6-12-2017

Virtual Reality Rhythm Game

Joshua Yi

Jordan Lai

SANTA CLARA UNIVERSITY

Department of Computer Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

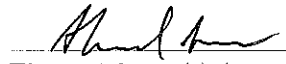
Joshua Yi, Jordan Lai

ENTITLED

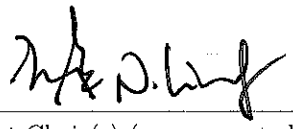
VIRTUAL REALITY RHYTHM GAME

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE AND ENGINEERING



Thesis Advisor(s) (use separate line for each advisor) — 6/12/2017 date



Department Chair(s) (use separate line for each chair) — 6/12/2017 date

VIRTUAL REALITY RHYTHM GAME

By

Joshua Yi, Jordan Lai

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Computer Science and Engineering

of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Computer Science and Engineering

Santa Clara, California

2017

Virtual Reality Rhythm Game

Joshua Yi, Jordan Lai

Department of Computer Engineering
Santa Clara University
2017

ABSTRACT

Virtual reality headsets such as the HTC Vive and Oculus Rift bring robust virtual reality technology in the hands of consumers. However, virtual reality technology is still a very new and unexplored domain with a dearth of compelling software that takes advantage of what virtual reality has to offer. Current rhythm games on the virtual reality platform lack a sense of immersion for the player. These games also require players to remain stationary during gameplay. Our solution is a game where players have to hit musical notes that appear in a trail around them. The trail will move in different directions and players have to move and turn around accordingly in order to hit every note and pass a song.

Keywords: Virtual Reality, Video Game, Rhythm Game

Table Of Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Problem Statement

The idea of virtual reality has been around for quite some time, but has not always been easily available to the public. Devices such as the Virtual Boy by Nintendo have promised a genuine virtual reality experience, but have been met with lackluster reception due to their technological shortcomings. Today, the technological capabilities we have allow us to create more fleshed-out virtual reality experiences than ever before. The introduction of virtual reality headsets such as the Oculus Rift and the HTC Vive have made virtual reality much more accessible for general audiences with their reasonable costs. The HTC Vive, in particular, is one of the first virtual reality systems to utilize the living room as a virtual space. The motion controllers act as the users hands within the virtual space. This being said, virtual reality technology is still a very new and unexplored domain, and only a small fraction of its potential has been utilized so far. As of right now there is a dearth of compelling software that takes advantage of what a virtual reality environment has to offer.

1.2 Current Solutions

Virtual reality has made it possible for avid players to experience their favorite genres in a new dimension. There are two virtual reality rhythm games currently on the market called Audioshield and Holodance. While these two games are both creative and interesting, they only bring out simpler integrations of what virtual reality can do as a whole. Furthermore, these two games fall short in creating an immersive virtual reality environment, resulting in games that could ultimately be recreated for consoles without virtual reality, such as the Xbox Kinect or Nintendo Wii. Since the only motion that is tracked for both games results from the two motion controllers, gameplay can be done with less movement comparable to games with more static environments. These games also seem to have their immersion hampered by the tethered nature of current virtual headset systems. The wire attached to the HTC Vive headset prevents players from turning around without any safety risk.

1.3 Proposed Solution

We propose a rhythm game that takes greater advantage of what virtual reality has to offer. Our project is a game in which players have to score points by hitting notes that appear in the virtual space around them. The notes pop up and the player must hit them at a certain point in order to score points before they disappear. What our project does that other rhythm games do not is that players will be moving and turning their body and legs in different directions instead of staying stationary. The trail of notes will move in different directions around the player and the player must move his or her head and body accordingly within the virtual space in order to hit all the notes. Players will be more physically active due to this. For our solution we also plan to utilize the motion controllers to greater potential. There are several different modes of play that we will explore which will require players to use the controllers in different ways. For one possible example, players will use the controllers as drumsticks to hit the notes. Another possible mode

would require players to use the controllers as pistols where they aim the reticle to hit the notes. A third possible option might have the controllers act as swords, and players must slash notes at the right time. Overall, we believe that the gameplay features in this game will ultimately provide players with a more enjoyable and physically active rhythm game experience. In addition to this, our project looks forward to future implementations of the HTC Vive involving wireless technology, allowing players to experience a wider degree of motion.

Chapter 2

Requirements

Before designing the virtual reality rhythm game, lists of requirements and design constraints were compiled. The requirements were split into functional and non-functional requirements. The requirements adhere to different priorities which are Critical, Recommended, and Suggested.

2.1 Functional

Critical

- The system will support motion tracking using the motion controllers and headset.
- The system will display notes that a player must interact with to play the game.

Recommended

- The system will display scores and ratings based on each note that is successfully "hit."

Suggested

- The system will provide different game modes that allow the player to experience different uses of the motion controllers.

2.2 Non-functional

Critical

- The system will provide an interactive virtual reality experience.
- The system will respond in a manner that feels natural in terms of tracking movement.
- The system will be playable in a manner that does not directly cause harm or illness to the player.

Recommended

- The system will display a menu that is easy to navigate through and is not confusing.
- The system will be visually appealing for players.

Suggested

- The system will have an easy and intuitive way of sorting through lists of songs.

2.3 Design Constraints

- The system will run on the HTC Vive.
- The system will be a video game.

Chapter 3

Design Concept

3.1 Use Cases

A use case lists the necessary steps to accomplish a specific goal. Each of the following use cases describe how the player interacts with the system to accomplish these goals which include preconditions, post conditions, and exceptions. To address the functional requirements, we have identified five use cases. Figure 3.1 illustrates the use cases defined in our system.

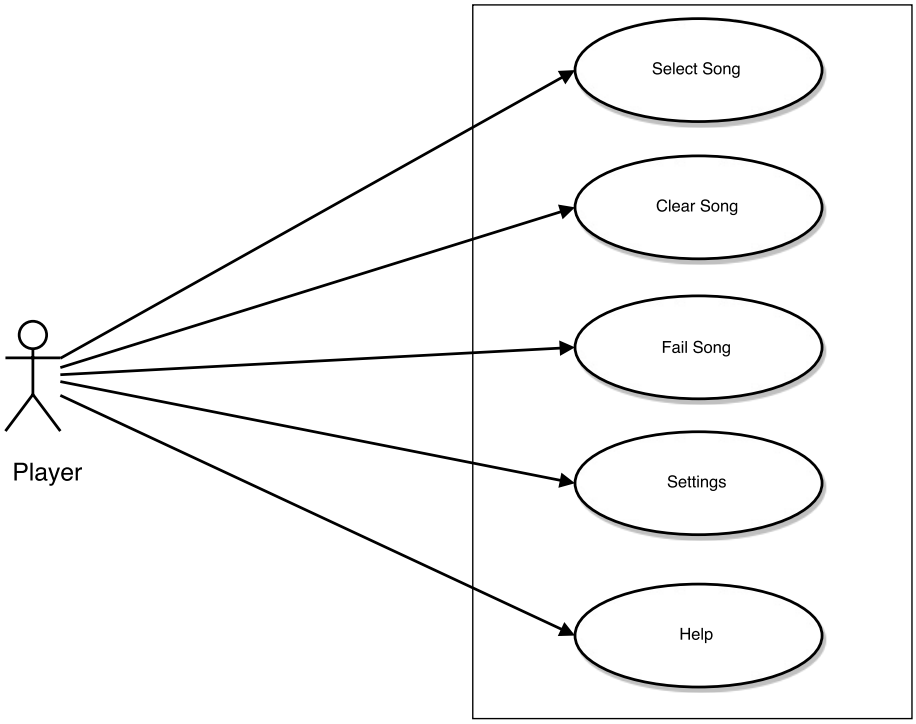


Figure 3.1: Use Case Diagram for the system.

