) - 4(easy)
2. How easy was it to change the room type? 1(hard) - 4(easy)
3. How well did the game establish goals? 1(hard) - 4(easy)
4. What goals did you establish while playing the game? (open response)
5. How easy was it to keep track of other players? 1(hard) - 4(easy)
6. How much did you have to physically move around while playing? 1(rarely) - 4(frequently)
7. Which activity was your primary focus during gameplay?
    - Watching Players
    - Building
    - Placing Enemies
    - Controlling Boss

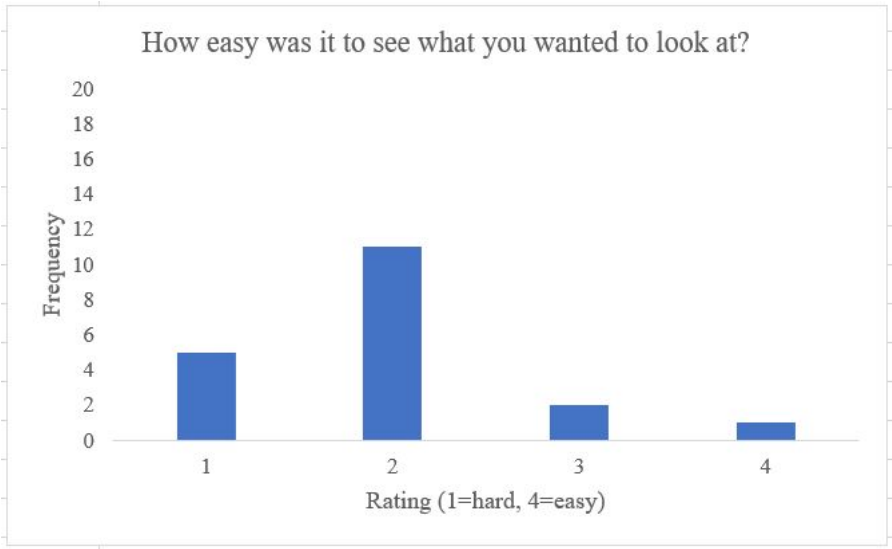
    - No particular focus
8. Did you spend more time building the maze, or monitoring the players? 1(building) - 4(monitoring)
9. How easy was it to control the boss? 1(hard) - 4(easy)
10. How did you feel about the background music? 1(disliked) - 4(easy)
11. How did you feel about the sound effects? 1(disliked) - 4(liked)
12. Is there anything you would like to see added to the game, or that you felt was missing? (open response)
13. Is there anything you felt should be removed from the game? (open response)
14. Please enter any comments or general suggestions below: (open response)

