THE DESIGN AND DEVELOPMENT OF AN AUDIO-ONLY RHYTHM GAME

by

Qingxiao Yu

Bachelor of Management, 2016

A Major Research Project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Digital Media

in the program of

Digital Media

Toronto, Ontario, Canada, 2017

© Qingxiao Yu, 2017

Author's Declaration for Electronic Submission of a MRP

I hereby declare that I am the sole author of this MRP. This is a true copy of the MRP, including any required final revisions.

I authorize Ryerson University to lend this MRP to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this MRP by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my MRP may be made electronically available to the public.

THE DESIGN AND DEVELOPMENT OF AN AUDIO-ONLY RHYTHM GAME

Master of Digital Media, 2017

Qingxiao Yu

Master of Digital Media, Ryerson University

ABSTRACT

The size of visually-impaired population in the world is large. They usually have a better hearing than normal people, but the lack of vision prevents them from enjoying video games. Audio games, as the main output of the type of game is sound, naturally become a game genre that visually-impaired people can play. The project designed and developed a rhythm audio game for visually-impaired people. The game has been successfully tested for different songs.

Acknowledgements

I would like to express my deepest appreciation to all those who supported me throughout the major research project. I am thankful for their aspiring guidance, invaluable constructive criticism and friendly advice during the project work.

I would like to express my gratitude to my supervisor Richard Lachman, program manager Michael Carter and program director Alex Ferworn. Also, I would like to thank the participants in the testing, who have willingly shared their precious time.

I express my warm thanks to Mr. Zhang and for his support.

Table of Contents

Author's Declaration	ii
Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	vi
1. Introduction	1
2. Related Works	2
3. Game Design	4
4. Development.....	7
5. Conclusions	15
References.....	15

List of Figures

FIGURE 1 Flow Chart	6
---------------------------	---

1. Introduction

Audio games, in contrast to video games, are computer games whose main output is sound rather than graphics. They can be accessible to blind or visually impaired players more easily than traditionally-design games (AudioGames.net, 2016). For gameplay that may not be primarily focused on the visual, audio games create a richer environment using sound, and providing possibilities for gameplay that not only need vision. Therefore, they are far more accessible to people with all levels of sight (AudioGames.net, 2016). That is why more and more people are showing interest in audio games, including sound artists, game accessibility researchers, game developers and video gamers. Now there are more than 500 audio games in various genres, consist of strategy games, puzzle games, card games, adventure games, role playing games, interactive fiction, action games, etc. (AudioGames.net, 2016).

Although mobile devices are becoming more powerful, with more memory, more processing power, and more multimedia functionalities, audio games for mobile platforms are rare. There are various kinds of benefits of audio games. It includes visually-impaired audiences in the play, and promote the innovation in trying to represent the game environment, characters and events, using audio and tactile feedback. In addition, it also provides an opportunity to explore and exercise other senses and to create more personalized experiences. By its audio-only experience, it increases the immersive experiences in games (Valente, Souza, & Feijó, 2009). Without visual, players have to rely on their own imagination, so it allows them to focus more on games.

The size of visually-impaired population is large. As of 2014, 285 million people around the world are estimated to be visually impaired, of which 39 million are blind and 246 million have low vision (World Health Organization, 2014). Approximately 500,000 Canadians are estimated to be living in severe vision loss that affects their quality of life, with more than 50,000 Canadians losing

their sight every year. This figure includes people without sight at birth, people who are legally blind, as well as people with less significant vision loss (cnib, 2015).

Therefore, the purpose of this project is through the study of existing rhythm games and audio games, design and develop a rhythm audio game for visually-impaired people.

The remainder of the paper is as follows. In Section 2 we discuss related rhythm games and audio games. In Section 3 we estimate various models of fundamental stock prices and present the results. In Section 4 we state the development process of the game. The final section offers conclusions.

2. Related Works

Rhythm games have become a popular trend in computer games recently. These games basically consist in simulating playing musical instruments (Jaime, Barbancho, Urdiales, Tardon, & Barbancho, 2016). Rhythm game is a subclass of action game. Games in this genre challenge the player's rhythm, such as *Guitar Hero*, *Taiko Drum Master* and *Just Dance*. Players need accurate and synchronized reaction to certain visual maps, which are pre-programmed sequences of buttons they need to press or actions they need to take in specific beats. This type also includes games that test player's sense of rhythm and pitch.