1. **Select a song**

   Actor: Player

   Goal: Choose a song to play

Preconditions:

- Player needs to start the game and have selected the "play" option.

Steps:

(a) The player chooses a song to play.

Postconditions:

- Player will be taken to the "adjust song settings" window.

Exceptions: None

2. **Clear Song**

Actor: Player

Goal: Player completes a level and gets a score back.

Preconditions:

- The player must have chosen a song to play and chosen the desired settings for the song.
- The player must have reached the end of the song without their life bar reaching zero percent.

Steps:

(a) The player hits or misses notes on the screen.
(b) The player avoids hitting zero on the life meter.
(c) Song has completed.

Postconditions:

- The player will be given a score based on how accurately they hit the notes throughout the song.
- The player will be given an option to either retry the song or return the song selection menu.

Exceptions: The player has hit zero on the life meter and taken to the "Fail song" window.

3. **Fail Song**

Actor: Player

Goal: To penalize players who fail to maintain a life bar over zero percent.

Preconditions:

- The player's life bar has reached zero at anytime between the start and end of the chosen song.

Steps:

(a) The player hits or misses notes on the screen.
(b) The player misses enough notes for the life meter to hit zero.

Postconditions:

- The player is immediately taken to the "Fail song" window.
- The player will have the option to either retry the current song or return to the song select menu.

Exceptions: None.

4. **Settings**

   Actor: Player

   Goal: Allows for players to adjust certain settings in the game.

   Preconditions:

   - The player must have selected the "Settings" button from the main menu or the song select menu.

   Steps:

   (a) The player chooses the "Settings" option from the main menu.

   Postconditions:

   - The settings menu will pop up, allowing for the player to customize his profile, delete save data, or change volume.

   Exceptions: None.

5. **Help**

   Actor: Player

   Goal: Help familiarize players with how the game works, as well navigating through the menu.

   Preconditions:

   - The player must have selected the "Help" button from the main menu or the song select menu.

   Steps:

   (a) The player chooses the "Help" option from the main menu.

   Postconditions:

   - The player will be given an explanation of how to select a song, adjust the settings, and play through the song.
   - The player will be given an additional option to view a tutorial on how to play.

   Exceptions: None.

3.2 Conceptual Model

3.2.1 Main Menu and Song Select

When the player starts up the game, they will see the main menu screen on their headset, shown in Figure 3.2, where the player can start the game, go the settings menu, or go the help menu. The options are displayed in blocks where the player can scroll left and right using the motion controllers to select an option. The top right corner displays a profile which contains the player's name, an avatar picture, and a level that indicates how long the player has been playing. The black strip at the bottom displays a description of the option currently selected.

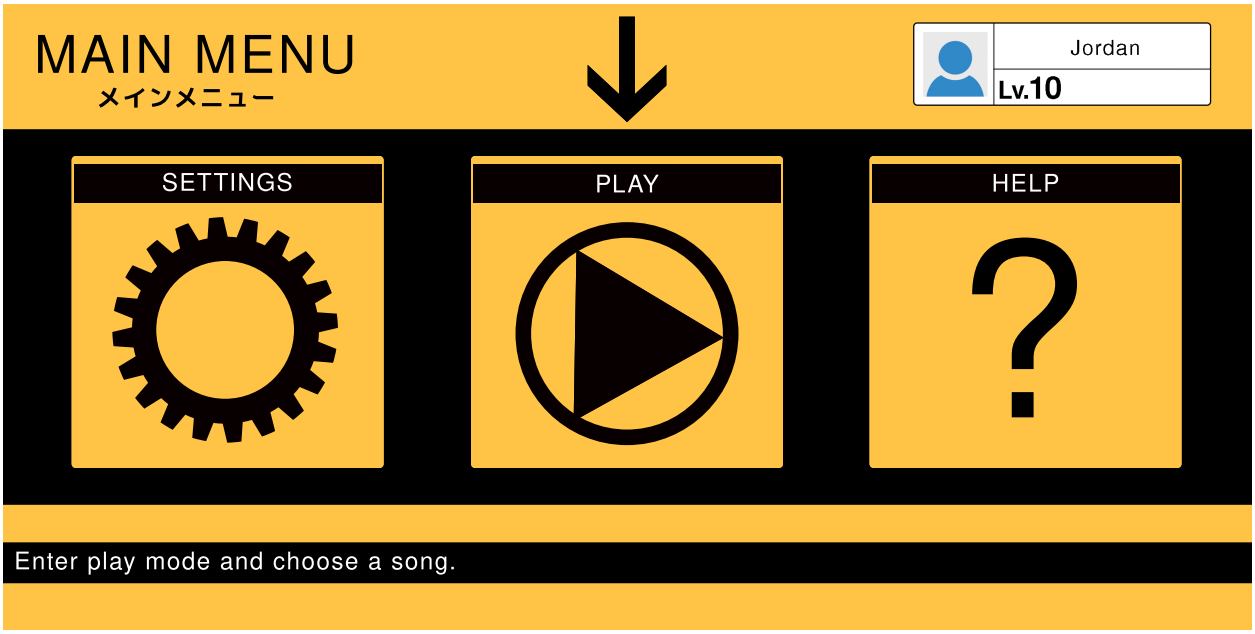


Figure 3.2: Main Menu screen for the player.

If a player decides to select the play option, he or she is brought to a music selection screen shown in Figure 3.3 where a song is chosen for play. Each song selection is shown as a block which includes the song title, song artist, music album, and the score for the last attempt. When the current song is hovered over, it will display its difficulty levels at the bottom from easiest to hardest. The player can select the difficulty for the song he or she wants to play. There is a back button in the bottom right corner that a player can press if he or she wants to go back to the main menu.