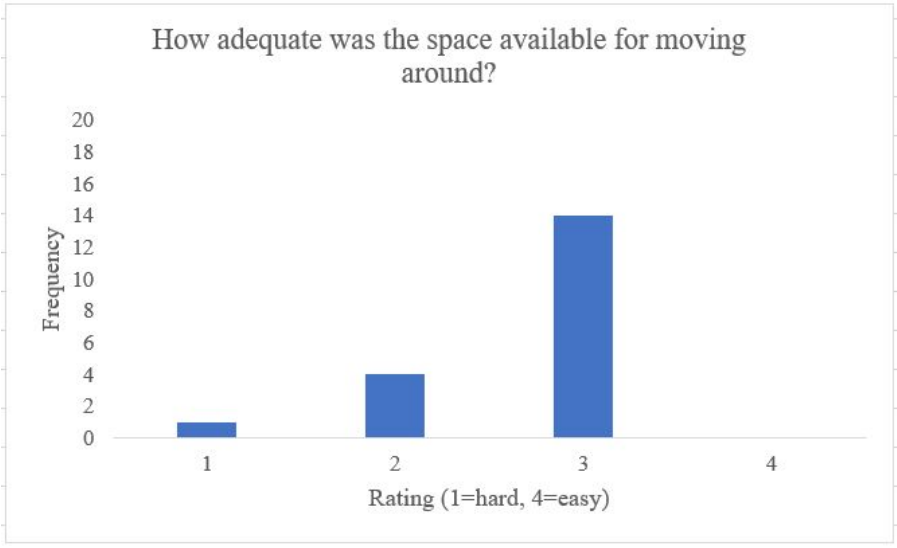
Appendix F. IMGD 2900 Results

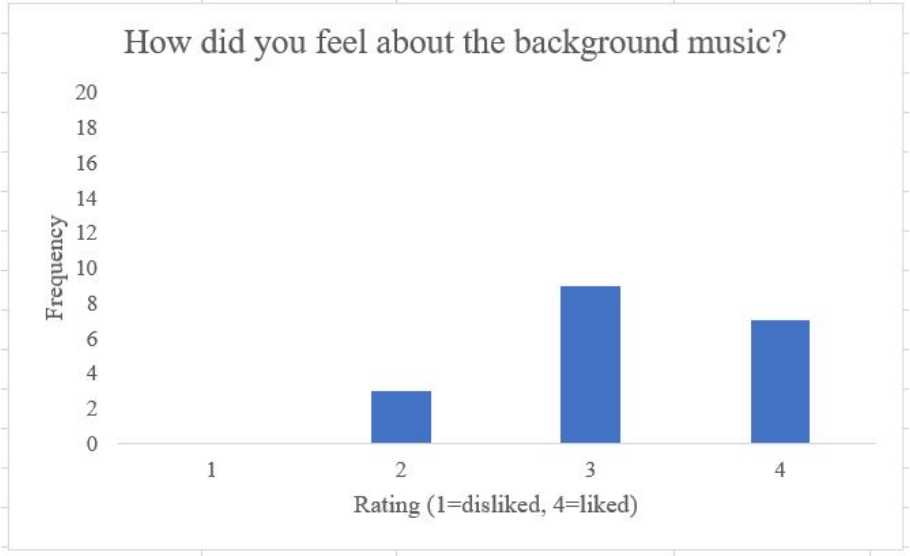
Controller Players

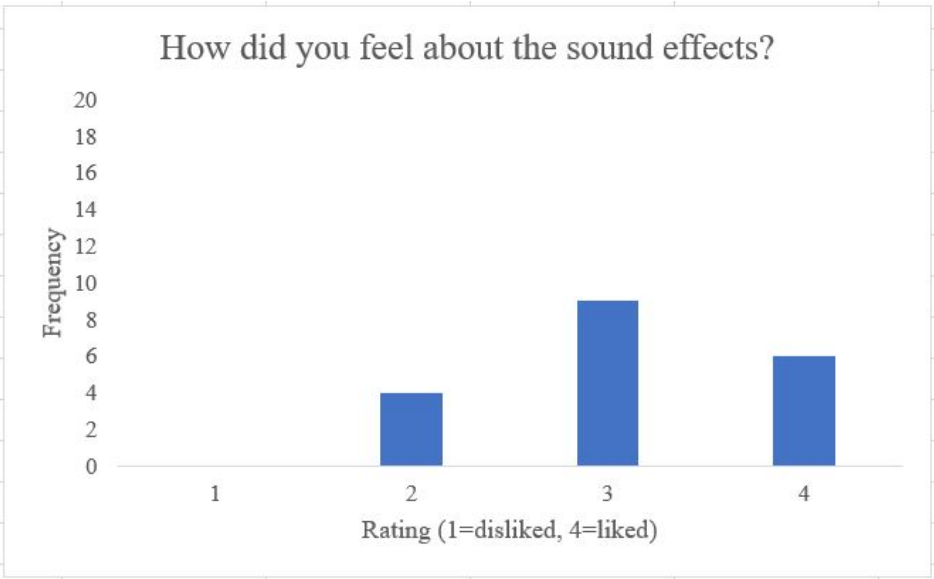


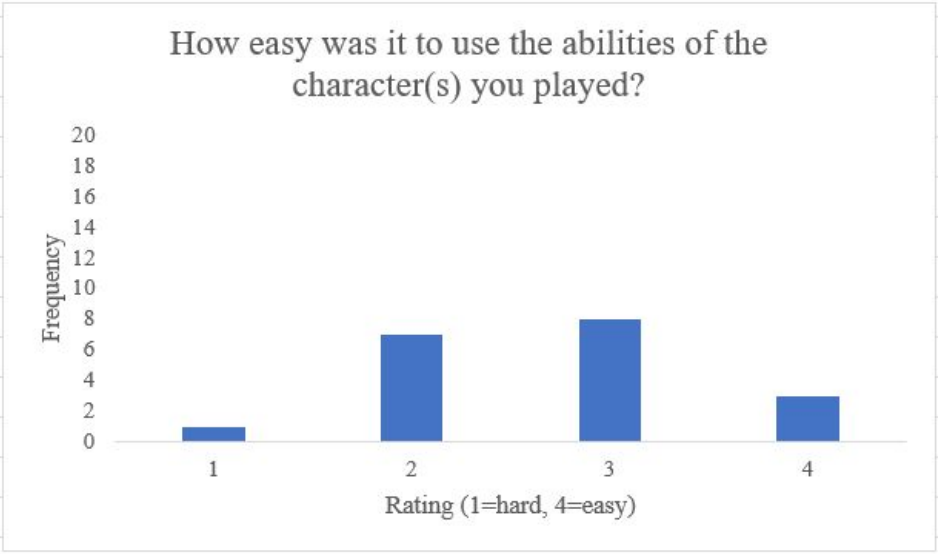


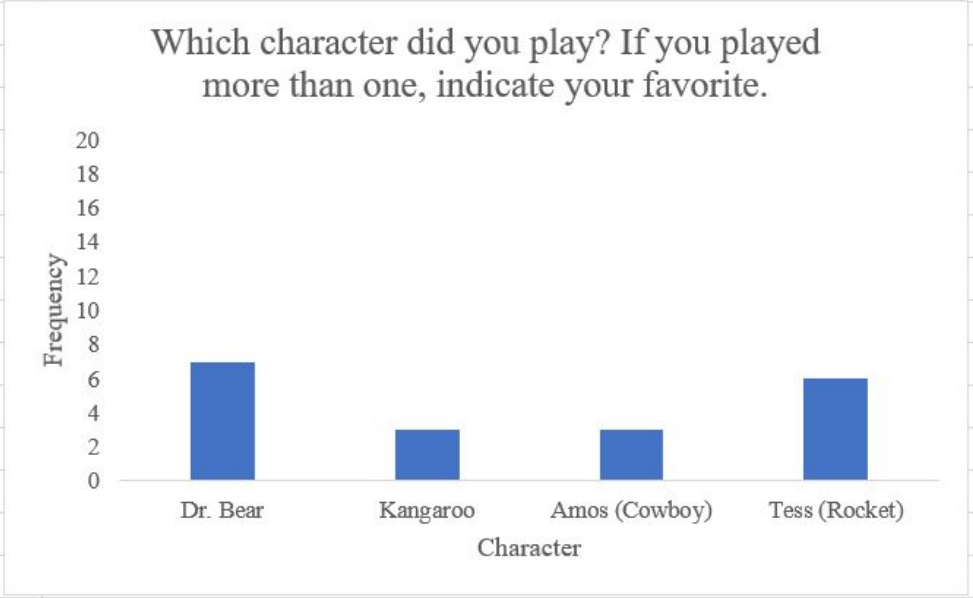

How easy was it to see what you wanted to look at?
Frequency
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
Rating (1=hard, 4=easy)

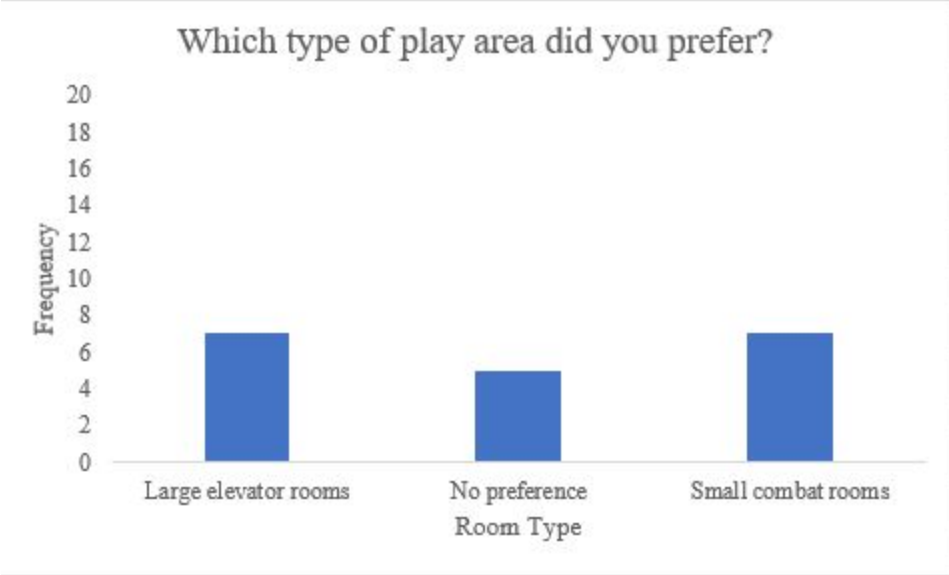

How adequate was the space available for moving around?
Frequency
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
Rating (1=hard, 4=easy)


How did you feel about the background music?
Frequency
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
Rating (1=disliked, 4=liked)


How did you feel about the sound effects?
Frequency
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
Rating (1=disliked, 4=liked)


How easy was it to use the abilities of the character(s) you played?
Frequency
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
Rating (1=hard, 4=easy)





Open Response Questions

Is there anything you would like to see added to the game, or that you felt was missing?

- Some tutorial for what the abilities of the characters did. Some were straightforwards, but others (like the lasso or the rug that Doctor Bear had) weren't clear. May have just been because this was a test version.
- Some feedback (either sound or visual) of when you take damage and when you damage an enemy. Otherwise it feels like the attacks do not have an impact.
- A better camera view for the levels
- Maybe onscreen control help for the toy players?
- Split-screen; seeing the combined view made play more difficult than reasonable

- I wish there was something to do when the VR player was building the level. Maybe more rooms that did stuff on their own without the VR player next to it.
- skill moves seemed to happen with some delay after a button press. more responsiveness would have helped combat
- more information on the story of the game. Was the skeleton our friend? What were we trying to do? Why were the other toys attacking us? Something to do while the skeleton builds the maze.
- More interaction with vr player
- Maybe make it a bit easier to see, as well as add some sound effects for other characters attacks.
- Add in animations for some of the attacks, like the bear feels stiff and lunging doesn't feel very good for his role.
- We need ULTs!!! Also it would be better if you can improve the abilities of the kangaroo.
- More moves or ways to see more drastic ways of damage you can do. hard to see if my attack had a huge effect.

Is there anything you felt should be removed from the game?

- Switching the camera when only one player moves out of the room. Maybe have the camera follow where the greater amount of players currently are.
- The room felt too big, but that might be because we didnt get to finish the whole thing
- The skeleton seemed out of place
- The platforming sections
- The rocket sound was kind of annoying

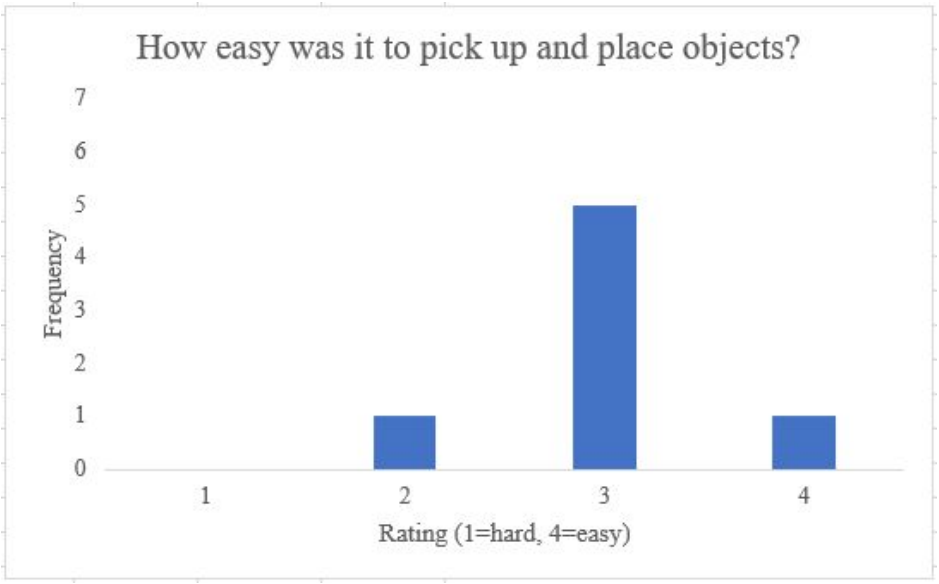
Please enter any comments or general suggestions below:

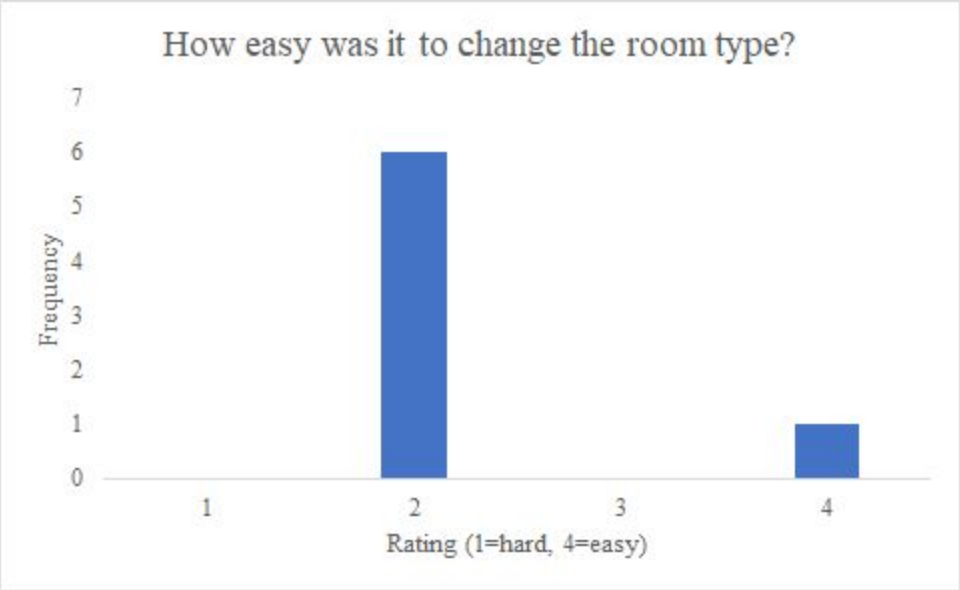
- The camera was a little wonky, and it was really easy for one player to move the camera away from everyone else. Maybe have it so that the camera will be biased towards areas with more players.
- The players were able to stack if they were standing against the wall (3 high) I'm not sure if that's supposed to be able to happen? The camera angles were also a little strange before the jumping puzzles, it was easy to die and hard to see correctly. Aside from that it was pretty fun!
- I had fun! I'd have loved to be able to jump on heads though.
- Had lots of technical difficulties so it was difficult to assess the game's full features and gameplay
- it was hard to use the keyboard to move around and attack.
- "I enjoyed that the VR user had to build a maze for the toy players. However, that did get a little frustrating for the toy players when the maze iddn't get built for awhile. There