In 2007, the Singapore-MIT Gambit Game Lab developed *AudiOdyssey*, an experimental rhythm game that is completely friendly to the visually impaired (Gambit, 2007). In addition, some games are not built specifically for the visually impaired, but because their gameplays quite rely on hearing, huge potential can be seen they become blind-accessible audio games. *Rhythm Doctor* produced by fizzd in 2014, inspired by *Rhythm Heaven*, is an example of these (fizzd, 2016).

AudiOdyssey is an experimental computer game designed to be accessible to visually impaired people and mainstream gamers. The user plays as a new coming DJ, whose mission is to make the clubbers dance (Gambit, 2007). It supports two controllers, keyboard or Wiimote (the remote-control device used by the Nintendo Wii), by pressing arrow keys on the keyboard or swaying Wiimote on correct timing and direction, players can build up a rich soundtrack gradually (Chandler, 2008). For example, if a player heard a clap on his right, he should hit the right arrow key or swing the remote to the right in time with the music. With a track matched, the player can continue adds more tracks, and more clubbers will dance with the music (Ellin, 2008).

As a rhythm game developed with explicit consideration of accessibility, *AudiOdyssey* successfully makes the visually-impaired and the sighted can enjoy the same level and quality of gameplay. Accessible installer is provided in addition to the normal installer, from installing to menu navigating to playing completely voiceover. The use of Wiimote creates a fun and natural control scheme, it is simple as well as direct. In general, it is an engaging game that relies more on high quality audio than visuals (Gambit, 2007).

However, when thinking of it as a normal rhythm game for people who do not have visual impairment, the quality of the game still exists many problems. Firstly, it's the quality of the art is far behind other similar games, and the entire game screen is in a small window, making it difficult to put players into its environment. No text description is provided during the game, thus for the general players who are got used to text prompts, it is confusing and difficult to understand. In addition, music quality needs to be improved, and the game is too short to be considered as a complete game.

In contrast, *Rhythm Doctor* is a high-quality, general-oriented hardcore rhythm game. It is a tough one-button rhythm game that heavily inspired by *Rhythm Heaven*, featuring an extremely tight margin of error compared to any other music game out there (fizzd, 2016). Players play the role of a doctor,

helping people who come to the hospital to recover normal, rhythmic heartbeat. They will hear the first six beats and finish the seventh beat by hitting space.

In the aspect of a rhythm game for normal players, Rhythm Doctor is stunning. The concept is easy and graspable enough to appeal casuals, but has a learning curve steep enough to tempt players to practice and perform better. The quality of the music is incredible, with unique beats and catchy tunes, as well as the intended glitches in the track add another depth to the rhythmic and sound-based capabilities of the song. Moreover, level difficulty is balanced, with early levels being easy to get start and the later levels taking a few tries to master. Overall, it is an excellent addition to the rhythm game genre.

However, the game still has some technical flaws. It is a web-based Flash game, with the advantage of not needing to be downloaded and installed. This makes it easy to start playing, but users may need to use a specific browser and update the Flash Player plug-in, otherwise the music may delay severely so that the game cannot be played normally (fizzd, 2016). Even though, there are still many players point out the game does not work on Linux as expected, with correct browser and plug-in. Although the gameplay offers the opportunity to be an accessible audio game, but it is not developed to be visually impaired friendly, and therefore a lot of details can be improved for increasing accessibility. Fizzd planned to make it more accessible (Zhou, 2015), but it has not been implemented yet. Unlike *AudiOdyssey*, the interface is not voiceover, and the mapping of some levels is too difficult to rely on hearing entirely.

3. Game Design

3.1 Gameplay Design

“Mania” mode has been widely used in almost all the major rhythm games. It came from Beatmania, which was a rhythm video game developed and distributed by Japanese game developer Konami and first released in December 1997. The game controls consist of five keys, each key with a corresponding vertical bar on-screen. The bars indicate the path which rectangular icons cascade down towards a horizontal line near the bottom of the screen. The player must hit the corresponding key when the icon matches with the line, which will trigger a preset sound sample and recomposes the song properly. Although the shape of bars or icons may be different, most rhythm games were used similar visual mapping gameplay to Beatmania, such as Guitar Hero and Taiko no Tatsujin.

Unlike these games, this game uses audio mappings instead of visual mappings. The gameplay of It relies on hearing only. Controls include tapping and holding down on the touch screen. In each level, series of hint beats are played along the music in two directions, right and left. The player reacts to the beats by tapping or holding down on the correct side of screen to score points. Taps and hold downs can be judged as either “Perfect”, “Good”, or “Miss”. The player must attempt to keep with the rhythm throughout the level, receiving a result at the end of the game depending on how well they did. Multiple difficulties are offered for some songs. In co-op mode, two players complete one song together by share a pair of earphones, one player each side, and controlling his own side of the screen. The players receive individual ranks and overall rank at the end of the song.

