

6-13-2019

Keyboard Hero

Connor Lucier

Follow this and additional works at: https://scholarcommons.scu.edu/cseng_senior

 Part of the [Computer Engineering Commons](#)

SANTA CLARA UNIVERSITY

Department of Computer Engineering

I HEREBY RECOMMEND THAT THE THESIS PREPARED
UNDER MY SUPERVISION BY

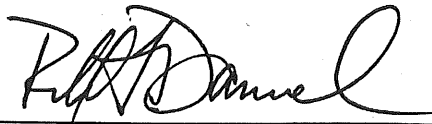
Connor Lucier

ENTITLED

KEYBOARD HERO

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

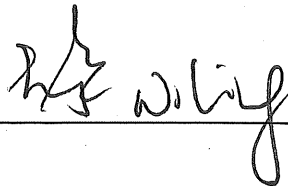
BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE & ENGINEERING



Thesis Advisor

6/11/19

date



Department Chair

6/12/19

date

KEYBOARD HERO

By

Connor Lucier

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Computer Engineering

of

SANTA CLARA UNIVERSITY

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

Santa Clara, California

June 2019

Senior Design Project: Keyboard Hero

Connor Lucier

Department of Computer Engineering
Santa Clara University
June 13, 2019

ABSTRACT

Learning the piano is a complicated process that involves a lifetime of practice. For many new musicians, however, this task is overwhelming and discouraging because of the many skills involved in playing the piano. To solve this problem, I aim to create a computer game compatible with MIDI keyboards that gives novice pianists an alternative to the typical learning process and gives experienced pianists a new and useful practice tool. The game I created is inspired by two popular video games, *Guitar Hero* and *Rock Band*, featuring instant accuracy feedback against imported MIDI tracks, high score tracking, and the ability to practice at slower tempos. This game will hopefully allow players of all skill levels to find new ways to have fun playing the piano.

Contents

1	Introduction	1
2	Requirements	2
2.1	Functional	2
2.2	Non-functional	2
2.3	Design Constraints	2
3	Use Cases	3
4	Activity Diagrams	5
4.1	Core Functionality	5
4.2	Additional Feature: Practice Mode	6
5	Technologies Used	7
5.1	Game Engine - Unity	7
5.2	Languages - C#	7
5.3	Unity Packages	7
6	Game Architecture	8
6.1	Architecture Diagram	8
7	Design Rationale	9
7.1	Rationale for Technologies Used	9
8	System Description	10
8.1	Navigating the Menus	10
8.2	Playing a Song	10
9	In-Game Images	12
10	Testing	18
10.1	Early Testing	18
10.2	Unit and Functional Testing	18
10.3	Integration Testing	18
10.4	Play Testing	18
10.5	Compatibility Testing	18
11	Difficulties Encountered	19
11.1	Note Creation & Timing	19
11.2	MIDI Keyboard Input	19
11.3	Standalone Builds & Package Issues	20
11.4	Song Importing	20

12 Future Work	21
12.1 Note Creation & Timing	21
12.2 MIDI Keyboard Input	21
12.3 Song Importing	21
12.4 Visual & Audio Effects	21
12.5 Improved Game Options	21
12.6 Additional Features	22
13 Lessons Learned	23
13.1 Working Alone	23
13.2 Setting & Sticking to Deadlines	23
13.3 Game Development	23
14 Societal Issues	24
14.1 Alternative Notation	24
14.2 Free-To-Use Decision	24
15 Conclusion	25
16 Appendix	26
16.1 Installation Guide	26
16.1.1 Game Installation	26
16.1.2 Manual Song Importing	26
16.1.3 Launch Settings	26
16.2 User Manual	26
16.2.1 Main Menu	26
16.2.2 Options Menu	27
16.2.3 Mode Selection Menu	27
16.2.4 Song Selection Menu	27
16.2.5 Song Player	28
16.2.6 Viewing Final Results	28

List of Figures

3.1	Use case diagram for students and teachers	3
4.1	Core functionality, featuring song player and main, options, and song selection menus	5
4.2	Core functionality with the addition of practice mode	6
6.1	Core game architecture	8
9.1	Main Menu	12
9.2	Options Menu	13
9.3	Mode Select Menu	13
9.4	Song Select Menu	14
9.5	Add Song Menu	14
9.6	Additional song options for practice mode	15
9.7	Song Player in quickplay mode with multicolor notes and standard key colors	15
9.8	Song Player in practice mode with single-color notes (red orange)	16
9.9	Song Player in quickplay mode with multicolor keys and notes	16
9.10	Song Complete Menu	17

1

Introduction

Aspiring musicians are often discouraged by the overwhelming task of learning an instrument. Learning the piano is a strong example of this predicament. Playing the piano requires that students play two different components of a song with each hand while reading two lines of music simultaneously. Learning these skills can be a frustrating and tedious challenge, which creates a barrier to entry for many new pianists.

Currently, few methods of teaching the piano without an instructor exist. For many students, the first step in the learning process is to buy a book. However, finding the right book for your specific needs can be a challenge in itself. Many casual, entry-level piano books spend too much time teaching about music theory and rhetoric instead of useful practice techniques to employ. In addition, most songs taught in these texts are unfamiliar to new musicians, putting a larger gap between students and their goals of learning their favorite songs.

My proposed solution to these pitfalls that pianists face was to gamify the learning process. Music-oriented video games have had strong success in the past decade, mostly coming from two popular series, *Guitar Hero*¹ and *Rock Band*.² Both of these series utilize simplified guitars to give players the opportunity to become a rock star without any prior experience. While these games entertain players of all ages and cater to many skill levels, they do not actually teach players how to play the guitar. In fact, talented guitarists often struggle to play these games at a high level, which makes the games poor tools for learning and practice of the instrument they are based on.

The piano, compared to the guitar, lends itself more naturally to the format of *Guitar Hero* and *Rock Band*. In these games, notes appear at the top of the screen and fall toward a single row of note heads along the bottom of the screen. However, this system becomes confusing with a standard six-string guitar, creating a problem that previously did not exist. While a MIDI keyboard can have up to eighty-eight keys, the arrangement of the keys in a single row makes this format much easier to digest for novices and veterans alike. Additionally, many popular songs across various genres and difficulties utilize less than half of this maximum number of keys, allowing players of various skill levels to play songs within their abilities without oversimplifying the instrument.

The core of the game includes a menu from which to select a song to play, an interface through which a player plays a selected song and receives live feedback, and a system for tracking local song high scores. In addition, I created a practice mode for the game, which allows players to play songs at slower tempos in order to learn before being scored on the song.

By focusing on playing real music with real instruments, my proposed game will help novice and advanced pianists develop their skills and learn new music more effectively than ever before. The gamification of the learning process for musical instruments, particularly the piano, could potentially help thousands of people learn to play their favorite music without the upfront cost of purchasing private lessons or books. Rather than frustrating students, this solution truly rewards players of all levels for accomplishing their original goal of making music.