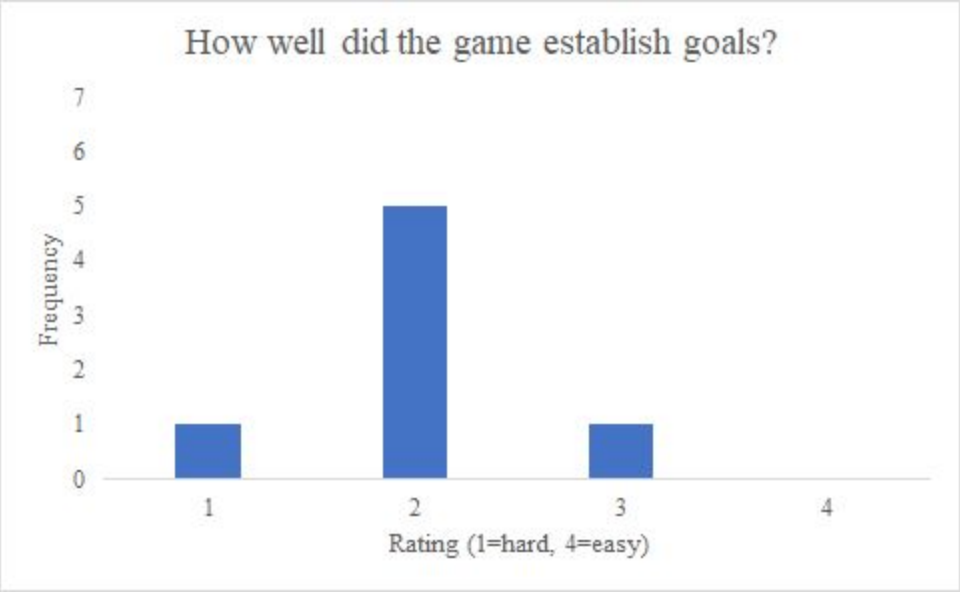
wasn't much for us to do besides jump around and pretend to shoot things. The character movement was rough and the TNT, lasso, and bullet aim didn't feel very lined-up."

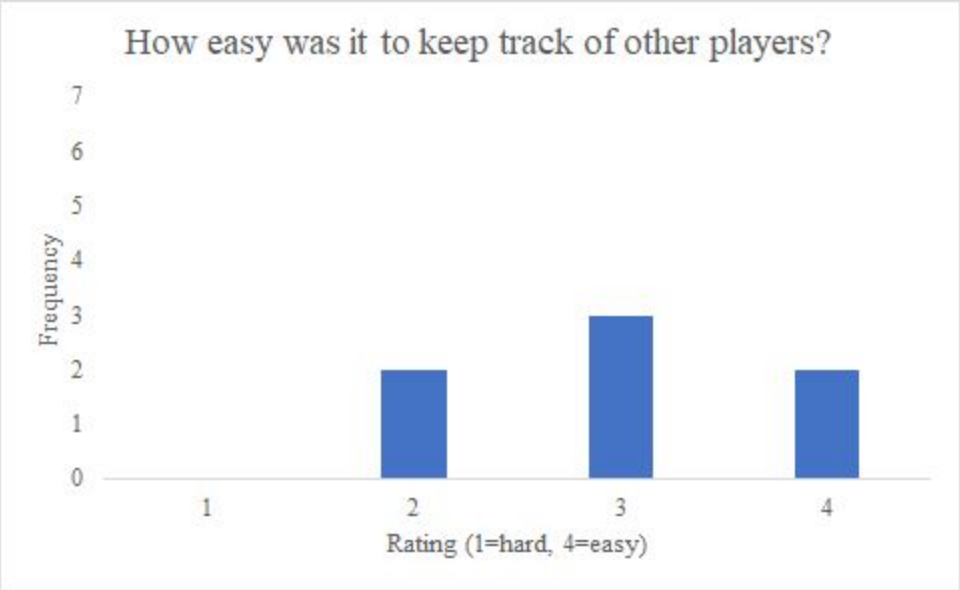
- it was hard to control my character, sometimes i would press shoot and nothing would happen.
- Balance the characters more I guess. I died nearly instantly on kangaroo but Bear was pretty OPOP (plz nerf)
- I like the character models.
- Tess OP, plz nerf
- Really enjoyed playing as the controller player. Would see about improving functionality on keyboard and mouse though. Aiming felt off, and abilities didn't feel too useful with the off aim.
- It was fun.