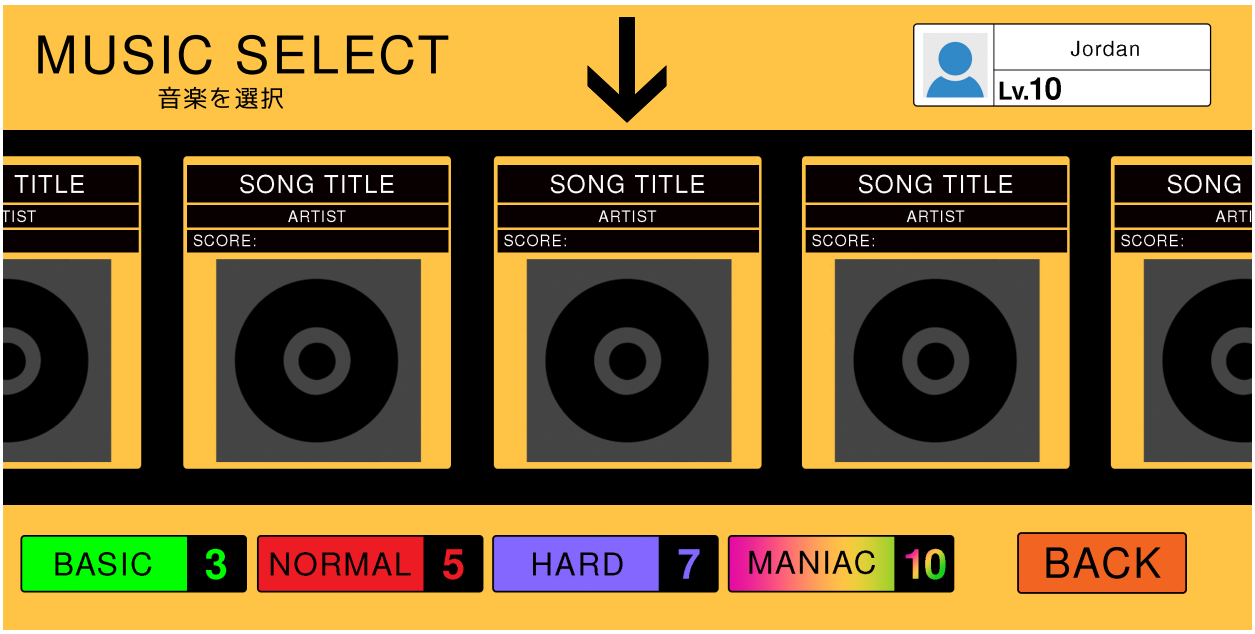


Figure 3.3: Music Selection screen for the player.

3.2.2 Gameplay

When a player chooses a song and difficulty level, the game will start as shown in Figure 3.4. Notes will pop up on the headset and a ring will outline each note. The ring will continue to get smaller and the player has to hit the note when the ring becomes the same size as the note. Based on the player's timing when he or she hits a note, a word will pop up based on their timing or accuracy. The messages that appear over the note hit are "critical", "great", "hit", and "miss" depending on the player's timing.

Figure 3.4: Conceptual gameplay model with HUD.

The heads-ups-display (HUD) displays the player's life meter and score. When a player successfully hits a note, the life meter will increase and the score will update. There will also be a combo count that shows the number of notes hit successfully in a row. When a player misses a note, the life meter will decrease, the score will not update, and the combo count will reset. If a player's life meter hits zero, the song will end prematurely and a song fail page will appear. If a player is able to finish the song without failing, a song clear page will appear.

Chapter 4

Design Architecture

4.1 Activity Diagram

Figure 4.1 is an activity diagram of our system in Unified-Modeling-Language (UML). This diagram reflects how users will interact with our game. It represents the flow from one activity to another, using rounded rectangles to represent actions, diamonds to represent decisions, black bars to represent splits and joins, a black circle to represent the start, and a black circle with a red circle around it to represent the end.

The flow of the player begins with the player at the main menu screen. The player then goes to the song select screen where they can choose the song. After the song is chosen, the gameplay begins with the first note popping up on the headset. Every note hit will increase the score and life meter. Every note missed will deduct the life meter. If the life meter is at zero then the game will stop with the player being shown a fail screen. Otherwise, the cycle will continue until the last note where the player will be shown the song clear page shortly afterwards.

4.2 Architectural Design

Figure 4.2 is a high level overview of the technologies that make up our game. The main architectural design of our game system can be broken down into two distinct parts. First is the mapping of notes to the environment around the player. All notes in a given song are mapped based on a time and location designated by a csv file. The game engine parses the corresponding csv file when a song is chosen into two separate arrays. The engine will then spawn notes at certain locations at the appropriate times designated by the arrays.

The second part of our system involves the player's interaction with the notes around them. Whenever a player interacts with a note, the virtual reality system will send back information for the game engine to determine the timing accuracy of the note hit. Through these reports back to our system, the player gets a score based on how accurately a note is hit.

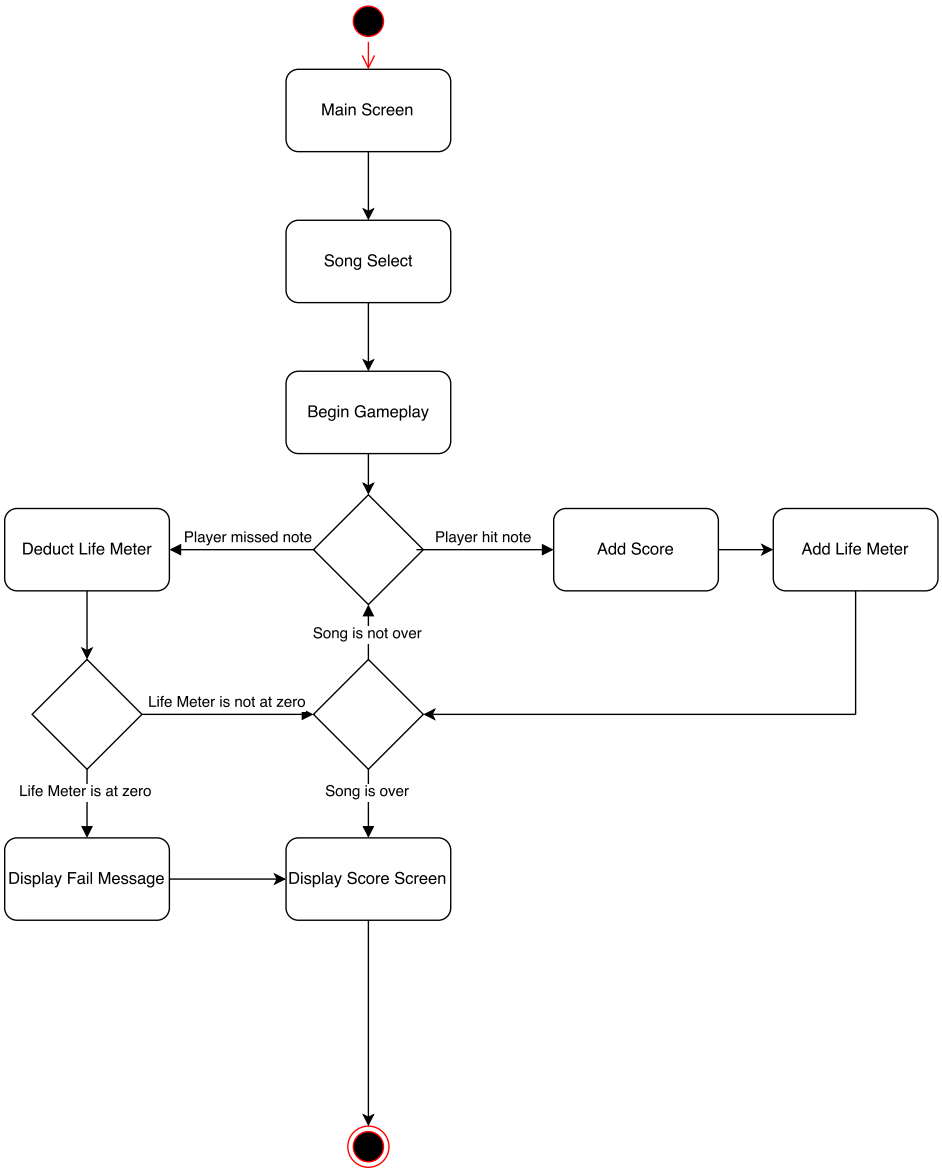


Figure 4.1: Activity Diagram showing the dynamics of the system for the player.