¹“The Beats to Beat: A History of Guitar Hero.” <https://historycooperative.org/the-beats-to-beat-a-history-of-guitar-hero/>

²“10 Years Ago: ‘Rock Band’ Changes The Music Video Game Industry.” <http://ultimateclassicrock.com/rock-band-anniversary/>

2

Requirements

This section outlines the functional and non-functional requirements of the proposed game as well as the design constraints of the project.

2.1 Functional

The game will:

- allow players to import MIDI files into their game
- provide a menu for players to select a song to play
- provide players an interface to play along with their MIDI tracks
- track the player's score and accuracy as they play along with the song
- store players' local high scores to be viewed in the song selection menu
- give players various customization options to optimize their playing experience

2.2 Non-functional

The game will be:

- intuitive – players should be able to navigate through the game without confusion
- low-impact – the game should be playable on high- and low-performance computers alike
- visually appealing – the game should draw players in visually

2.3 Design Constraints

The game must be:

- playable on both Windows and Mac OS
- compatible with MIDI controllers and digital pianos

3

Use Cases

This game will be used by two different actors: students and teachers. Speaking technically, teachers themselves are also students, but they could use this game to help their students learn to play music in a new and interesting way. All features of the game are available to students and teachers of all skill levels – it is up to each player to decide how best to play the game. Figure 3.1 shows the core use cases of the game.

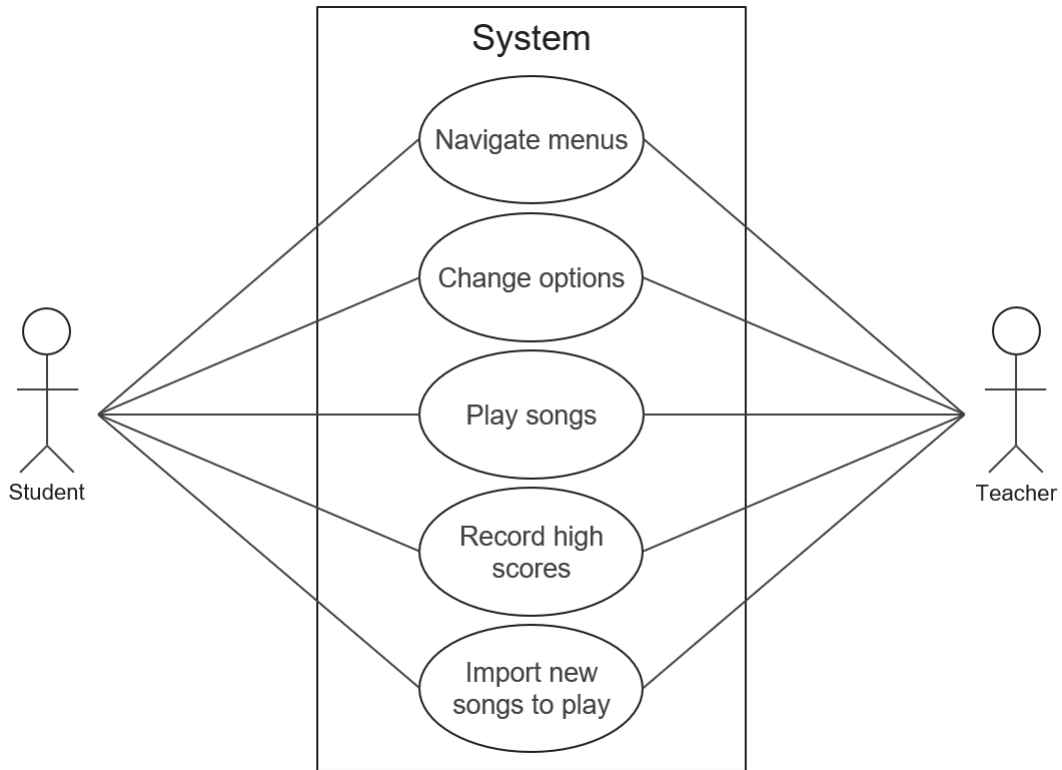


Figure 3.1: Use case diagram for students and teachers

- **Navigate menus**
 Goal: allow clients to select a game mode, select a song to play, and exit the game
 Actors: all
 Precondition: player opens the game
 Postcondition: none
 Exceptions: game exits unexpectedly
- **Change options**
 Goal: allow clients to customize their game
 Actors: all
 Precondition: player navigates to the options menu
 Postcondition: game options are saved
 Exceptions: game exits unexpectedly
- **Play songs**
 Goal: play along with source MIDI file and receive accuracy and score feedback
 Actors: all
 Precondition: player selects a song to play
 Postcondition: player is presented with final accuracy and score for the song played
 Exceptions: game exits unexpectedly
- **Record high scores**
 Goal: allow players to track their progress as they improve
 Actors: all
 Precondition: player finishes playing a song
 Postcondition: player is able to see their highest score on a given song from song selection menu
 Exceptions: high score file(s) are deleted or lost
- **Import new songs to play**
 Goal: give players the opportunity to play more songs
 Actors: all
 Precondition: game is properly installed
 Postcondition: newly-imported song(s) are visible in song selection menu
 Exceptions: song file(s) is corrupted or lost

4

Activity Diagrams

The flow of the game can best visualized with an activity diagram. The first diagram shows the core game experience with no extra features implemented. The second, however, shows the complexity added to the game by practice mode, the additional feature I built into the game.

4.1 Core Functionality

Figure 4.1 shows the basic flow of the game in its simplest form. The player begins at the main menu and is presented with the options to play songs, change options, or quit the game. Selecting 'Options' takes the player to a menu that will allow them to configure various settings for playing songs. Choosing 'Play' will take the player to the Song Selection Menu, which will show them a list of their songs and their highest score on each song. Clicking 'Exit' will quit the game.

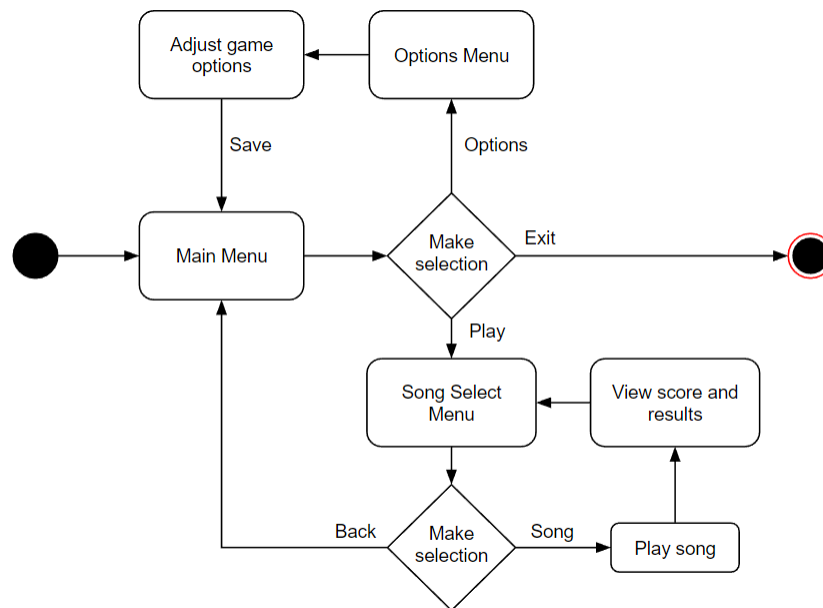


Figure 4.1: Core functionality, featuring song player and main, options, and song selection menus

4.2 Additional Feature: Practice Mode

Figure 4.2 shows the flow of the game after adding practice mode. As is visible in the diagram, the addition of this feature adds complexity to the game, creating some additional challenges for the implementation and testing of the game.