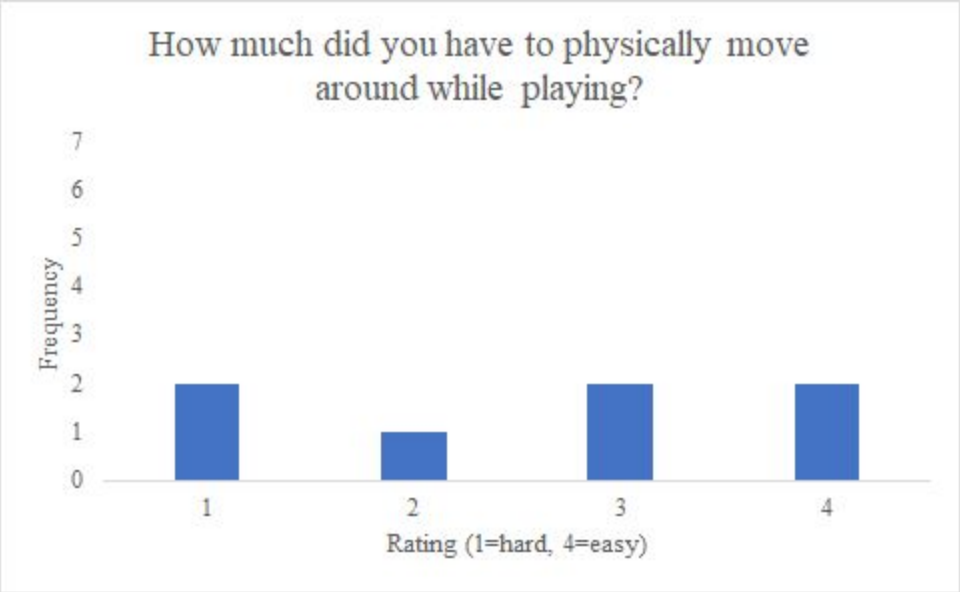
VR Players

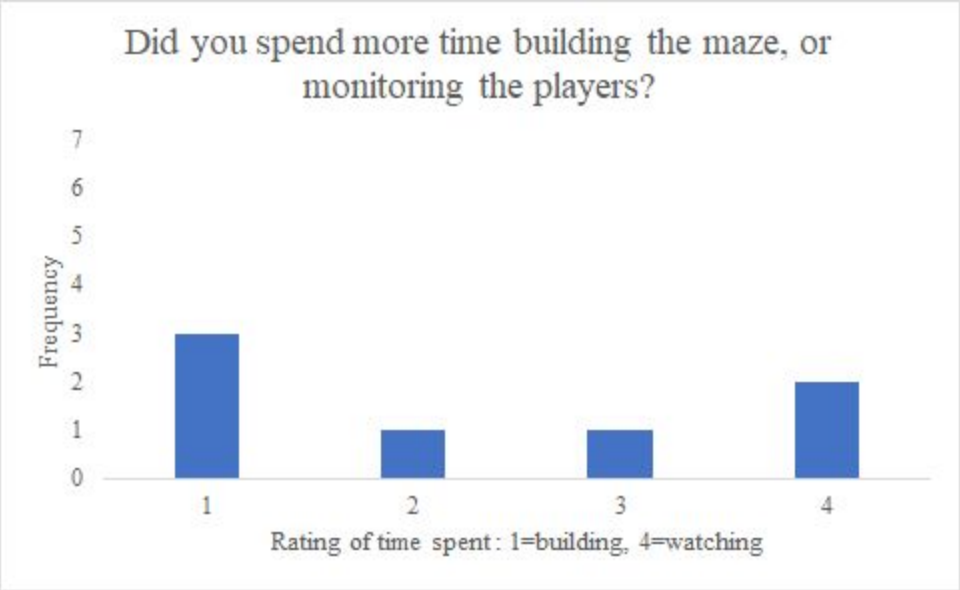


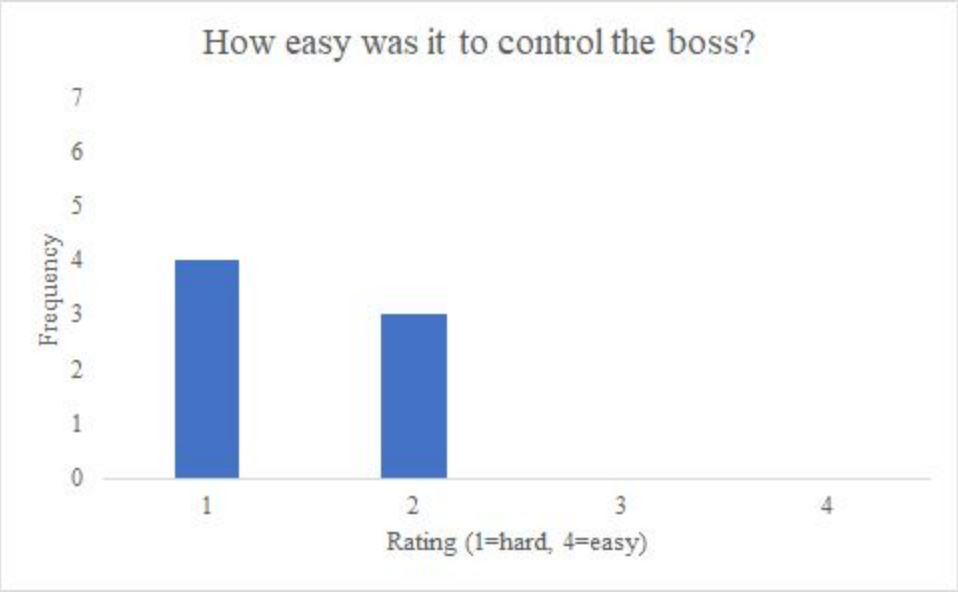

How easy was it to change the room type?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)

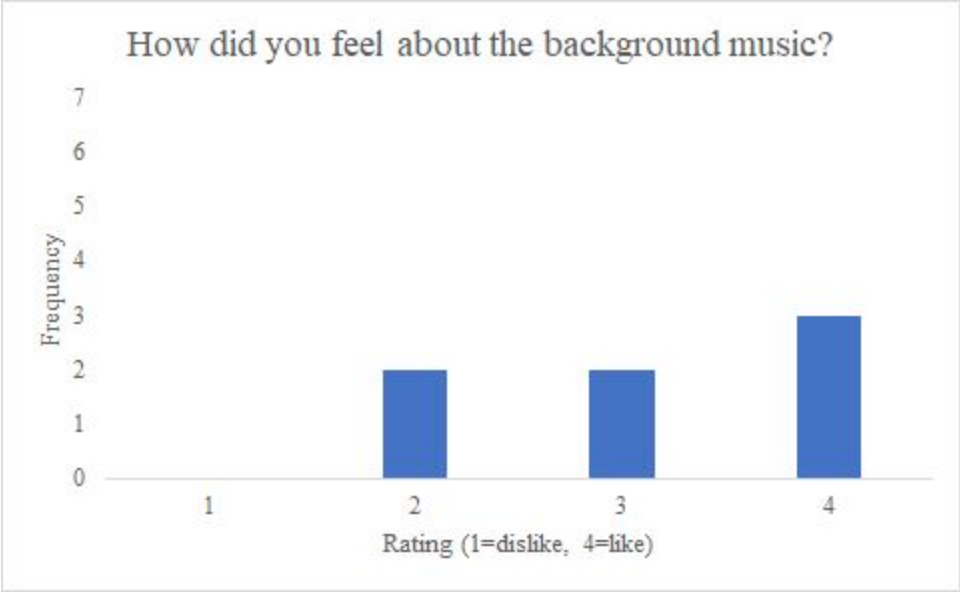

How well did the game establish goals?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)

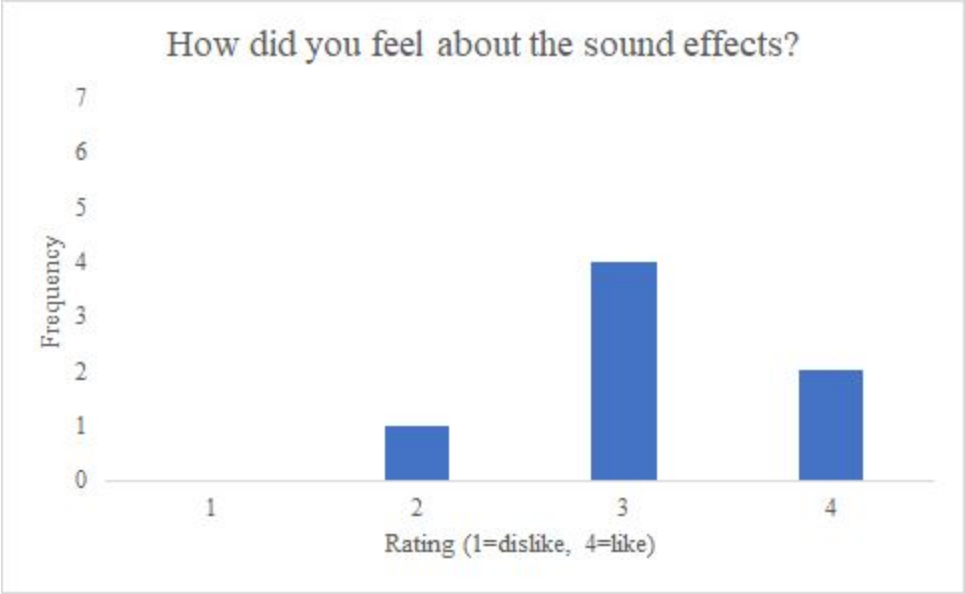

How easy was it to keep track of other players?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)

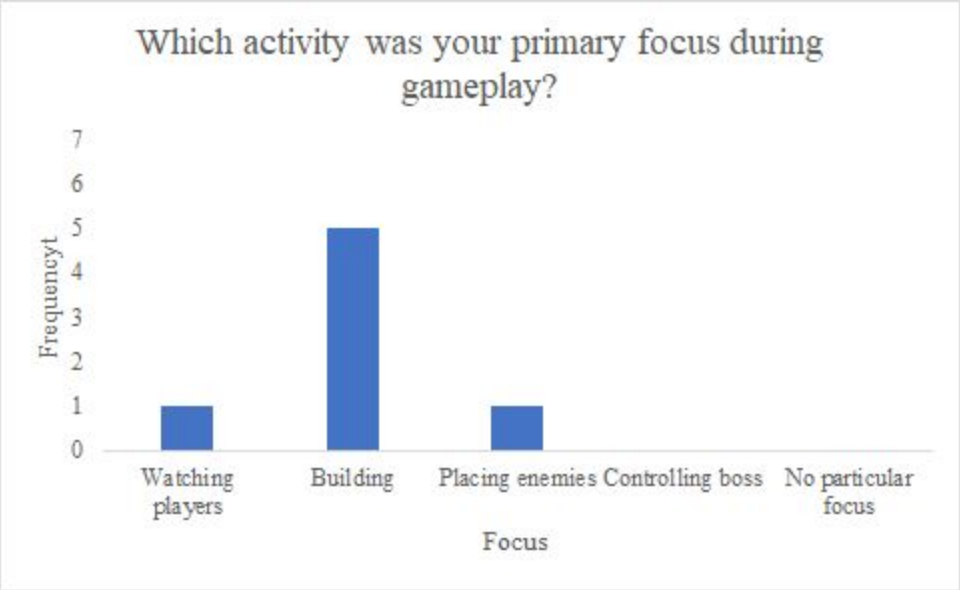

How much did you have to physically move around while playing?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)


Did you spend more time building the maze, or monitoring the players?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating of time spent: 1=building, 4=watching


How easy was it to control the boss?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)


How did you feel about the background music?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=dislike, 4=like)


How did you feel about the sound effects?
Frequency
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=dislike, 4=like)



Open Response Questions

What goals did you establish while playing the game?

- Throw as much stuff at the players as quickly as possible/destroy the players
- move stuff around until it clicks somewhere
- I was just trying to building things for players to keep up
- Pick things up and put them together
- Place rooms for the toys, move traps when they were near it
- making a path for the players
- Have fun. I would've appreciated to learn earlier that the goal was to set up challenges for the final goal room, rather than kinda improvising it on the spot.

Is there anything you would like to see added to the game, or that you felt was missing?

- More available options to throw into an area, rather than just one object.
- I couldn't really hear any sound effects
- More ways to move as the boss. Wasn't hard to change levels but did not understand if I really was changing anything. Add a way I can interact more.