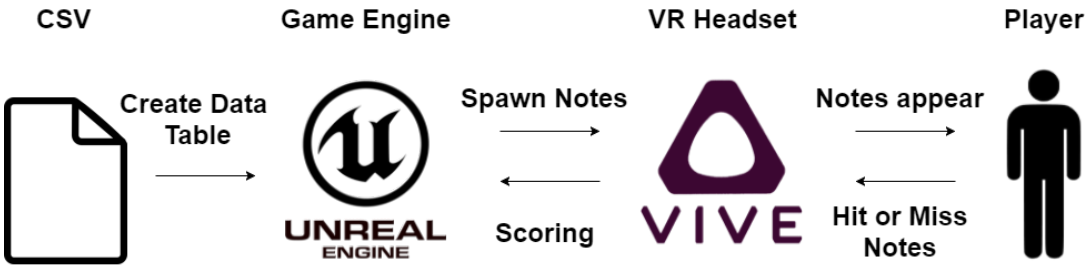


Figure 4.2: Diagram describing the technology stack for our game.

4.2.1 Loading the Game

The game can be broken down into two separate levels or spaces which are the gameplay level and menu level. The menu level shown in Figure 4.3 is where the player can interact with the menu widget in the virtual space. When a player changes settings or selects a song, the menu level will communicate to the gameplay level shown in Figure 4.4 to load the appropriate csv file, music event, and setting variables when the player hits the play button.

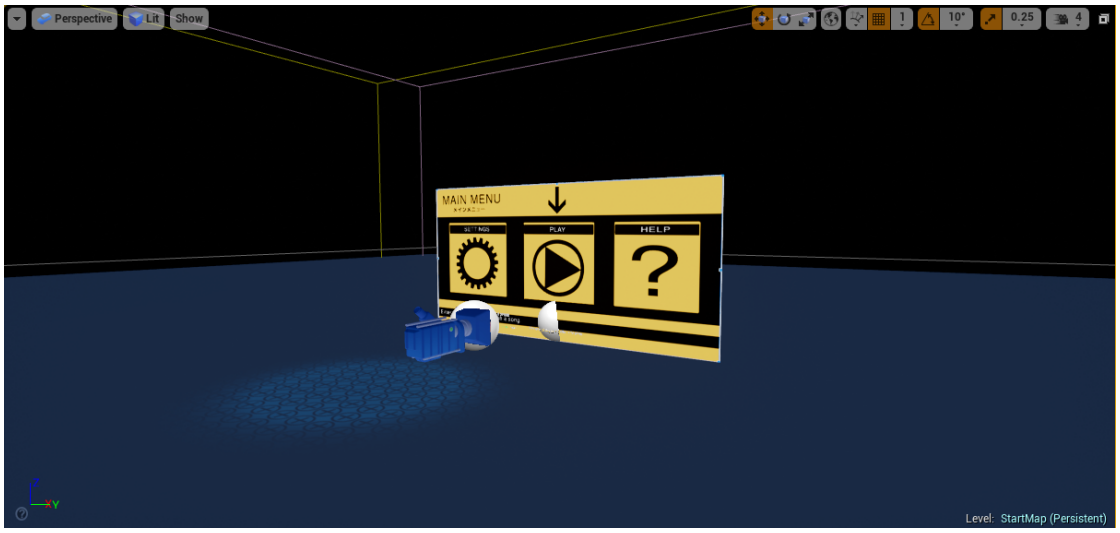


Figure 4.3: Screenshot of the menu level in the editor.

Figure 4.4: Screenshot of the gameplay level in the editor.

While the gameplay level is loading, the game engine will create spawn points and load the appropriate timing file. Figure 4.5 is close up view of the gameplay level where a player will be interacting in. There are 64 red arrows arranged in four rings around the player. The red arrows are a visual representation of each coordinate a note can spawn on. The game engine gets the relative transform of each arrow and inserts each transform into an array.

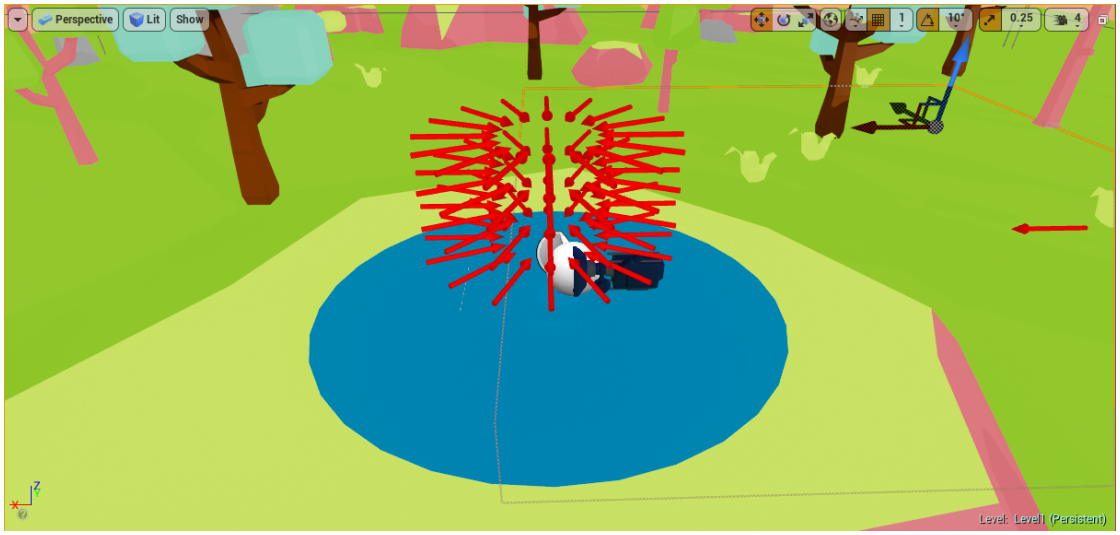


Figure 4.5: Screenshot of the area in the gameplay level where a player will be interacting in.

After creating the transformation array, the engine will load a timing file. A timing file is a csv file containing two columns of data which are time and location. The time column is an ordered list of integer values in milliseconds of when the engine should call the function to spawn a note. The location column is a list of numbers that tell the engine which arrow to spawn a note at.