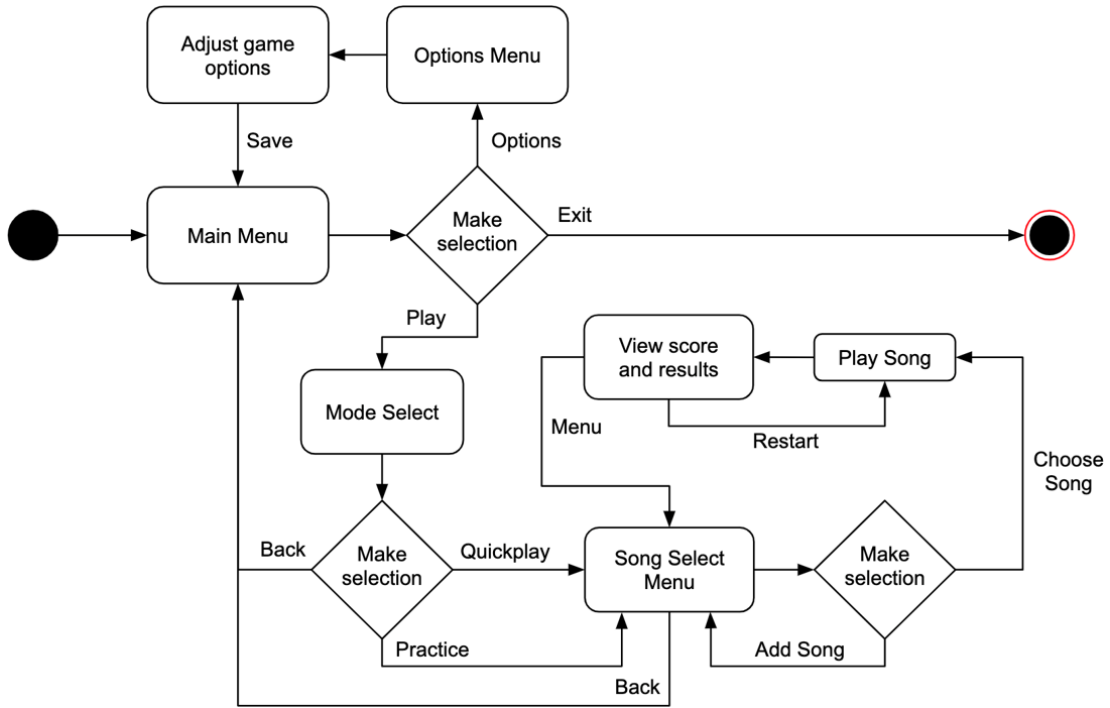


Figure 4.2: Core functionality with the addition of practice mode

5

Technologies Used

This chapter discusses the various tools I used to develop this proposed game.

5.1 Game Engine - Unity

Unity is a free, widely-used game engine that has all the necessary functionality I need to develop the various features of this game. Based on my initial research, Unity does not support MIDI interaction out of the box, but various developers in the Unity community have created free extension libraries that helped accomplish my goals for this project.

5.2 Languages - C#

C# is the supported scripting language of the Unity engine. It is a language that I am comfortable working with given my prior work and project experience.

5.3 Unity Packages

For this project, I utilized two existing, third-party Unity packages to help handle MIDI file interaction and keyboard input, called Audio Helm and MIDI Jack. These packages were essential to the success of the project because I was working alone, as I simply did not have enough time to build everything I needed from scratch. While each of these packages had their limitations, using them allowed me to focus on building the game itself rather than building the tools I needed to build the game.

6

Game Architecture

This chapter describes the overall game architecture and visualizes the interactions of the game's various components.

6.1 Architecture Diagram

As is the case with many video games, this project will consist of an event-based architecture, with the central point of the system being Unity, the game engine that handles and simulates all the game physics, lighting, and visual effects in real time. Using Unity's editor, I will be able to create multiple scenes for the various menus and interfaces necessary for the game. The menu controllers are implemented using Unity's scene management tool and player preferences. The remaining components shown in Figure 7.1 all have to do with the song player interface. I have divided the functionality of this scene into four components: song, note, keyboard input, and UI/statistics controllers. The song controller parses the MIDI source file and informs the note controller of when notes should be created. The input controller determines when a key has been pressed, which allows the UI and statistics controller to update the current score, streak, and accuracy as the user plays a song.

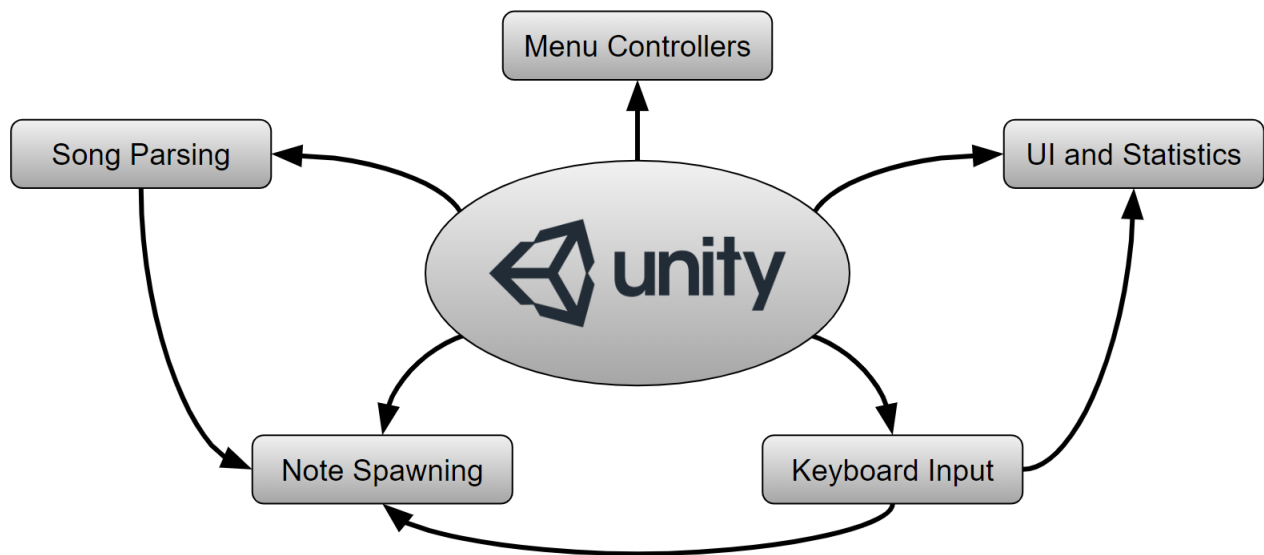


Figure 6.1: Core game architecture

7

Design Rationale

This chapter provides reasoning behind the technologies selected to build the game.