3.2 User Interface Design

3.2.1 Flow

There are six scenes in the game: start, select song, select mode, gaming, individual mode result, and co-op mode result. The overall flow is as following.

FIGURE 1 Flow Chart

3.2.2 Visual and Audio Design

As a game supporting visually-impaired players, the game has a special concept in UI design. With eyes closed, the buttons and complex hierarchies of normal mobile apps and games are impossible to use. In order to simplify the control, in each page, the screen is roughly divided into two parts, left and right, each contains only one function. The instructions are given to the player by the voices go in left and right ear. The theme of visual design is dark color and simple, encouraging players to play in an audio-only way. The game also provides a similar experience for users playing with their eyes opening, replacing complex visual effects interfere with audio content.

For example, in the select song page, the right side of the screen allows the player slides up and down to select a song, when the player stops on a song, a preview and song information will be played so that the player can determine if it is what he want; then click on the left side to select the song and enter the following page.

4. Development

4.1 Development Environment

The game was developed with Unity Version 2017.1.0f3 Personal (64bit) on Mac OS X Version 10.11.5. The Unity game engine is the most popular third-party game development software amongst developers. 34% of top 1000 free mobile games are Made with Unity. It also supports multiple platforms, including iOS, Android, Windows, Mac and other game consoles (Unity, 2017), makes the game easy to cross platforms.

The textures, icons and other images used in the game were drawn in Adobe Photoshop CC 2015. The start animation was created by Adobe After Effects CC 2015. The voiceover in the game was

provided by Google Translate, recorded by Apowersoft Audio Recorder for Mac v2.5.0 (Build 06/07/2017). All the audio, including sound effects and music, were created and processed by Logic Pro X Version 10.2.0. It provides rich software instruments and loops that meet the requirement of development.

4.2 Level Development

4.2.1 Music Selection

There are four songs in the game, two of them are loyalty free music from Jingle Punks Music and conte-de-fees.com. The other two are original music produced for the game. The gameplay fits any music genre, but the selected songs were simple and relaxing.

4.2.2 Music Preparation

First, a metronome was used to calculate the BPM (beat per minute) of each song. Once the BPM was known, a Logic Pro X project was created and adjusted to match the BPM of the song. Secondly, the song was imported and the starting position of it was adjusted to match the beats of the project. Then, a duplicate song track was trimmed and added the voice that included song information and level difficulty, and then bounced to an audio file. It was used as the preview in the song selection scene.

4.2.3 Mapping

First, the soundtracks that required for the hint beats were created. The hint beats of taps used software instrument tracks, and the hint beats of hold down notes used audio tracks. Left and right notes each need one software instrument track and one audio track.

Software instrument tracks were using Electronic Drum Kit – Drum Machine Designer – Analog Circuits, the notes were B1. This is a crisp percussion sound with a high distinction with music. After that, the directional effect was added to the track. This effect was achieved by adding the

Imaging-Direction Mixer Plug-in to the tracks in Logic Pro X. For the left tracks, the direction parameters were adjusted to -90.0 degrees and the right sides were adjusted to 90.0 degrees, which gives the direction to the sounds on the track. The spread parameters were remained at 1.00, so that the sound is gathered and not too scattered. Then, according to the beat of the music, the time points of taps, hold down notes, and the hint beats were added. Finally, the music and the hint beats were bounced and exported for playing in the game.

4.2.4 Recording and Process Note Data

Following the mapping, note data was recorded manually into Microsoft Excel. Note data are divided into two columns, left and right, corresponding to the left and right notes in the game. Each note data contains two time points, the start time and the end time, the start time is the time the player should press the button, and the end time is the time the player should release it. For the taps, the start and end times are the same; for hold down notes, the end time is later than the start time. Each side of the note data for each song will be converted to a $[n, 2]$ two-dimensional array, where n is the number of the notes in that side.

4.2.5 Solving the Sound Latency on Android Platform

Because of the music playback mechanism of Android systems, using the Unity engine to play music and sound effects directly causes a significant delay. The delay is various depending on the device. In a rhythm game that focus on beats and time, it is intolerable.