4.2.2 Timing

Once the gameplay level has finished loading, the in-game timer will start and check the current index of the time array on every frame. When checking the time array, the comparison function will see if the timer is three seconds before the value in the current index of the time array. This will give enough time for the player to see the note and interact with it. Figure 4.6 is what a note looks like in the editor. The note is a bright, blue sphere with a ring around it. The ring shrinks in size to indicate when a player should hit the note. If a player does not hit a note after 3.5 seconds, the note will self-destruct.

One of the biggest hurdles for any rhythm game is making sure that the actions occurring in the game match with the music that the player is listening to. The challenge behind this lies within a games frame rate. While a game would ideally report playhead position changes gradually in consistent steps, more often than not these position changes are reported back in varying sized chunks. This sudden change in position can cause the gameplay to de-sync with the music, rendering a rhythm game unplayable. To remedy this, we implemented our own function to accommodate sudden changes in framerate, while also making sure the actions in game matched with the music. In order to achieve this, our function compares the last reported playhead position from the game engine in the form of a time stamp to the current value of the in-game timer. If the two happen to be different, the function takes the average of the two to smooth out any sudden framerate changes until it smooths out.

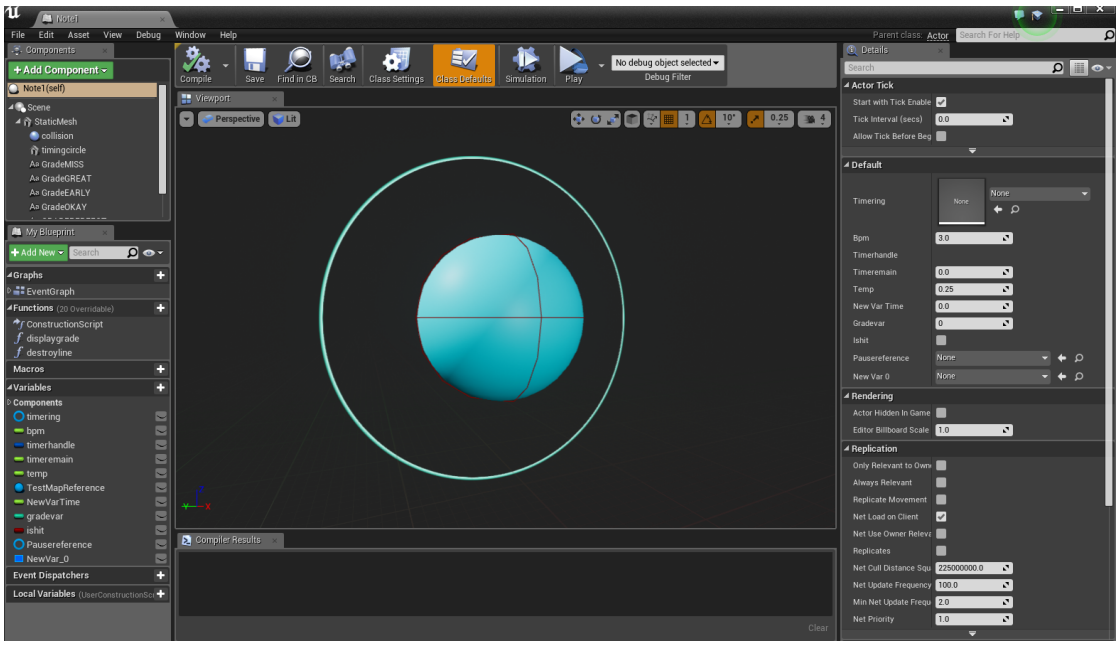


Figure 4.6: Screenshot of a note in the editor.

4.2.3 Game Logic

During gameplay, the game has two functions which handle the game logic. The event listener listens for player input and other in-game actions such as pausing. When a player hits a note, the event listener will grab the note ID/index and the exact time when it was hit in milliseconds. If a player misses a note, the event listener will grab the note ID/index along with broadcasting to the event handler a boolean that the note was not successfully hit.

The event handler takes the note ID/index and time from a event listener broadcast and matches the ID with the timing file. The event handler also grabs the time corresponding with the right ID and compares it with the time from the event listener. Based on the difference between both times, the event handler will assign a temporary string with a grade value ranging from perfect to fail. Using a switch case statement, the event handler will call functions that modify the score, combo count, and life meter based on the grade.

Chapter 5

Design Rationale

5.1 Technologies Used

The system uses a variety of game development technologies to meet all the requirements. The system is implemented using Unreal Engine 4 and C++. The system is designed to run on the HTC Vive.

- HTC Vive is a virtual reality headset developed by HTC and Valve Corporation. The headset is designed to utilize "room scale" technology to turn a room into a 3D virtual space via sensors. Users are able to navigate naturally in the virtual world and use motion controllers to manipulate objects and experience immersive environments.
- Unreal Engine 4 is a game engine developed by Epic Games. Although primarily developed for first-person shooters, it has been used in a variety of other genres such as role-playing games or first person shooters. Unreal Engine features a high-degree of portability and is a tool used by many game developers today.
- C++ is a general-purpose programming language. It has imperative, object-orientated, and generic programming features. C++ is used as the scripting language for Unreal Engine.
- FMOD is a video game sound effects engine developed by Firelight Technologies. FMOD supports playback of a wide variety of audio formats. FMOD is available for Unreal Engine 4 as a plugin.

5.2 Justification for UI

Our user interface is made to be simple and easy for players to navigate around. The main menu consists of three options labeled clearly so that players will have an easy time learning, configuring, and playing the game. The player can choose any of these options using the motion controllers. The following options are given to the player in the main menu.

- Settings: The player can configure appearance and gameplay settings from this option.
- Play: The player will be taken to the song selection menu and can choose a song to play.
- Help: Players that are either unfamiliar with the game or want to learn more about how the menu and gameplay works can choose this option. From here, the player can learn how to navigate through the different menus, as well as getting a tutorial for the gameplay.

The song selection menu is laid out so that players can view songs available to play by scrolling left or right using the motion controllers. After choosing a song, the appropriate information regarding the song difficulty is shown to the player. From this point, the player can also adjust settings specific to the song as they desire before transitioning to actual gameplay.

5.3 Design Rationale for Technologies Used

The following outlines the justification for using these technologies.

- We used the HTC Vive as our main virtual reality platform of choice because of its "roomscale" technology. The ability to move around and interact in a virtual space imposed on a physical play area is required for our game. The HTC Vive is the only headset to offer "roomscale" technology as of November 2016. The Oculus Rift is bound to get its own "roomscale" technology additions soon but not close enough for us to justify considering it.
- We chose to use Unreal Engine 4 for its development flexibility and features. Unreal Engine is also available to everyone for free and has the complete C++ source code available online. Unreal Engine is also one of the few game engines that is compatible with virtual reality headsets.
- We chose to use C++ because it is the scripting language used for Unreal Engine 4.
- We chose to use FMOD because it allows us to retrieve the playhead position of a song and it loads metadata banks at the beginning of the game instance. The base Unreal Engine 4 implementation does not have these capabilities. As mentioned earlier, the ability to acquire the current playhead position served a vital role within our timing function.