7.1 Rationale for Technologies Used

- **Unity**

Unity is a free game engine with an active community and development team. Additionally, Unity games can be compiled and played on both Windows and Mac OS, which satisfies the constraints of the project. The engine itself has been refined significantly over the years, resulting in an easy-to-use environment that still retains much of the complex functionality required for game development. Another option I considered was Unreal Engine. However, the greater capabilities and complexity of Unreal is intended for life-like 3D games, which this game is not. Because of the simple nature of this game, Unity seemed like the best option for development.

- **C#**

C# is the coding language supported by the Unity engine. Conveniently, I have spent a fair amount of time coding in C# at various companies previously. This makes the combination of these two tools a comfortable and sufficient choice for implementing the design specified in previous sections.

- **Audio Helm**

Audio Helm is one of the only MIDI sequencing libraries of its kind built specifically for Unity. The ability to load MIDI files into a sequencer was exactly what I needed for this project and saved me a lot of time and headache.

- **MIDI Jack**

MIDI Jack, like Audio Helm, was one of the only Unity packages built to find and receive input from MIDI devices attached to the computer. This was an equally important task for my game, and finding this package was a major help in the development of my project.

8

System Description

This chapter provides a detailed overview of the game and its features.

8.1 Navigating the Menus

Upon starting the game, the player is taken to the main menu, from which they have three choices: play, options, and exit. The exit button simply exits the game. The options button takes the player to the options menu, and the play button takes the player to the mode selection menu before moving to the song selection menu.

The options menu gives players four main options to change about their game: volume, note speed, note colors, and key colors. The note speed controls determine how fast notes will drop from the top of the screen toward the bottom. This will change how physically long each note appears as well as how many notes appear on screen at one time. An advanced player may want to play with a higher note speed in order to more accurately see the proper time to play each note. The note color options include a gradient of colors named “multicolor” as well as all the individual colors within the gradient. These allow players to change how the notes look on their screen when they play. Finally, there are two options for key colors: standard black & white and the same multicolor gradient.

The mode selection menu allows the player to select practice mode or quickplay mode. In practice mode, score will not be calculated. However, practice mode allows the player to practice any part of the song at slower speeds in order to learn more effectively. In quickplay mode, the speed of the song is fixed, but the player earns a score by playing the entire song.

From the mode selection menu, the player is taken to the song selection menu, from which players can add songs from within the game and choose a song to play. When practice mode is enabled, the player must also choose a speed and starting point in the song after selecting a song to play. After playing songs in quickplay mode, the player’s high score and corresponding star rating will appear in this menu as well. From here, the player is taken to the song player itself.

The player’s star rating is calculated based on the maximum score possible for that song with no additional score multiplier. Three stars is awarded just for completing any song. Four stars indicates that the player reached a minimum 75% of this score threshold. Five stars indicates that the player reached at least 90% of the threshold.

8.2 Playing a Song

Once in the song player, the player has a couple seconds to get ready before the song begins playing. After that, notes will begin falling from the top of the screen toward the keys along the bottom. As the player plays keys on their own piano keyboard, the on-screen keys will change color to indicate that the key is pressed. A note that is hit will glow while the player holds the note, and a note that is missed will dim to indicate that the player has in fact missed it. The

player will also find a progress bar at the top of their screen to keep track of where they are in the song as well as live score, streak, and accuracy statistics at the bottom of their screen if they are not in practice mode.

9

In-Game Images

This chapter showcases the game’s appearance and functionality through in-game screenshots. These include the various menus and the song player with various game mode and color configurations for the keys and notes. One thing to note is that the main, options, and mode select menus all feature randomly generated “notes” as an added visual effect in the background while playing the game.