Is there anything you felt should be removed from the game?

- Being able to teleport outside the room

Please enter any comments or general suggestions below:

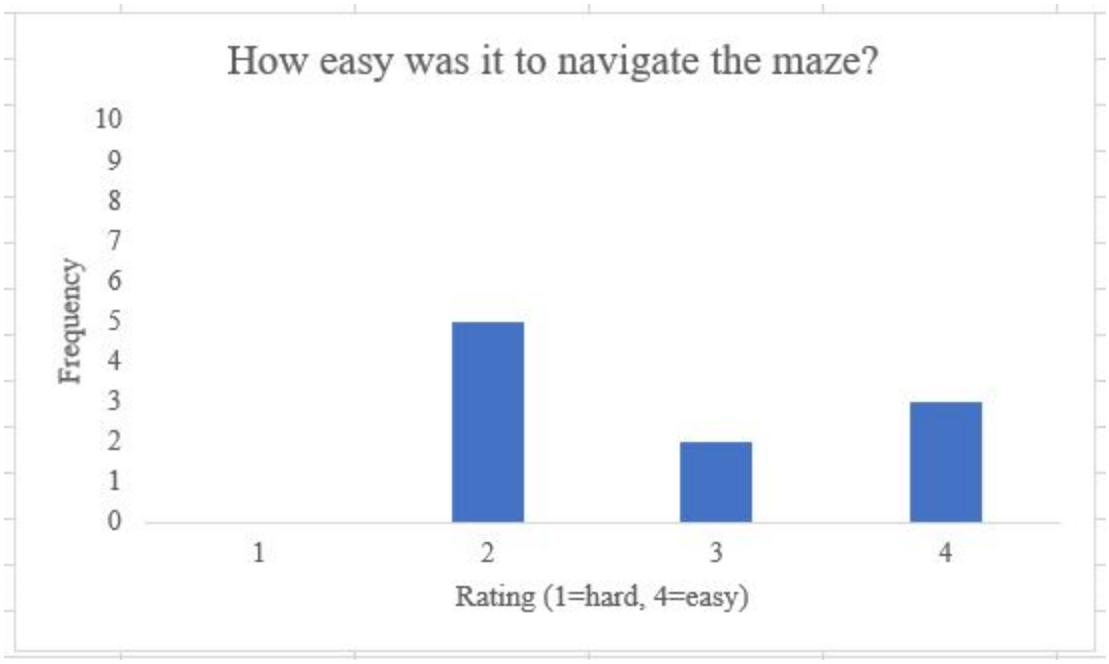
- Maybe I was missing something but I felt like I needed a wall or some time to build things up before I let players go through I felt like I was always trying to building things and make enemies to keep up with players. I also had trouble telling exactly what my

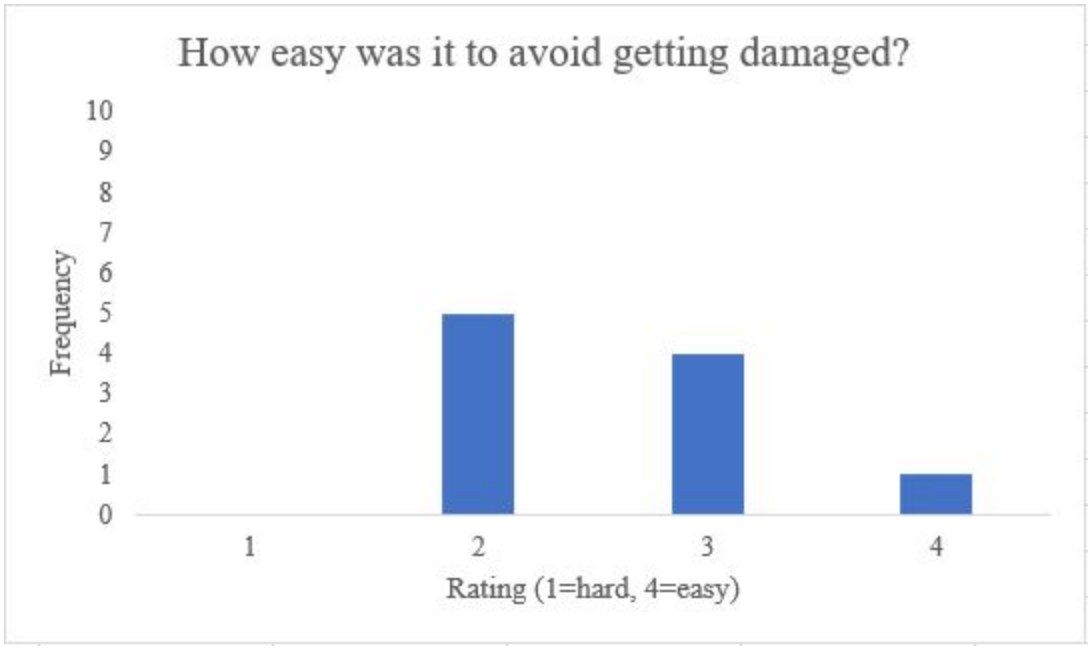
objective was I am not sure if it was ever introduced and I missed it I felt like a DM almost in DND making areas for players to explore. A lot of my issues could be fixed with a better tutorial Also you can probably watch the video back but sometimes I would see my hand float off. I also liked the background music but it did get repetitive after a while I wish i got to test the boss because it would be interesting to see it change when I become the boss or something.

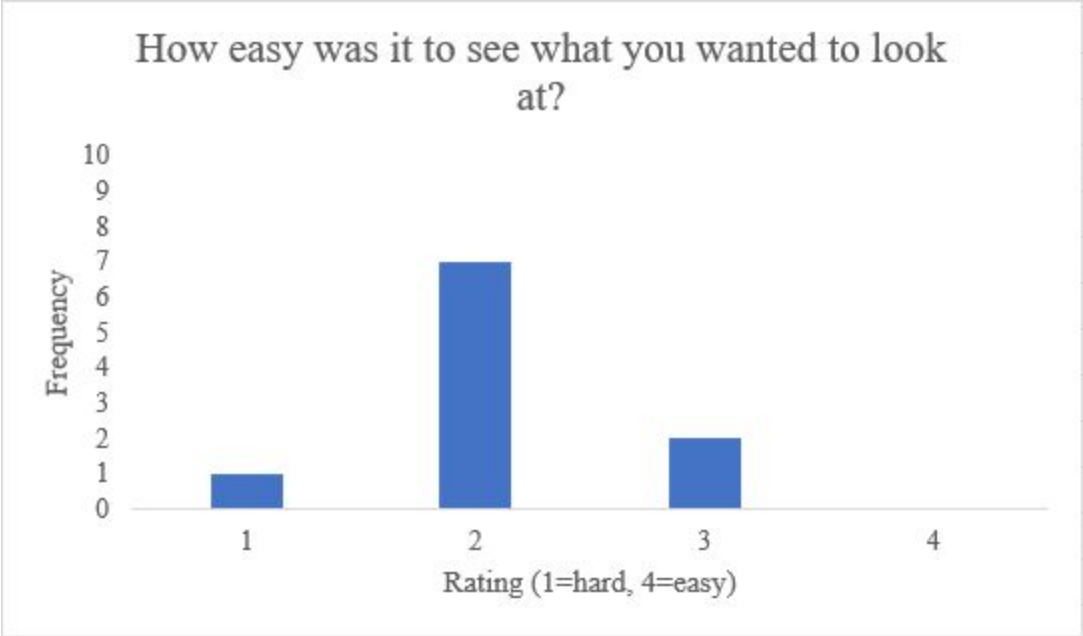
- It's pretty cool but kinda buggy I think.
- Waiting too long for levels to appear and a lot of watching.