Chapter 6

Test Plan

The testing phase is integral to the software development process because its main purpose is to find bugs and problems within the software. Our test plan was multi-step process that consisted of various phases designed to deliver a final product with little to none imperfections.

6.1 Interface Testing

This phase consisted of Graphic User Interface (GUI) testing. The goal was to make sure the GUI was presented correctly in the headset and that the interaction between the menus flowed smoothly. We also made sure that the menu navigation was working correctly with motion gestures and feedback. For this section, we asked volunteers for feedback on the layout of our interface in order to provide an intuitive and easy to navigate interface for users.

6.2 Verification Testing

The next phase was a complete back-end test with verification steps to make sure the game logic works correctly. These steps are:

- Verify that players transition from each different screen accordingly.
- Verify that the notes display on the headset when the player begins a song.
- Verify that the notes do not de-sync with the audio during gameplay.
- Verify that players can interact with the notes accordingly.
- Verify that the scoring system for our game is consistent and reasonable.

6.3 Integration Testing

This phase was a complete application test of every component combined. The goal was to find more possible bugs throughout the application and to test the application for performance issues. Latency and framerate issues needed to be carefully tested as these problems can negatively affect the player's experience.

6.4 Motion Testing

In this phase, we asked volunteers to test our game and give feedback based on the gameplay. Each volunteer tested each song and said if they felt motion sickness during playing. We made notes of what portions of the song caused potential sickness and what songs were too difficult or easy for the player.

Chapter 7

Risk Analysis

We created a risk analysis table in order to evaluate risks that could happen during development and their consequences. This table provides risks, their consequences, their impact and severity, and ways to mitigate each risk.

Risk	Consequences	Probability	Severity	Impact	Mitigation
Sickness or Emergency	Deadlines getting delayed or certain parts of the system not finished on time	0.5	7	3.5	Get good sleep and schedule extra meetings to work
Time	System may not get entirely finished	0.4	8	3.2	Set early deadlines and prioritize features
Bugs	System not working as intended	1	2	2	Make test plan early and begin testing in early stages
Data Loss	Losing work and falling behind	0.02	9	0.18	Keep multiple backups of project and save often
New Technologies	Causes delays	0.3	4	1.2	Focus on necessary aspects of technology and follow tutorials
Breaking Equipment	Causes delays and requires money to replace or fix	0.05	8	0.4	Carry equipment in bag and have a companion to watch the player

Chapter 8

Project Management

8.1 Development Timeline

For our development timeline, we organized the timeline by the following sections: Conception Phase, Elaboration Phase Part One, Elaboration Phase Part Two, Operational System, Delivery Phase, and Testing. The timeline is also divided into three sections for each school quarter. For each task, members involved in completing it are labeled by cell color. The legend has the color corresponding to each member of the team.

Joshua Yi	
Jordan Lai	
All Members	

Fall Quarter	1	2	3	4	5	6	7	8	9	10
Conception Phase										
Problem Statement										
Requirements Gathering										
Elaboration Phase, Part I										
Budget										
Design Document										
Begin Project Development										
Operational System										
Object Interaction										
Music Engine										
Menu Interface										

Figure 8.1: Development timeline for the project in fall quarter.

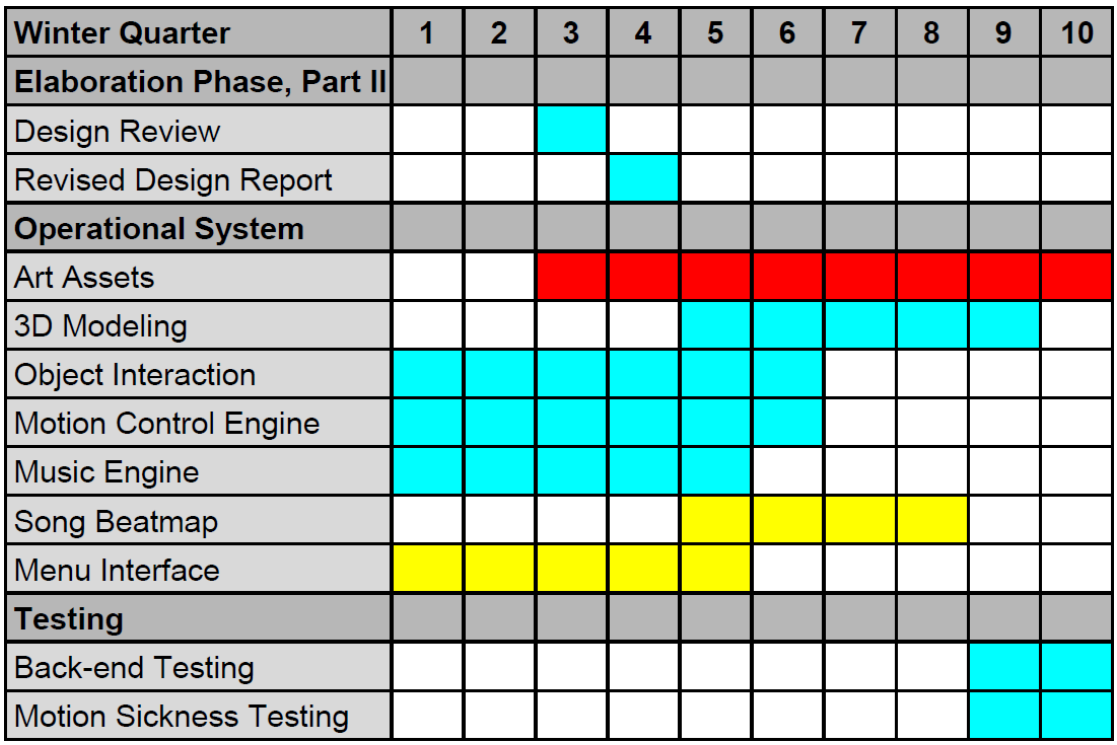


Figure 8.2: Development timeline for the project in winter quarter.

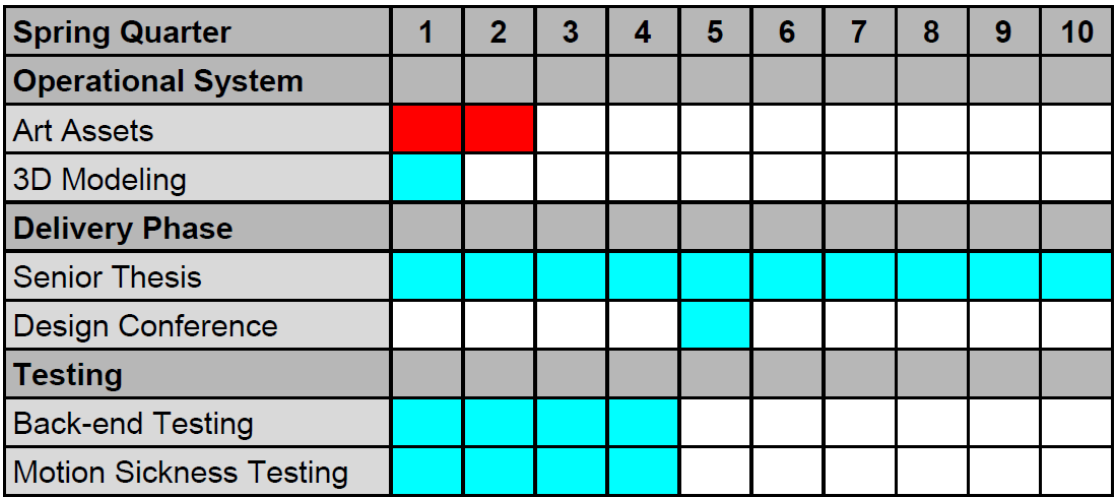


Figure 8.3: Development timeline for the project in spring quarter.