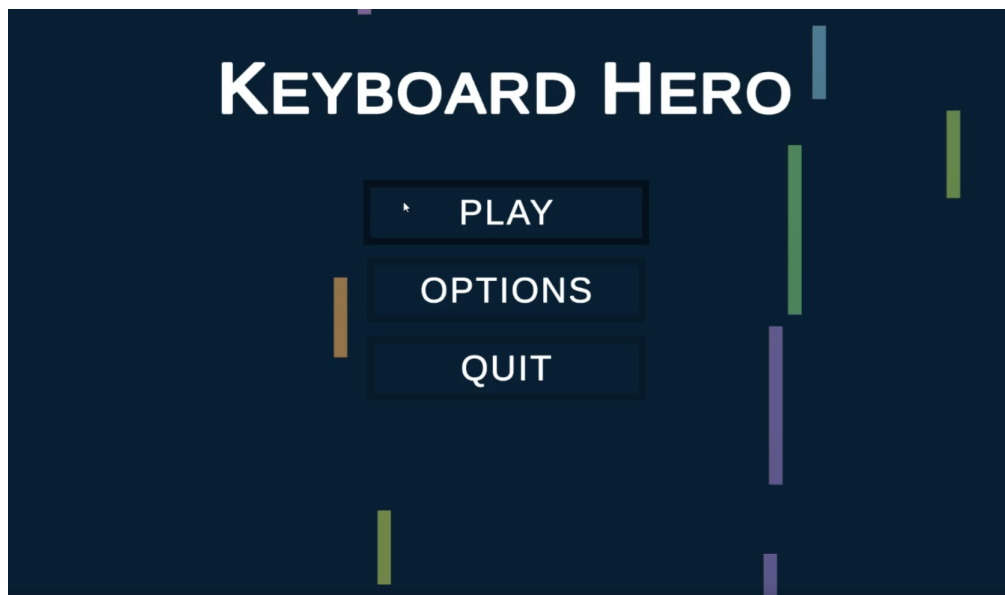


Figure 9.1: Main Menu

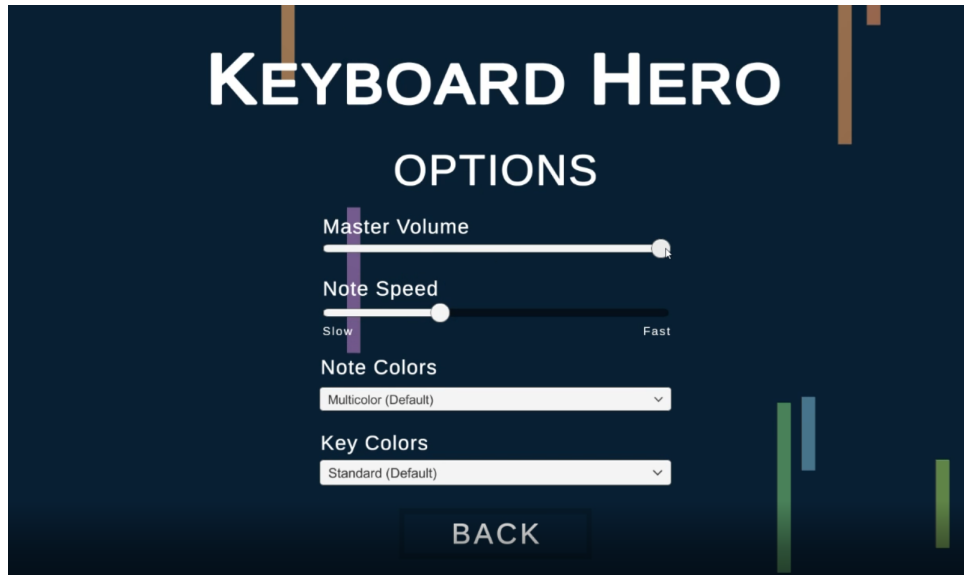


Figure 9.2: Options Menu



Figure 9.3: Mode Select Menu

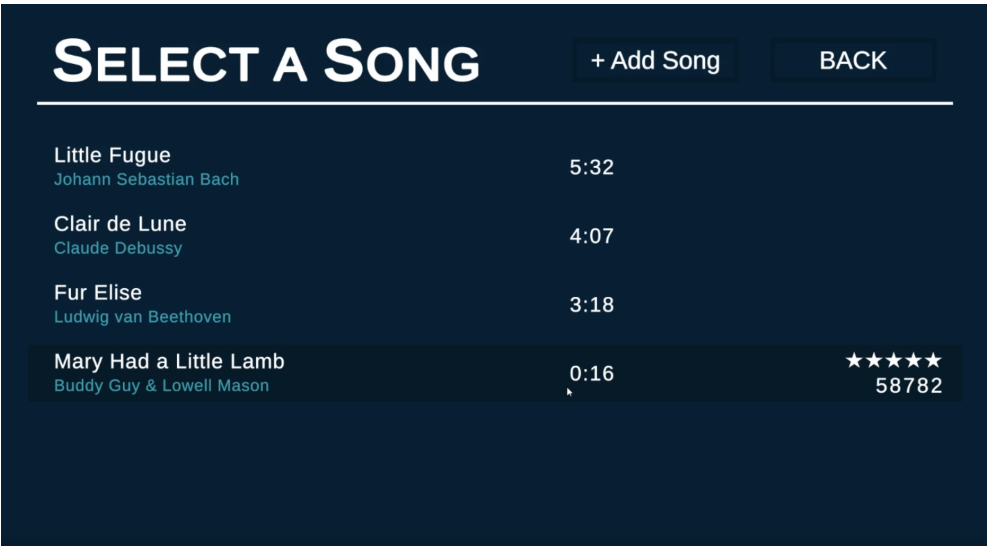


Figure 9.4: Song Select Menu

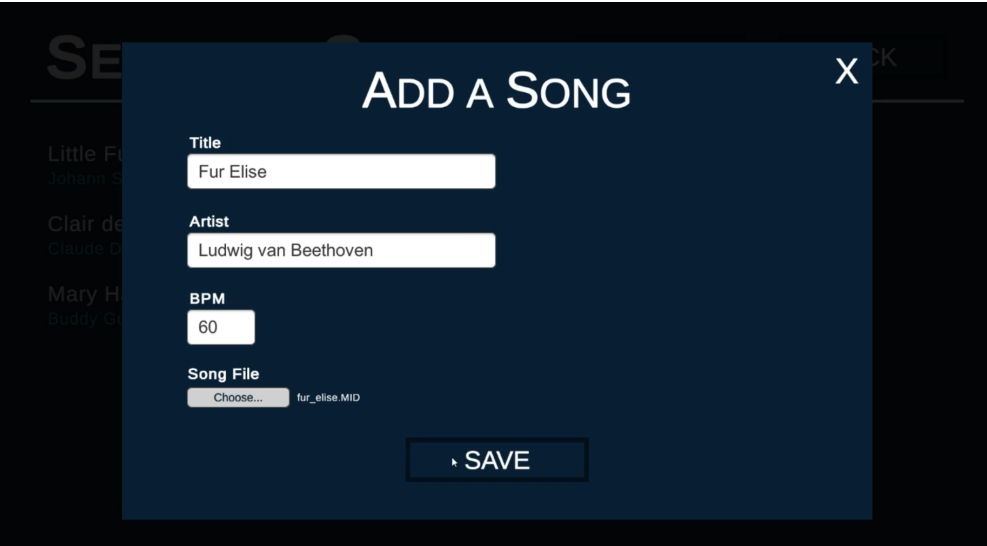


Figure 9.5: Add Song Menu

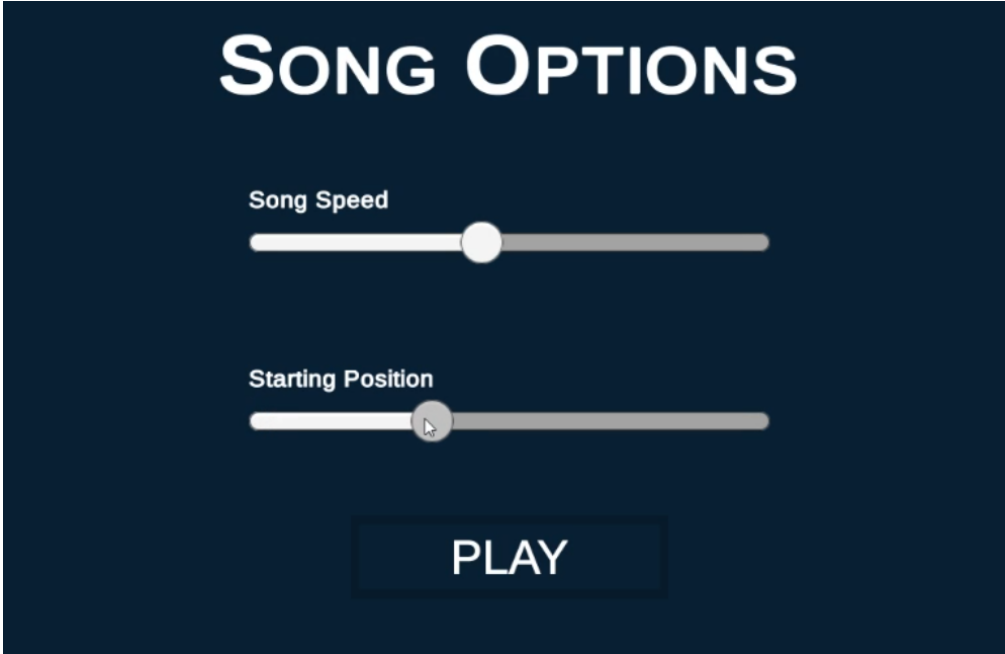


Figure 9.6: Additional song options for practice mode

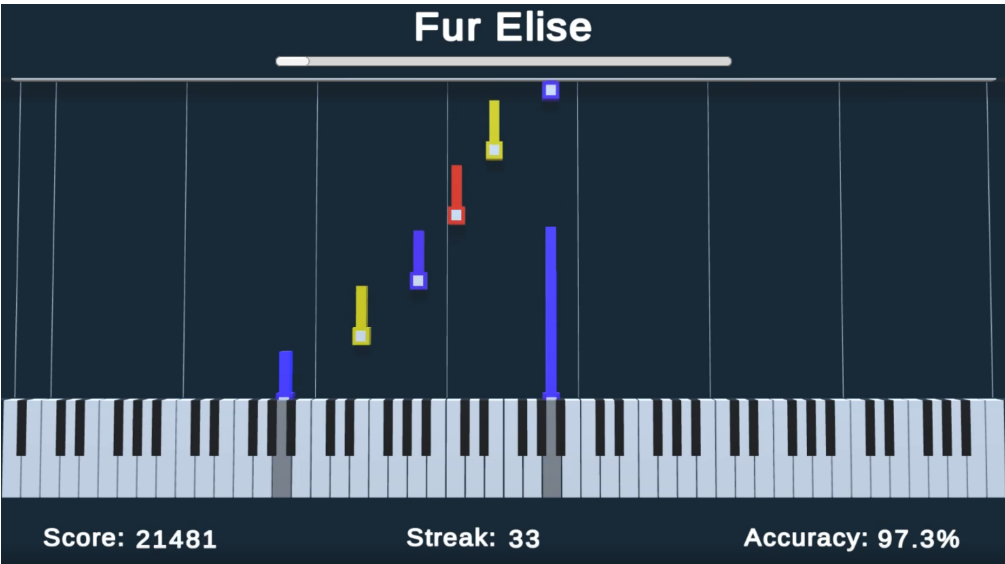


Figure 9.7: Song Player in quickplay mode with multicolor notes and standard key colors



Figure 9.8: Song Player in practice mode with single-color notes (red orange)

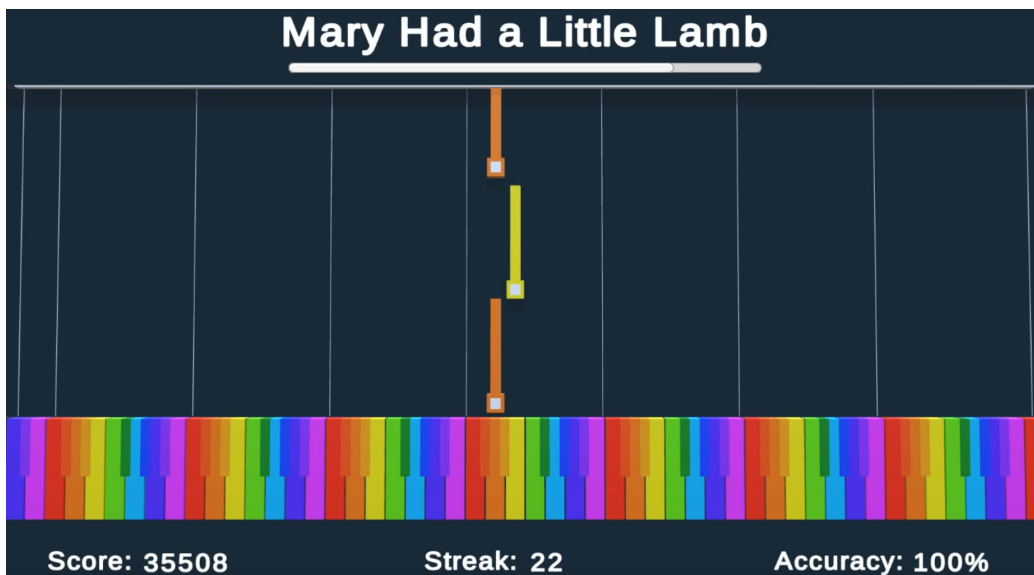


Figure 9.9: Song Player in quickplay mode with multicolor keys and notes

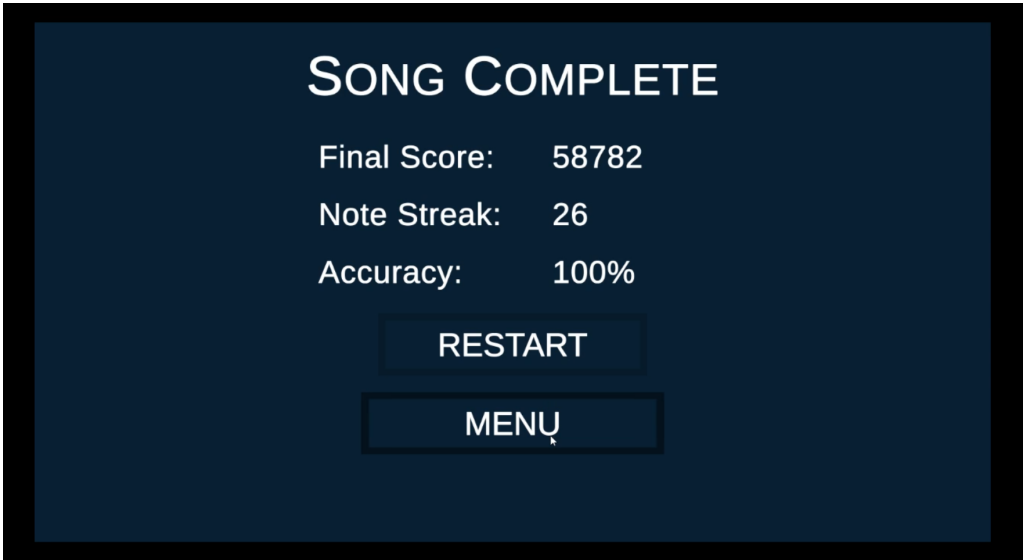


Figure 9.10: Song Complete Menu

10

Testing

This section documents how I went about testing the game as I developed it. Sticking to this plan was vital in fulfilling and verifying all the requirements laid out for the game.

10.1 Early Testing

The convenient part of modern game development is the ease with which a developer can test their game. As such, I was able to run the game and perform simple functionality testing as code is written. By testing new functionality in small pieces, I was able to save time that would otherwise be spent debugging large sections of code.

10.2 Unit and Functional Testing

The first stage of testing involved testing individual components and small groups of components of the system. For example, the song parser and note generator, when working together, produced notes that fell in time with a selected MIDI file. After this was working as intended, I moved on to handle the input from the keyboard to determine whether the player was playing the song properly.