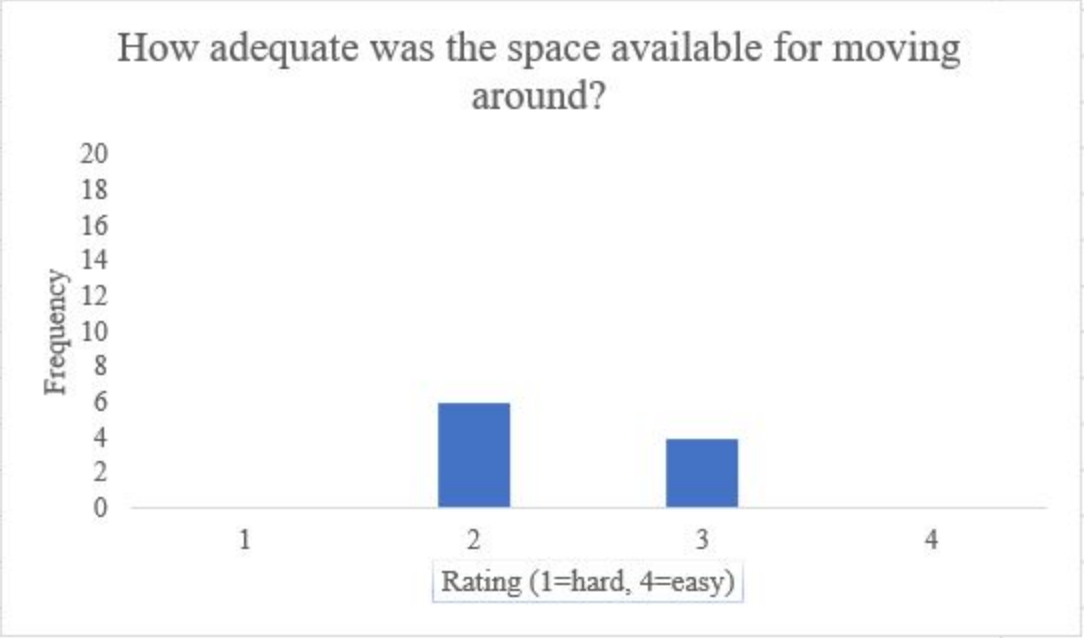
Appendix G. IMGD 3900 Results

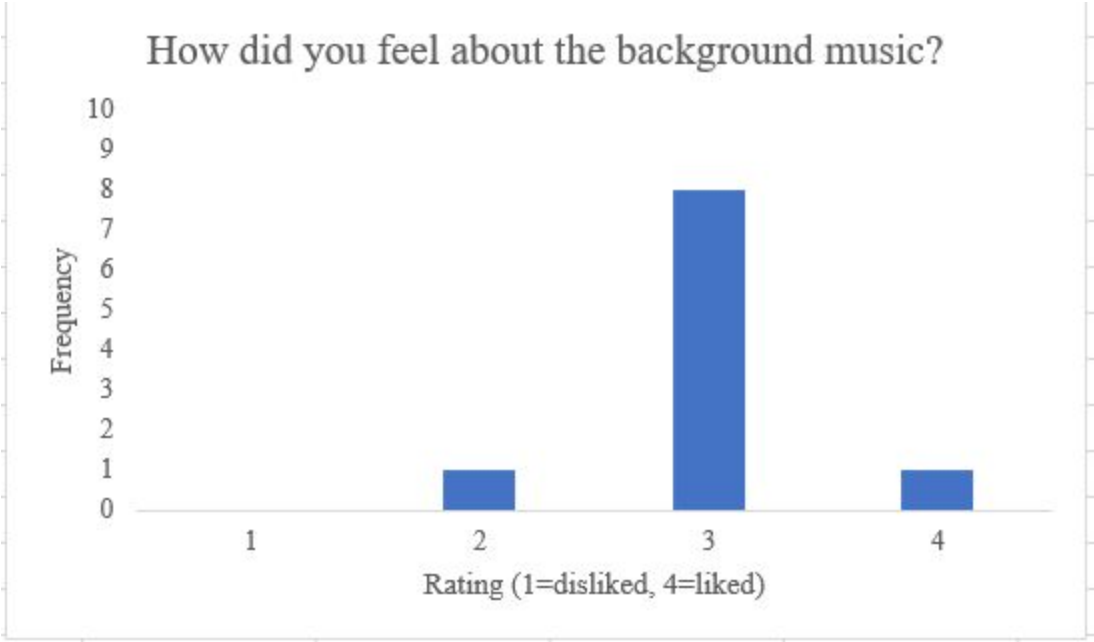
Controller Players

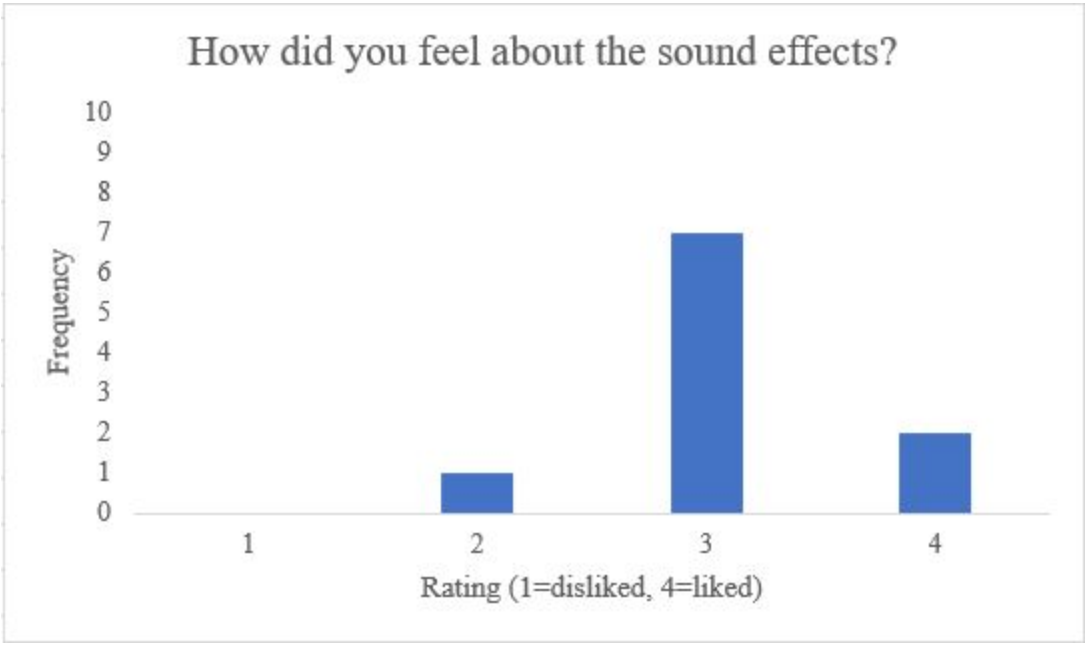


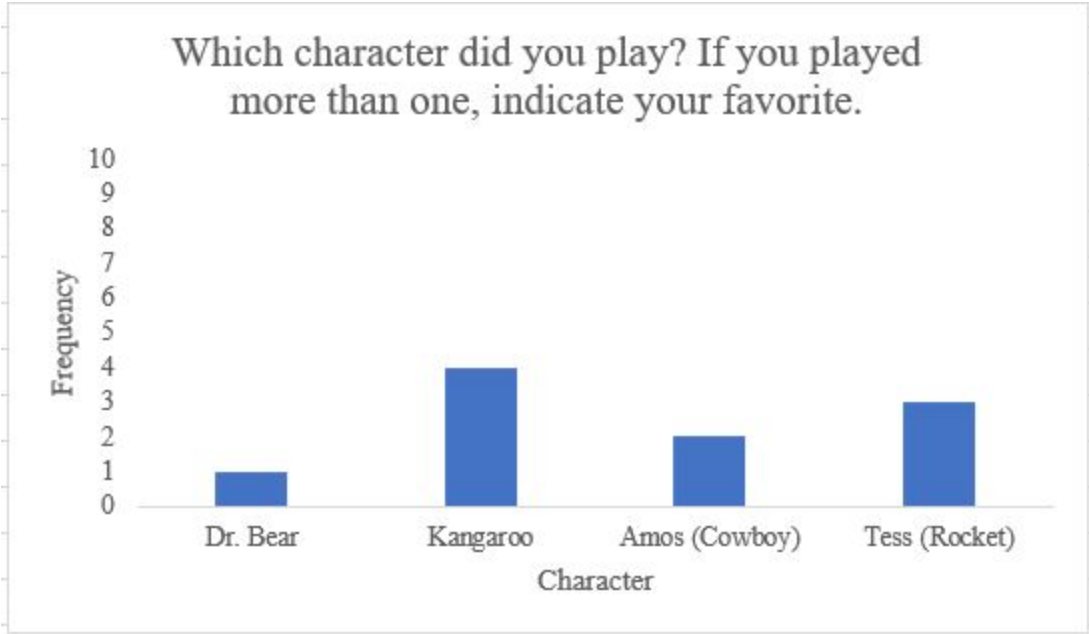


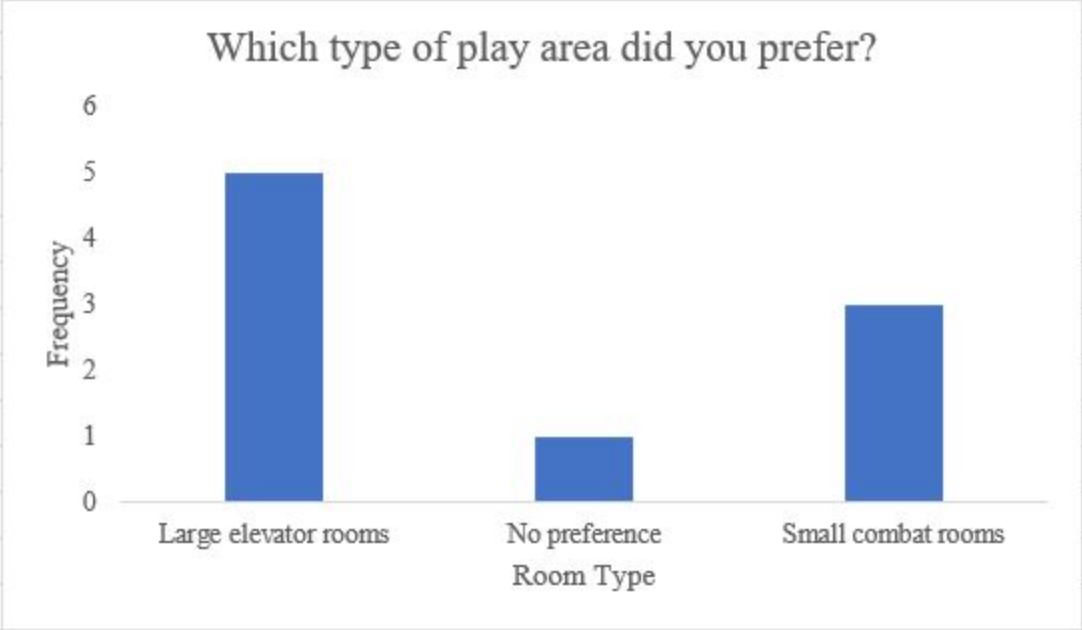

How easy was it to see what you wanted to look at?
Frequency
10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)


How adequate was the space available for moving around?
Frequency
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
Rating (1=hard, 4=easy)


How did you feel about the background music?
Frequency
10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=disliked, 4=liked)


How did you feel about the sound effects?
Frequency
10
9
8
7
6
5
4
3
2
1
0
1
2
3
4
Rating (1=disliked, 4=liked)





Open Response Questions

Is there anything you would like to see added to the game, or that you felt was missing?