8.2 Project Testing

During actual development of the game, we use a number of testing methodologies to make sure we did not run into any major problems along the way. We did unit testing on the modules we were working on which were the menu level and gameplay level. We did a black box test on the menu level to make sure the menu was functioning and that the motion controllers were interacting with the menu properly. On the gameplay level, we did black box testing to make sure notes were properly spawning along with motion controller interaction.

In terms of white box testing, we were able to preview how our game worked at any given time thanks to Unreal Engine 4s built in VR preview function. This was crucial in making sure that in game objects such as menus, notes, and the HUD were all functioning and appearing as planned without having to compile the game every time we wanted to test it. In addition to this, the simulate function shown in figure 8.4 allowed for us to monitor game variables to see if they were changing as expected.

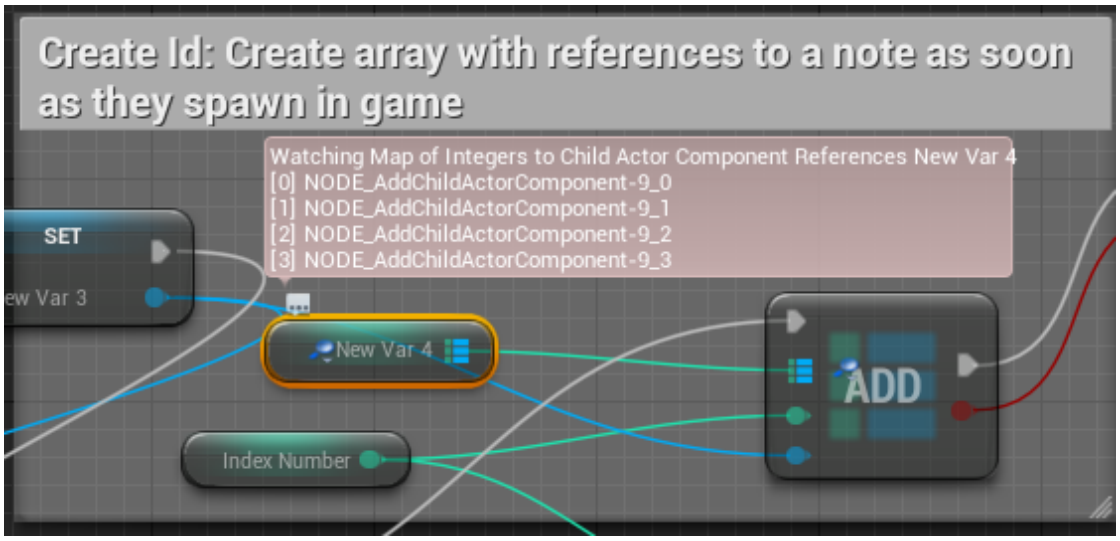


Figure 8.4: Screenshot of using Unreal Engine's simulation capabilities for testing.

8.3 Societal Issues

Whether it is a game or an application, taking social aspects into consideration is always important. As engineers, we have to act in an ethical manner when designing and developing a game.

8.3.1 Ethical Issues

The main ethical issue with our project is copyrighted material regarding music. Because we are not interested in obtaining a license for copyrighted music, our alternative options is to either use creative commons licensed music or to just produce our own music.

8.3.2 Social Issues

With our game, we hope to bring interest for virtual reality to not only gamers but also the general public. Giving an exhilarating virtual reality experience will help more people adopt the virtual reality platform. Kindling this interest can potentially bring more quality experiences along with potential innovations.

8.3.3 Political Issues

Our game is not political in nature, so this topic does not apply here.

8.3.4 Economic Issues

When the game is ready for release, the main issue we have to face is dealing with royalties regarding Unreal Engine 4 and potential digital distribution platforms.

8.3.5 Health and Safety

Ensuring the health and safety of players is our top concern regarding this game. When designing the timing files, we had to implement our own design rules to prevent harm and injury. We wanted to make sure that players can see the next note within their field of view. We also made sure that the distance between notes did not cause players to make sudden or erratic movements. Because of the tethered headset, we prevented note patterns that curved in a single direction which would cause the wire to disorient the player.

8.3.6 Manufacturing Issues

Our game is not a physical product, so this topic does not apply here.

8.3.7 Environmental Issues

Our software does not cause environmental issues, so this topic does not apply here.

8.3.8 Usability

Making the menu easy to use and intuitive was very important to us. We made sure that menu navigation was seamless. In terms of gameplay, we made sure that the heads up display was not causing obstruction with the player's vision. We also made sure that the heads up display had enough information for the player to know what is going on during the gameplay such as score and combo count. We added a help menu so that players could learn the features of our game easily.

8.3.9 Lifelong Learning

Lifelong learning was crucial in order for us to bring the game to a playable state. The knowledge we gained from our classes alone was not enough for us. We had to take an extensive look through the official documentation and tutorials in order to get a solid grasp on the game engine functionality.

8.3.10 Compassion

Our game hopes to bring all kinds of gamers together with an innovative and enjoyable experience.

Chapter 9

Our Project

9.1 The Final Application

Catchbeat is a virtual reality rhythm game developed for the HTC Vive using Unreal Engine 4. When players open up the game, they will spawn in the main menu level where they can use their motion controllers as a pointer to navigate through the main menu.

9.1.1 Main Menu

When a player begins the game, they will first be taken to the menu map, which contains our main menu as widget. As shown in Figure 9.1, the player can choose to see the song select menu, the settings menu, or the help menu. Using the motion controller with the black pointer indicator, the player can hover over any of the menu icons to choose where to navigate to next.