The following two ways were attempted to reduce the sound latency:

- (1) Change the relevant settings in Unity. In the Unity Editor click Edit - Project Settings - Audio, and change the setting of DSP Buffer Size to best latency.
- (2) Use the Android Native Audio plug-in to handle music and sound effect in the game. It provides easy access to the native Android audio system for low-latency audio playback.

The plug-in allowed a significant reduction in delay. With a minor lag adjustment in the judgement process, the audio playback effect has been able to meet the requirement of the game.

4.3 Judgement

The gaming scene consist of two large buttons in the left and right. The game makes a judgement each time the player press and release each button. When the player is pressing, or releasing a button, the current game time is recorded and compared with the time points in the note data array. If the current time is in a certain range around the precise time, the player receives a “Good” or “Perfect” judgement, if not it receives a “Miss” judgement. The key codes are as follows.

The following script was linked to the start scene.

```
//Pre-load sound effects with Android Native Audio when the game starts;
```

```
AndroidNativeAudio.makePool(1);
```

```
SEButtonLpress = AndroidNativeAudio.load ("SmashClapL.wav");
```

```
SEButtonRpress = AndroidNativeAudio.load ("SmashClapR.wav");
```

```
SEButtonLrelease = AndroidNativeAudio.load ("SmashTimL.wav");
```

```
SEButtonRrelease = AndroidNativeAudio.load ("SmashTimR.wav");
```

```
SEButtonLmiss = AndroidNativeAudio.load ("LoFiGlitchL.wav");
```

```
SEButtonRmiss = AndroidNativeAudio.load ("LoFiGlitchR.wav");
```

The following script was linked to the gaming scene and runs immediately when the scene starts.

```
//Get the scene start time.
```

```
StartTime = Time.time;
```

The following script was linked to the left. The one for the right button is similar.

```
//Note data, here is the example of song 3.
```



```

double[,] NoteData = new double[12,2]{{16.20, 16.20}, {19.81, 19.81}, {23.40, 23.40}, {25.20,
27.00}, {34.20, 34.20}, {45.00, 46.81}, {54.00, 54.00}, {63.00, 63.00}, {81.00, 81.00}, {84.60,
84.60}, {88.20, 88.20}, {90.00, 91.81}};

//Detect if the press or release is a “Miss”

private bool isExist = false;

//Lag control

private double LagTime = 0.12;

//Time range control

private double GoodLowerLimit = -0.3;

private double GoodUpperLimit = 0.3;

private double PerfectLowerLimit = -0.1;

private double PerfectUpperLimit = 0.1;

//When button is pressed

public void OnPointerDown (PointerEventData eventData) {

    //Play Sound Effect

    AndroidNativeAudio.play(SEButtonLpress);

    isExist = false;

    //Compare the time of pressing with the time points in the note data

    for (i = 0; i < 12; i++) {

        //If the pressing time is within the “Good” range

        if (Time.time - StartTime - LagTime - NoteData [i, 0] > GoodLowerLimit &&

Time.time - StartTime - LagTime - NoteData [i, 0] < GoodUpperLimit) {

            isExist = true;

```

```

        //If the pressing time is within the “Perfect” range
        if (Time.time - StartTime - LagTime - NoteData [i, 0] > PerfectLowerLimit
        && Time.time - StartTime - LagTime - NoteData [i, 0] < PerfectUpperLimit) {
            JudgeTextL="Perfect!";
            LeftPerfectCount = LeftPerfectCount + 1;
        }
        else {
            JudgeTextL="Good!";
            LeftGoodCount = LeftGoodCount + 1;
        }
        break;
    }
}

//When button is released
public void OnPointerUp (PointerEventData eventData) {
    isExist = false;
    for (i = 0; i < 12; i++) {
        //If the pressing time is within the “Good” range
        if(Time.time - StartTime - LagTime - NoteData [i, 1] > GoodLowerLimit &&
        Time.time - StartTime - LagTime - NoteData [i, 1] < GoodUpperLimit){
            isExist = true;

```

//Judge whether it is a single tap or a hold down note. If the start time and the end time of the note is same, means it is a tap.

```
    if (NoteData [i, 1] == NoteData [i, 0]) {}  
    else {  
        AndroidNativeAudio.play(SEButtonLrelease);  
    }  
    if (Time.time - StartTime - LagTime - NoteData [i, 1] > PerfectLowerLimit  
&& Time.time - StartTime - LagTime - NoteData [i, 1] < PerfectUpperLimit) {  
        JudgeTextL= "Perfect!";  
        LeftPerfectCount = LeftPerfectCount + 1;  
    }  
    else {  
        JudgeTextL= "Good!";  
        LeftGoodCount = LeftGoodCount + 1;  
    }  
    break;  
}  
}  
}
```

4.4 Game Result

The game uses the counts of “Perfect”, “Good”, and “Miss” to calculate the score. First, the score of each side is calculated separately, and then the weighted average of both sides is the total score.