- Maybe a simpler more consistent camera system, like always top down (the top down camera parts were the only time i could clearly see what was happening). It also did not feel like a maze (if that was the goal) maybe give the vr player more time at start
- It was really hard to tell what my abilities actually did from the icons that were there.
- A more explicit tutorial, maybe an intro to explain the story/premise

- idle and walk animations
- Platforming felt inconsistent, if that could be tweaked to feel better, it might improve everything.
- It might be more fun if you could always see what the giant vr person was doing

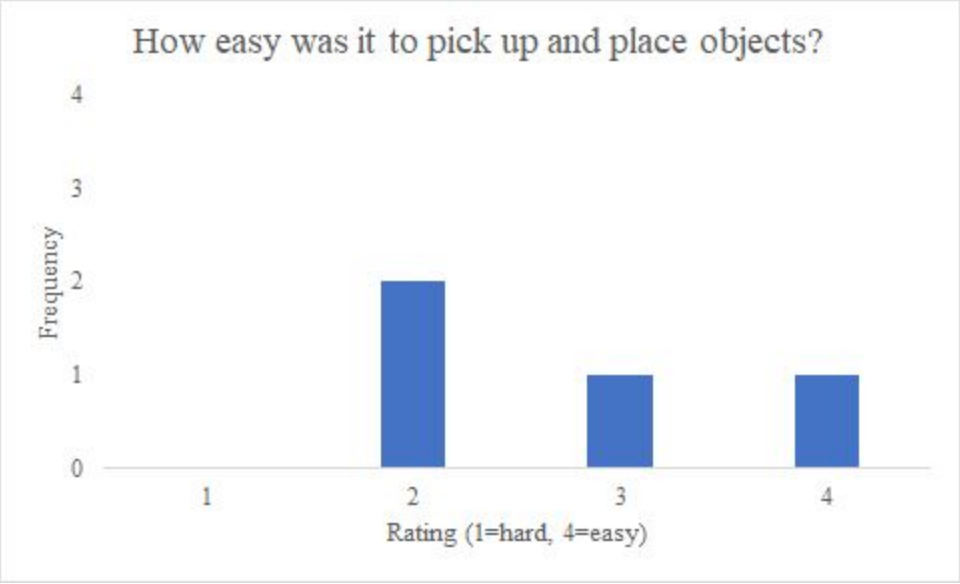
Is there anything you felt should be removed from the game?

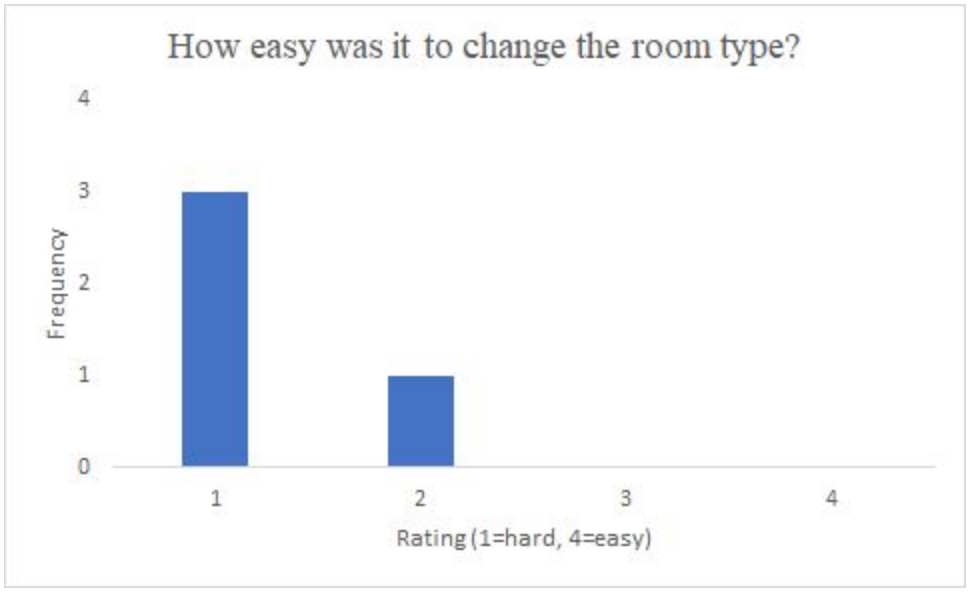
- I'm not sure how well the platforming sections gelled with the game? I had way more fun in the battles.
- Ability to damage your teammates was a bit frustrating

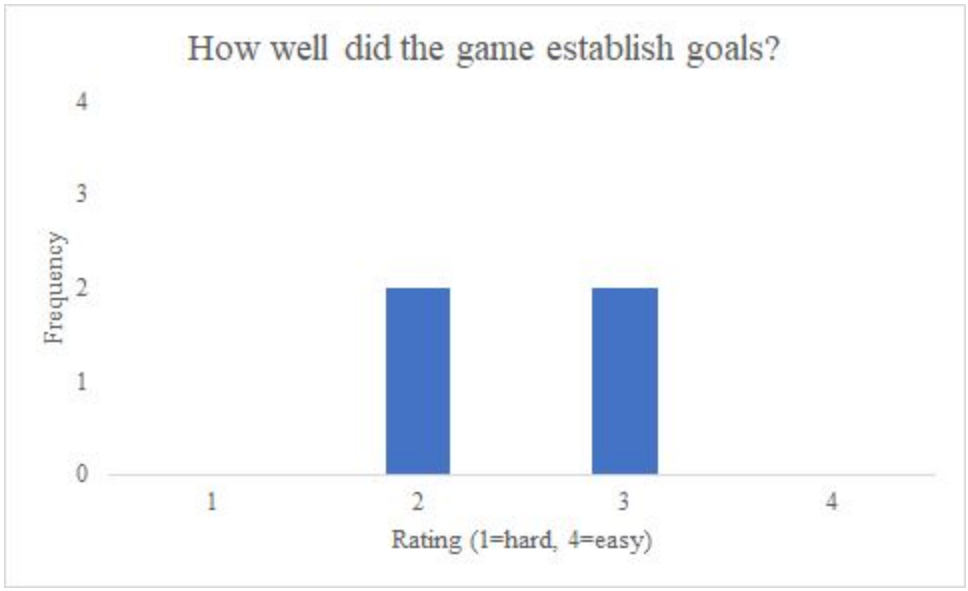
Please enter any general comments or suggestions below:

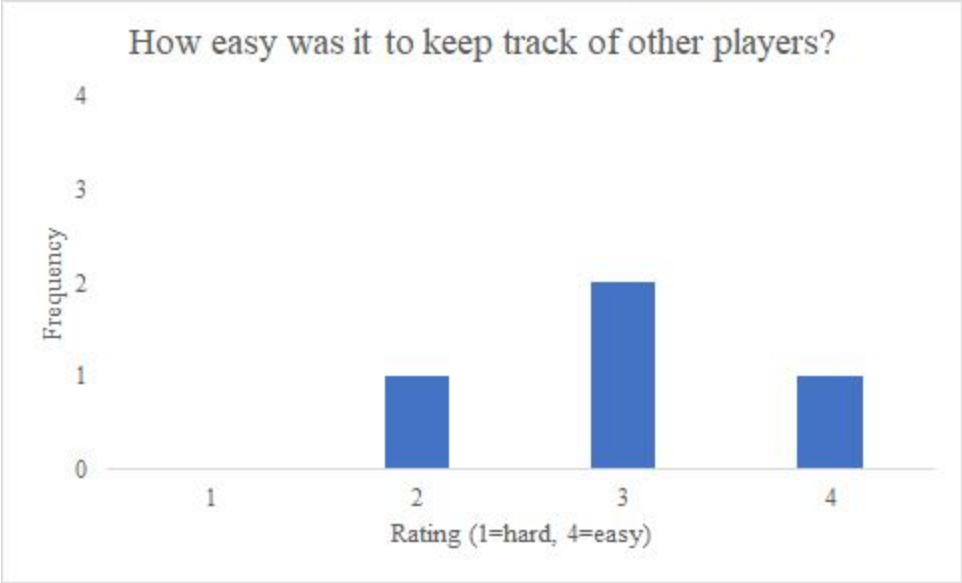
- First of all, I wasn't able to hear the sound. So, disregard my rating for the music and sound effects.
- It can be frustrating to finish a section and simply wait for the VR player to decide what to add, especially at the end of a hallway section waiting on little tiny platforms.
- Revive mechanic was really good! Also, my character just disappeared during the elevator room and broke the game.