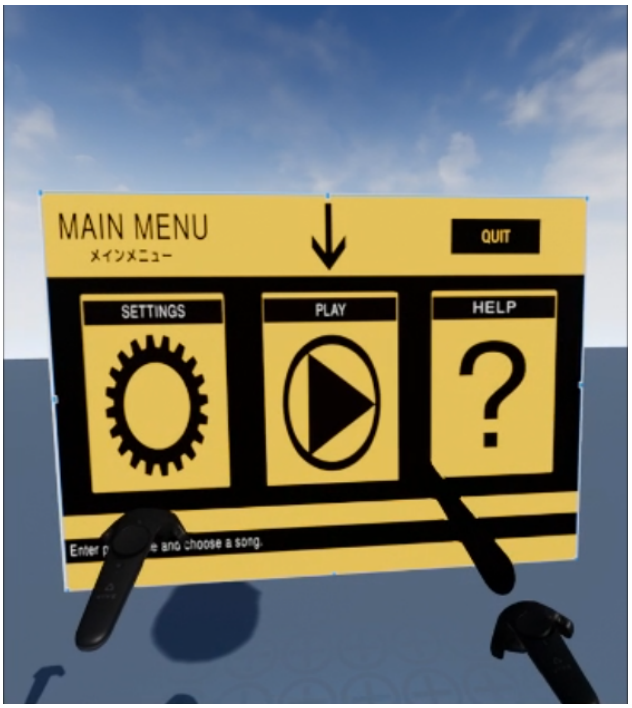


Figure 9.1: Main menu widget in-game.

9.1.2 Song Selection Menu

By hovering over and selecting a song, a player can hear an audio sample of the track selected. Once a player has selected a track to play, they can then choose a difficulty, as shown at the bottom of the menu. Once both a track and difficulty have been selected, the player can press the play button to be taken to the next level where the gameplay takes place.

Figure 9.2: Song Selection menu in-game.

9.1.3 Settings

The settings menu shown in figure 9.3 gives players the choice to alter a number of different gameplay mechanics. Note size allows players to adjust how large notes will appear once they begin gameplay. Location refers the the size of the play area in which notes appear around the player. Scale is a secondary sizing option which alters both the note size and location size. Besides these three options, the player is also given several other choices such as changing the music volume, deleting their saved scores, toggling no-fail mode, and resetting to the default settings.

9.1.4 Help

Players that are new to the game or need further explanations can get help by selecting the help option from the main menu. As shown in figure 9.4, the help menu provides various explanations for players. All aspects of the game are explained to the player, from navigating between menus, choosing a song, and playing the game.

9.2 Gameplay

After pressing play from the song select menu, players will be taken to the actual gameplay section of the game. Once players load in, the chosen song will begin to play, and the game engine will begin to spawn notes in the play area. As shown in figure 9.5, each note has a ring around it after spawning. The ring will

Figure 9.3: Settings menu in-game.

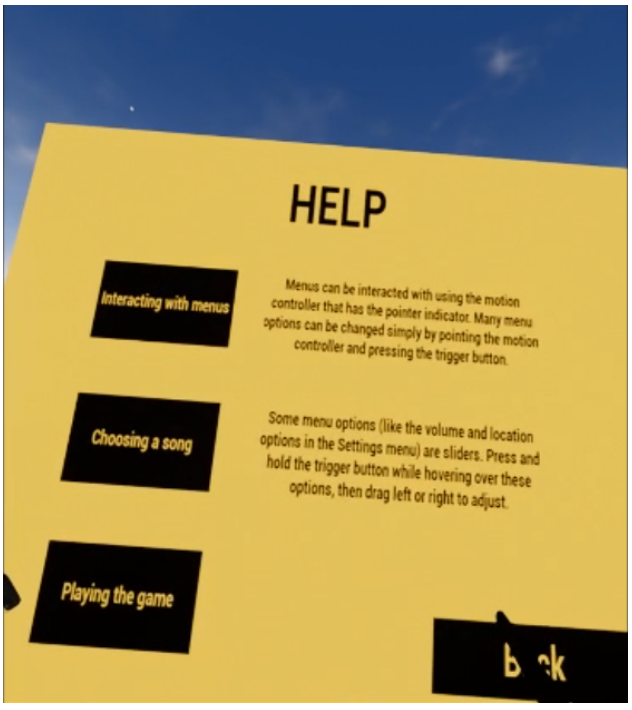


Figure 9.4: Help menu in-game.

shrink until it makes contact with the note, which indicates the ideal time to hit the note with the motion controller. Based on the players timing when they hit a note, a word pops up indicating the accuracy of their timing. The messages that appear over the note hit are perfect, great, okay, or miss depending on how accurate the player was.

Figure 9.5: In-game screenshot of gameplay.

The heads-up-display shows the players current score and combo count. When a player successfully hits a note, the combo count will increase and the score will update. When a player misses a note, the score will not update, and the combo count will reset. Missing too many notes will result in the player failing a song. Reaching the end of a song without failing results in the player clearing it, in which the final score is displayed to the player. Should the player need a break while in the middle of gameplay, they can pause the game by pressing the menu button on either motion controller. Once the game is paused, the player is also given a number of other options such as returning to the main menu or restarting the song.

9.3 Lessons Learned

During the development of our game, we learned two lessons that allowed us to put our designs to fruition: learning the documentation early and the necessity of getting feedback early.

Before working on this project, our team had very little experience with Unreal Engine 4 and game engines in general. During the elaboration phase, we came up with a architectural design without having a good grasp of the software. When it came to implementation, we ran into a lot of hiccups that prevented us from proceeding into later development phases. After learning Unreal Engine's intricacies we were able to come up with a comprehensive architectural design plan that had very little implementation pitfalls.

Getting feedback early was also crucial for meeting our development goals. From the feedback we received, we noticed two major problems with our gameplay. The first was that players had a hard time following the stream of notes as they appeared and sometimes hit notes unintentionally, causing them to receive bad accuracy ratings. The second problem was that players had issues colliding with the actual physical environment around them. To remedy the first issue, we implemented a rule within our game that only allowed players to make contact with the next two notes that were about to disappear. In addition to this, we added pink neon lines as a particle effect to guide players as the notes appeared. As for the second problem, we provided players with options to adjust the play area as needed. The early feedback helped us refine and revise our systems early so that we would not have to run into future problems trying to integrate these features.

9.4 Future Work

Moving forward, we are looking to add more features and gameplay mechanics. The first gameplay mechanic we want to add is more types of notes. These notes will have different requirements for success such as requiring the left hand for collision or right hand. We are also looking towards implementing different types of interaction with the notes such as shooting the notes with the motion controllers as guns or slashing the notes with the motion controllers as swords.

Other short term goals include refining the aesthetics and graphics of our game, code optimization, and compatibility with the Oculus Rift. In the long term, we want to create tools that allow players to easily import their own music and be able to create custom timing files. These features will help enhance the user experience along with ensuring the longevity of the game and its enjoyment factor.

Chapter 10

Conclusion

The lack of compelling software that takes virtual reality to its fullest is a concern for the potential consumer. Rhythm games on the virtual reality platform lack a sense of depth in regards to movement and immersion. Our solution is rhythm game that forces players to move their bodies in different directions in order to beat a song. Our game is the first interactive experience that does not tie a player to a static position in an interactive environment. We are also the first rhythm game that utilizes a mechanic to indicate the timing of a beat independent of location. We hope that our solution inspires upcoming software developers to go beyond the limitations of static gameplay and interaction, providing content that brings out the whole new world of immersion that is virtual reality. We also hope that our solution inspires upcoming software developers to flex their creative muscles and produce quality entertainment that is enjoyable and engaging to players.