The key codes are as follows.

```

//Left side scores

//SongLcount is the number of notes on the left side

if (LeftPerfectCount + LeftGoodCount + LeftMissCount <= SongLcount) {

    LeftScore = (LeftPerfectCount + LeftGoodCount * 0.8) / SongLcount;

}

else {

    LeftScore = (LeftPerfectCount + LeftGoodCount * 0.8) / (LeftPerfectCount +
LeftGoodCount + LeftMissCount);

}

//Right side scores

if (RightPerfectCount + RightGoodCount + RightMissCount <= SongRcount) {

    RightScore = (RightPerfectCount + RightGoodCount * 0.8) / SongRcount;

}

else {

    RightScore = (RightPerfectCount + RightGoodCount * 0.8) / (RightPerfectCount +
RightGoodCount + RightMissCount);

}

//Overall scores

OverallScore = (LeftScore * SongLcount + RightScore * SongRcount ) / (SongLcount +
SongRcount);

```

4.5 Build and Test

The game used Android Studio Version 2.3.3 with Android 5.0 (Lollipop) SDK to build. It was tested on HTC One 802t and Samsung Galaxy Tab S2. It ran smoothly and fully functional without any error on both devices.

5. Conclusions

The project designed and developed a rhythm audio game for visually-impaired people. A gameplay, which the player listens to hint beats and reacts to them by tapping or holding down on the correct side of screen to score points, is designed and tested. The game is produced by Unity Engine, and available on Android systems. In the project, it is also discovered that using Android Native Audio plug-in is the solution of Unity Engine audio latency on Android systems. Future research will focus on improving the gameplay to attract more players, and invite visually-impaired players to test and provide feedbacks.

References

- AudioGames.net. (2016, October 17). *Game List*. Retrieved October 17, 2016, from Audiogames.net: <http://www.audiogames.net/list-games/>
- AudioGames.net. (2016). *Welcome at AudioGames.net!* Retrieved October 17, 2016, from AudioGames.net: <http://www.audiogames.net>
- Chandler, D. (2008, June 23). *AudiOdyssey: A Game for the Visually Impaired*. Retrieved October 12, 2016, from Dr.Dobb's: <http://www.drdobbs.com/architecture-and-design/audiodysey-a-game-for-the-visually-impa/208800207>
- cnib. (2015). *Fast Facts about Vision Loss*. Retrieved October 17, 2016, from cnib: <http://www.cnib.ca/en/about/media/vision-loss/pages/default.aspx>
- Ellin, A. (2008, December 26). *See Me, Hear Me: A Video Game for the Blind*. Retrieved October 12, 2016, from The New York Times: http://www.nytimes.com/2009/01/04/education/edlife/ideas-blindvideo-t.html?_r=1

- fizzd. (2016, August 8). *Rhythm Doctor*. Retrieved October 17, 2016, from Newgrounds:
<http://www.newgrounds.com/portal/view/608878>
- Gambit. (2007). *AudiOdyssey*. Retrieved October 12, 2016, from Gambit:
<http://gambit.mit.edu/loadgame/audiodysey.php>
- Jaime, J., Barbancho, I., Urdiales, C., Tardon, L. J., & Barbancho, A. M. (2016, April). A new multiformat rhythm game for music tutoring. *Multimedia Tools and Applications*, 4349-4362.
- Unity. (2017). *Company Facts*. Retrieved from Unity: <https://unity3d.com/public-relations>
- Valente, L., Souza, C. S., & Feijó, B. (2009, March). Turn off the graphics: designing non-visual interfaces for mobile phone games. *Journal of the Brazilian Computer Society*, 15(1), 45-58.
- World Health Organization. (2014, August). *Visual impairment and blindness*. Retrieved October 17, 2016, from World Health Organization:
<http://www.who.int/mediacentre/factsheets/fs282/en/>
- Zhou, G. (2015, April 19). *Rhythm Doctor (updated song completion + producer question and answer)*. Retrieved October 18, 2016, from bilibili:
http://www.bilibili.com/video/av2238073/index_2.html