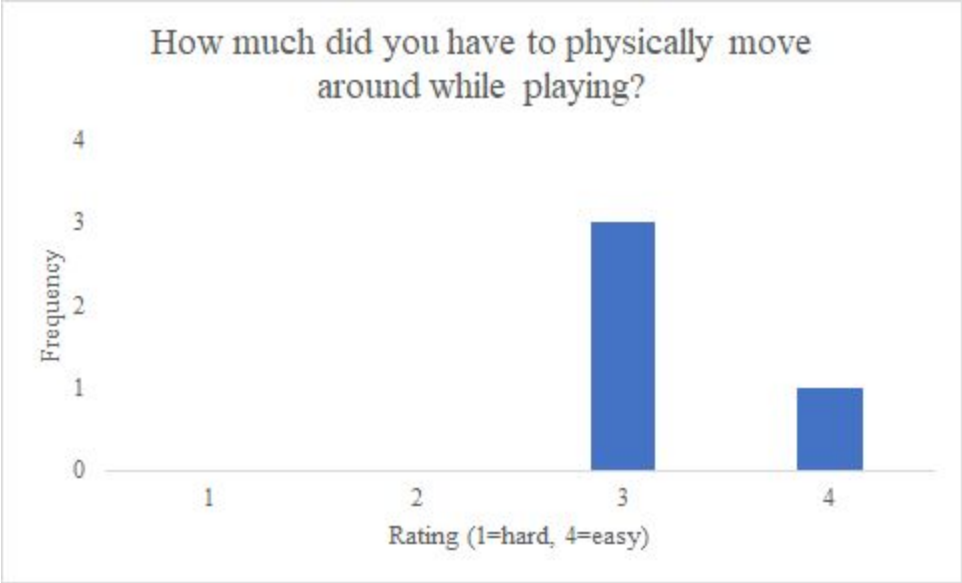
VR Players

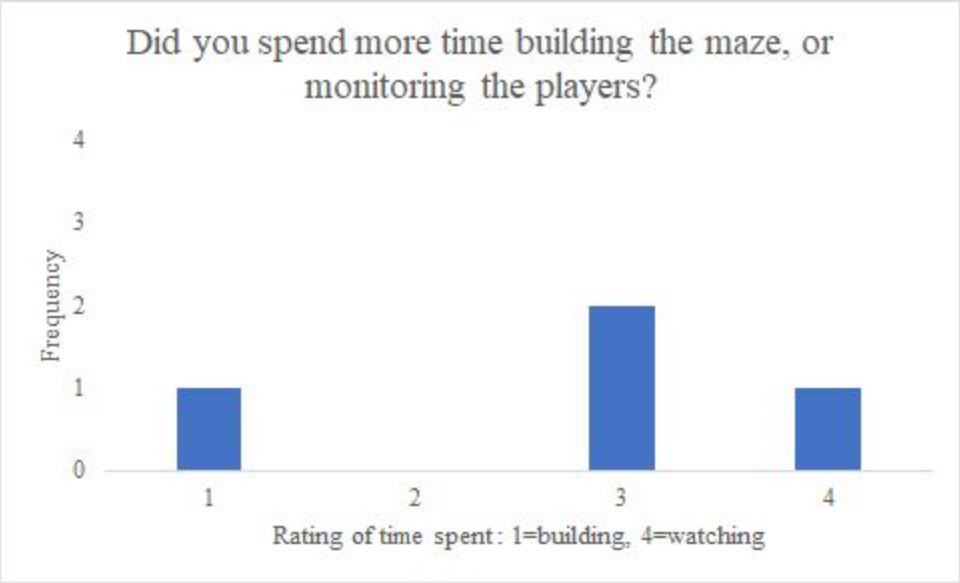


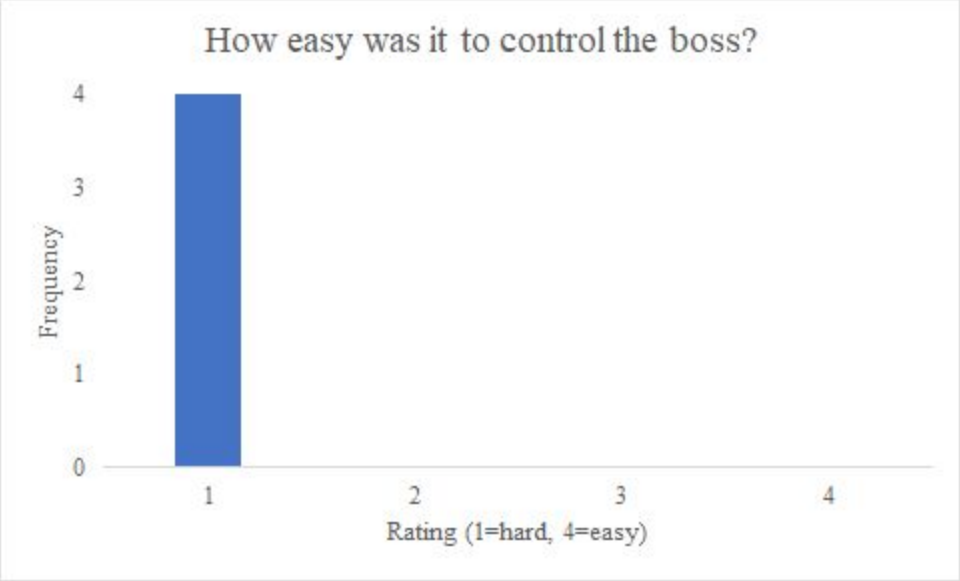

How easy was it to change the room type?
Frequency
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)

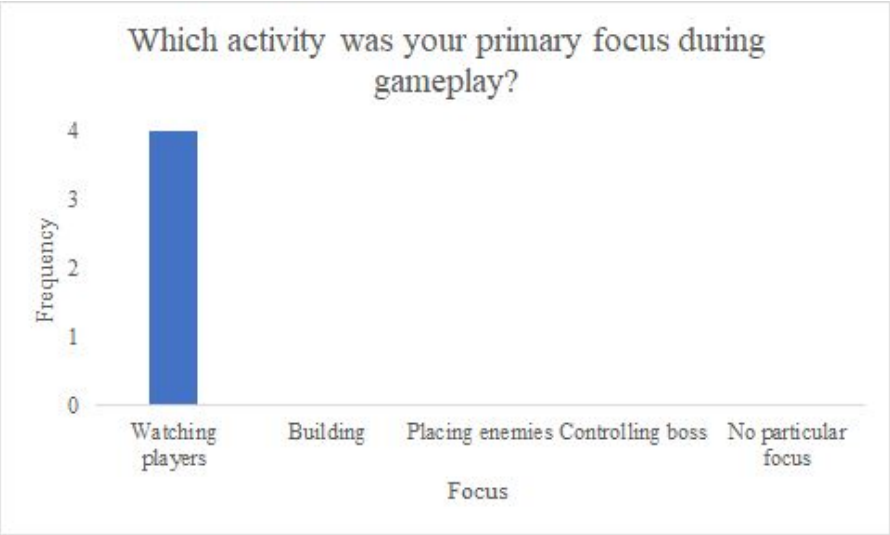

How well did the game establish goals?
Frequency
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)


How easy was it to keep track of other players?
Frequency
0
1
2
3
4
1
2
3
4
Rating (1=hard, 4=easy)


How much did you have to physically move around while playing?
Frequency
0
1
2
3
4
1
2
3
4
Rating (1=hard, 4=easy)


Did you spend more time building the maze, or monitoring the players?
Frequency
4
3
2
1
0
1
2
3
4
Rating of time spent: 1=building, 4=watching


How easy was it to control the boss?
Frequency
4
3
2
1
0
1
2
3
4
Rating (1=hard, 4=easy)



Open Response Questions

Is there anything you would like to see added to the game, or that you felt was missing?

- Polish menus and controls. Balance
- You can spawn monsters?
- More ways of telling the player what to do/ if what you're doing has an impact

Is there anything you felt should be removed from the game?

- A table would pop up from time to time.

Please enter any comments or general suggestions below:

- During combat, I had no idea what my abilities or attacks were. The range of them weren't very well defined.

Appendix H: Third-Party Assets

- Humble Fantasy Game Dev Bundle, by Rexxard
- Cartoon FX Free, by Jean Moreno
- Simple Wall Lamp, by Flatriver
- Skybox, by Clod
- SteamVR Plugin, by Valve
- VR Samples, by Unity Technologies
- Freesound users:
    - Westington
    - Huggy13ear
    - D-Sav
    - MadamVicious