10.3 Integration Testing

Once large components of the game were completed, I was able to navigate through the game and verify that the requirements are met for each section. For example, I tried navigating from the main menu to the options menu, changed game options, and proceeded to play a song to ensure the changes were reflected in the game.

10.4 Play Testing

Once a working version of the game was finished, I had other people test my game. Finding players of multiple skill levels was one of the more difficult tasks in the testing process. Their experience helped guide the refinement process as I worked to achieve my non-functional requirements. Namely, the UI style and color scheme I used was driven by user feedback, but I also worked to make the game rewarding for skilled players but also accessible for new players.

10.5 Compatibility Testing

In order to meet the design constraints of this project, the game needed to run on Windows and Mac operating systems. This was important because players on any computer or laptop should have the same opportunity to play the game. Additionally, computers with less powerful hardware should be able to play and enjoy the game as much as someone with a powerful gaming computer. To accomplish this, I tested the game on my Windows PC as well as my Mac laptop. Luckily, the Unity Engine has no problem supporting both operating systems.

11

Difficulties Encountered

This section discusses some of the difficulties I encountered during the development process of the game and its features.

11.1 Note Creation & Timing

Parsing a source MIDI file and playing it at runtime was simple enough because of the intuitive design of the Audio Helm library. However, creating notes that appear before they are to be played was one of the first challenges I had to solve. To accomplish this first task, I had to implement a song delay to read through the file and create the notes ahead of time. This way, the player is properly able to see notes falling before they are intended to play them.

The next challenge I encountered was the first limitation of the Audio Helm library. The library indexed the MIDI data in terms of sixteenth notes, meaning that 1 unit of time in every song was equal to one sixteenth note multiplied by the beats per minute (BPM) of the song. Because of this, notes that were shorter than a sixteenth note such as thirty-second notes, sixty-fourth notes, etc. were not indexed properly, as integers are typically used when indexing. This led to many notes being created simultaneously when they should not be. To solve this, I had to find the shortest note duration in the song and “multiply” the entire song by a scaling factor to make that note equal to 1 unit of time. I also had to multiply the song’s BPM by the same factor so that the playback of the song would sound the same as the original.

One final challenge I encountered was how to make the notes line up perfectly in time with the song itself. This started as a simple physics problem: the height at which the note is created is constant, and the speed at which it falls is also constant. The time in seconds, therefore, until a note reaches the point at which it should be played, is also constant. However, due to the varying BPM of each song, the song delay had to be calculated at runtime for each song. In addition, I had to create note objects in a special way within the Unity Editor so that the “origin” of each note was along the bottom edge rather than in the center of the note. This way, I could use the calculations above to line up the start of the note in the song with the closest edge of the note created on screen

11.2 MIDI Keyboard Input

The next area in which I had trouble with while developing the game was the input from the keyboard, mostly due to the limitations of the MIDI Jack library. MIDI Jack is incredibly accurate at detecting whether a note on your device is being pressed or not. However, it is far less reliable at determining the exact instant when a key is pressed. Because of this, I was forced to create a small loophole in my game: if a player was to press all the required keys down simultaneously for the duration of any song, they would receive a perfect score. In reality, the player would only be cheating themselves in doing this. However, this remains an issue that should be addressed in future work.

11.3 Standalone Builds & Package Issues

Another area that gave me trouble in creating this game was actually building the game and allowing players to play it. The packages I was using in particular had some code that was, at least initially, intended for use in the Unity editor only, not in standalone builds. To address these issues, I had to contact the developers of each package to learn how to use the code outside of the editor before I could successfully distribute the game.

11.4 Song Importing

The final issue that I encountered with my game was the ability for players to import MIDI files into the game to play from within the game itself. At first, this task seemed fairly straightforward, as Unity has its own file dialog built into the editor. However, when running a standalone build of a game, this file dialog is no longer accessible. As such, I had to search for another way to allow file importing within the game. While some options exist, many did not fit the needs of my project. I was able to find one third-party package that accomplished this task, but using it actually broke the functionality of the menus in the rest of the game. As such, I am continuing to search for an alternative that will help simplify the process of adding songs to the game.

12

Future Work

Due to working alone and the time frame of this project, I was not able to implement all the features that I had hoped to in this game yet. Some existing functionality could be improved upon, and additional features could be added to the game.

12.1 Note Creation & Timing

As mentioned in the previous section, note creation and timing is something that could be improved upon in this game. Namely, I believe that I may have solved the problem of timing backwards. Because the song delay (in seconds) and the note speed should be constant across all songs, the height at which notes are created should be the variable in the equation, rather than fixing the height and varying the song delay itself. This may be the root cause as to why some songs line up better than others.

12.2 MIDI Keyboard Input

In future work, I would hope that a solution to the loophole I describe previously would be found. For now, a method of cheating remains in the game for which I have no immediate method to mitigate or punish.

12.3 Song Importing

A custom song import dialog would be the best solution to some of the difficulties described earlier, but time did not allow for me to create my own. The game would feel much more cohesive with a custom-made file dialog.

12.4 Visual & Audio Effects

One of the biggest drawbacks of working alone on this project was that I was unable to make the game as visually appealing and rewarding as *Guitar Hero* or *Rock Band*. These games rely heavily on visual and audio effects to give players instant feedback. As of now, the only visual effects I was able to incorporate into the song player were the glowing or fading of notes as they are hit or missed.

12.5 Improved Game Options

Another area that I was unable to get to in this game was advanced game options. Unity allows players to customize their graphics settings before launching the game, but adding the ability to change these options at runtime is very helpful for players as they determine what settings are best fit for their computer. In addition, the addition of sound effects and other noises would allow for more robust volume options. Future work could see these options incorporated into another section of the options menu.

12.6 Additional Features

Some of the features that I wanted to explore in this game but never got around to were custom song creation, “Star Power,” and online score sharing.

One of my first ideas for an additional feature to add to the game was custom song creation. With this game as a teaching tool, I thought it would be extremely helpful for teachers and other skilled players to be able to record themselves and share their creations with other players should they want to. This was an ambitious idea, but I believe it would make the game even more viable going forward.

Star Power, as implemented in *Guitar Hero* and *Rock Band*, is a temporary boost to the player’s score multiplier as a reward for correctly playing certain sequences of notes throughout the song. This feature would add a lot of fun and excitement to the game, but was not something I could accomplish alone in the time I had.

Finally, I hope that this game will eventually integrate with a website that would serve as a repository for sharing song files with other players as well as tracking high scores on popular songs. This would be an incredible feature to add, but would require an immense amount of work to accomplish.

13

Lessons Learned

This project has been an incredible learning experience as an aspiring game developer and as an engineer. I hope to apply these lessons to my future projects.

13.1 Working Alone

The experience of working alone on this project was quite unique compared to other senior design projects, and doing so illuminated both the benefits and drawbacks of completing a large-scale project without a team. Firstly, working alone increased my overall workload, as I was entirely responsible for every deliverable throughout the year. While this seems like a major drawback, I also did not have to worry about picking up slack for others on any assignment, coordinating times to meet, and organizing my presentations in a way that allowed everyone to have time to talk. In addition, the consistency of the writing style throughout all this project's documents is evident, which can become a problem in large teams.

The biggest drawback I faced by not working on a team was a lack of time to explore all the possibilities this project still has to offer. Given a strong teammate or two, I am confident that the game could have reached a much more polished state in the time given. However, I am still very happy with the progress I made and the product I have created.

13.2 Setting & Sticking to Deadlines

Having a carefully laid out timeline for working on this project was extremely helpful throughout my two quarters of development. By dividing up large goals into smaller tasks, I was able to make visible progress each week. While it was difficult to stay on top of every deadline I set forth, I was able to adapt and change my schedule to accommodate the ebb and flow of the development process.

13.3 Game Development

This project was a great introduction to game development as a process. One of the major takeaways I had from this project was that moving from a game that is functional to a polished product is about 90% of the total work required to make a successful game. Much of the work that I did in the past several months has been on bug fixing, refining, and re-engineering existing functionality in the game. I had to suspend development on a lot of the features I wanted to explore due to the limited time I had to complete the project, but in doing so I gave myself more time to make the existing game more functional.

14

Societal Issues

This section discusses the societal and ethical implications of this project, including decisions made about the release of the game.

14.1 Alternative Notation

This game brings an existing video game music notation to the piano. While this notation system is massively different from sheet music, this game is not intended to replace sheet music. Standard musical notation offers much more valuable information about how to play a piece of music than the system used in the game, and as such, the game is intended to provide another method of learning their favorite songs for those who struggle reading sheet music.

14.2 Free-To-Use Decision

The decision to release the game for free was an early choice I made. The inspiration for this game came from a fan-built game based on *Guitar Hero 3: Legends of Rock* called *Clone Hero*, a Windows and Mac version of the game with added functionality to allow players to add songs and playlists created by other players. This game is free and open to the public because it was built by players for players. Similarly, I chose to provide this tool for free to anyone who desires to use it rather than to seek profits from selling the game, with the hope that a strong community of players may form and drive the game's future development.

15

Conclusion

This report has documented the requirements, use cases, design, implementation, testing, challenges, and lessons learned from the development of a new, music-based video game for pianists of all skill levels. I hope that this report is a good starting point for future work on the project or another game of its kind.

16

Appendix

This appendix contains two sections. First is an installation guide that instructs players on how to properly install the game. The second is the user manual that walks a user through how to use and play the game.

16.1 Installation Guide

This section provides a comprehensive guide of how to install the game, add songs manually, and configure the game before launching.

16.1.1 Game Installation

After obtaining the ZIP file for the game, simply extract the contents to a location on your machine. Once the game is extracted, proceed to the next step.

16.1.2 Manual Song Importing

If you have additional songs to add to the game, ensure that the MIDI file you want to add is in a folder along with a .ini file named 'song.ini' before moving it to your game installation's Songs directory. On windows, this folder is located in the subdirectory of your game's installation. On Mac, the Songs directory is inside the *KeyboardHero.app/Contents* folder.

16.1.3 Launch Settings

Upon launching the game for the first time, a Unity menu appears which allows you to select the graphics quality, resolution, and window mode you would like to use for the game. After customizing these settings to your liking, you can also select to skip the dialog when launching unless explicitly specified.

16.2 User Manual

16.2.1 Main Menu

From the main menu, you can:

- Click the Play button
- Click the Options button
- Click the Exit button

Clicking the Play button takes you to the mode selection menu. Clicking the Options button takes you to the options menu. Clicking the Exit button simply closes the game.

16.2.2 Options Menu

In the options menu, you can:

- Change master volume
- Change note speed
- Change note colors
- Change key colors
- Return to the main menu

By using the sliders and dropdown menus provided, you can customize the four main game options from the options menu. Once you have customized the game to your liking, you can click the Back button to return to the main menu.

16.2.3 Mode Selection Menu

The mode selection menu allows you to:

- Click the Quickplay button
- Click the Practice button
- Return to the main menu

In the mode selection menu, you can select one of the two modes of play or click the Back button to return to the main menu. The Quickplay mode allows you to play songs at full speed and receive scores for your performances. The Practice mode allows you to slow down songs to a comfortable speed while learning, but you do not receive a score for playing songs in this mode.

16.2.4 Song Selection Menu

From the song selection menu, you can:

- Select a song to play
- Add a new song to your game
- Return to the main menu
- View current high scores on your songs

Once you have reached the song selection menu, you can choose a song from your list of imported songs to play. In addition, you can add another song right from in the game, so long as you have a MIDI file downloaded and ready to import. Clicking the Add Song button will open a dialog which asks for the song's title, composer, and BPM. Finally, the dialog allows you to select the file you wish to import. Once you have entered the required information, you can save the song to your library, which automatically returns you to the song selection menu, where you will see your newly added song. From here, you can again choose a song to play or return to the main menu.

In addition to selecting songs, you are also shown your current high scores on each song in your library in the song selection menu. If you have never completed a certain song or only practiced it, your high score will be blank. Otherwise, your highest score and a star rating will appear in line with the song title. The star rating associated with your score is based on the maximum score possible calculated for that song without a streak multiplier. If you are able to score over 80% of this calculated score, you earn four stars on that song. Additionally, you earn five stars for scoring 90% of this score or higher on that song.

In practice mode only, selecting a song does not immediately take you to the song player. Instead, you are prompted to select a speed and starting position in the song before beginning to play. This brief menu allows you to practice a tricky part of a song you are learning as slowly as you need to.

16.2.5 Song Player

In the Song Player, you can:

- Play the selected song
- Pause and resume the game
- Restart the song
- Return to the song selection menu

In the song player, you will see the title of the song you selected to play at the very top of your screen. Below that, a progress bar will indicate how far along you are in the song. The middle of the screen holds the playing area, including the visual keyboard and octave lines. As you play your selected song, notes will appear at the top of this playing area and fall toward the corresponding keys below. Your goal is to hit all the notes as they reach the top of the key below and hold them for the duration indicated by the physical length of the note. You receive the most points for hitting several notes in sequence, increasing your score multiplier as your note streak increases. Finally, your statistics are shown along the very bottom of your screen, showing your current score, note streak, and accuracy.

By pressing the Escape key on your computer's keyboard, you can pause the game at any point while playing a song. Pausing the game will stop your progress in the song and open up the pause menu. From this menu, you can resume the game when ready, restart the song or section of the song you are practicing, or return to the song selection menu to choose a different song. Pressing the Escape key again will automatically resume the game without the need to press any buttons on your screen.

16.2.6 Viewing Final Results

After completing a song in the song player, a summary appears that shows you your final stats on the song you played. This shows you your total score, highest note streak, and your overall accuracy. From here, you can choose to play the song again or return to the song selection menu.